Chapter three

System design and simulation

3.1 Structure.

3.2 Building Frame.

3.3 Gears.

3.4 Pulleys.

3.5 Wheels.

3.6 Inertia.

3.7 Robot Drive.

System design and simulation

3.1 Structure

In this chapter we try to talk about the component which will use in designing our robot, and the items used in building our robot frame, containing the beams, bricks, gears, sensors, motors, wheels…etc. beside to all of the that the brain of the robot which represent in the NXT kit, and type of programming used in our robot. Here in the figure 3.1 shown, the flowchart obviously shows you the function of the robot, and how it works with the program downloaded on the kit "robot brain".

The block diagram in Figure 3.2 also show the component of the robot and its functions, the figure show how the rechargeable battery supply the NXT kit with 9V, which is embedded with a Microcontroller which needs a 5V to operate so that the kit has its own resistance to make drop voltage for proper operating of the Microcontroller.

The sensors that work on 5V, should be connected to the Input ports of the Microcontroller, each sensor has 3 wires, two of them for the power supply and the other one for the electrical signal that which has been changed from physical quantity to electrical quantity, the wires that connect them together called RJ12, we will use for about 5 sensors "sound, light, touch, ultrasonic and rotational sensor (encoder)" each sensor has his own function to do.

The outputs are the motors which are connected to the output ports of the Microcontroller, we have three outputs, and all of them are DC-Servo motors, these motors have a rotational sensor "encoder" embedded in, the motors work on 9V power supply and two of them have 6 wires while the third not, the cable which will connect them called RJ12, two wires for the power supply for the motor and another tow for the power supply of the motor while the last two, one of them to measure the distance in the forwarding, while the last one for measure the distance in backward mode, you will see the motors simulated on the MATLAB, the speed is controlled through the PWM of the H-bridge converter with assistant of the gears and wheels.

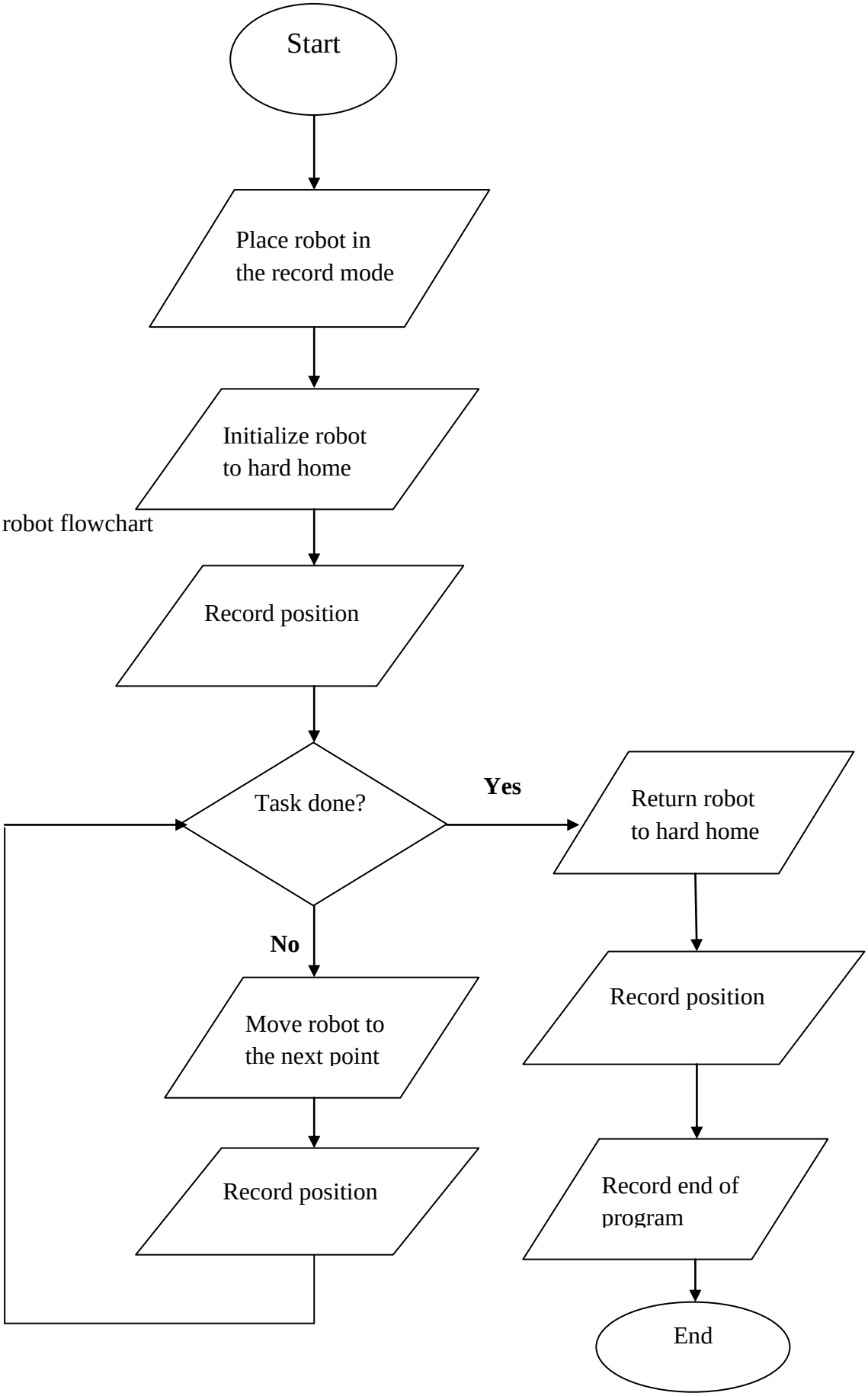

Figure 3.1: robot flowchart

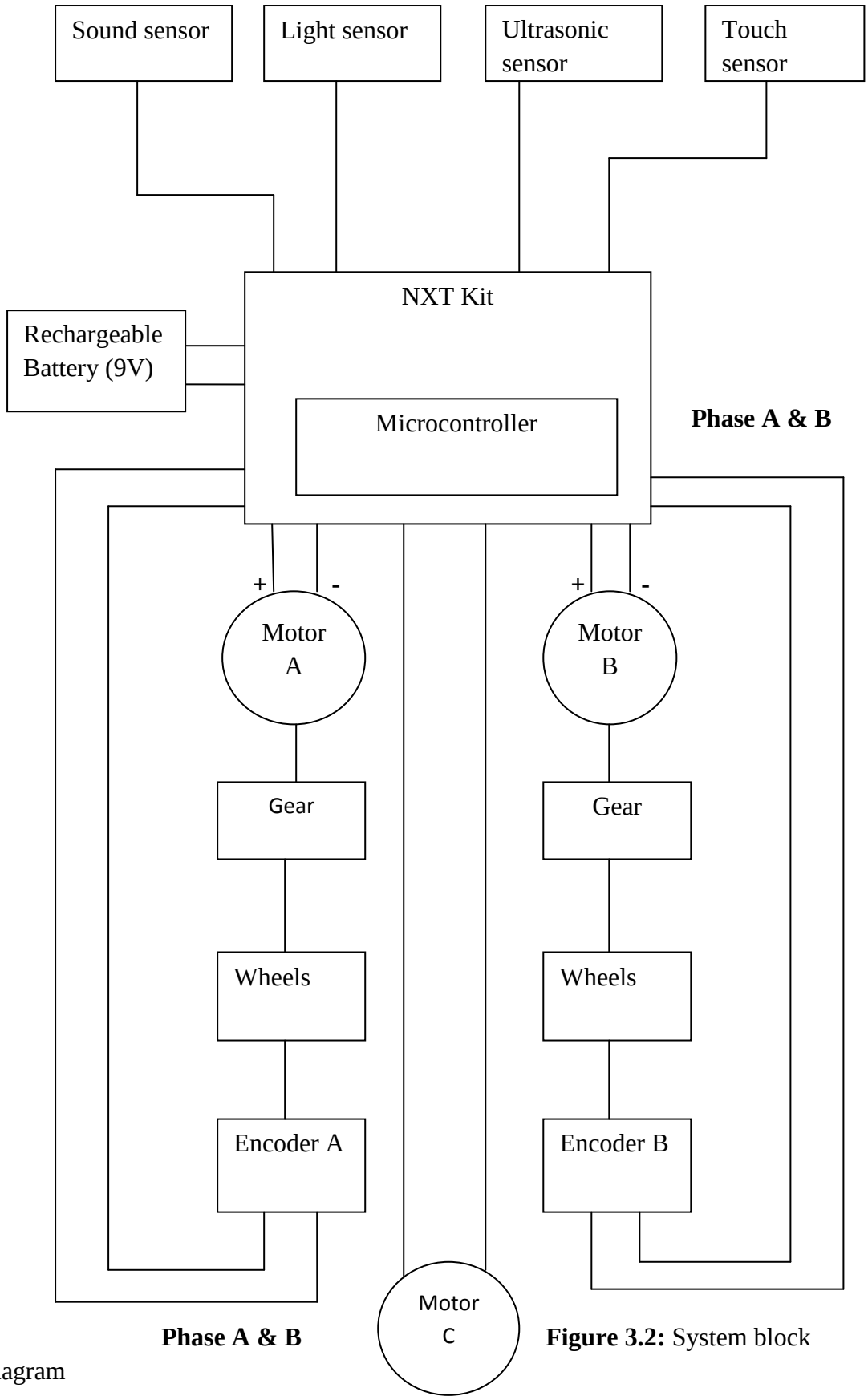

Figure 3.2: System block diagram

3.1.1 Bricks, plates, and beams

Bricks, plates, and beams are not as glamorous as the NXT brick, motors, and sensors. But they are the fundamental components that are used to build the frame that supports the NXT, counteracts the motor's forces, and holds the sensors precisely in place. Master some robot fundamental frames, which give the success for the project, ignoring them will force you to spend more time for repairing your robot than you build it.

