Table of Contents

Illustration Index

Chapter 6: Sensors

Introduction

Human being has 5 senses to measure signals from the environment to interact with the world. The senses are: sight, hearing, taste, smell and touch.

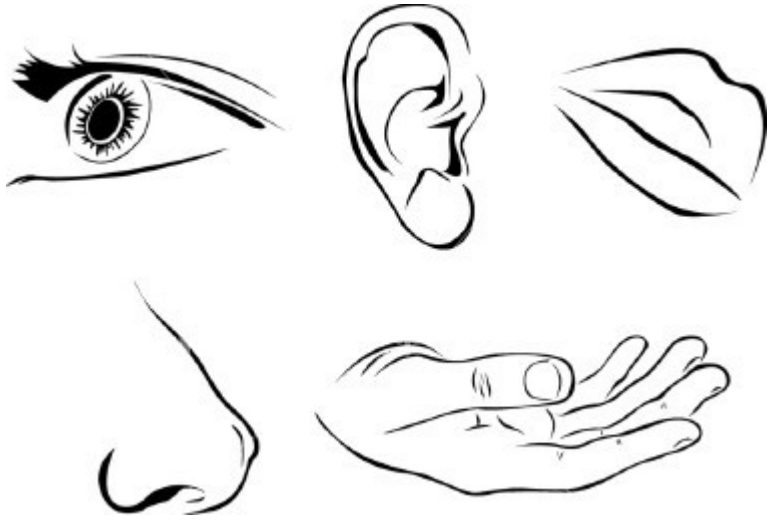


Illustration 1: Five senses

In robotics, it is possible to add sensors in your robots to get data from environment or internal state of that creation to intereact. Robots developed with NXT brick has the opportunity to use 4 sensors connected to the brick. If you classify the nature of the information which sensors measure, you have 2 groups:

1. “exteroceptors” for the measurement of its environmental (external, from the robot point of view) parameters.
2. “proprioceptors” for the measurement of the robot’s (internal) parameters

Exteroceptors sensors

Exteroceptors are sensors that measure the positional or force-type interaction of the robot with its environment.

Proprioceptors sensors

Proprioception in robotics means sensing the internal state of the robot or a part of it . For example the

posture of a mechanical manipulator, leg or other jointed mechanism or the battery level.

Exteroceptors sensors in NXT

Introduction

Any Lego Mindstorms KIT included the following exteroceptor sensors:

1. Ultrasonic Sensor
2. Touch Sensor
3. Sound Sensor

Illustration 2: Lego Mindstorms NXT 2.0 Kit

Besides, it is possible to buy new exteroceptor sensors from providers as Mindsensors, Dexter Laboratories and HiTechnics. In this ebook, we will explain the following sensors:

1. Compass sensor
2. GPS Sensor
3. NXTCam

Ultrasonic sensor

The Ultrasonic Sensor helps your NXT robot measure distances and "see" where objects are.

Illustration 3: Ultrasonic sensor

The Ultrasonic Sensor uses the same scientific principle as bats: it measures distance by calculating the time it takes for a sound wave to hit an object and return – just like an echo.

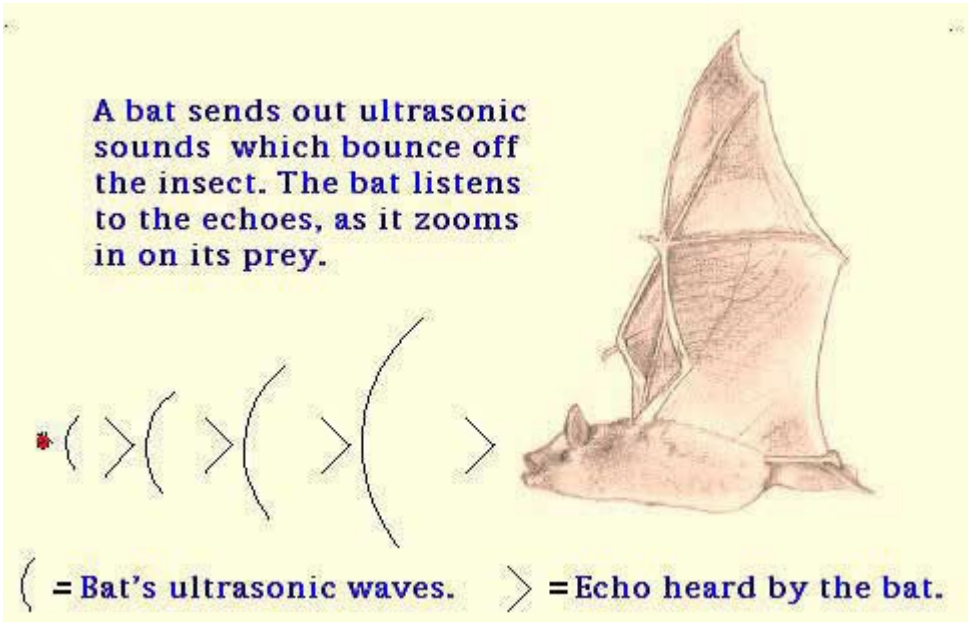


Illustration 4: Bat system to detect insects using ultrasonic waves

Large sized objects with hard surfaces return the best readings. Objects made of soft fabric or with curved shapes like a ball or are very thin or small can be difficult for the sensor to detect.

To explain how to use Ultrasound Sensor with NXJ, we will explain the example SonicTest.java that you can find in the folder SonicTest.

```
public class SonicTest {

	public static void main(String[] args) throws Exception {
		UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);

		while(!Button.ESCAPE.isPressed()) {
			LCD.clear();
			LCD.drawString(sonic.getVersion(), 0, 0);
			LCD.drawString(sonic.getProductID(), 0, 1);
			LCD.drawString(sonic.getSensorType(), 0, 2);
			Thread.sleep(200);
			LCD.drawInt(sonic.getDistance(), 0, 3);
			LCD.refresh();
			Thread.sleep(500);
		}
	}
}
```

To use Ultrasound Sensor, you have to create an instance of the UltrasonicSensor Class.

```
UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);
```

You have to indicate in what port you have pluged the sensor. In the example Ultrasound is pluged in your NXT brick. In this example Ultrasonic sensor was connected in the Port 1.

If your program needs to know the distance from the sensor to any object then you need to use the method:

```
sonic.getDistance()
```

Light Sensor

This is one of the most popular sensor in educative robotics projects. This sensor enables your robot to distinguish between light and dark, as well as determine the light intensity in a room or the light intensity of different colors.

