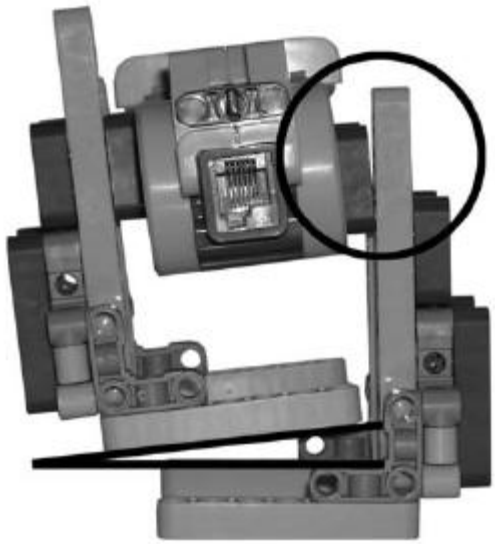


Figure 2-2. *The biped from behind (with the NXT on top removed) shows the looseness of the structure at hip level. The ankle is quite rigid instead.*

It's easy to run into similar problems when working with LEGO parts. Having such a loose structure is a problem that can arise, but don't worry. You can solve it as indicated in the schematic shown in Figure 1-2*c*, by adding the wedges in the inner side of the feet beams. The result of our biped is shown in Figure 2-3.

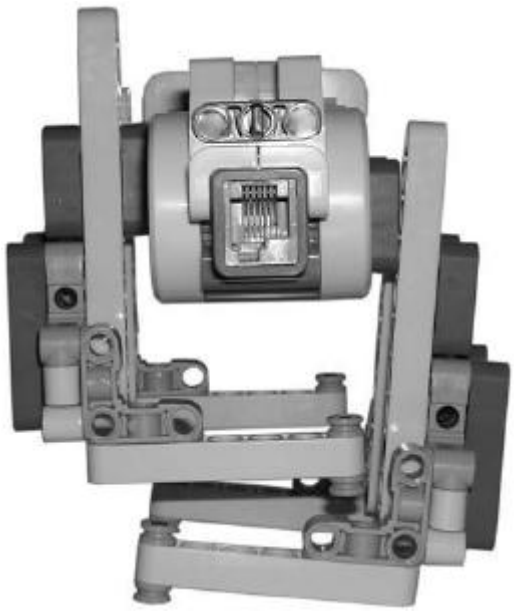


Figure 2-3. *Putting wedges in the inner side of both feet compensates for the leg joints' slackness.*

Even after the wedge additions, the hip joint still tends to be quite loose (it's made with long gray pins, which connect the cam to the leg). To solve this tricky issue, I adopted the hip tendons shown in Figure 1-2*d*, a matter that could have been obscure to you just after reading Chapter 1. Don't worry though, it will become clear now.

Compare Figure 2-4 (before the treatment) with Figure 2-5 (after the treatment). Notice how this elegant solution with the tendons made from LEGO steering links with ball joints, prevents the legs from bending. This last idea of creating tendons is particularly good because it does two things for the price of one: it solves the looseness problem and allows us to connect to the NXT brick in an original way.

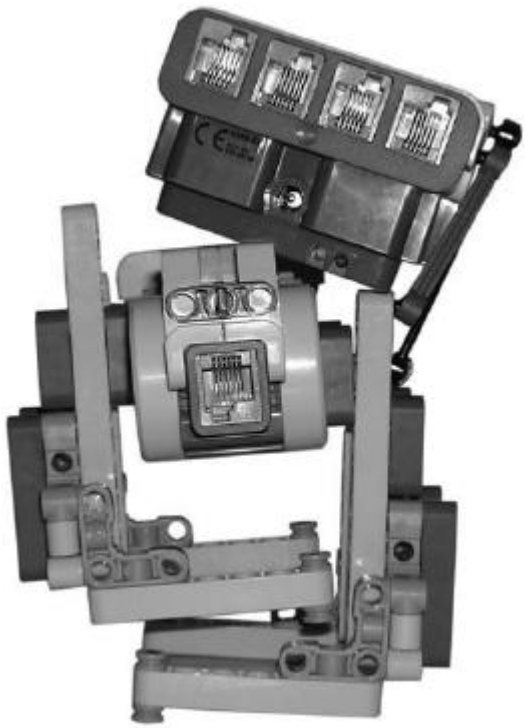


Figure 2-4. *Tendons are not attached yet.*

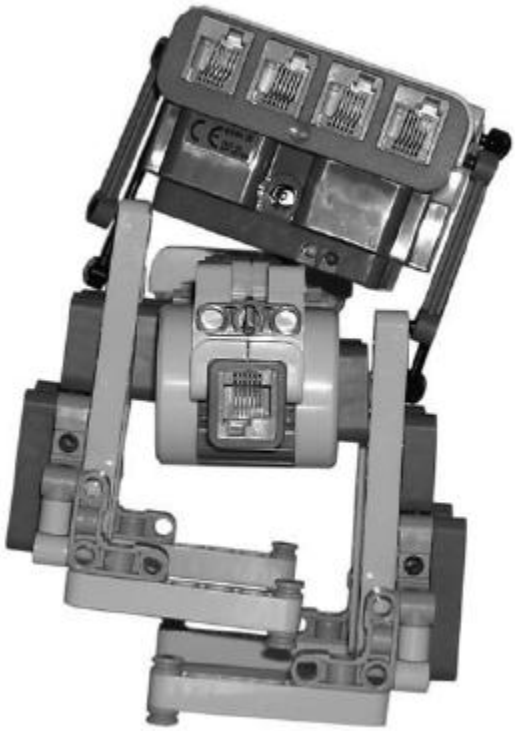


Figure 2-5. *Tendons are now attached.*

Because the tendons are connected to the legs, they swing the NXT in harmony with Quasimodo's gait, and the NXT seems as light as a butterfly. Don't forget that our beloved programmable brick acts as a hump here! I came up with this COG shifting mechanism almost without noticing it, and I must admit this combination of technique, inspiration, and luck is rare. Such a mix makes this robot special. In its simple shape, it summarizes a lot of theory about an unusual way of walking. Ah, I almost forgot: Quasimodo can only walk straight. To create a biped that turns, read the following chapters.

Introducing NXT Technology

Before going on, it's worth introducing the LEGO MINDSTORMS NXT technology briefly. In your NXT retail set, you have LEGO parts, of course, but also some electronic devices that make the NXT system special: three interactive servomotors, a Touch Sensor, a Light Sensor, an Ultrasonic Sensor, and the NXT programmable brick itself. In addition, you have a user guide, and the LEGO software CD-ROM, which allows you to program the NXT using the NXT-G graphical programming language.

The LEGO elements are well assorted, so that you can start creating every kind of robot at once, without having to look for additional spare parts. The set includes LEGO TECHNIC studless elements, except for a few parts. Unlike the common LEGO studded bricks, you do not have to place one brick on another, like building a wall, but you have to start thinking more three-dimensionally, attaching beams and liftarms using pins.

The *NXT servomotors* are different from the common LEGO motors. They are interactive, meaning that they include a Rotation Sensor (optical encoder) that allows you to control interactively the shaft position with 1 degree of resolution, and to set the rotation speed from –100 to 100. A whole shaft rotation is equal to 360 degrees.

The *Touch Sensor* gives your robots the sense of touch: it detects when it is pressed or released, returning a Boolean reading that can be 1 or 0. The *Light Sensor* can distinguish between light and dark colors, measuring the amount of light reflected by the surface illuminated by its LED; it can also measure the light intensity in the environment with the LED off. The *Sound Sensor* makes your robot hear, measuring the sound intensity in decibels. Its readings go from 4 in a silent room to 100, corresponding to people shouting or loud music.

The *Ultrasonic Sensor* enables your robot to see obstacles, measure distances, and detect movement. This digital sensor measures the distance from an object like a bat does, calculating the time needed by an ultrasonic sound wave to hit the object and return. It can measure distances from 0 to 255 centimeters, with an error of **5** 3cm.

Finally, the brain of your robot is the *NXT brick*. It is a microcomputer, programmable with a PC, that lets your LEGO robots come alive, just like JohnNXT (see Chapter 8). You can connect the NXT brick to your PC using a USB cable or Bluetooth. Bluetooth wireless communication is useful if you want to control your robots remotely, or just program it without annoying cables around. You can also connect more NXTs using Bluetooth, to make big complex robots. The NXT has three output ports for attaching motors and four input ports to connect sensors; it has a large dot-matrix screen to display text, numbers, and images. Also, your robots can produce sounds, because the NXT features a loudspeaker to play tones and WAV-like sound files. Two microprocessors are at the base of the NXT brick. The main processor is an Atmel ARM7 (like the one you might have in your mobile phone), and works at 48 MHz, on 32 bits. This allows your robots to deal with large numbers, making calculations at a high speed. The NXT has 256KB of nonvolatile memory; you can store files into it and they won't be erased, even if you remove the batteries.