3.1.1.1 Bricks

Bricks are made out of ABS plastic. They are injection molded to very exacting tolerances (0.002mm)[1]. The top of the brick is covered with cylindrical plastic bumps called studs. The bottom of the brick has cylindrical holes or tubes. When you snap two bricks together, the tubes deform slightly around the studs, locking the two firmly together.

Figure 3.3: Plastic bricks

3.1.1.1.1 Dimensions

The common practice is to refer to bricks using their dimensions: width, length, and height (though height is often left off when referring to standard sized bricks). When doing this, the width and length dimensions are given in studs. The piece below is a 2 x 4 brick. Bricks are based on the metric system. The 2 x 4 brick above is 16mm wide, 32 mm long, and 9.6mm high (ignoring the studs on top). That works out to 1 stud = 8mm. It also means that bricks are 1.2 studs high. This asymmetry can lead to design and building difficulties as will be discussed later.

[1] From "Jin Sato's Lego Mindstorms The Master's Technique"

3.1.1.2 Plates

Plates are essentially short bricks. They are 1/3 the height of standard bricks--3.2mm or 0.4 studs. Plates use the same naming convention as bricks. Some plates have through holes aligned with the backside tubes. They are referred to as Technical plates, or less obscurely, plates with holes. The holes accept axles and connector pins and make the Technical plates much more useful.

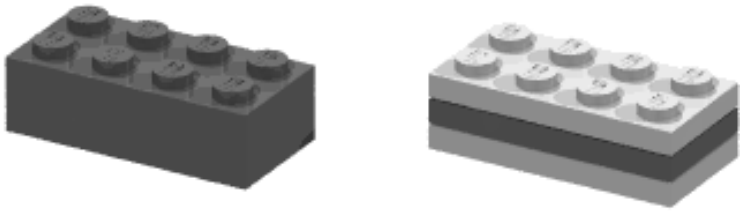

Figure 3.4: three plates = one brick high

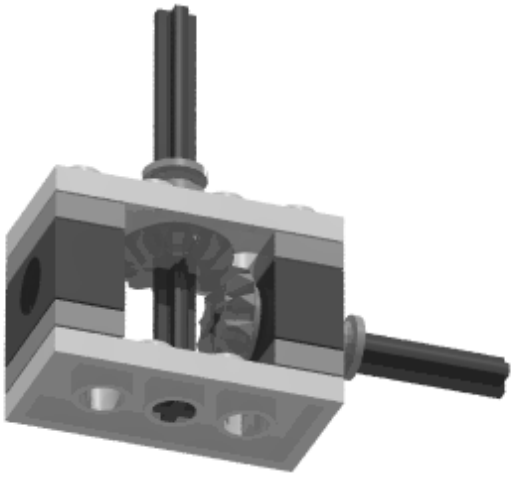

Figure 3.5: simple gear box using Technical plates

3.1.1.3 Beams

A series of complex models for older children to build. Central to Technical are the new beams which are 1x bricks with holes in their sides. The holes are spaced at one-stud intervals and centered between the studs on the top of the beam. The beams can be stacked on top of each other just like bricks. In addition, connector pins can be placed in the side holes allowing the beams to be assembled side by side. The number of assembly techniques available using the new parts is staggering.

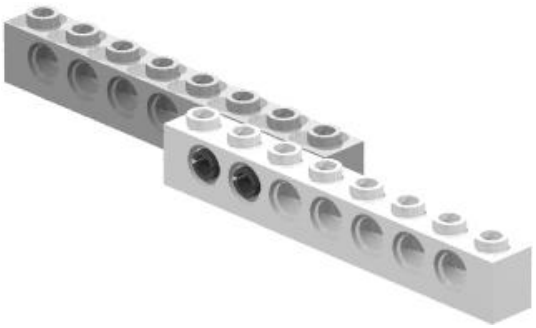

Figure 3.6: technical beams

3.2 Building frame

A robot needs some sort of frame. The frame gives the robot its shape. It provides mounting points for sensors and reacts the forces generated by motors and gears. It is like our skeleton, which gives us our shape, supports our organs, and reacts the forces generated by our muscles. A good frame is strong, lightweight, and holds together even after much use.

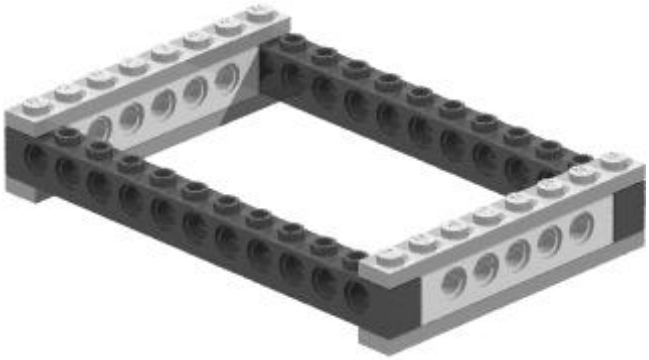

Figure 3.7: simple frame

Figure 3.7 shows a simple frame made out of beams and 1x8 plates. It's strong, lightweight, and the dimensions are appropriate for the base of a robot platform. But it is not very rigid. A gentle push on opposing corners causes the frame to twist out of shape. Eventually the corner connections work loose, and the frame falls apart.

The problem is that the plates do not lock the corners at right angles. There is a small amount of clearance between the ends of the 1 x 6 beams and the sides of the 1 x 12 beams. This allows the studs to act as hinges. Replacing one or more of the 1 x 8 plates with 2 x 8 plates makes the frame much more rigid.

This same technique can be used to build tall frames that are lightweight and strong. Figure 3.8 shows a tall frame that uses friction pins at the corners to attach the vertical and horizontal members. Notice that the 1-2-1 (1 Beam-2 Plates-1 Beam) technique is used to get the proper spacing.

Figure 3.8: A Tall frame

3.2.1 Diagonal bracing

It's very easy to get locked into the mindset that horizontal and vertical are the only ways to build. The shape looks like grid. But diagonal connections are possible as well. Diagonal bracing is trickier to implement than perpendicular cross bracing. Cross bracing can be used on any assembly where the dimension is evenly divisible by two studs, but it can't be used anywhere else. With diagonal bracing there are more solutions, but their derivations are not as obvious.

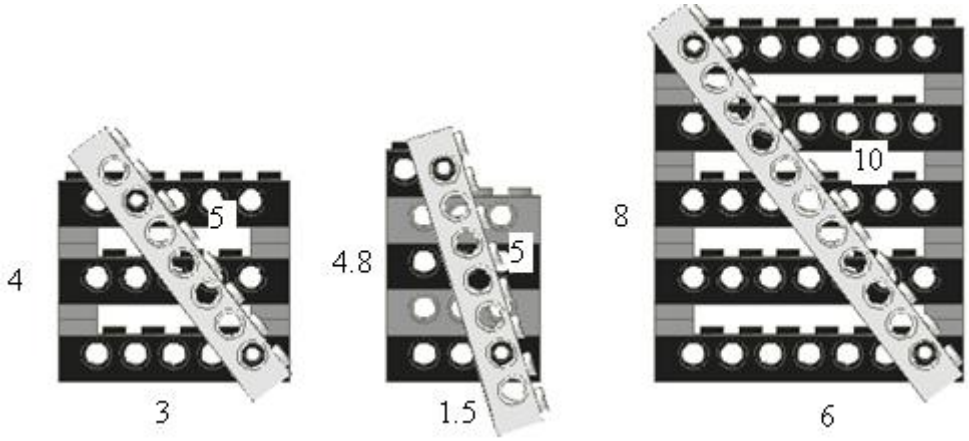

Figure 3.9: diagonal cross bracing

You can find diagonal bracing solutions through experimentation. Lay the diagonal brace on the part, and move it about until you find a place where the holes line up. It can also be done analytically using the Pythagorean Theorem for right

triangles. “The sum of the squares of the legs of a right triangle equals the square of the hypotenuse.” This is often written as “$C = \sqrt{A^2 + B^2}$.” The two legs are the base (width across the bottom) and the height. The hypotenuse is the diagonal beam. Diagonal bracing is possible if the hypotenuse is close to an integer number (less than 0.05studs difference).

3.3 Gears

Eventually you will want to make your robot move. The 9 volt motors provide the motive power, but they may not run at the right speed or be powerful enough. It may also be too difficult to position the motors where they can be directly attached to the wheels. All these problems can be solved using gears.

Gears are generally used for one of the following reasons:

1. to transmit torque from one angle to another.
2. to increase and decrease the speed of rotation.
3. To reverse the direction of rotation.
4. To move rotational motion to a different axis.
5. To change rotary motion to linear motion.
6. To keep the rotation of tow axles synchronized.

3.3.1 Spur gears

A spur gear is used when shafts must rotate in the same plane. In a spur gear the teeth are straight and parallel to the shaft. They are by far the most common type of gears, and they are what most people picture when you mention gears. Our robot includes four different sized spur gears in the Robotics Invention System.