Illustration 5: Light sensor

An example using this sensor:

```
import lejos.nxt.*;

public class LightTest {
  public static void main(String[] args) throws Exception {
    LightSensor light = new LightSensor(SensorPort.S1);

    while (true) {
      LCD.drawInt(light.getLightValue(), 4, 0, 0);
      LCD.drawInt(light.getNormalizedLightValue(), 4, 0, 1);
      LCD.drawInt(SensorPort.S1.readRawValue(), 4, 0, 2);
      LCD.drawInt(SensorPort.S1.readValue(), 4, 0, 3);
    }
  }
}
```

Touch sensor

The Touch Sensor gives your robot a sense of touch. The Touch Sensor detects when it is being pressed by something and when it is released again

Illustration 6: Touch sensor

An example using this sensor is the following:

```
import lejos.navigation.TachoNavigator;
import lejos.nxt.*;

public class FindCatch{

      public static void main(String [] args)throws Exception {
            TachoNavigator tn=new TachoNavigator(5.5f,11.2f,Motor.B,Motor.C,false);
            UltrasonicSensor uss = new UltrasonicSensor(SensorPort.S3);
            TouchSensor ts=new TouchSensor(SensorPort.S1);
            LightSensor ls=new LightSensor(SensorPort.S4);

            int distance =30;
            for (int i=0;i<6;i++) {
                  Sound.beep();
                  try { Thread.sleep(500); } catch (Exception e) {}
            }
```

```
Motor.C.setSpeed(400);
Motor.B.setSpeed(400);
Motor.C.backward();
Motor.B.forward();
while (Button.readButtons()==0) {
      float dist = uss.getDistance();     //distance calculate
      if (dist <= distance) {
            tn.stop();

            Motor.A.setSpeed(200);
            Motor.A.forward();
            try {
                  Thread.sleep(2000);
            } catch (Exception e) {

            }
            tn.setSpeed(400);
            tn.forward();
      }else if(ts.isPressed()){    //if touch is pressed
            Sound.beep();

            tn.stop();
            LCD.drawString("Object Found", 3, 4);
            try {
                  Thread.sleep(3000);
            } catch (Exception e) {

            }
            Motor.A.setSpeed(200);
            Motor.A.backward();
            int colorval=ls.readValue();//learn color value

            try{
                  Thread.sleep(7000);
            } catch (Exception e) {
```

```
}
if(colorval<=22){  // if it is blue
      LCD.clear();
      LCD.drawString("Blue Object", 3, 4);
      tn.travel(-10);
      tn.rotate(180);
      tn.travel(60);
      tn.goTo(50,0);
      Motor.A.forward();
      tn.rotate(-10);
      Motor.A.backward();
      tn.goTo(0,0);

      try {
            Thread.sleep(2000);
      } catch (Exception e) {

      }
}
else if(colorval>22){//if it is red
      LCD.clear();
      LCD.drawString("Red Object", 3, 4);
      tn.travel(-10);
      tn.rotate(180);
      tn.travel(60);
      tn.goTo(-50,0);
      Motor.A.forward();
      tn.rotate(-10);
      Motor.A.backward();
      tn.goTo(0,0);
      LCD.clear();
}
else{
      LCD.clear();
      LCD.drawString("where is object",0, 0);
      System.shutDown();
```

```
                                }
                            }
                        }
                    }
}
```

Sound sensor

The Sound Sensor can detect both decibels [dB] and adjusted decibel [dBA]. A decibel is a measurement of sound pressure.

Illustration 7: Sound sensor

dBA: in detecting adjusted decibels, the sensitivity of the sensor is adapted to the sensitivity of the human ear. In other words, these are the sounds that your ears are able to hear.

dB: in detecting standard [unadjusted] decibels, all sounds are measured with equal sensitivity. Thus, these sounds may include some that are too high or too low for the human ear to hear.

The Sound Sensor can measure sound pressure levels up to 90 dB

An example using sound sensors is the following:

```
public class Listen implements SensorPortListener {
      String changed = "State changed";
      String val = "Value:";
```

```
  String oldVal = "old Value:";
  String free = "Free Mem:";
  SoundSensor sound = new SoundSensor(SensorPort.S1);

  public static void main (String[] aArg)
  throws Exception
  {
        Listen listen = new Listen();
        listen.run();
        Button.ESCAPE.waitForPressAndRelease();
        LCD.clear();
        LCD.drawString("Finished", 3, 4);
        LCD.refresh();
        Thread.sleep(2000);
  }

  public void stateChanged(SensorPort port, int value, int oldValue)
{
        if (port == SensorPort.S1 && sound.readValue() > 50)
        {
        LCD.clear();
        LCD.drawString(changed,0,0);
        LCD.drawString(val, 0, 1);
        LCD.drawInt(value,7,1);
        LCD.drawInt(sound.readValue(), 12, 1);
        LCD.drawString(oldVal, 0, 2);
        LCD.drawInt(oldValue, 11, 2);
              LCD.drawString(free, 0, 4);
            LCD.drawInt((int)(Runtime.getRuntime().freeMemory()),10,4);
        LCD.refresh();
        }
}

  private void run()
  throws InterruptedException
  {
```

```
            SensorPort.S1.addSensorPortListener(this);
        }
}
```

Compass sensor

The NXT Compass Sensor contains a digital magnetic compass that measures the earth's magnetic field and calculates a heading angle. The Compass Sensor connects to an NXT sensor port using a standard NXT wire and uses the digital I2C communications protocol. The current heading is calculated to the nearest 1° and refreshed 100 times per second.

Illustration 8: Compass sensor

A compass can be a valuable sensor that your robot can use to navigate the world. The Dinsmore 1490 compass is a cheap and durable unit that is relatively easy to interface to. It provides 8 headings (N, NE, E, SE, S, SW, W, and NW). 4 signals, N,S,E,W are read by the Basic Stamp II controller to determine the robot's heading.

```
import lejos.nxt.*;
import lejos.nxt.addon.*;

/**
 * Simple test of compass sensors.
 *
```

```
 * Works with Mindsensors and HiTechnic compass sensors.
 *
 */
 public class CompassTest {

      public static void main(String[] args) throws Exception {
            CompassSensor compass = new CompassSensor(SensorPort.S1);
            while(!Button.ESCAPE.isPressed()) {
                  LCD.clear();
                  LCD.drawInt((int) compass.getDegrees(), 0, 0);
                  LCD.refresh();
                  Thread.sleep(500);
            }
      }
}
```

Compass sensor is sensor which is not included into Lego Mindstorms NXT so you have to import the class from the package lejos.nxt.addon.*.

The way to get the degrees from the sensor is compass.getDegrees()

This kind of sensor is necessary to many projects which use local navigation.

GPS

A GPS sensor is a powerful sensor for robots because NXT robots are able to read data from GPS (Global positioning system)

Global positioning system.

The Global Positioning System (GPS) is a space-based global navigation satellite system (GNSS) that provides location and time information in all weather, anywhere on or near the Earth, where there is an unobstructed line of sight to four or more GPS satellites. It is maintained by the United States government and is freely accessible by anyone with a GPS receiver with some technical limitations which are only removed for military users.

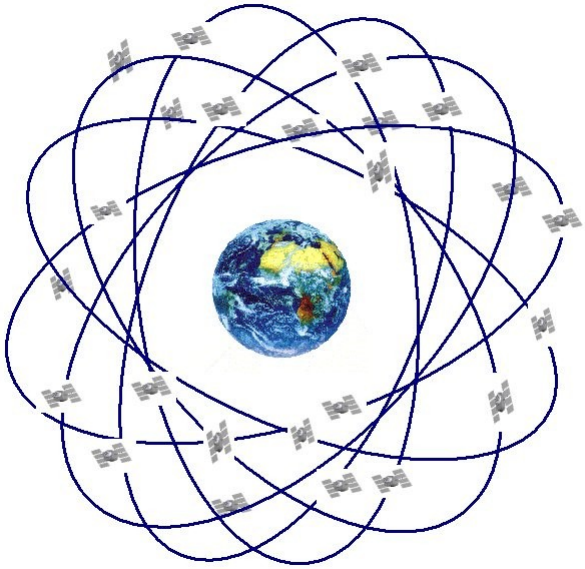


Illustration 9: Global Positioning System, GPS

How to use a GPS with your NXT brick?