Oh, I forgot! The NXT needs six AA batteries to work, but can also be powered by the LEGO Li-Ion rechargeable battery. For other details, you can always consult the NXT User Guide included in your retail set.

Meeting the NXT-G Software

Before starting programming at full throttle in NXC, it will help you to get familiar first with NXT programming. The first programming environment you will meet is the official LEGO NXT-G software, the one you should have installed from the CD-ROM included in the NXT retail set. If you haven't done so yet, install this software now.

Connecting the NXT for the First Time

If you've never connected your NXT to a PC before, perform the following steps. You can connect the NXT using the USB cable or using Bluetooth. To find out if your Bluetooth dongle or integrated device is compatible with the NXT, check the LEGO MINDSTORMS official page at http://mindstorms.lego.com/Overview/Bluetooth.aspx.

1. Open the NXT-G software and create a new blank program, by clicking the Go button in the Start New Program panel, as shown in Figure 2-6.

Figure 2-6. *The startup panel of the NXT-G software*

2. In the NXT controller (see Figure 2-7), click the top left button to open the NXT window, shown in Figure 2-8.

Figure 2-7. *The NXT controller, which you can find in the right bottom of the software main window*

Figure 2-8. *The NXT window, without any NXT devices listed*

3. Connect the NXT to the PC using the USB cable. Otherwise, you can turn on the Bluetooth radio of the NXT brick using its menu. Be sure also to turn on the NXT Bluetooth visibility; use the NXT User Guide as a reference. Then, click the Scan button on the NXT window in Figure 2-8. If you're using a USB connection, it should find the NXT at once; scanning takes a bit more time, if you're searching the NXT using Bluetooth. Once the NXT has been found, it shows up in the list, as you can see in Figure 2-9.

Figure 2-9. *The NXT named "book" has been found using Bluetooth.*

4. Click the Connect button, and you're asked for a passkey in the Enter Passkey dialog (see Figure 2-10). Use the default number 1234 and click OK. The NXT makes a sound, asking for the passkey. Confirm the connection on the NXT brick, using the same passkey as before.

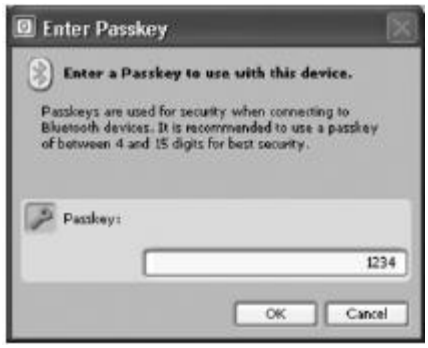


Figure 2-10. *Dialog asking for the passkey for the Bluetooth connection*

5. Once the connection is established, the NXT window looks like the one shown in Figure 2-11; the NXT is reported twice in the list in that figure, because both USB and Bluetooth found it. Once an NXT has been connected, it will be present in the list unless you remove it by clicking the Remove button. As you can see, in this panel you can rename your NXT by typing a new name in the appropriate text field and clicking the Enter button. This window also shows some diagnostic information, such as the battery level, the free FLASH memory amount, and the firmware version. Close the panel and you can start programming your robot.

Figure 2-11. *The NXT is connected.*

Programming Quasimodo Using NXT-G

Quasimodo has only one motor and one Ultrasonic Sensor. The robot walks straight until it sees a near obstacle; it moans and backs up until the sight line is free again, repeating this in a loop. You can code this simple sequence of actions using NXT-G software; the result is shown in Figure 2-12.

In Figure 2-12*a* you can see how the program looks when completed. Build the program as follows:

1. Start adding the Loop (*b*) and set it to run forever.
2. Add a Motor block (*c*), with the following settings:
   - Port: A
   - Direction: Forward
   - Power: 55
   - Control: Motor Power checked
   - Duration: Unlimited
3. Add a Wait block (*d*), with the following settings:
   - Control: Sensor
   - Sensor: Ultrasonic Sensor
   - Port: 4
   - Until distance < (less than) 20 (cm)
4. Add a Motor block (*e*), with the following settings:
   - Port: A
   - Direction: Stop
   - Next Action: Brake
5. Add a Sound block (*f*), with the following settings:
   - Action: Sound File
   - Control: Play
   - Volume: 75
   - File: Woops
   - Wait for Completion checked
6. Add a Motor block (*g*), with the following settings:
   - Port: A
   - Direction: Reverse

- Power: 55
- Control: Motor Power checked
- Duration: Unlimited

7. Add a Wait block (), with the following settings:
   - Control: Sensor
   - Sensor: Ultrasonic Sensor
   - Port: 4
   - Until distance > (greater than) 30 (cm)

The program is ready to be downloaded to the NXT. As shown in Figure 2-7, you can click the Download button (bottom left) or the Download and Run button (center) on the NXT controller. You can also save the program for future use.

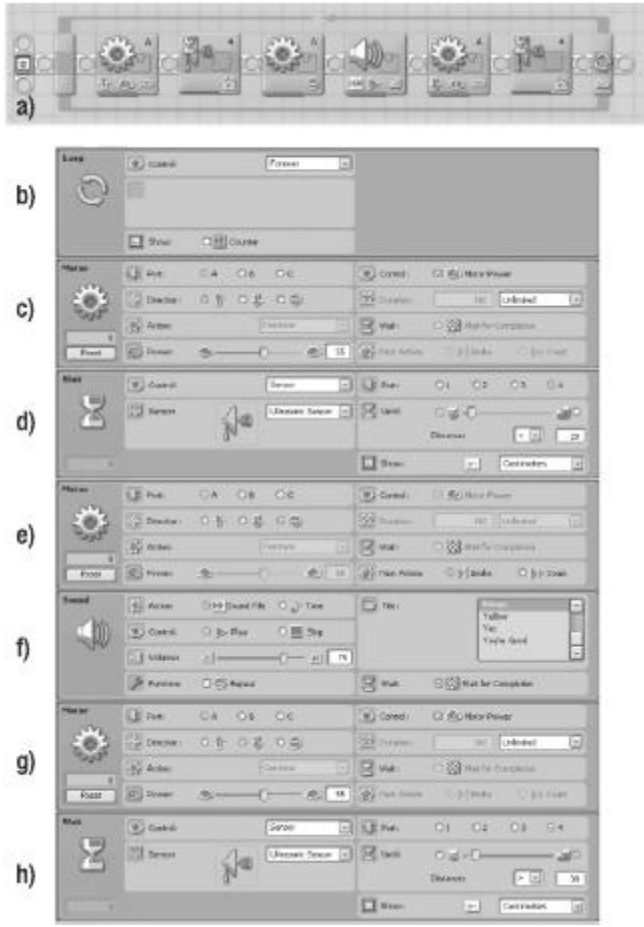


Figure 2-12. *The NXT-G program for Quasimodo*

Now that you know how to connect the NXT to a PC and how to write and download programs using NXT-G software, you can take the next step. From now on, we write the programs for the robots only in the NXC language, using BricxCC as the Integrated Development Environment (IDE). To get started with BricxCC, read Appendix A.

The Shortest Program in the Book

You can translate the program for Quasimodo described earlier into the NXC code in Listing 2-1. Try to single out the correspondences between the NXT-G blocks and the NXC statements.

Listing 2-1. *The Program for Quasimodo*

```
task main ()
{
  SetSensorLowspeed(IN_4);
  while (true)
  {
    // walk forward
    OnFwdReg(OUT_A,55,OUT_REGMODE_SPEED);
    // wait for an obstacle to come near
    while (SensorUS(IN_4)<20);
    // stop walking
    Off(OUT_A);
    // moan
    PlayFile("uuu.rso");
    Wait(800);
    // walk backward
    OnRevReg(OUT_A,55,OUT_REGMODE_SPEED);
    // wait for the obstacle to get far
    while (SensorUS(IN_4)>30);
  }
}
```

■ **Note** The `#include "NXCDefs.h"` statement was used at the top of NXC programs to tell early Beta versions of the NXC preprocessor to include this header file, which was needed to translate every NXC statement into low-level assembly language for the NXT. The latest versions of the NXC compiler (Beta 29 and later) no longer need this preprocessor directive because that file is automatically included at the time of program compilation.

In your program, you use only the main task. Remember that you'll have to put a task named main in every NXC program you'll write. This task is the first one to be executed when the program is run, so you must put it inside all the actions you want your robot to do. Here, you set the input port to 4 to manage the Ultrasonic Sensor using this code:

```
SetSensorLowspeed(IN_4);
```

You start an infinite loop (at least, until the batteries run out, the automatic NXT power-off feature turns it off, or you just stop the program), inside of which you can make the robot do whatever you want. How is this infinite loop done?