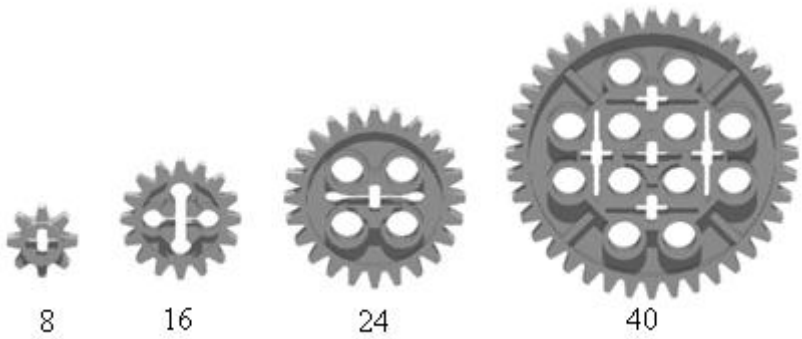

Figure 3.10: spur gears

Gears are normally referred to by their type and the number of teeth. Take, for example, an 8 tooth spur gear. Sometimes a kind of shorthand notation is used where "tooth" is replaced with "t" and the type is not specified for spur gears (because they are so common). For example, a 40 tooth spur gear would be referred to as a 40t gear.

3.3.1.1 Gear spacing

Sizes of spur gears are shown in table 3.1 It is interesting to note that the ratio of the radii is equal to the ratio of the tooth count (8/24 = 0.5/1.5 = 1/3). This is because all the different sized spur gears have the same sized teeth--even the little 8t gear with its involutes profile gear teeth. Having the same sized teeth allows the gears to mesh properly.

Table 3.1: spur gear sizes

Teeth	8	16	24	40
Radius (studs)	0.5	1	1.5	2.5

Knowing the radius of a gear and the number of teeth, we can calculate the size of each tooth. This information can be useful to know when using a spur gear with a gear rack.

Circumference = 2 x Pi x Radius (3.1)

Circumference = Tooth Size x Tooth Count (3.2)

Size x Count = 2 x Pi x Radius (3.3)

Size = 2 x Pi x Radius / Count (3.4)

16 tooth = 2 x Pi x 1 / 16

= Pi / 8 studs

= 0.392 studs or 3.14 mm

Check it out for yourself. The teeth for each spur gear really do evaluate to the same size! There are three different types or shapes of spacing the gears such that the horizontal spacing, vertical spacing, diagonal spacing, which they are presented in the figures below

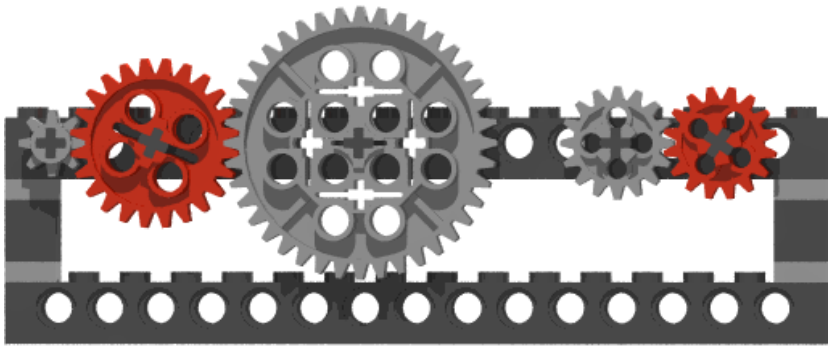

Figure 3.11: Horizontal spacing

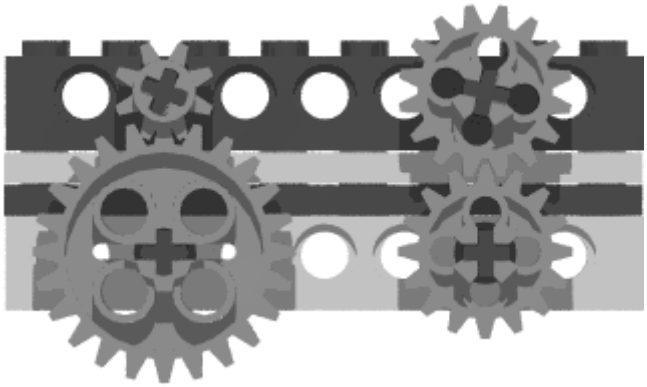

Figure 3.12: Vertical spacing

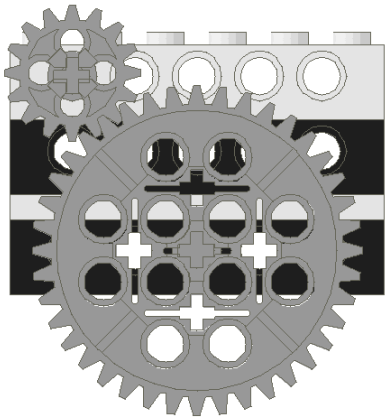

Figure 3.13: diagonal spacing

3.3.2 Gear ratio

Gear ratio is how much the output shaft of a gearbox turns for a given rotation of the input shaft. In figure 3.14, we have a gearbox consisting of two gears: an 8t gear on the input shaft and a 24t gear on the output shaft. If the 8t gear rotates one full revolution then eight of its teeth would pass through the starting line. Because the two gears are meshed, eight of the 24t gear's teeth would also pass the starting line. Since the teeth are evenly distributed around the circumference of the gear, the 24t gear turns 8/24ths or 1/3 of a revolution.

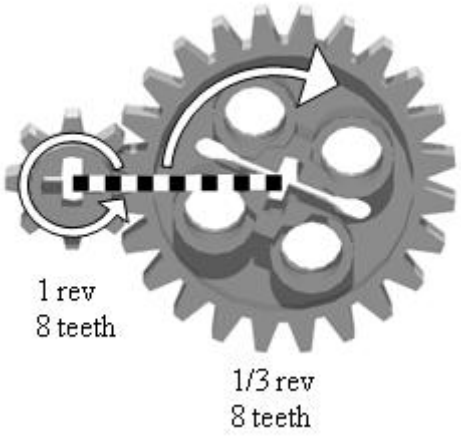

Figure 3.14: Gear ratio

Using the rotation information we can calculate the gear ratio. Following popular convention, it is expressed as a ratio of whole numbers.

Gear ratio = 1 : 1/3

= 3 : 1

The 3:1 gear ratio tells us that the input shaft (attached to the 8t gear) has to complete three full revolutions for the output shaft (attached to the 24t gear) to rotate all the way around just once. Using gears to slow down rate of rotation or decrease the amount of rotation is called gearing down. If we were to switch the 8t and 24t gears around, the output shaft would spin three revolutions for each revolution of the input shaft. This is gearing up, and the gear ratio would be 1:3.

You may have noticed that the gear ratio is the inverse of the ratio of the number of gear teeth. The reason for this is easier to see if we recalculate the gear ratio using an

input shaft rotation of only 1 tooth. In the case of the 8t gear driving the 24t gear, the input shaft would turn 1/8 of a revolution and the output shaft 1/24 of a revolution.

3.3.3 Torque

Torque is a force that tends to rotate or turn things. You generate a torque any time you apply a force to the handle of a wrench. This force creates a torque on the nut, which tends to turn the nut. If the nut is too tight, you either pull harder (more force), or get a longer handled wrench (more distance). From the wrench example, we see that torque is a product of force and distance. The units used when measuring torque reflect this. In the U. S. we measure torque in foot-pounds (ft-lbs). Newton-meters (Nm) is the unit of torque in the metric system.

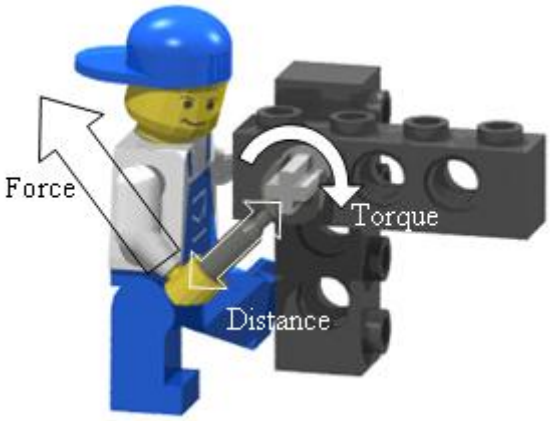

Figure 3.15: Torque relation with the distance and force

Gears operate by transmitting forces at the teeth of the gear. In the 24t gear is generating a force against the teeth of the 40t gear. The force (f) is equal to the torque (t1) applied to the 24t gear divided by the radius (r1). The force transmitted by a gear is inversely proportional to the gear's radius. The larger the radius of the gear, the less force it will generate for a given torque.

From the discussion earlier, we know that the gear ratio is the inverse of the ratio of the gears' radii. This allows us to use the gear ratio to calculate the torque amplification of a gear system. Using the gear ratio for the example above:

Gear ratio = r2 : r1 (3.5)

= r2 : r1

= 5 / 3

t2 = t1* r2 / r1

= t1 * gear ratio

3.3.4 Speed

When using simple machines like gears (or any kind of machine for that matter) you never get something for nothing. In our prior example, we used gears to increase torque. What we traded to get the torque increase was speed. The output shaft may turn stronger, but it also turns slower.

If we measure the angles in radians (1 radian = 180 degrees / Pi = 1 revolution / (2 x Pi)), the tooth velocity of a gear (v) is equal to the angular velocity (ω) times the radius (r). There is a proportional relationship between the radius and the tooth velocity. At a given angular velocity, the teeth of a larger gear will travel faster than the teeth of a smaller gear.