With leJOS is very easy to manage information from a GPS device. In LeJOS project exists 2 ways to manage GPS data:

1. Using the package javax.microedition.location
2. Using the package lejos.gps

Using the package javax.microedition.location

In JavaME exist a package designed to manage Location services. In LeJOS we have a package to use with this kind of sensors. In the following example we connect with a Bluetooth GPS Receiver to receive data from GPS.

```
import javax.microedition.location.Coordinates;
import javax.microedition.location.Criteria;
import javax.microedition.location.Location;
import javax.microedition.location.LocationException;
import javax.microedition.location.LocationProvider;

import lejos.nxt.Button;
```

```
import lejos.nxt.LCD;
import lejos.util.Delay;

public class KmlRecorder2 {

	private static LocationProvider lp = null;
	private static Location l = null;
	private static Coordinates current = null;
	private static double course = 0;

	private static final int oneSecond = 1000;

	private static boolean getConnection(){

		boolean connected = false;

		Criteria criteria = new Criteria();

		// Get an instance of the provider:
		try {
			System.out.println("1. Connecting with a BT GPS ");
			lp = LocationProvider.getInstance(criteria);
			System.out.println("2. Connected");

			connected = true;

		}catch(LocationException e) {
			System.err.println(e.getMessage());
		}

		return connected;
	}

	/**
	 * @param args
	 */
```

```
public static void main(String[] args) {

	long iteration = 0;

	if(!getConnection()){
		Button.waitForAnyPress();
		System.exit(0);
	}else{

		System.out.println("3. Extracting data");

		WaypointRecorder wr = new WaypointRecorder();
		wr.setMode(WaypointRecorder.MODE_INTERACTIVE);
		wr.start();

		Delay.msDelay(oneSecond);

		while(!Button.ESCAPE.isDown()){

			//Get a location in every iteration
			try {
				l = lp.getLocation(-1);
			} catch (Exception e) {
				System.err.println(e.getMessage());
				continue;
			}

			//Get a coordinate object
			current = l.getQualifiedCoordinates();

			wr.updateTimestamp(l.getTimestamp());
			wr.updateCoordinate(current);

			iteration ++;

			System.out.println("Iteration: " + iteration);
```

```
                        if(Button.ENTER.isDown()){
                              wr.update();
                        }

                        Delay.msDelay(oneSecond);

                        LCD.clearDisplay();
                  }

                  wr.close();

            }

            credits(2);
            System.exit(0);
      }

      private static void credits(int seconds){
            LCD.clear();
            LCD.drawString("LEGO Mindstorms",0,1);
            LCD.drawString("NXT Robots  ",0,2);
            LCD.drawString("run better with",0,3);
            LCD.drawString("Java leJOS",0,4);
            LCD.drawString("www.lejos.org",0,6);
            LCD.refresh();
            //try {Thread.sleep(seconds*1000);} catch (Exception e) {}
      }

}
```

Using the package lejos.gps

The package lejos.gps allows the developer to manage gps information from the following NMEA Sentences:

GGA - essential fix data which provide 3D location and accuracy data.

GSA - GPS DOP and active satellites

GPV - GPS Satellites in view

RMC - NMEA has its own version of essential gps pvt (position, velocity, time) data. It is called RMC

VTG - Track Made Good and Ground Speed.

With data from this NMEA sentences it is possible to return Date objects to manage date information or Coordinates objects to calculate distances.