Normally, using the following statement, the program will execute the code you put inside the **{ }** braces, while the Boolean condition is true, checking this condition at every loop iteration.

```
while ( condition )
{
   //your code here
}
```

Well, using while (true), the program executes what's inside the braces forever, because its condition never becomes false. The true constant means, in fact, true (not hard to believe!), and the loop goes on forever.

The first three lines of code inside the loop start the motor attached on output port A at 55 percent of full speed, regulating its speed precisely:

```
OnFwdReg(OUT_A,55,OUT_REGMODE_SPEED);
while (SensorUS(IN_4)<20);
Off(OUT_A);
```

The code then waits for the sensor reading to become less than 20cm (an obstacle comes near), using a while loop without the **{ }** braces. Finally, the motor is stopped.

Normally, a while loop is written as shown before, with some code inside the braces. However, in this case you're interested in waiting for something, not in repeating some conditional action. So you start the motor and you wait (doing nothing) until the Ultrasonic Sensor sees something near; the motor will run until it is explicitly stopped by the Off statement. This waiting is accomplished simply by writing the desired condition (as the sensor reading) inside the while loop **( )** brackets, with no additional code inside the **{ }** braces. So, you can remove those last braces and just put in the semicolon after the condition. When an obstacle is seen, the program flow breaks out the while loop, and the Off statement is executed, stopping the motor.

■ **Note** Remember, the `while ( condition );` waits for the condition to become false.
The `until ( condition );` waits for the condition to become true.
This implies that the `while(condition)` is equivalent to an `until(!condition)`, where **!** is the unary logic negation operator NOT. Remember that every line of code must end with a semicolon (**;**).

The central chunk of code simply plays a file; here, I chose a self-made audio file that I recorded, compressed, and converted to the NXT standard format: RSO. However, you can use any file you want, or you can simply comment out these statements by putting // before the lines of code.

■ **Note** The compiler ignores lines preceded by // or enclosed inside the /* and */ couple; for example:

```
//this is a comment until the end of line
statement;
statement;
/* this is a comment
   on multiple lines
*/
```

Warning! The /* */ cannot be nested! So, the following is wrong:

```
/* extern comment
/* inner comment
   whatever
*/ end of inner comment
*/ end of extern comment
```

■ **Tip** You can find the custom sounds I made with the book's source code. You can find the RSO default sound files in the \engine\Sounds folder of your official NXT retail or educational version software (for example, C:\Program Files\LEGO Software\LEGO MINDSTORMS NXT\engine\Sounds). You must download the sound files into the NXT memory for the program to find and play them. You can download sounds to the NXT following the directions in Appendix A.

The Wait statement does what it says: it waits for a given number of milliseconds. If you want to play a file without any other action, Wait is needed because the program flow does not wait for a sound file to be played until the end, and it goes on executing the next instruction even if the sound has not been completely played (this holds for PlayTone too).

The third and last chunk of code starts the motor in reverse to let the robot walk away from the obstacle:

```
OnRevReg(OUT_A,55,OUT_REGMODE_SPEED);
while (SensorUS(IN_4)>30);
```

Another while loop is now used to wait for another condition to become false; you want to wait for the Ultrasonic Sensor reading to become greater than 30cm. As soon as the obstacle is far enough, the action sequence starts from the beginning of the loop.

Although simple, this first program is useful as an introduction to the NXC language, and to demonstrate how you can translate a program made with NXT-G blocks using a few lines of code, which are much faster to write. You've seen some basic flow-control statements of the NXC language, as while/until loops, and some NXT-specific Application Programming Interface (API) functions, to move motors and to read sensors.

Building Your Belfry Hunchback

It's time to build this model now. Quasimodo is the simplest robot of the whole book. It will take just few minutes to build this robot, made only out of parts from the NXT retail set. You can check the bill of materials in Figure 2-13 and Table 2-1.

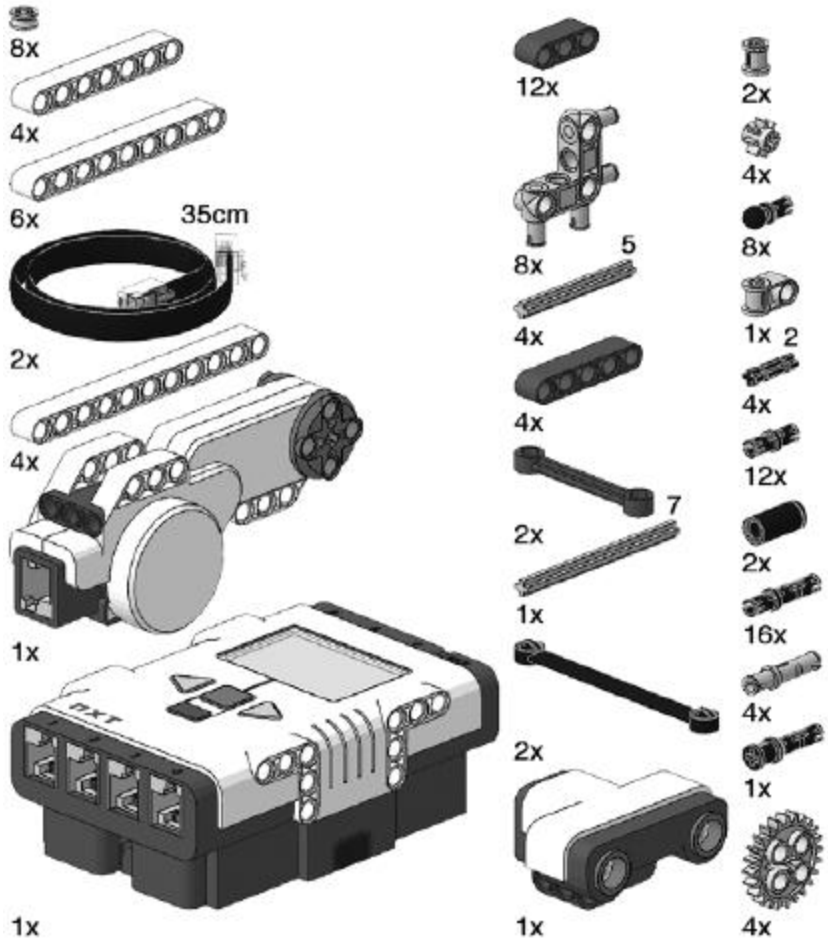
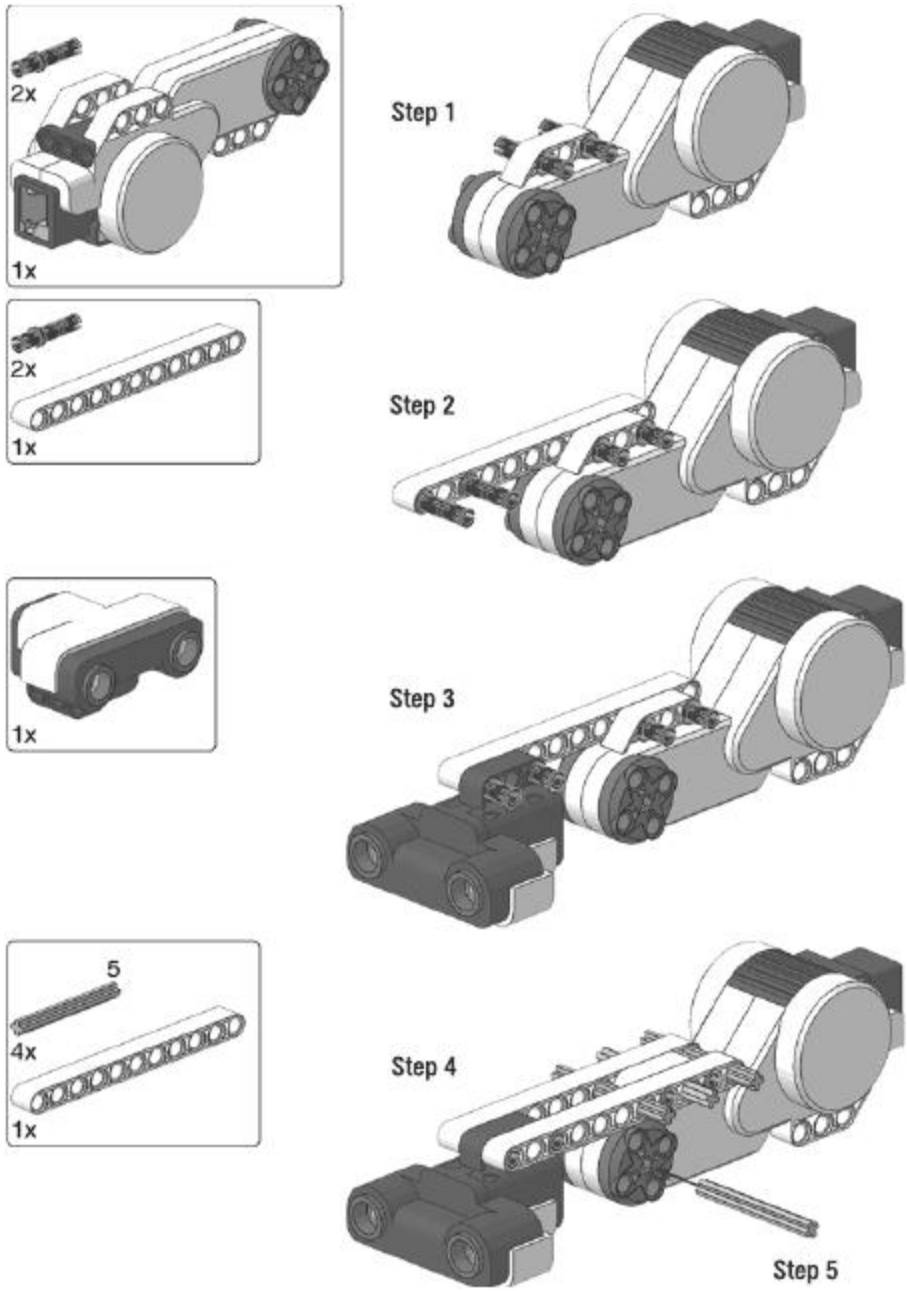