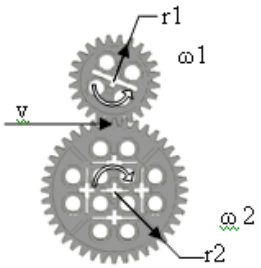

Figure 3.16: the relation between the speed and the ratio

$$V = \omega 1 \times r1 \quad (3.7)$$

$$\omega 2 = v / r2 \quad (3.8)$$

$$\omega 2 = (\omega 1 \times r1) / r2 \quad (3.9)$$

$$\omega 2 / \omega 1 = r1 / r2. \quad (3.10)$$

The larger gear spins more slowly than the smaller gear. The ratio of the angular velocities is equal to the inverse of the ratio of the radii. Knowing this we can calculate the angular velocity using the radius ratio directly.

3.3.5 Backslash

The backlash of a gear train is the amount the input shaft can rotate without moving the output shaft. Backlash is caused by the gears not meshing perfectly Figure 3.17. In this example, when gear A reverses rotation, the tooth on gear B goes from being loaded on the left side to being loaded on the right side. Because of the gap between the teeth, A will be able to rotate a small amount before B notices the change in direction.

Backlash introduces discontinuity, uncertainty, and impact in mechanical systems. This makes accurate control difficult. Positioning accuracy is also compromised due to backlash. A large amount of backlash makes a robot feel sloppy and gives an overall impression of poor engineering (you want to impress those judges).

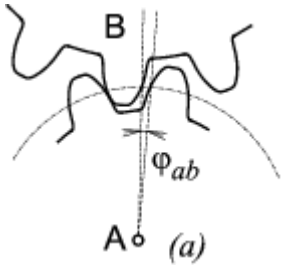

Figure 3.17: Backslash caused by poor meshing

In industry, backlash is reduced by using specially mated gears that are designed to mesh perfectly. Gears are designed to work with a wide variety of gears of different sizes and designs. This limits how perfectly they can fit together. Luckily, there are other ways to minimize backlash and its effects.

Reducing backslash:

1. Place rotation sensor near the output shat, this minimizes the effect backslash has on the sensor readings.
2. Use large gears. Backslash is less noticeable in larger gears.
3. Minimize the number of gears in a gear train, the larger the number of meshes, the greater the backslash.
4. The backslash increases when you gear up and decreases when you gear down.

An interesting alternative is to prevent backlash from occurring. You can take the play out of a gear train by applying a torque to the output shaft, trading some input torque for more precision. If the torque is great enough, it prevents backlash from happening when changing direction.

3.4 Pulleys

A pulley is a wheel with a groove about its diameter. The groove, called the race, accepts a belt which attaches the pulley to other pulleys. As the pulley rotates, friction forces pull on the belt putting it in tension. The belt transmits the force to the

other pulley causing it to rotate.

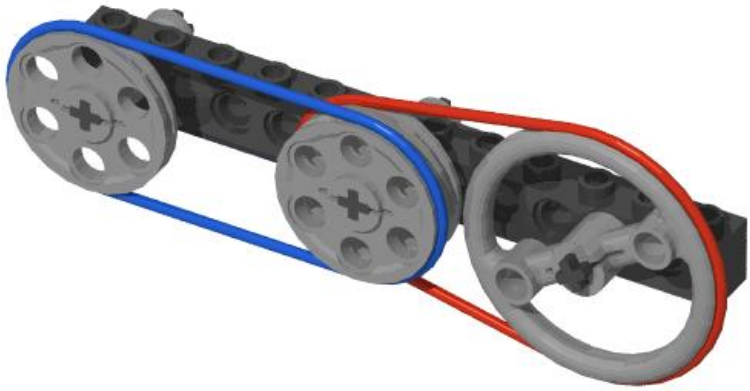

Figure 3.18: Pulleys and belts

Look to these four sizes of pulleys below, it's possible to "gear up" and "gear down", the reduction ratio of pulleys are determined by the ratio between their diameters. The trick is determining exactly where to measure the diameters

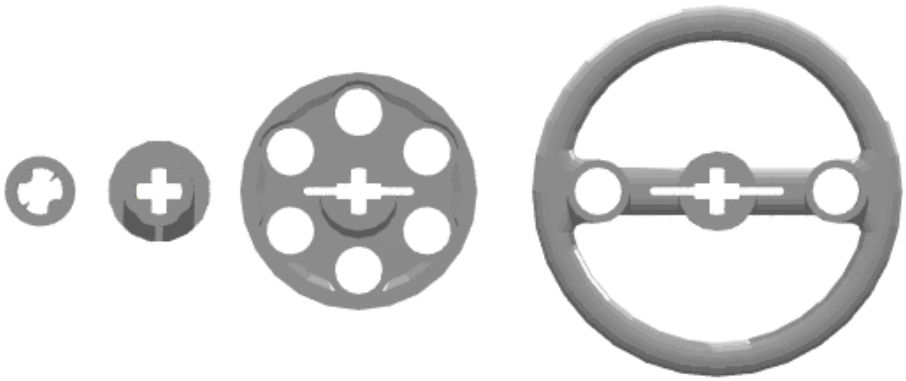

Figure 3.19: Pulleys with different diameters.

3.5 Wheels

Challenges are involved moving around and picking up things, moving around and dropping things, moving around and triggering things. All this moving around requires some sort of mobile platform, and most of the mobile platforms use wheels.

Wheels support the weight of the robot and transmit the power of the motors to the ground. What wheels you use will affect your robot's speed, power, accuracy,

and ability to handle variations in terrain. Wheel choices will have a profound effect on your robot's success or failure.

3.5.1 Sizes

There are three sizes of solid rubber tires which all fit on the same plastic wheel and three sizes of balloon tires that fit on differently sized hubs. Dimensions in millimeters are stamped on the sidewall of the balloon tires. Dimensions for the solid tires are not available; therefore, approximate dimensions are supplied in the figure below.

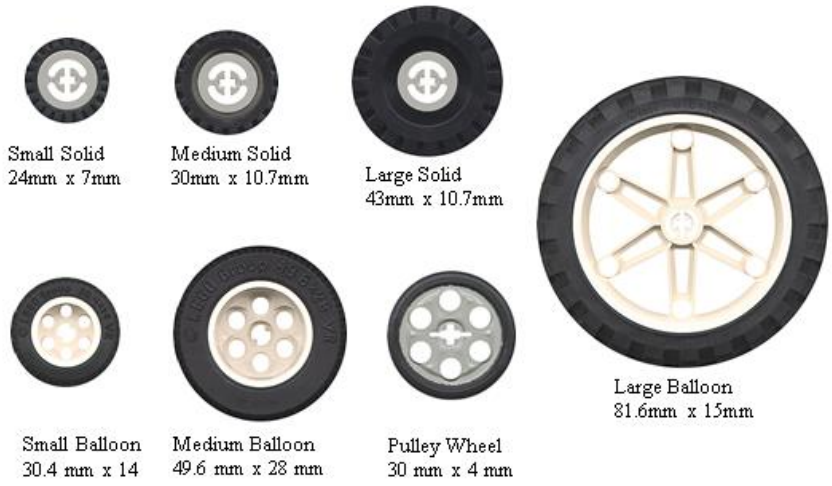

Figure 3.20: robot wheel sizes

The diameter of the wheels chosen will, in a large part, determine how fast and powerful your robot is. Large wheels will make a robot run faster but decrease its towing capacity. Small wheels will give you more power but with a corresponding decrease in speed.

$$\text{Circumference} = \text{Pi} * \text{diameter} \quad (3.11)$$

$$v = \omega \times \text{Pi} \times d \times a \quad (3.12)$$

ω: the angular velocity, d: the diameter of the wheel, v: the linear velocity, a: gear ratio.

3.6 Balance

Proper balance is very important in robot design. For your robot to move in a predictable and repeatable manner, all wheels must be in contact with the ground at all times, and the weight carried by each wheel must be consistent. Improper balance

will result in a robot that tips over easily or lifts its wheels when accelerating or turning.

Balance is dependent upon two factors: wheel base and center of gravity. Center of gravity (CG) is the point within your robot where there are equal amounts of mass above and below, equal amounts to the left and right, and equal amounts fore and aft. For stability calculations, we pretend that all the mass of your robot is concentrated in this tiny spot. The wheel base is the region outlined when you play connects the dots with the points where your robot's wheels touch the ground.

3.6.1 Inertia

Determining the vertical component of the center of gravity is more difficult than finding the left/right center or the fore/aft center. So why bother doing it? After all, if you know that the CG (central of gravity) is near the center of the wheelbase, then the robot is guaranteed to be stable, right? Well, not always. Figure 3.21 shows how climbing an incline moves the effective CG. The robot with the lower CG is still stable, but climbing the incline moves the CG behind the rear wheels on the robot with the higher CG, causing it to tip over backwards.

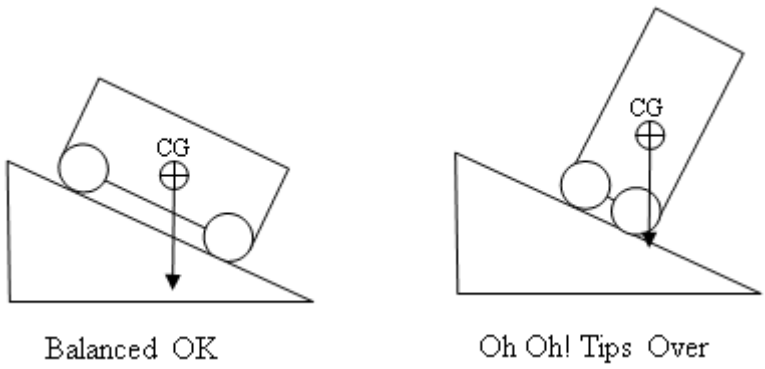

Figure 3.21: an incline moves to effective CG

3.6.2 Finding the center of gravity

Determining your robot's CG is an informative and educational activity. There are many ways this can be done, ranging from the dry and boring (summing the moment of inertia for each component and dividing by the total mass) to the slightly frightening (hanging the robot from a string). A fairly safe and easy way to locate the CG is the balance method.