A GPS is perfect if you have a robot which operates in a outdoor area. GPS devices are sensitives to the place where it operates. You can measure the quality of the data from any GPS device if you see the number of satellites and error data which you can get from the following methods:

```
import lejos.addon.gps.*;
import lejos.nxt.*;
import lejos.nxt.comm.*;
import lejos.util.Stopwatch;
import lejos.util.TextMenu;

import java.util.*;
import java.io.*;
import javax.bluetooth.*;
import javax.microedition.location.*;

/**
 * This example show how to:
 *
 * + Connect with a GPS Device with a NXT brick with leJOS
 * + Get Data from GGA NMEA Sentence
 * + Get Data from RMC NMEA Sentence
 * + Get Data from VTG NMEA Sentence
 * + Get Data from GSV NMEA Sentence
 * + Get Data from GSA NMEA Sentence
 * + Use JRS-179 Objects
```

```
 * + Use Date Objects with leJOS
 *
 *
 * This example is experimental. It is necessary to test more time
 *
 * Click on left and right button to show to show more data about GPS.
 *
 * @author BB
 * @author Juan Antonio Brenha Moral
 */
public class BTGPS{
      private static String appName = "GPS";
      private static String appVersion = "v6.8";

      //Inquire code
      private static int cod = 0; // 0 picks up every Bluetooth device regardless
of Class of Device (cod).

      //Bluetooth
      private static RemoteDevice GPSDevice = null;
      private static GPS gps = null;
      private static InputStream in = null;

      //GPS Pin
      private static final byte[] pin = {(byte) '0', (byte) '0', (byte) '0', (byte)
'0'};

      //GPS Data
      private static Date connectionMoment;
      private static Date now;
      //private static Satellite ns;
      private static Coordinates origin;
      private static Coordinates current;

      public static void main(String[] args) {
```

```
        //Detect GPS Device
        boolean GPSDetected = false;
        GPSDetected = discoverBTDevices();

        if(GPSDetected){
              //Connect with GPS Device
              int connectionStatus = 0;
              connectionStatus = connectGPS();

              if(connectionStatus == 2){
                    //Show data from GPS Receiver
                    showData();//GUI
              }else{
                    if(connectionStatus == -1){
                          LCD.drawString("No connection", 0, 7);
                    }else if(connectionStatus == -2){
                          LCD.drawString("Something goes bad", 0, 7);
                    }
                    try {Thread.sleep(2000);} catch (Exception e) {}
              }
              LCD.refresh();
        }else{
              LCD.drawString("No detected GPS", 0, 3);
              LCD.refresh();
              try {Thread.sleep(2000);} catch (Exception e) {}
        }
        credits(2);
        System.exit(0);
  }//End main

  /**
   * This method, show all BT Devices with BT Services enable
   * User choose a GPS device to connect
   *
   * Developer note: This method has a bug when you click in exit button twice
   */
```

```
	static boolean discoverBTDevices(){
		boolean GPSDetected = false;

		LCD.clear();
		LCD.drawString("Searching...", 0, 0);
		LCD.refresh();
		//Make an BT inquire to get all Devices with BT Services enable
		Vector devList = Bluetooth.inquire(5, 10,cod);

		//If exist GPS Devices near
		if (devList.size() > 0){
			String[] names = new String[devList.size()];
			for (int i = 0; i < devList.size(); i++) {
				RemoteDevice btrd = ((RemoteDevice) devList.elementAt(i));
				names[i] = btrd.getFriendlyName(true);
			}

			TextMenu searchMenu = new TextMenu(names,1);
			String[] subItems = {"Connect"};
			TextMenu subMenu = new TextMenu(subItems,4);

			int selected;
			do {
				LCD.clear();
				LCD.drawString("Found",6,0);
				LCD.refresh();
				//Menu 1: Show all BT Devices
				selected = searchMenu.select();
				if (selected >=0){
					RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(selected));
					LCD.clear();
					LCD.drawString("Found",6,0);
					LCD.drawString(names[selected],0,1);
					LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
					//Menu 2: Show GPS Device
```

```
                              int subSelection = subMenu.select();
                              if (subSelection == 0){
                                    GPSDetected = true;
                                    GPSDevice = btrd;
                                    break;
                              }
                        }
                  } while (selected >= 0);
            }else{
                  GPSDetected = false;
            }

            return GPSDetected;
      }

      /**
       * This method connect with a RemoteDevice.
       * If the connection has success then the method create an instance of
       * the class GPS which manages an InputStream
       *
       * @return
       */
      static int connectGPS(){
            int result;
            Bluetooth.addDevice(GPSDevice);

            BTConnection btGPS = null;
            btGPS = Bluetooth.connect(GPSDevice.getDeviceAddr(), NXTConnection.RAW,
pin);

            if(btGPS == null){
                  result  = -1;//No connection
            }else{
                  result = 1;//Connection Sucessful
            }
```

```
		try{
			in = btGPS.openInputStream();
			gps = new GPS(in);
			//gps.updateValues(true);

			result = 2;//
		}catch(Exception e) {
			result = -2;
		}

		return result;
	}

	/**
	 * Show the example GUI
	 */
	static void showData(){
		LCD.clear();
		//int sentenceCount = 0;

		Stopwatch sw;
		sw = new Stopwatch();

		//boolean flag = true;
		int NSAT = 0;
		int GPSDataQuality = 0;
		int checkTime = 10000;

		//Circular System
		int GPSScreens = 8;
		int GPSCurrentScreen = 1;

		LCD.drawString(appName + " " + appVersion, 0,0);

		//FirstConnection
		boolean firstMomentFlag = false;
```

```
            while(!Button.ESCAPE.isDown()){
                  NSAT = gps.getSatellitesTracked();
                  GPSDataQuality = Math.round((NSAT * 100)/4);

                  LCD.drawString("        ", 9, 0);
                  LCD.drawString(GPSDataQuality + "%", 9, 0);
                  LCD.drawString("OK", 13, 0);

                  if(sw.elapsed() >= checkTime){
                        sw.reset();
                        if(GPSDataQuality >=8){
                              Sound.twoBeeps();
                        }else if(GPSDataQuality >=4){
                              Sound.beep();
                        }else{
                              //Sound.buzz();
                        }
                  }

                  if(!firstMomentFlag){
                        Date tempDate = gps.getDate();
                        int hours = tempDate.getHours();
                        int minutes = tempDate.getMinutes();
                        int seconds = tempDate.getSeconds();
                        connectionMoment = new Date();
                        connectionMoment.setHours(hours);
                        connectionMoment.setMinutes(minutes);
                        connectionMoment.setSeconds(seconds);


                        origin = new
Coordinates(gps.getLatitude(),gps.getLongitude(),gps.getAltitude());

                        //Repeat the operation until you have valid data:
                        if(
```

```
                        (seconds != 0) &&
                        (gps.getLatitude() != 0)){

                        firstMomentFlag = true;
                  }
            }

            now = gps.getDate();
            current = new
Coordinates(gps.getLatitude(),gps.getLongitude(),gps.getAltitude());



            //Circular System
            if (Button.LEFT.isDown()){
                  if(GPSCurrentScreen == 1){
                        GPSCurrentScreen = GPSScreens;
                  }else{
                        GPSCurrentScreen--;
                  }
            }

            if (Button.RIGHT.isDown()){
                  if(GPSCurrentScreen == GPSScreens){
                        GPSCurrentScreen = 1;
                  }else{
                        GPSCurrentScreen++;
                  }
            }

            //Reset
            if (Button.ENTER.isDown()){
                  GPSCurrentScreen  =1;
            }

            if(GPSCurrentScreen == 1){
                  showGGAUI();
```

```
                }else if(GPSCurrentScreen == 2){
                      showRMCUI();
                }else if(GPSCurrentScreen == 3){
                      showVTGUI();
                }else if(GPSCurrentScreen == 4){
                      showGPSTimeUI();
                }else if(GPSCurrentScreen == 5){
                      //By Security
                      if(gps.getSatellitesTracked() >= 4){
                            showSatTableUI();
                      }
                }else if(GPSCurrentScreen == 6){
                      showSatUI();
                }else if(GPSCurrentScreen == 7){
                      showSatIDUI();
                }else if(GPSCurrentScreen == 8){
                      showCoordinatesUI();
                }

                LCD.refresh();
                try {Thread.sleep(1000);} catch (Exception e) {}
          }
     }

     /**
      * Show GGA Basic Data from GPS
      */
     private static void showGGAUI(){
          refreshSomeLCDLines();
          LCD.drawString("GGA", 0, 2);

          LCD.drawString("Tim " + now.getHours() + ":" + now.getMinutes() + ":" +
now.getSeconds() + "", 0, 3);
          LCD.drawString("Lat " + gps.getLatitude(), 0, 4);
          LCD.drawString("" + gps.getLatitudeDirection() , 15, 4);
          LCD.drawString("Lon " + gps.getLongitude(), 0, 5);
```

```
            LCD.drawString("" + gps.getLongitudeDirection() , 15, 5);
            LCD.drawString("Alt " + gps.getAltitude(), 0, 6);
            LCD.drawString("Sat " + gps.getSatellitesTracked(), 0, 7);
            LCD.drawString("QOS " + gps.getFixMode(), 6, 7);
            LCD.refresh();
      }

      /**
       * Show RMC Data from GPS
       */
      private static void showRMCUI(){
            refreshSomeLCDLines();
            LCD.drawString("RMC", 0, 2);

            LCD.drawString("Dat " + now.getDay() + "/" + now.getMonth() + "/" +
now.getYear() + "", 0, 3);
            LCD.drawString("Com " + gps.getCompassDegrees(), 0, 4);
            LCD.refresh();
      }

      /**
       * Show VTG Data from GPS
       */
      private static void showVTGUI(){
            refreshSomeLCDLines();
            LCD.drawString("VTG", 0, 2);

            LCD.drawString("Spe " + gps.getSpeed(), 0, 3);
            LCD.refresh();
      }

      /**
       * Show Time Data from GPS
       */
      private static void showGPSTimeUI(){
            refreshSomeLCDLines();
```

```
            LCD.drawString("GPS time data", 0, 2);


            LCD.drawString("Tim " + now.getHours() + ":" + now.getMinutes() + ":" +
now.getSeconds() + "", 0, 3);
            LCD.drawString("Dat " + now.getDay() + "/" + now.getMonth() + "/" +
now.getYear() + "", 0, 4);
            LCD.refresh();
      }


      private static void showSatTableUI(){
            refreshSomeLCDLines();
            LCD.drawString("Sat table", 0, 2);



            Satellite ns1 = gps.getSatellite(0);
            Satellite ns2 = gps.getSatellite(1);
            Satellite ns3 = gps.getSatellite(2);
            Satellite ns4 = gps.getSatellite(3);

            LCD.drawString(" PRN Ele Azi SRN",0,3);
            LCD.drawString("1 " + ns1.getPRN(),0,4);
            LCD.drawString("2 " + ns2.getPRN(),0,5);
            LCD.drawString("3 " + ns3.getPRN(),0,6);
            LCD.drawString("4 " + ns4.getPRN(),0,7);
            LCD.drawString("" + ns1.getElevation(),5,4);
            LCD.drawString("" + ns2.getElevation(),5,5);
            LCD.drawString("" + ns3.getElevation(),5,6);
            LCD.drawString("" + ns4.getElevation(),5,7);
            LCD.drawString("" + ns1.getAzimuth(),9,4);
            LCD.drawString("" + ns2.getAzimuth(),9,5);
            LCD.drawString("" + ns3.getAzimuth(),9,6);
            LCD.drawString("" + ns4.getAzimuth(),9,7);
            LCD.drawString("" + ns1.getSignalNoiseRatio(),13,4);
            LCD.drawString("" + ns2.getSignalNoiseRatio(),13,5);
            LCD.drawString("" + ns3.getSignalNoiseRatio(),13,6);
            LCD.drawString("" + ns4.getSignalNoiseRatio(),13,7);
```

```
		LCD.refresh();
	}

	/**
	 * Show Sat Data
	 */
	private static void showSatUI(){
		refreshSomeLCDLines();
		LCD.drawString("Sat quality data", 0, 2);

		LCD.drawString("Mode " + gps.getSelectionType(), 0, 3);
		LCD.drawString("Value " + gps.getFixType(), 8, 3);
		LCD.drawString("NSat " + gps.getSatellitesTracked(), 0, 4);
		LCD.drawString("PDOP " + gps.getPDOP(), 0, 5);
		LCD.drawString("HDOP " + gps.getHDOP(), 0, 6);
		LCD.drawString("VDOP " + gps.getVDOP(), 0, 7);
		LCD.refresh();
	}

	/**
	 * Show Sat ID
	 */
	private static void showSatIDUI(){
		refreshSomeLCDLines();
		LCD.drawString("Sat detected", 0, 2);

		int SV[] = gps.getPRN();

		int cols[] = {0,4,8,12};
		int rows[] = {3,4,5};

		LCD.drawString("" + SV[0], cols[0], rows[0]);
		LCD.drawString("" + SV[1], cols[1], rows[0]);
		LCD.drawString("" + SV[2], cols[2], rows[0]);
		LCD.drawString("" + SV[3], cols[3], rows[0]);
		LCD.drawString("" + SV[4], cols[0], rows[1]);
```

```
            LCD.drawString("" + SV[5], cols[1], rows[1]);
            LCD.drawString("" + SV[6], cols[2], rows[1]);
            LCD.drawString("" + SV[7], cols[3], rows[1]);
            LCD.drawString("" + SV[8], cols[0], rows[2]);
            LCD.drawString("" + SV[9], cols[1], rows[2]);
            LCD.drawString("" + SV[10], cols[2], rows[2]);
            LCD.drawString("" + SV[11], cols[3], rows[2]);
            LCD.refresh();
      }

      /**
       * Show Sat Data
       */
      private static void showCoordinatesUI(){
            refreshSomeLCDLines();
            LCD.drawString("GPS Session", 0, 2);

            LCD.drawString("Ini " + connectionMoment.getHours() + ":" +
connectionMoment.getMinutes() + ":" + connectionMoment.getSeconds() + "", 0, 3);
            LCD.drawString("Now " + now.getHours() + ":" + now.getMinutes() + ":" +
now.getSeconds() + "", 0, 4);

            LCD.drawString("Dis " + Math.round((float)origin.distance(current)), 0,
5);
            LCD.drawString("Azi " + Math.round((float)origin.azimuthTo(current)),
0, 6);
            LCD.drawString("Com " + gps.getCompassDegrees(), 0, 7);
            LCD.drawString("N", 8, 6);
            LCD.drawString("N", 8, 7);
            LCD.refresh();
      }

      /**
       * Clear some LCD lines
       */
      private static void refreshSomeLCDLines(){
            LCD.drawString("                    ", 0, 2);
```

```
            LCD.drawString("                    ", 0, 3);
            LCD.drawString("                    ", 0, 4);
            LCD.drawString("                    ", 0, 5);
            LCD.drawString("                    ", 0, 6);
            LCD.drawString("                    ", 0, 7);
      }

      private static void credits(int seconds){
            LCD.clear();
            LCD.drawString("LEGO Mindstorms",0,1);
            LCD.drawString("NXT Robots  ",0,2);
            LCD.drawString("run better with",0,3);
            LCD.drawString("Java leJOS",0,4);
            LCD.drawString("www.lejos.org",0,6);
            LCD.refresh();
            try {Thread.sleep(seconds*1000);} catch (Exception e) {}
      }
}//End Class
```