Figure 2-13. *Quasimodo bill of materials*

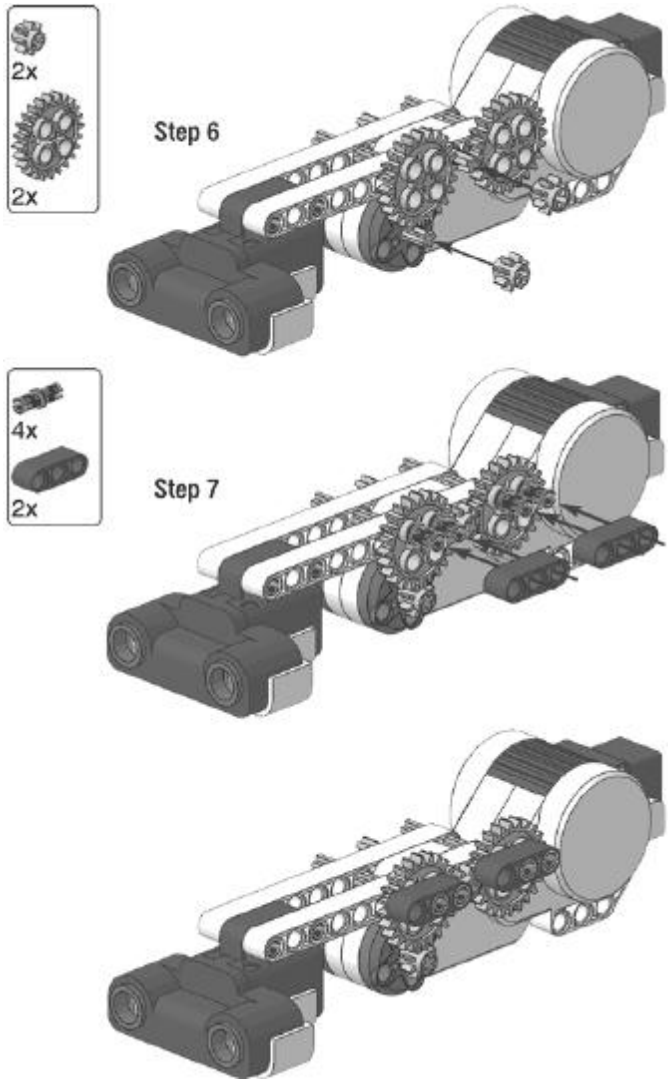
Table 2-1. *Quasimodo Bill of Materials*

Quantity	Color	Part Number	Part Name
8	Light gray	32123.DAT	TECHNIC Bush 1/2 Smooth
4	White	32524.DAT	TECHNIC Beam 7
6	White	40490.DAT	TECHNIC Beam 9
2		55805.DAT	Electric Cable NXT 35cm
4	White	32525.DAT	TECHNIC Beam 11
1		53787.DAT	Electric MINDSTORMS NXT Motor
1		53788.DAT	Electric MINDSTORMS NXT
12	Dark gray	32523.DAT	TECHNIC Beam 3
8	Light gray	55615.DAT	TECHNIC Beam 5 Bent 90 (3:3) with 4 Pins
4	Light gray	32073.DAT	TECHNIC Axle 5
4	Dark gray	32316.DAT	TECHNIC Beam 5
2	Dark gray	2739B.DAT	TECHNIC Steering Link
1	Light gray	44294.DAT	TECHNIC Axle 7
2	Black	32293.DAT	TECHNIC Steering Link 9L
1		56467.DAT	Electric MINDSTORMS NXT Ultrasonic Sensor
2	Light gray	3713.DAT	TECHNIC Bush
4	Light gray	3647.DAT	TECHNIC Gear 8 Tooth
8	Black	6628.DAT	TECHNIC Friction Pin with Towball
1	Light gray	6536.DAT	TECHNIC Axle Joiner Perpendicular
4	Black	32062.DAT	TECHNIC Axle 2 Notched
12	Black	4459.DAT	TECHNIC Pin with Friction
2	Black	75535.DAT	TECHNIC Pin Joiner Round
16	Black	6558.DAT	TECHNIC Pin Long with Friction and Slot
4	Light gray	32556.DAT	TECHNIC Pin Long
1	Black	32054.DAT	TECHNIC Pin Long with Stop Bush
4	Light gray	3648.DAT	TECHNIC Gear 24 Tooth

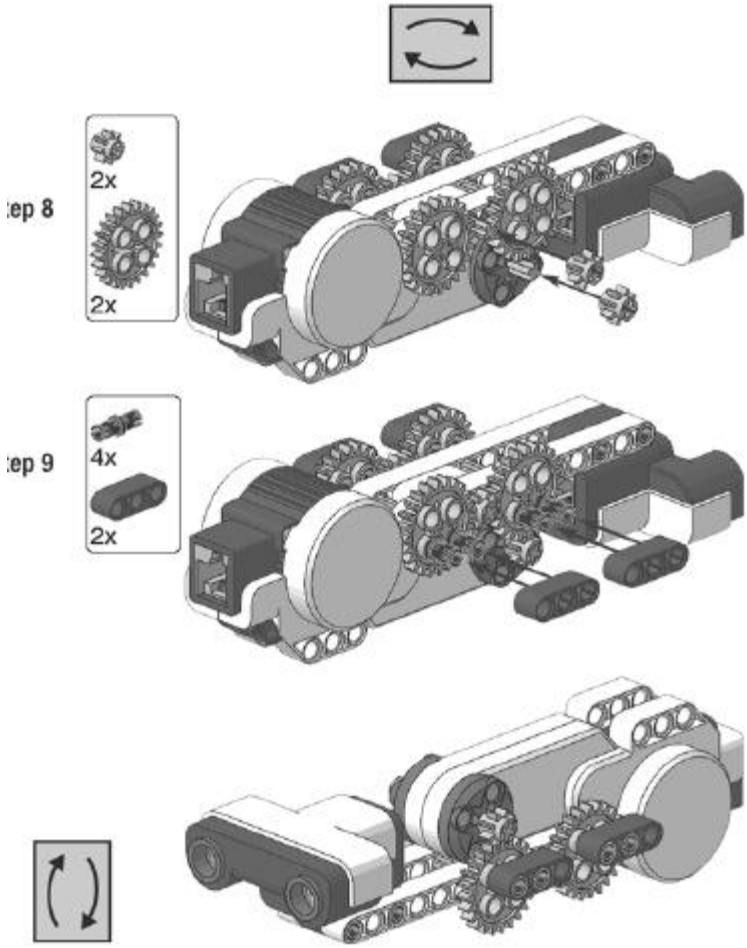
118 parts total (all included in the NXT retail set)



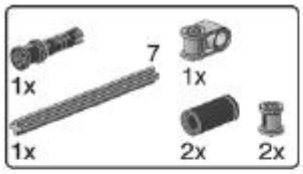
Start building Quasimodo's body. As you can see, the motor is laid horizontally with respect to the ground. In Step 2, use an 11-long beam. In Step 3, add the Ultrasonic Sensor that forms the robot head, and then insert the 5-long axles into their places. You must insert one of the 5-long axles in the motor shaft hole.