To determine the CG using the balance method you need to locate the balance point on the lateral, longitudinal, and vertical axes. Each balance point defines a plane upon which the CG resides. The CG is located at the intersection of the three planes.

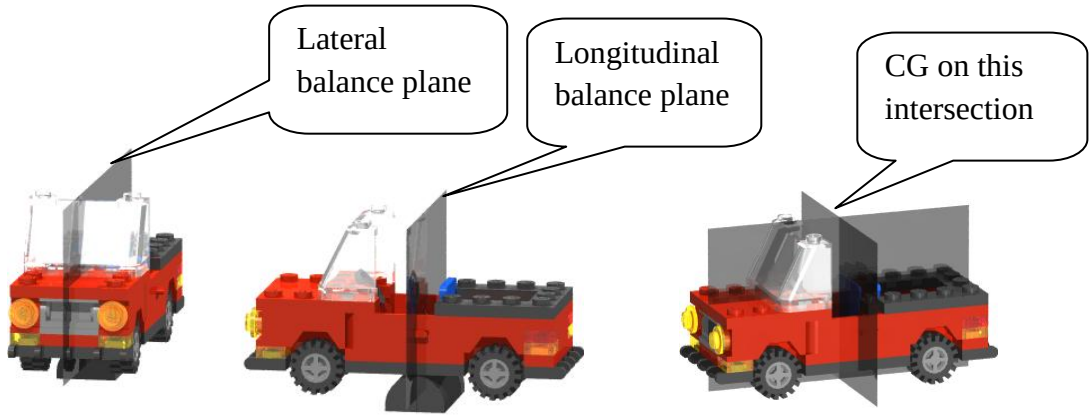

Figure 3.22: finding CG using balance method

To find the balance points, you need a fulcrum that will support the weight of the robot while still allowing it to tip easily. Look to the intersection is the fulcrum in the figure above. Place the robot on the fulcrum such that the fulcrum is parallel to the balance plane you are trying to locate. Slowly adjust the position of the fulcrum until the robot balances. This is the balance point. Repeat for the other two axes to find the CG.

3.7 Robot drive

3.7.1 Sensors

Without sensors, a robot is just a machine. Robots need sensors to deduce what is happening in their world and to be able to react to changing situations. This section introduces a variety of robotic sensors, explaining electrical use and practical application. Please do not be limited by the ideas contained in this section! The sensor applications presented here are not meant to be exhaustive, but merely to suggest some of the possibilities. Assembly instructions.

A standard plug configuration has been developed to connect sensors to the robot board (the brain) board, as shown in Figure 3.23 Notice that one pin is removed from the plug, making it asymmetric and therefore polarized. This means that once the plug is wired correctly, it cannot be inserted into a sensor port backwards. This makes the plug much easier to use. The sensor is connected to the plug with three

wires. Two of the wires are used to supply power from the battery or any power supply to the sensor. These are the wires labeled \+5v" and \Gnd." The third wire, labeled \Signal" is the voltage output of the sensor. It's the job of the sensor to use the power wires (if necessary) and return its \answer", as a voltage.

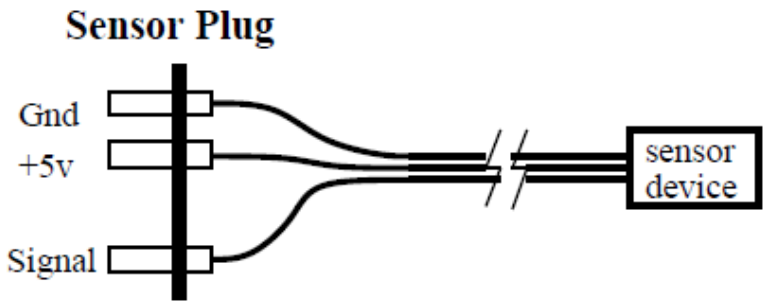

Figure 3.23: wiring a general sensor

3.7.1.1 Touch sensor

The most primitive, but often very useful, sensor is a touch switch. It is simply a pushbutton or other momentary switch that is mounted on a robot so that when the robot runs into something, the switch is triggered. The robot can detect that it has made contact with some object. Touch sensors used as collision detectors are imperfect in that it is hard to design a touch switch and bumper mechanism that can detect contact of any object from any angle. Creative mechanical design in implementing the bumper mechanism is important.

3.7.1.2 Bumpers

A bumper is a device that notifies your robot that you ran into an obstacle. When struck, the bumper moves and presses or releases a touch sensor, notifying your robot of the impact. Bumpers are the most common use for a touch sensor. This may be due to the Constructor using touch sensors primarily in this role, or it could be because bumpers are such useful devices.

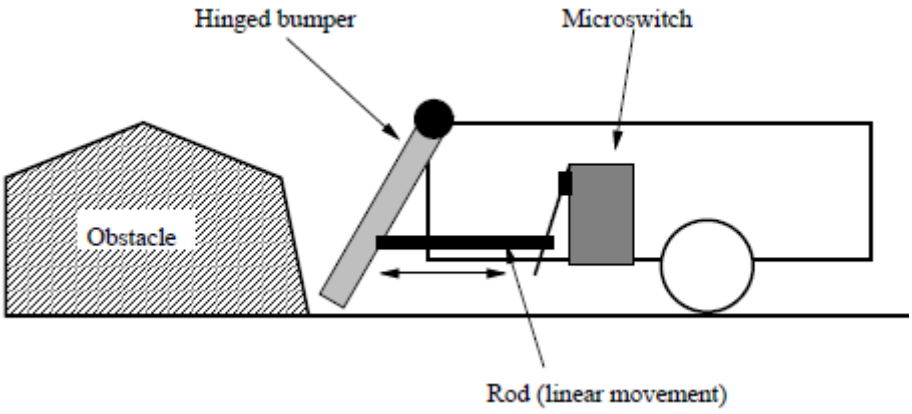

Figure 3.24: bumpers coupled to touch sensor

A bumper which shown in figure below assembly usually consists of four parts: the bumper, a sensor, a return mechanism, and a supporting framework. The bumper component is the part that receives the impact and converts it to motion that can be detected by the sensor. The sensor component is usually a touch sensor, but bumpers can be built using rotation sensors or light sensors

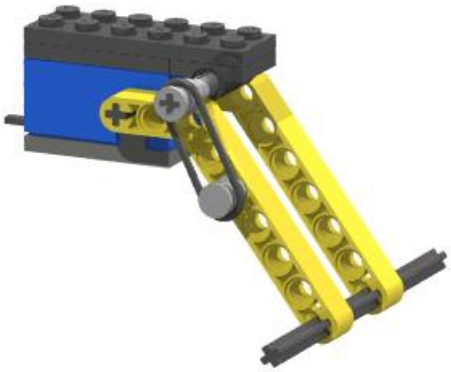

Figure 3.25: bumper uses rotation sensor

3.7.1.3 Light sensor

The light sensor contains a red light emitting diode (LED) and a photo transistor. The red LED illuminates the area in front, and the photo transistor measures the intensity of the reflected light. The light sensor returns a value in the range of 0 to 100%. The lowest reading ever seen is 20%, which could be taken at midnight with the lights off. You can get to 100% by aiming the sensor at the sun on a clear day or by holding it within a few inches of a 100W light bulb. In normal operation, the sensor values tend to be between 30 and 60%.

Figure 3.26 light sensor

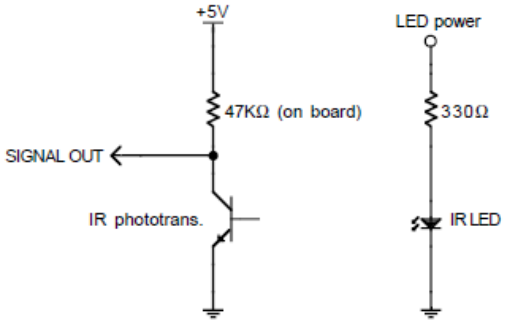

Figure 3.27: Light sensor with infrared emitter and photo transistor

3.7.1.4 The Robot Encoders

The encoder is a sensor attached to a rotating object (such as a wheel or motor) to measure rotation. By measuring rotation your robot can do things such as determine displacement, velocity, acceleration, or the angle of a rotating sensor. A typical encoder uses optical sensor(s), a moving mechanical component, and a special reflector to provide a series of electrical pulses to your microcontroller. These pulses can be used as part of a PID feedback control system to determine translation distance, rotational velocity, and/or angle of a moving robot or robot part.

For instance, if you have a wheel rotating, and you want to measure the time it takes to rotate exactly 40 degrees, or know when you have traveled X distance, you would use an encoder. The sensor would be fixed on your robot, and the mechanical part (the encoder wheel) would rotate with the wheel. The output of an encoder would be a square wave, so if you hook up this signal to a digital counter or microcontroller you can then count the pulses. Knowing the distance/angle between each pulse, and the time from start to finish, you can easily determine position or angle or velocity or whatever. Encoders are necessary for making robot arms, and very useful for acceleration control of heavier robots. They are also commonly used for **maze navigation**.