If you study with detail the code then you will see that the method stablish a Bluetooth connection and return an InputStream object which is used by a GPS Object to process NMEA sentences.

NXTCam

Introduction

NXTCam is the new Lego Mindstorm sensor designed to add artificial vision features. NXTCam is not a usual sensor as Lego Ultrasonic sensor. Before plugging this sensor, it is necessary to train it.

The NXTCam provides following capabilities:

- Track up to 8 different colorful objects at 30 frames/second
- Configure the NXTCam using USB interface on Windows XP, Windows Vista.
- Supports two tracking modes: Object tracking and Line tracking.
- Provide real-time tracked object statistics (number of objects, color of objects, bounding box coordinates or line coordinates) through a standard NXT sensor port.
- Tracked image resolution of 88 x 144 pixels at 30 frames/second

- Perform full-resolution (176 x 144) pixels color image dumps to PC via USB port.
- Low power consumption (the entire system only draws 60 mA)
- Uses NXT compatible I2C protocol for communications.
- Supports Auto Detecting Parallel Architecture (ADPA) for NXT sensor bus. This means that NXTCam can coexist with LEGO or third party digital sensor on the same NXT port. ADPA support enables user to employ several sensors on the same port without the need of external sensor multiplexer, reducing the overall size without compromising the functionality.

Illustration 10: NXTCam sensor

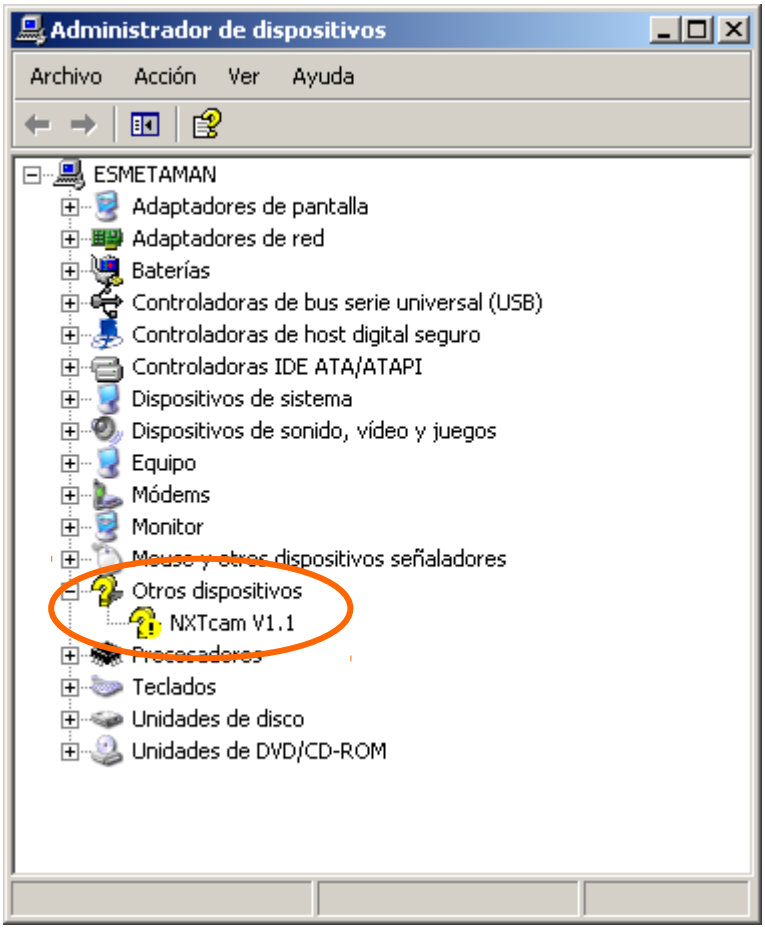
Mindsensors has created a sourceforge project to de create a tool that you need to install and use to train your NXTCam. http://nxtcamview.sourceforge.net/

In this section we are going to explain:

1. Install NXTCam driver
2. Install NXTCamView software
3. Learn to use NXTCamView
4. Learn the class NXTCam in NXJ
5. Learn NXTCam API
6. Learn to train NXTCam with NXTCamView
7. Detect a object with NXJ
8. Create a Radar robot with NXTCam

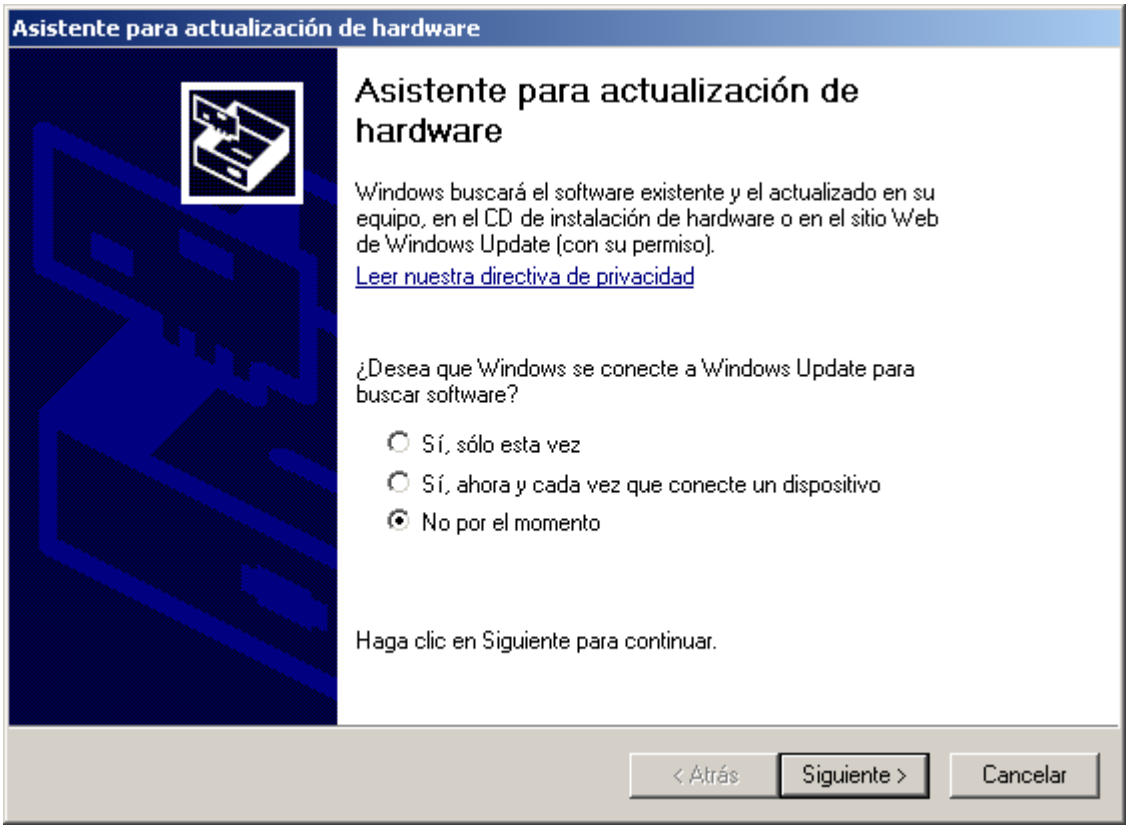
Install NXTCam Driver

The first time when you connect your NXTCam sensor with your computer, you notice in Device Manager Windows, that this USB device is not recognized by your Operating System:

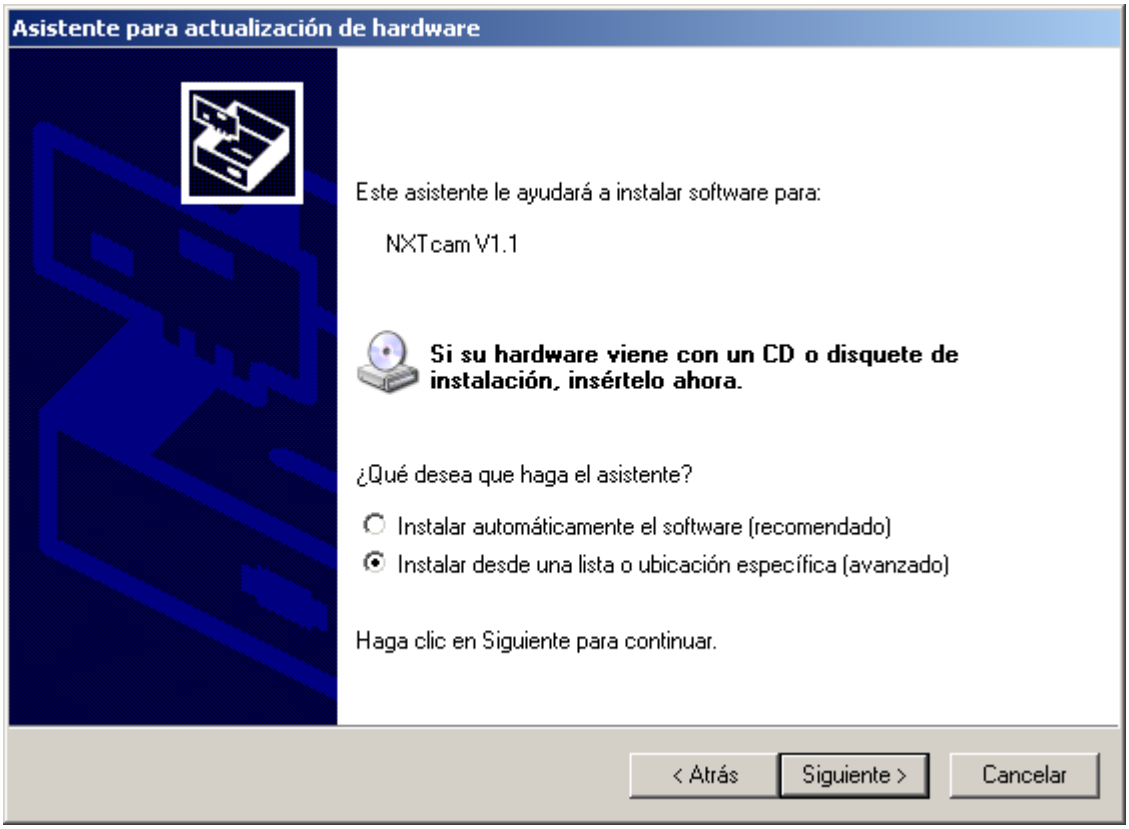


To associate this device with the correct driver, it is necessary to indicate Windows the location of NXTCam Drivers using the assistant.