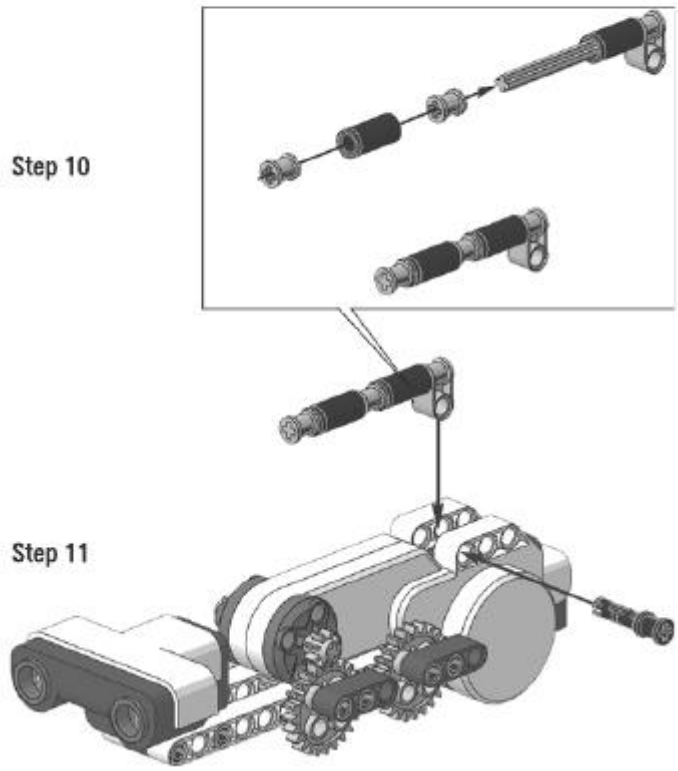


Add the first row of gears, the black pins, and then the 3-long beams. Be sure to align the cams correctly, so that they are parallel. To do this, check the gear meshing. The cams will make the leg move in a circle.

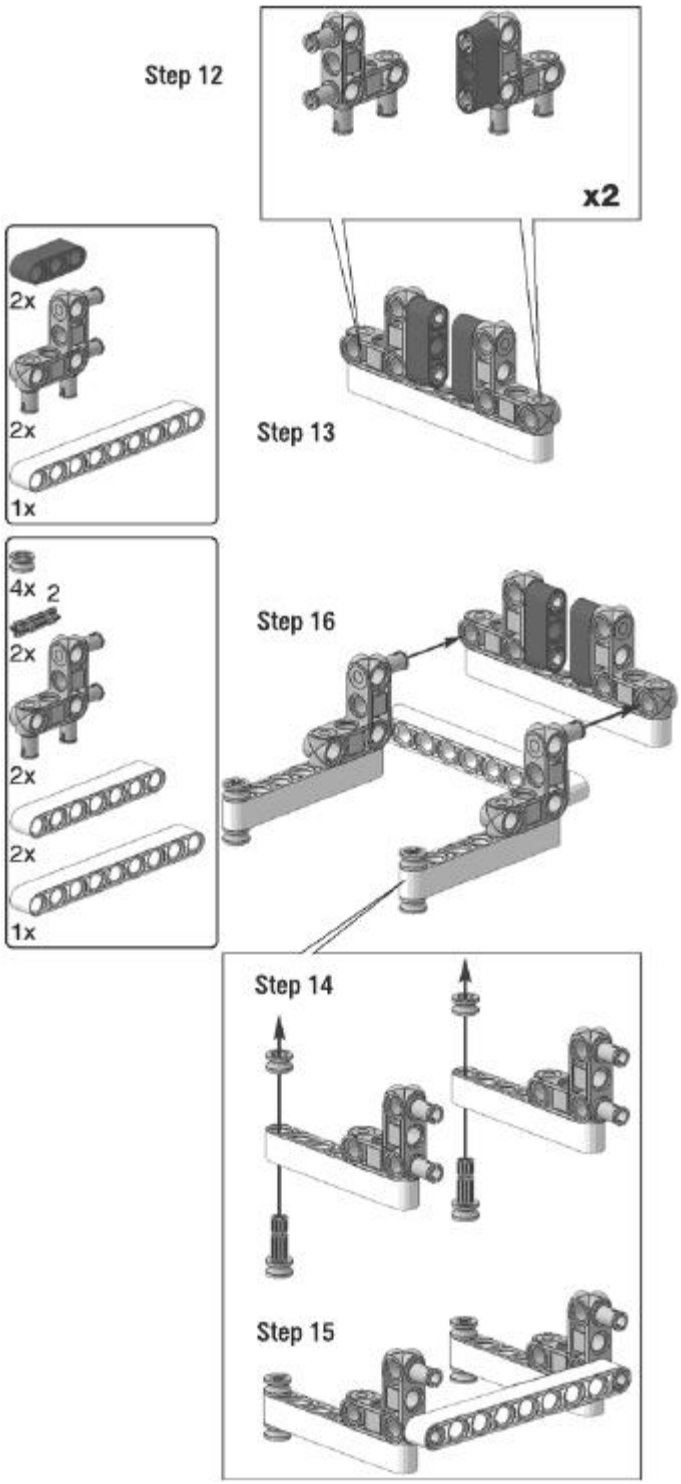


Rotate the model and place the second row of gears, as you did for the ones already in place. Then add the cams on this side, making sure to place them 180 degrees out of phase; that is, rotated half a turn with respect to the other side ones.

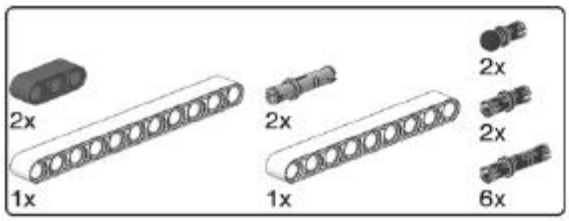


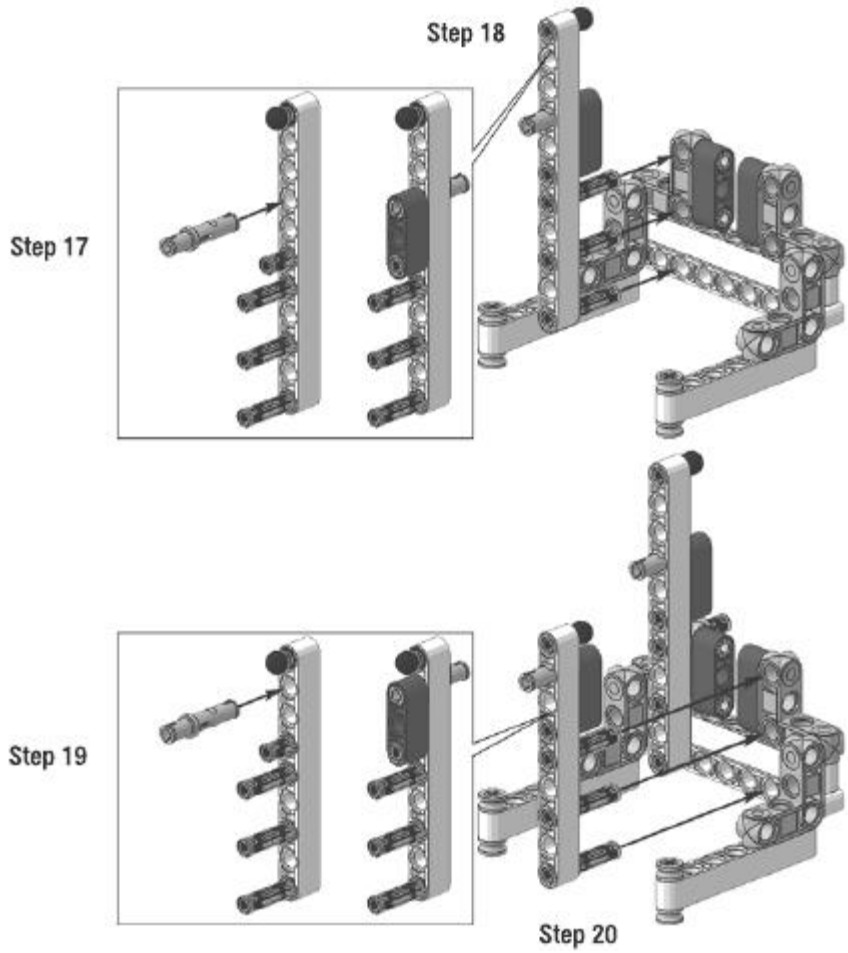


The robot main body is complete. Skip this step if you are going to use the NXT rechargeable battery pack instead of regular batteries. Build the sub model of this step if you're going to use normal batteries. In fact, the NXT battery pack protrudes one LEGO unit out of the normal NXT profile; to get this extra battery thickness, needed to have the robot walk smoothly, you have to build this offset part.