Calculating the robot motion with an encoder:

To do this calculation you need more information, such as **wheel diameter** and **encoder resolution** (number of clicks per 360 degrees, or counts per revolution). Starting off, you should know two things - wheel circumference and counts per revolution. Dividing the two, you can easily figure out the distance your robot travels between each encoder click:

Wheel circumference / counts per revolution = distance traveled per encoder count. Eq3.13

Now velocity is just distance divided by time . . . So using the answer in the above equation, divide that by the time passed determined from your microcontroller timer:

Distance traveled per encoder count / time = velocity. Eq3.14

After you know distance and velocity, you must then run a PID feedback control algorithm so that your robot can match a desired (pre-determined) distance and velocity.

Figure 3.28: Rotary Encoder

The Encoder can measure the distance or speed in both directions as we discussed in **Section2.4.** The Figure 3.29 below shows the Encoder simulation on MATLAB using the XPC target tools in addition to real time tools, while the two other figures are the results

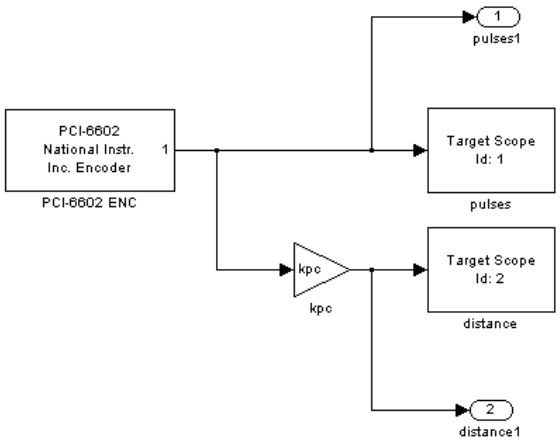

Figure 3.29: Encoder simulation on MATLAB

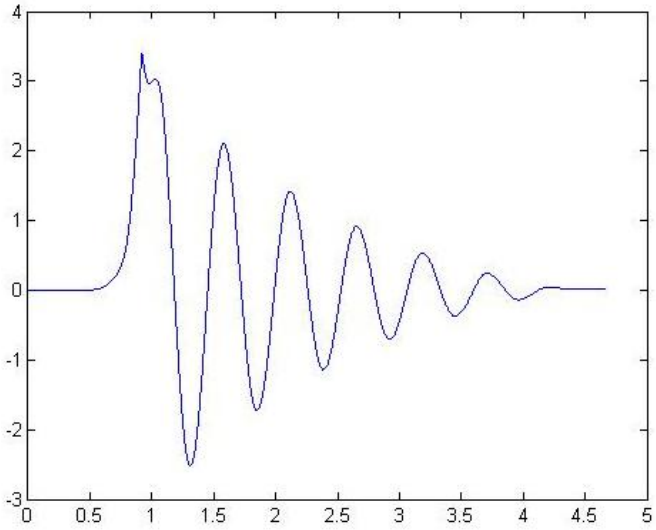

Figure 3.30: distance vs. time

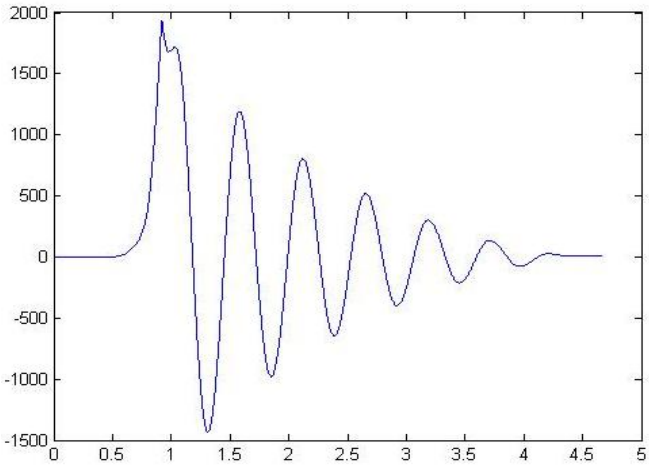

Figure 3.31: pulses vs. time

3.7.1.5 Ultrasonic sensor

This kind of sensor used to measure the distance from the robot by sending ultrasound wave, these waves couldn't been heard by human being according to range that we could hear between 20HZ to 22KHZ, after sending the waves from the transmitter of the Ultrasonic sensor they will be back to the receiver if there is an obstacle in front if not no waves come back and the robot keep moving forward, this sensor work on 5volt power supply and according to this equation we can measure the distance, in addition to the MATLAB simulation using XPC target tools.

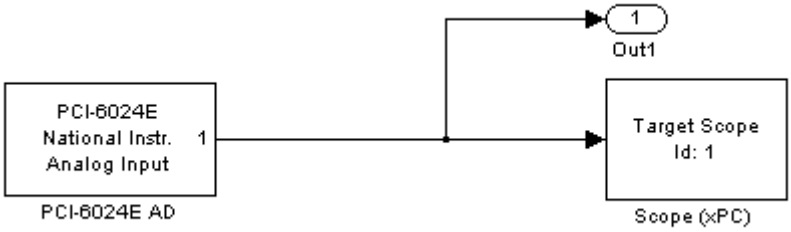

Figure 3.32: Ultrasonic sensor simulation

Speed = 2*Distance / Time. (3.15)

Distance = Speed * Time / 2. (3.16)

3.7.2 Motors and controlling modes

In this section we try to talking about the motors which we will use in our robot design, we will talk about the DC permanent servo motors with controlling encoder.

Servo are intended to be used in close loop speed and positional control systems where performance requirements are such that cannot be achieved by normal dc motor, the servo are designed to achieve good dynamic performance and steady state accuracy, its designed to achieve the same performance in both direction, high torque to inertia ratio, low friction and smooth ripple, free torque, in some servo, inertia is reduced by reducing the diameter and increasing length for the same rating. Small servos are usually with permanent magnet type. In this project we will use two small servos for two steering wheels.

As we mentioned formerly in the second chapter about the DC servo motor, we here try to clear the idea obviously, the figure 2.1.1 in the second chapter explain obviously the construction of the motor.

$$u = E + R_a.i + L_a.(dI_a / dt) \quad (3.1)$$

$$T_e = K_e.i \quad (3.2)$$

$$j = j_m + j_l \quad (3.3)$$

$$T_f = sin(\omega).T_f + K_f.\omega \quad (3.4)$$

$$T_e = T_f + T_l + j.(d\omega / dt) \quad (3.5)$$

Where

T_e *: The electrical torque*

T_f *: The friction torque*

T_l *: Resistant torque*

j_m *: The motor inertia*

K_e *: Torque constant*

K_f *: The viscous friction*

j_l *: Load inertia*

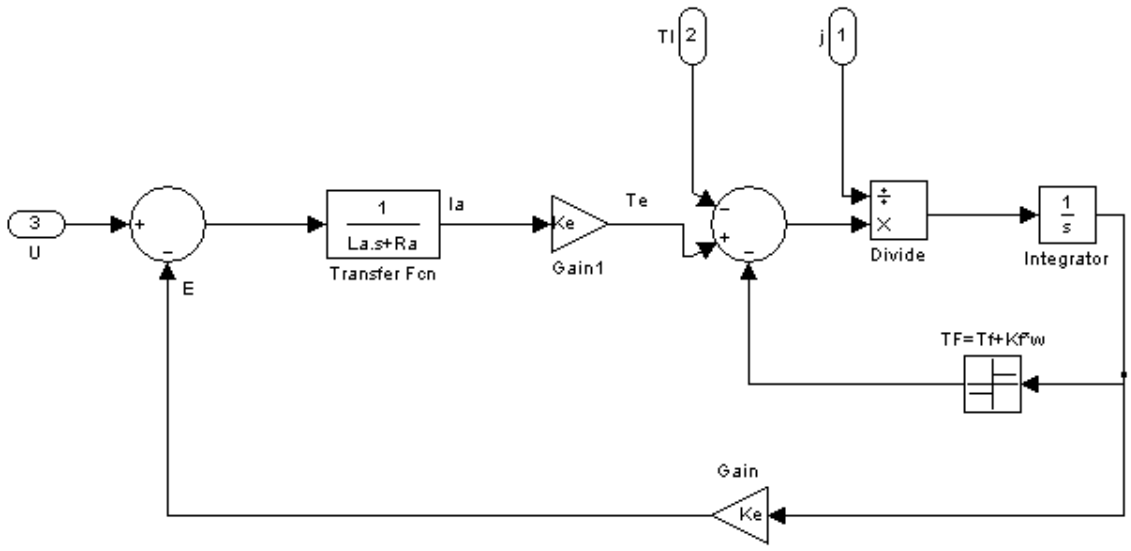

Figure 3.33: electric circuit of the armature

3.7.2.1 DC-Servo permanent magnet Motor simulation

Here we try to simulate the DC permanent magnet motor with driving mechanic load which we used in our project using the mat lab, referring to the figure 3.7.2.1 and using the equation below, and referring to the eq.(2.1)

By the simulation shown in this figure we connect the constant blocks to the inputs *U (10 volts), j(0.05 Kg.m^2) and T_l(4 N.m),* the internal parameters R_a*(0.5 ohm),* L_a*(10 mH)* K_e*(0.5 N.m/A)* and K_f*(0.001 N.m.s).*

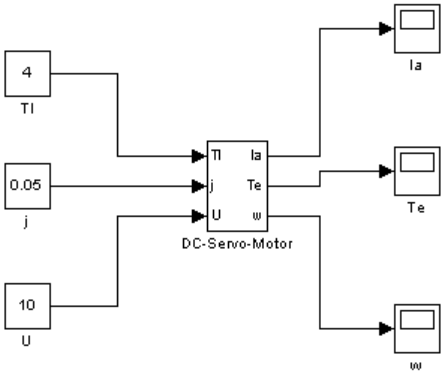

Figure 3.34: the permanent magnet DC-Servo motor simulation on MATLAB

The model after using the PID controller shown in figure below

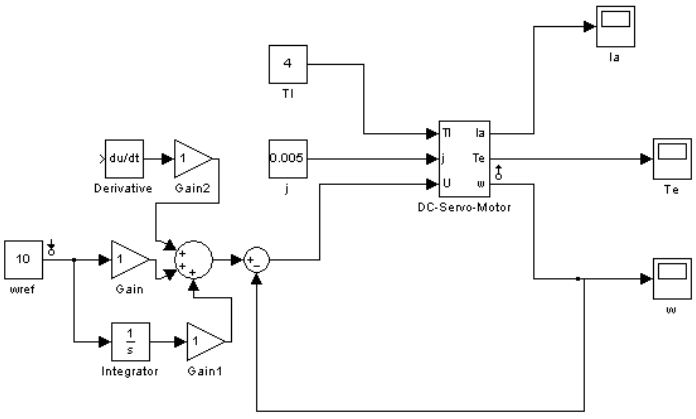

Figure 3.35: the model using PID controller

3.7.2.2 Speed control

In controlling the DC-motors there are three common methods, The **External resistance** which is useless in controlling robot motor because it has power dissipation with affect on Motor current, the second way is **Armature voltage** with is useless here. The speed control of our robot is depending on usage the H-bridge Technique, because it's useful in our application.

The NXT has three modes of controlling the motor--Off, On, and Float. In Float mode, the motor is not driven and can spin freely. It behaves just as it would if the connecting leads were removed. In Off mode, the NXT internally shorts the contacts of the output connector. This has a braking effect on the attached motor causing it to quickly stop spinning. In On mode, the NXT sends the motor either +9V or -9V to make the motor spin clockwise or counter clockwise. Pulse width modulation (PWM) is used to provide eight power level settings.

Pulse width modulation (as shown in **Figure3.30)** is an inexpensive way to control power output. Instead of setting the output voltage (voltage regulators are expensive), the NXT quickly switches the power on and off. Different power levels are achieved by varying the percentage of time that the power is on (this is called the duty cycle). On the NXT, the minimum power setting is zero, which corresponds to a duty cycle of 12.5% (the power is on 12.5% of the time). At level 7, power is supplied continuously (100% duty cycle).

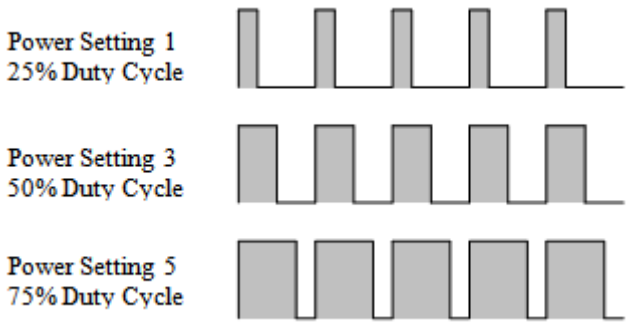

Figure 3.36: PWM duty cycle

The figure below shows the speed characteristic with time and step input after Linearize the model, the next figure also show the speed response to time with j=0.005.

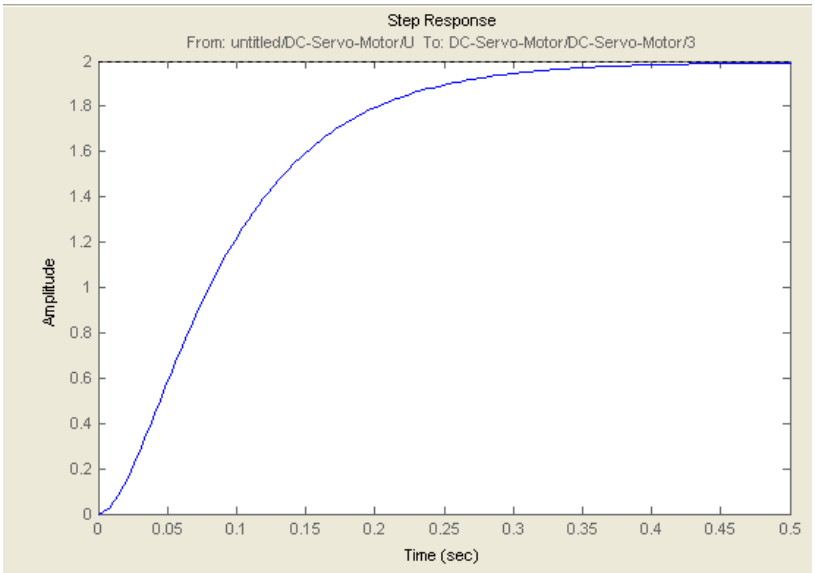

Figure 3.37: speed response versus the time at j=0.05.

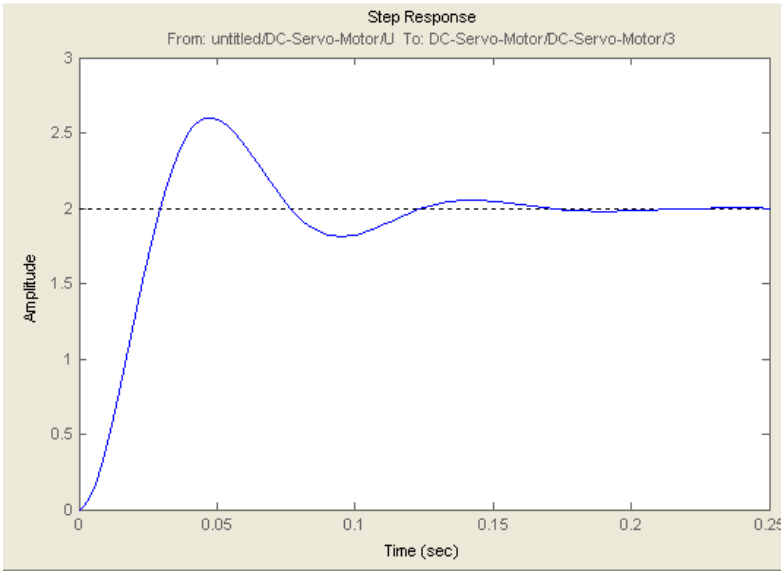

Figure 3.38: speed vs. time at j=0.005.

The figure below shows the connection of the DC-Servo motor on the H-bridge and simulates the system on the MATLAB, using the PWM and the PID controller.

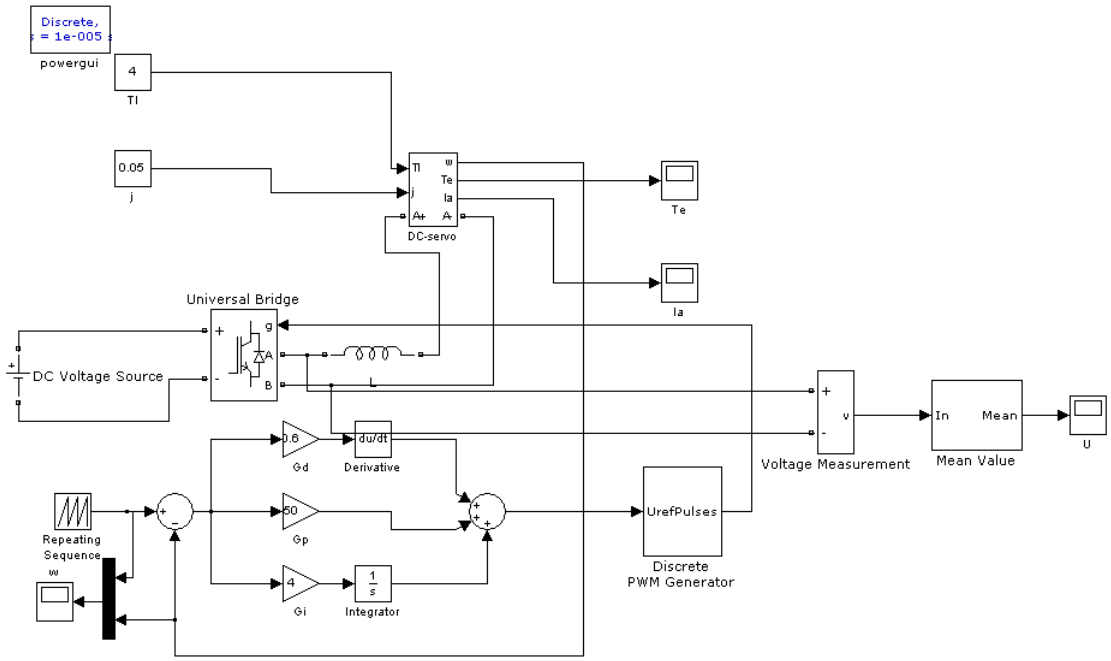

Figure 3.39: System Simulation on MATLAB

3.7.2.3 Motor Starting

Maximum current that dc motor can safely carry during starting is limited by the maximum current that can be commutated without sparking, for normally designed machines, twice the rated current can be allowed to flow and for specially designed machines it can be 3.5 times.