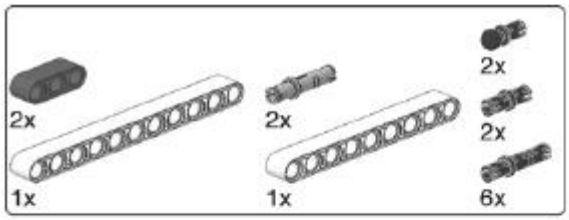


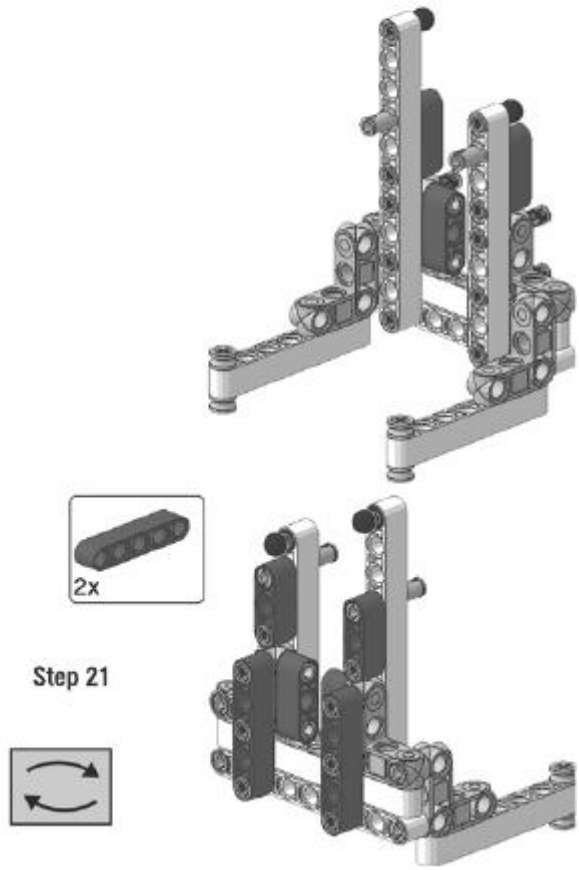
Start building the left leg.



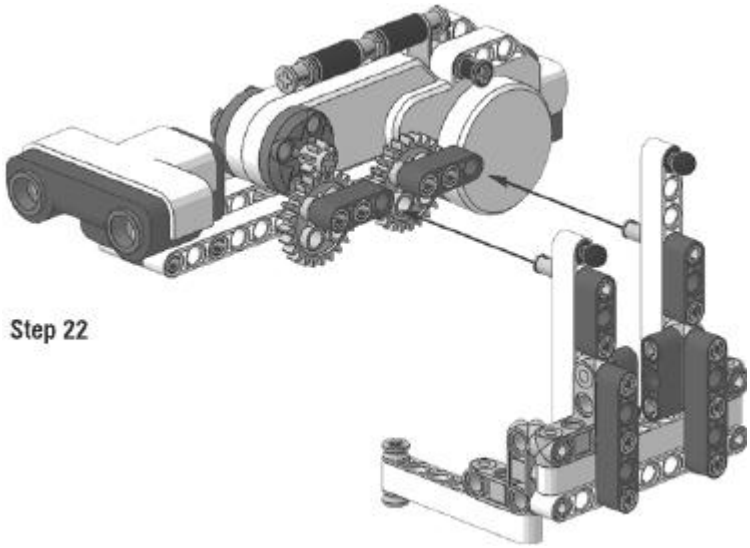


Assemble the vertical beams of the leg. The first is 11 holes long; the second is 9 holes long.

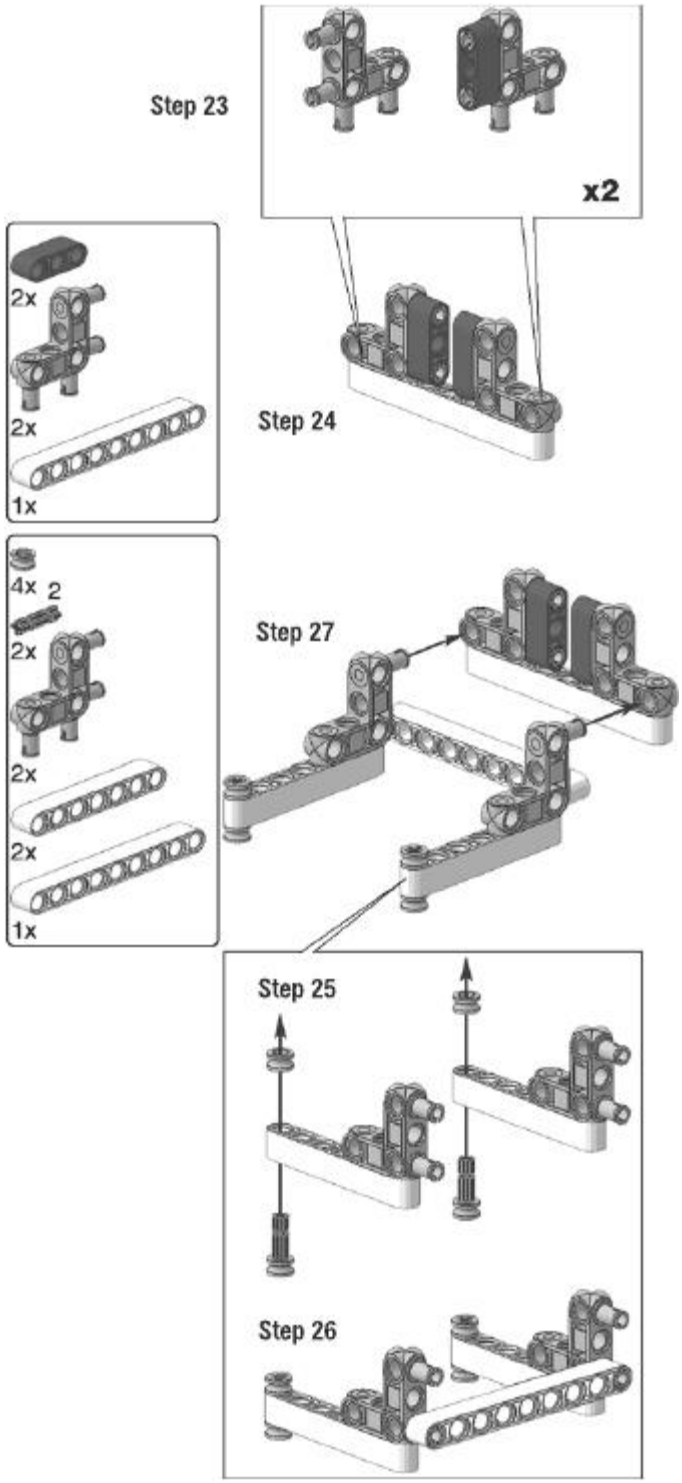




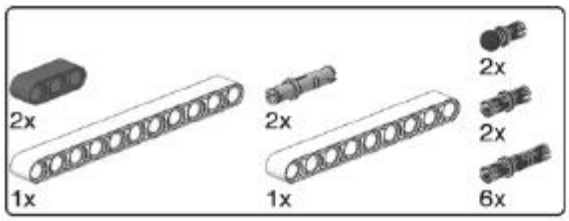
The left leg is completed.

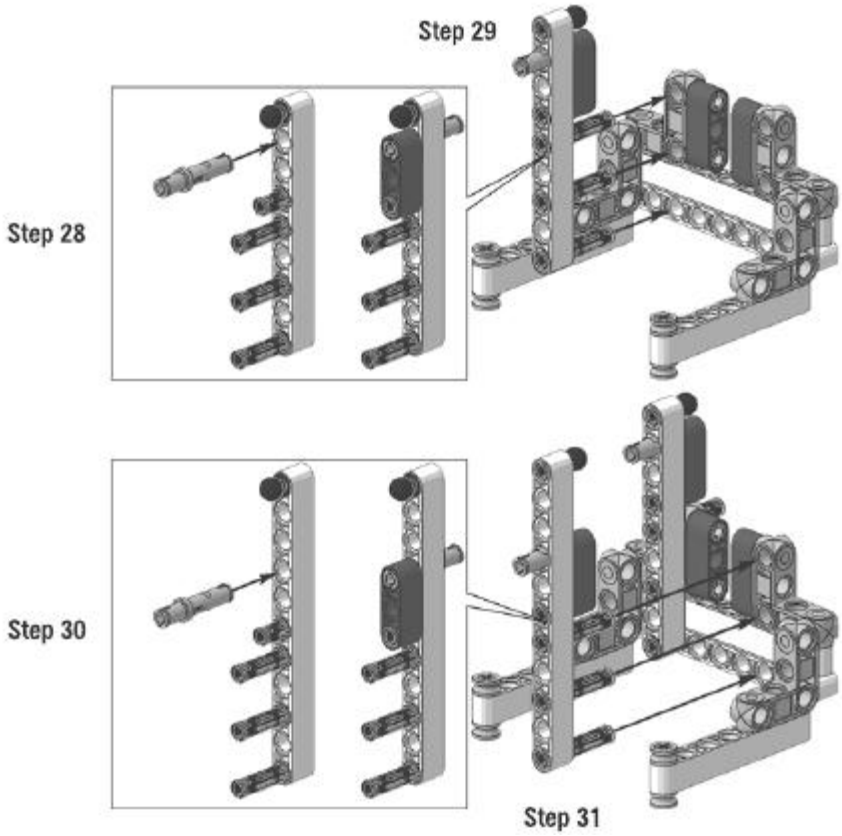


Attach the left leg to the robot body, inserting the gray pins into the cam's free holes.

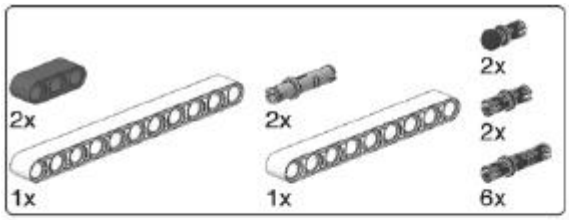


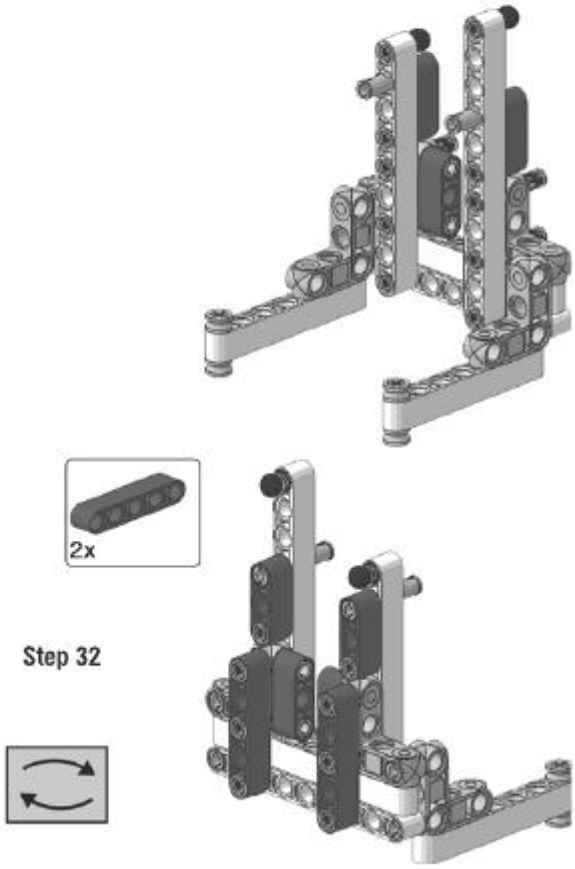
Start building the right leg.



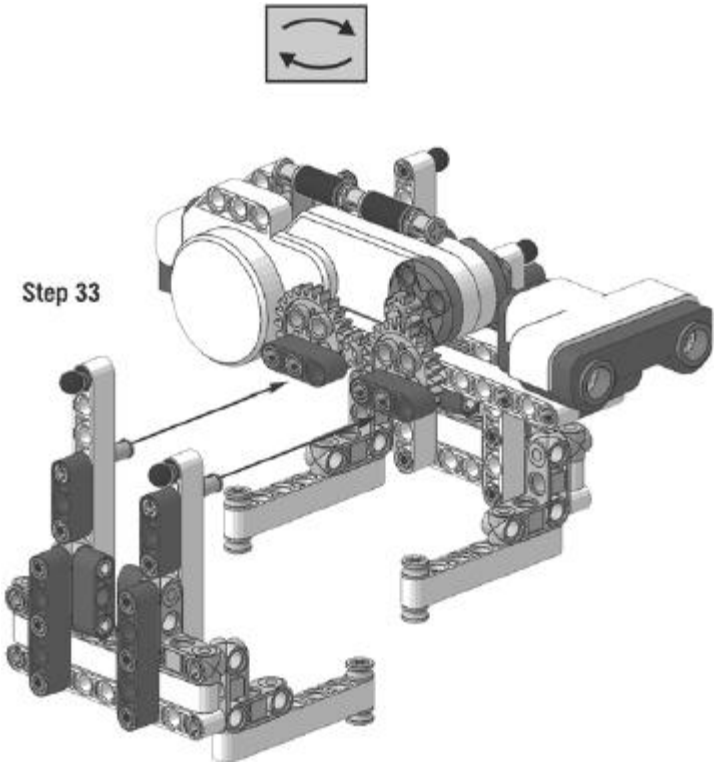


Assemble the vertical beams of the leg. The first is 9 holes long, and the second is 11 holes long; this is the only difference between the two leg assemblies.

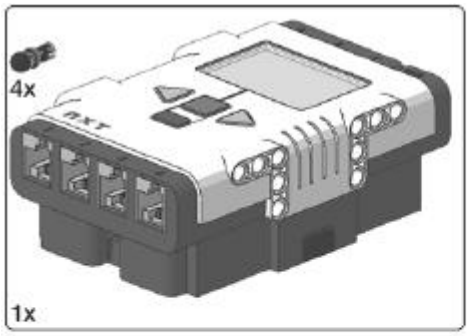


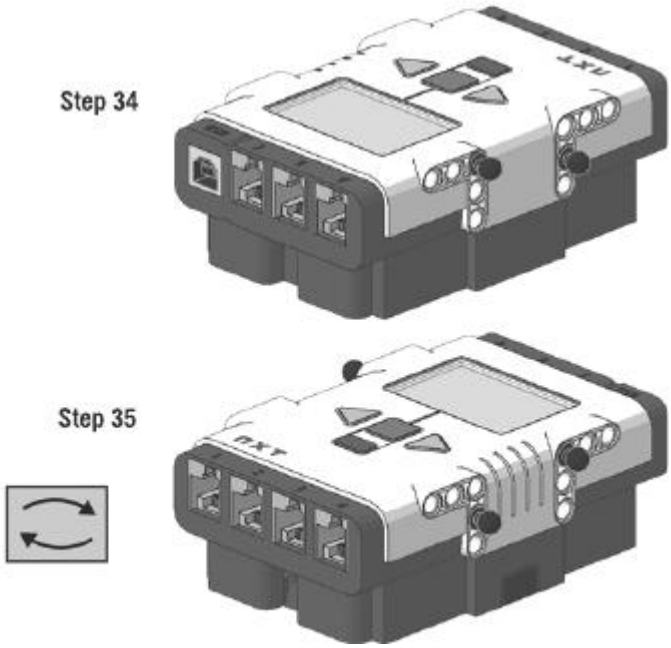


The right leg is completed.

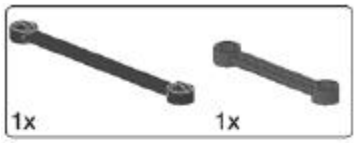


Turn the model and attach the right leg to the robot. Notice that, if you built the cams correctly, the legs are out of phase—one forward, the other backward. Quasimodo should now stand on its own feet.

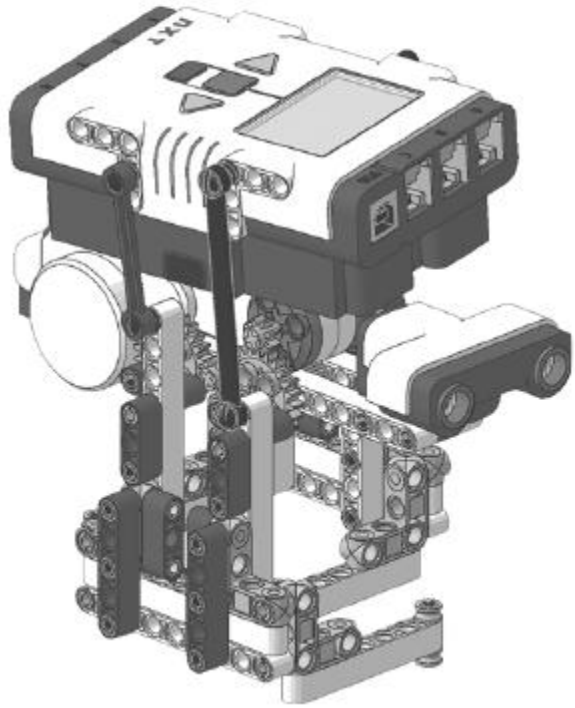




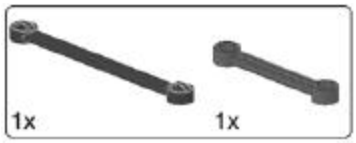
Here you are building the NXT subassembly that forms the hump.