If a dc motor is started with full supply voltage across its terminals, a very high current will flow, which may damage the motor due to heavy sparking at the commutator and heating of winding. There for it's necessary to limit the current to a safe value during starting. When motor speed controlled by armature voltage control as mentioned before the controller which controls the speed can also be used for limiting the motor current during starting. In absence of such controller, a variable resistance is used by connecting it in series with the armature resistance and by changing the value of the resistance we can make a soft start; **Figure 3.31** shows the current behavior after simulation on MATLAB.

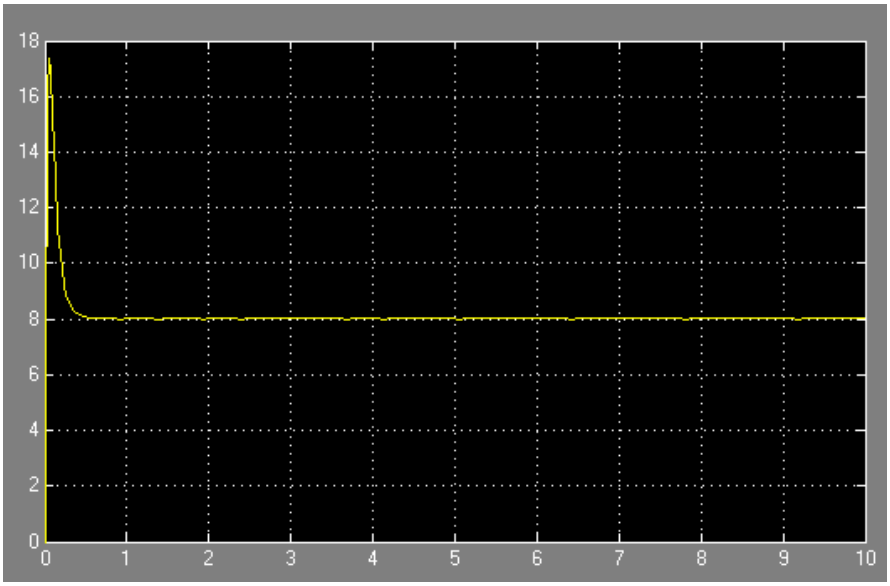

Figure 3.40: current behavior of the motor.

3.7.2.4 DC-Motor Braking

In braking the motor works as a generator developing a negative torque which opposes the motion. It is of three types, each way has its own benefits and draw backs, so I will go over each for you to decide which is best.

3.7.2.4.1 Controls Method:

This method requires an encoder placed onto a rotating part of your DC motor. You will have to write an algorithm that determines the current velocity of your motor, and sends a reverse command to your H-bridge until the final velocity equals zero. This method can let your robot balance motionless on a steep hill just by applying a reverse current to your motors.

3.7.2.4.2 Mechanical Method:

The mechanical method is what is used on cars today. Basically you need something with very high **friction** and wear resistance, and then push it as strongly as possible to your wheel or axle. A servo actuated brake works well.

3.7.2.4.3 Electronic Method:

This method is probably the least reliable, but the easiest to implement. The basic concept of this is that if you short the power and ground leads of your motor, the inductance created by your motor in one direction will power your motor in the opposite direction. Although your motor will still rotate, it will greatly resist the rotation. No controls or sensors or any circuits overheating. The disadvantage is that the effect of braking is determined by the motor you are using. Some motors brake better than others.

3.7.3 Motor source

The source of our robot motor is a rechargeable battery, 6 AA batteries these battery delivered DC voltage (9V), Robots may be powered by a variety of methods. For a small robot, however, battery power delivers a number of advantages over any other method. Batteries are cheap, relatively safe, small, and easy to use. Also, there are many different types of batteries, each with its own tradeoffs.

3.7.4 Robot electronics

The NXT is our robot brain, a tiny computer shaped, at the core of the NXT is a microcontroller running at 5 to 20MHZ with 32K of RAM. This may seem anemic when compared to modern desktop computers with 2GHz 32 bit processors and 512 Mbytes of memory, but it's as powerful as the Apple II computer I learned to program on and significantly more powerful than the computers that sent men to the moon.

The microcontroller is used to control three voltage outputs, three sensor inputs, and an infrared serial communications port. Snap-on wire leads are used to connect the NXT to motors, lamps, touch sensors, light sensors, rotation sensors, etc… The serial communication port is used to download programs from a PC and can be used to communicate between two NXT's.

The NXT is powered by batteries. Older versions of the NXT also had a 9V DC input plug for use with an optional AC adapter. If you have this model of NXT, the transformer will pay for itself in saved battery costs. Having two or three sets of good quality rechargeable batteries is also a good idea. The rechargeable alkaline batteries seem to work best with the NXT.

3.7.4.1 Programming

The NXT is programmed using special software that resides on your PC. The program is translated into byte codes and transferred to the NXT via an infrared serial link (using the IR tower). This method of programming--writing code on one type of computer to be run on another type of computer--is called cross development. A computer without a keyboard and monitor, like the NXT, is called an embedded system. Programs are written by picking instruction blocks and snapping them together.

Figure 3.41: The NXT programmable kit

3.8 Table design

The table with the robot will be set on it is made of wood with the proper dimensions(2.5m X 1.20m), in addition to that the lines painted on the table which considered as the robot guider such as the point that the robot will initialize and the homing the site of the obstacle, …etc.

3.9 Conclusion

The mobile robot is useful everywhere, they are so important to diminish the fault of position accuracy, so that in our robot we use the servo motors which they have good performance in position control by closed loop system, and give you very good results in position accuracy, we should notice the gears meshing in order to avoid the error estimation of the speed, in addition to avoid the sensor errors.

This robot due to cost of building it, you can use it in many application in our practical life in addition to the industrial aspects but you need to improve the size in addition to high power level, this robot in the future could be able to instead of the laborers in the factory and do works the laborers themselves couldn’t do.

3.10 Recommendations

There are several problems with using encoders for robot position control. First, just because your wheel rotates does not mean your robot is moving. Ever driven a car in snow? Error can quickly build up. This is why it is not recommended to use encoders for position feedback of your robot (such as in mazes). Second, encoders have a finite accuracy.

There are several things you need to watch out for when designing encoders. First, keep ambient light out of your sensor, such as sunlight. If light shines in to your sensor, it could potentially read false clicks. High resolution encoders for velocity control can take a lot of computational system time, so it is better to use a **digital counter IC** to count encoder clicks than to have your microcontroller count clicks. Your controller can read the counter value serially when it pleases instead.

Instead of using the NXT intelligent brick, you can use another IC's that do the job, you can use the PIC18F4550 Microcontroller with pins and using the datasheet you can the function of each pin, in addition to that you need H-bridge IC, and you need an Isolation IC to protect the PIC from being damaged, such as the Opt-isolator ILD74, but the disadvantage of these IC's is so hard to be programmed, the stepper motor could be used instead of using the Servo.

In this project you can connect a wireless camera in order to take photos of anything you want so that you will use a cable to connect the receiver of the camera with the NXT brick, and program the robot to do the proper task.

In order to get more efficiency and to let this work to be used in the indusrial factories we should improve the actuating to deliver more power for the heavy loads, such as improve the manipulator to work on hydraulic power or in DC 100 volt motor

To make this Robot automatically without pressing on the switch button to work, you can cotrol him by the remote by sending a signal through the remote control to the Robot and the Robot response to the order, or you can use the Bluetooth Technique.

3.11 Additional results

There are many kinds of gears in the life, but we got enough to talk about the spur gears because used widely in decreasing and increasing the speed, the gears are so close to the electrical transformer principles.

While attaching the gears together we should avoid the backslash problem which results from poor meshing of the gears. Backslash produce discontinuity and affect on the mechanical component and doesn't give the best accuracy for controlling so that to solve this problem we try to use big gears, as we mentioned formerly in the gears section.

In our robot we can use treads instead of using wheels, the treads are popular in robots, and they have good traction on the rough terrain, they can cross ravines that would stop the wheeled vehicle, but unfortunately treads have number of disadvantages, they have fairly poor traction on the smooth surfaces. Also there is a lot power loss due to continuous bending and straitening of the treads as they roll around the sprocket wheels, for these reasons mentioned we use the wheels which give the robot the good speed it needs on any surfaces.

In programming on the NXT kit it is turned on, the NXT is receptive to messages coming in on its infrared serial port. With all the activity and congestion at competitions, it is quite possible that another team could unintentionally overwrite the programs stored in your NXT. Care should be taken to block the IR port when not in use.

It appears that the light sensor we could use is very sensitive to ambient light, which is disconcerting because you seldom have control over the lighting. So be careful when construction the robot using light sensors.

References

1. Fred G.Martin, The 6.270 Builder's guide second edition,The Massachusetts Institute of Technology, December 1992.

2. Thomas R. Kurfess Ph.D., P.E, Robotics and automation handbook, Printed in the United States of America, CRC Press LLC, 2000 N.W. Corporate Blvd., Boca Raton, Florida.

3. Gordon mccomb and myke predko, Robot builder's bonanza, The McGraw-Hill Companies, united states of America. 2006.

4. Ben-Zion sandler, Robotics designing the mechanisms for automated machinery second edition, Ben-Gurion University of the Negev, Beersheva, Israel.

Hyper links

http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf

http://services.eng.uts.edu.au/cempe/subjects_JGZ/ems/ems_ch12_nt.pdf

http://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/

http://mindstorms.lego.com/en-us/Products/default.aspx/