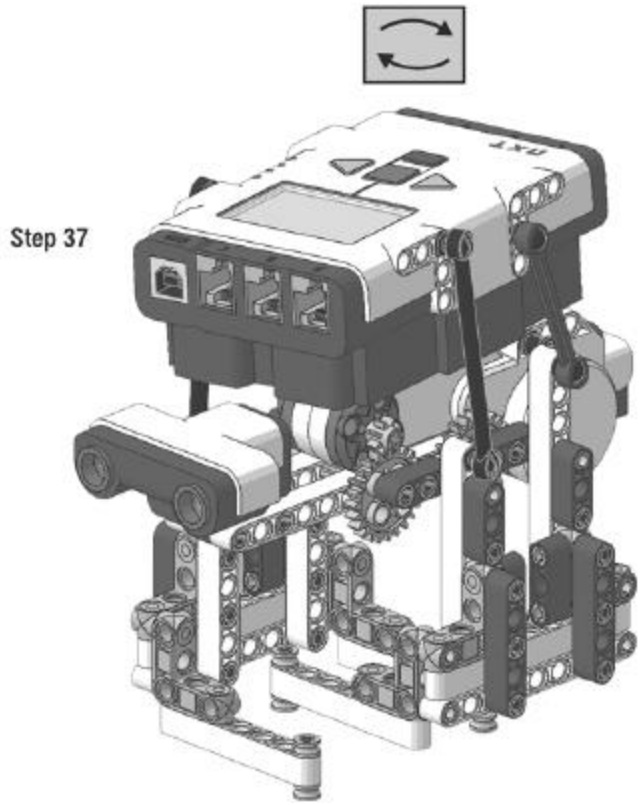


Step 36

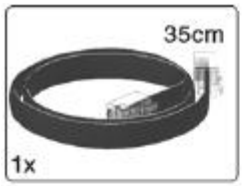


Put the NXT on the robot's top and attach two hip tendons to the right leg.

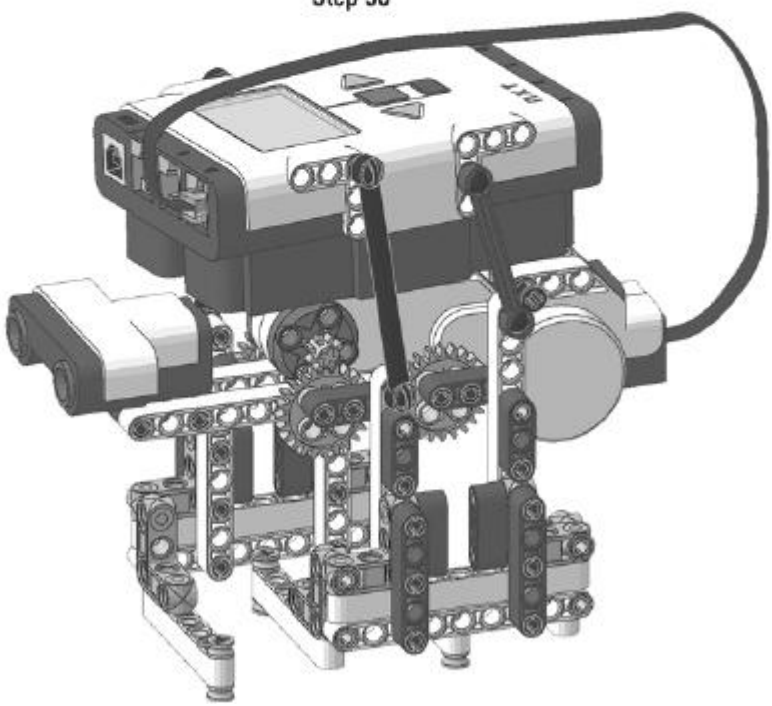




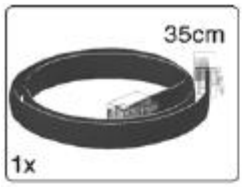
Turn the model and attach the other two tendons.

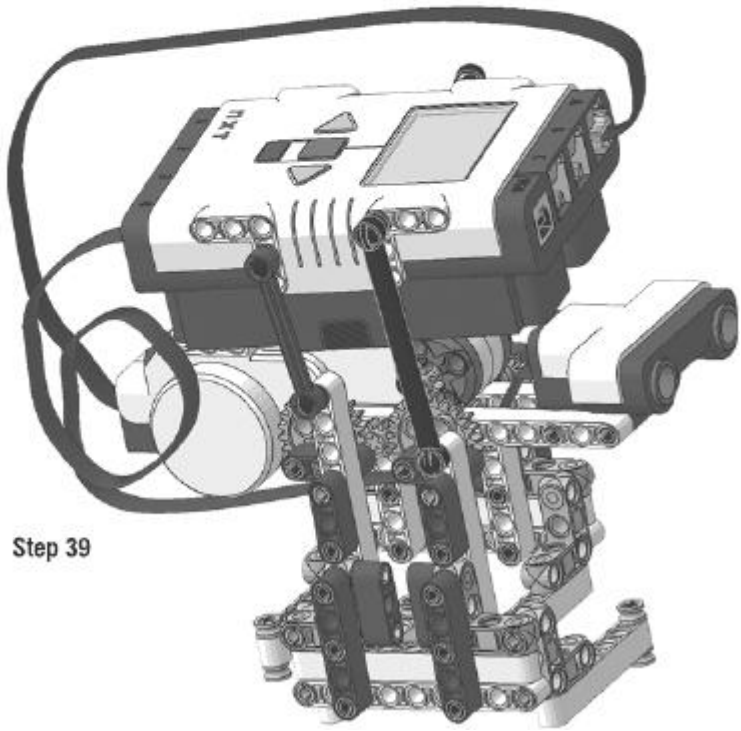


Step 38

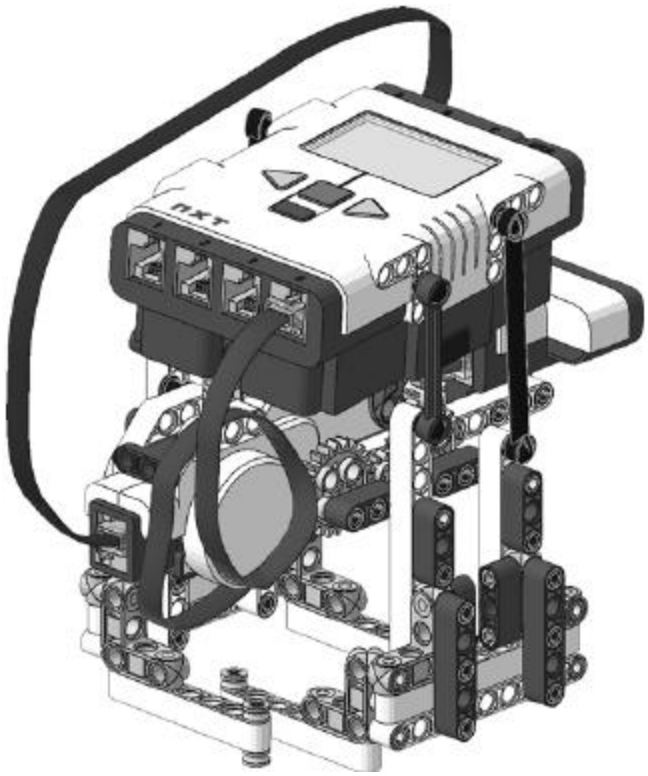


Connect the motor to the NXT output port A with a 35cm (14 inch) long cable.

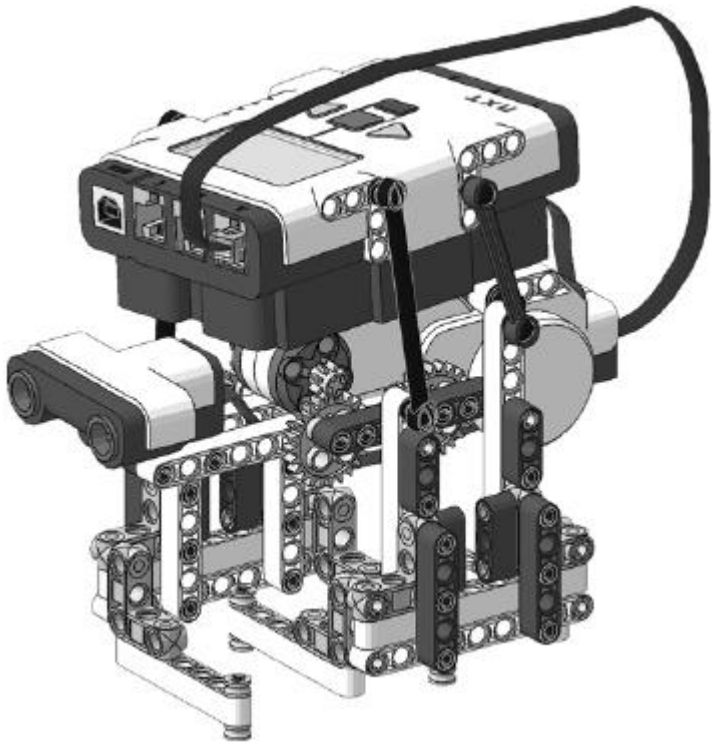




Connect the Ultrasonic Sensor NXT input port 4 with a 35cm (14 inch) cable. Try to make the cable pass between the main body beams, as shown.



A view of the robot's back, showing the sensor cable attachment



QUASIMODO