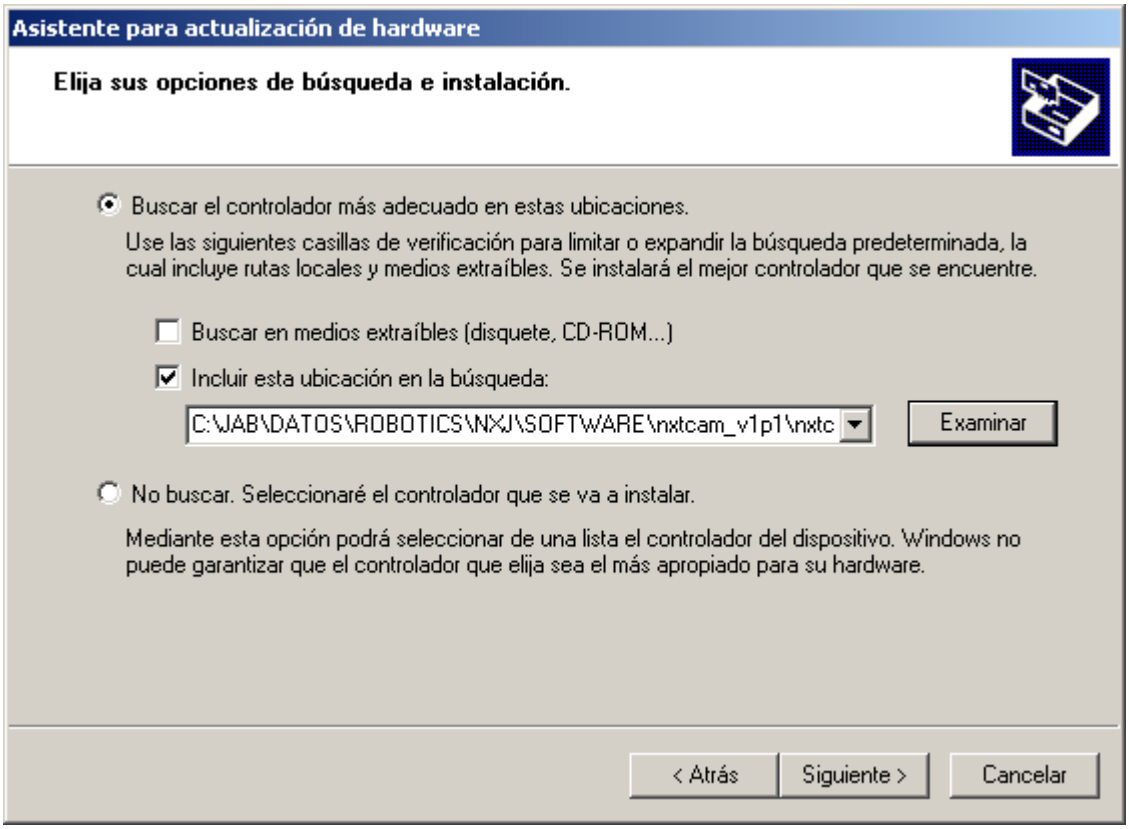
Step1: Choose the option to not use Windows Update to discover NXTCam driver.
You can download the driver from Mindsensors website. Use the following URL:



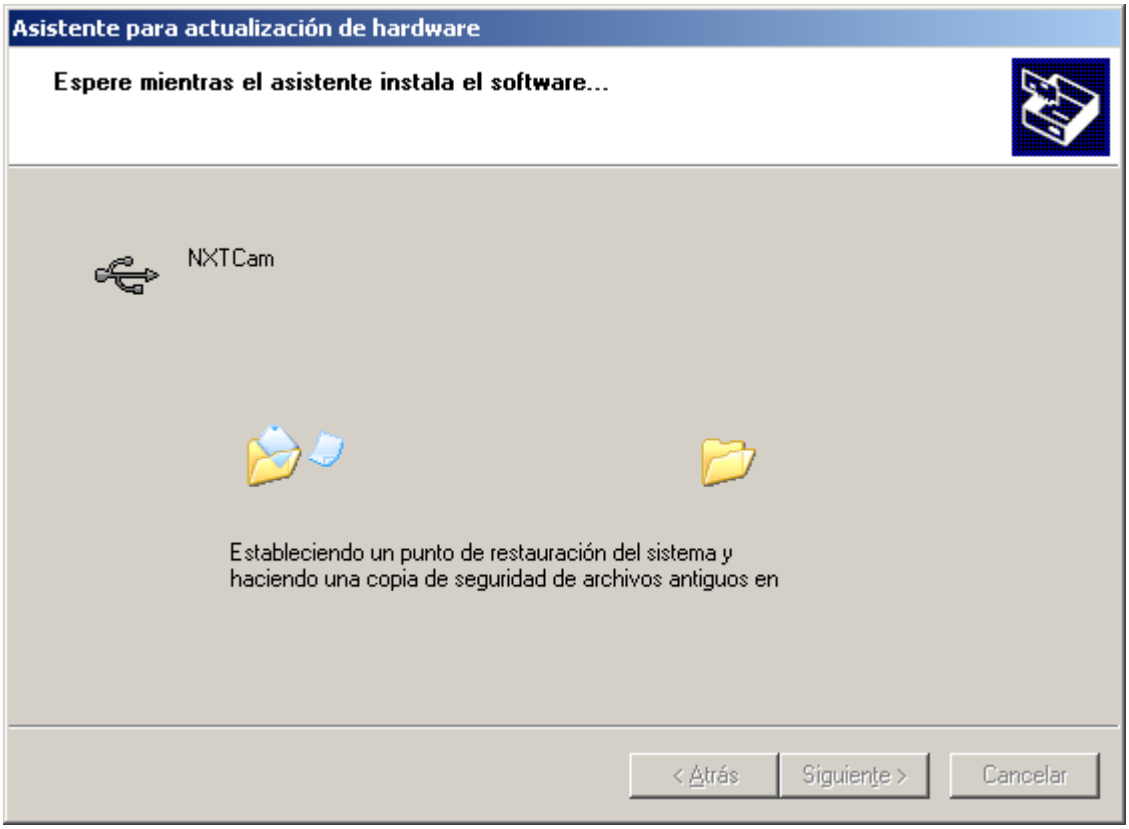
Step2: Edit the path where you have stored NXTCam driver.
Indicate that you know the location where you have stored the driver.



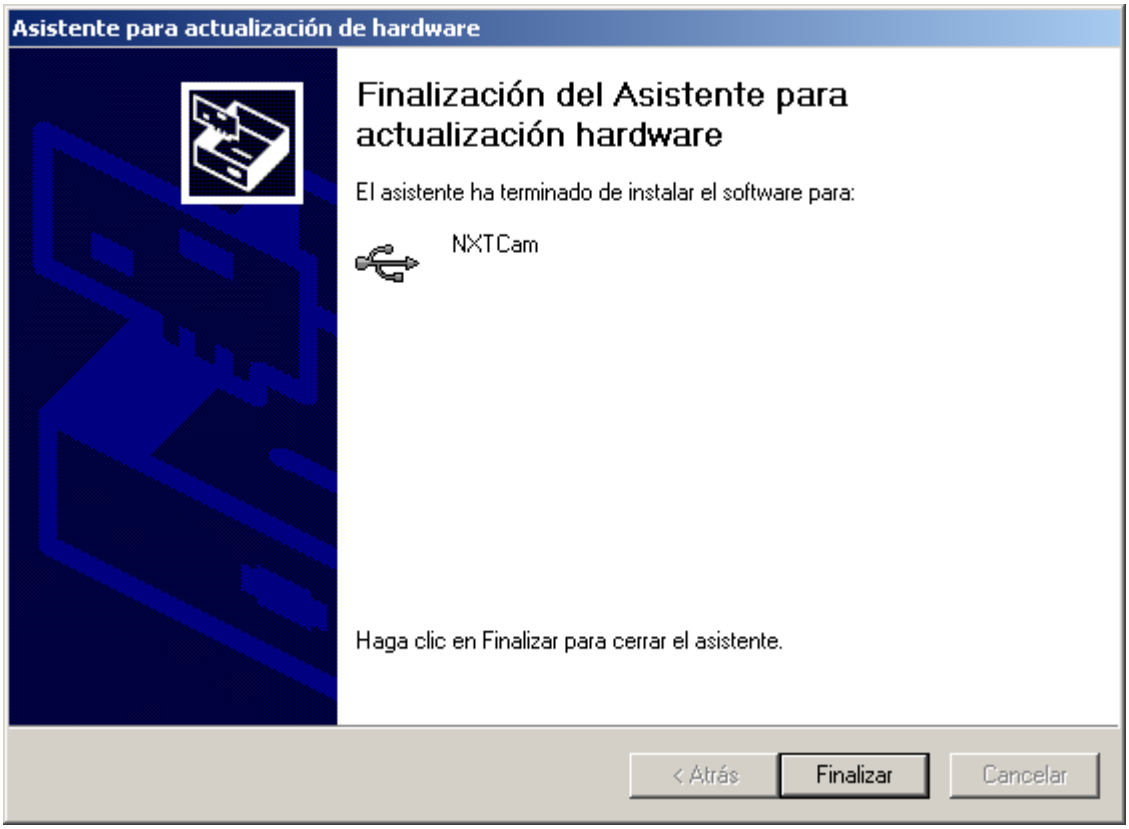
Type the path:



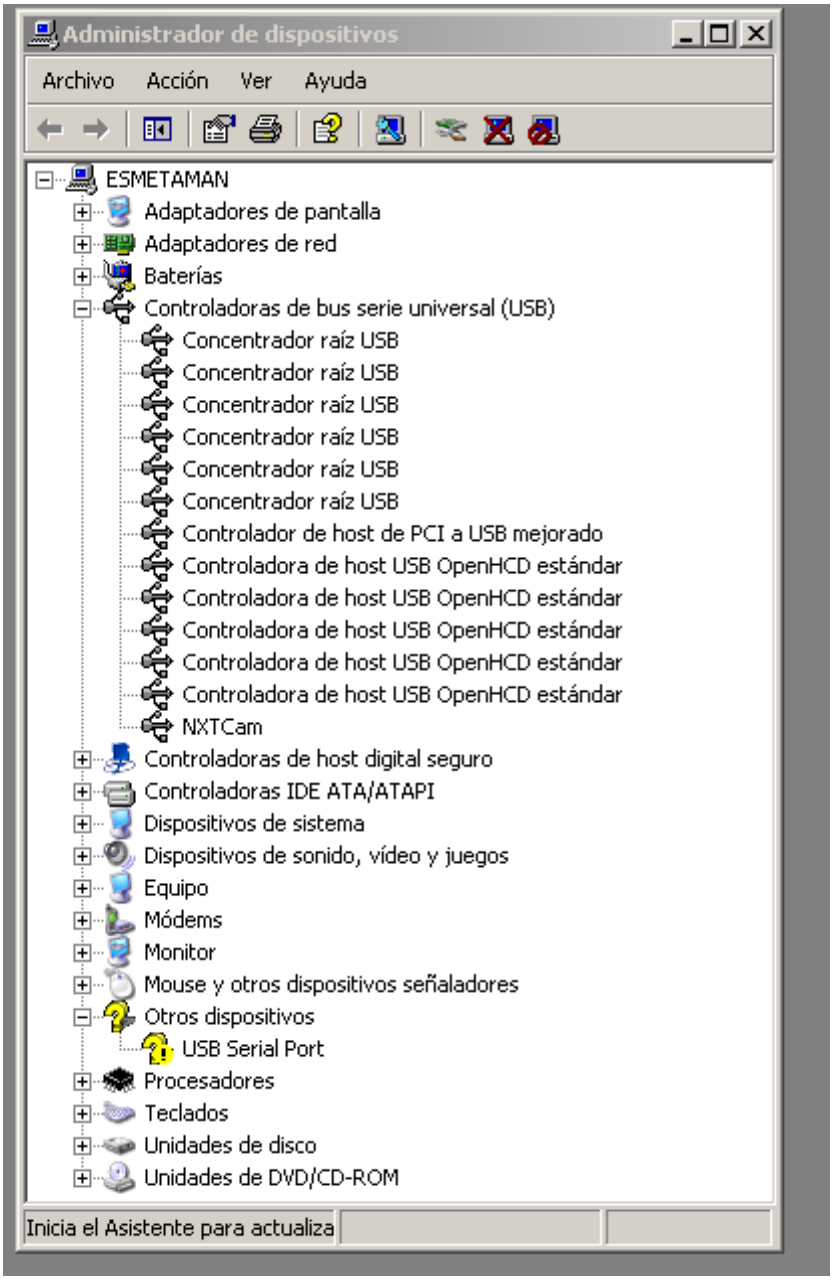
When you click in Next button, Windows will associate NXTCam with the driver.



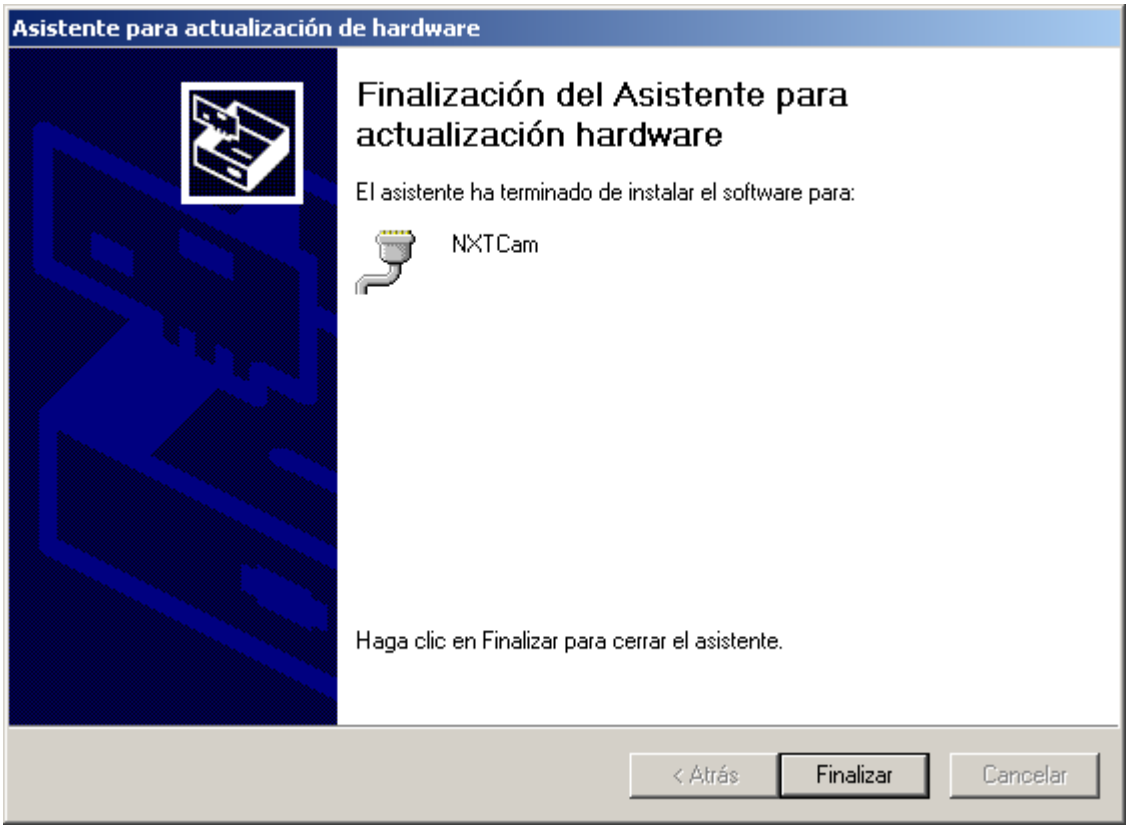
In this moment, your device has been recognized.



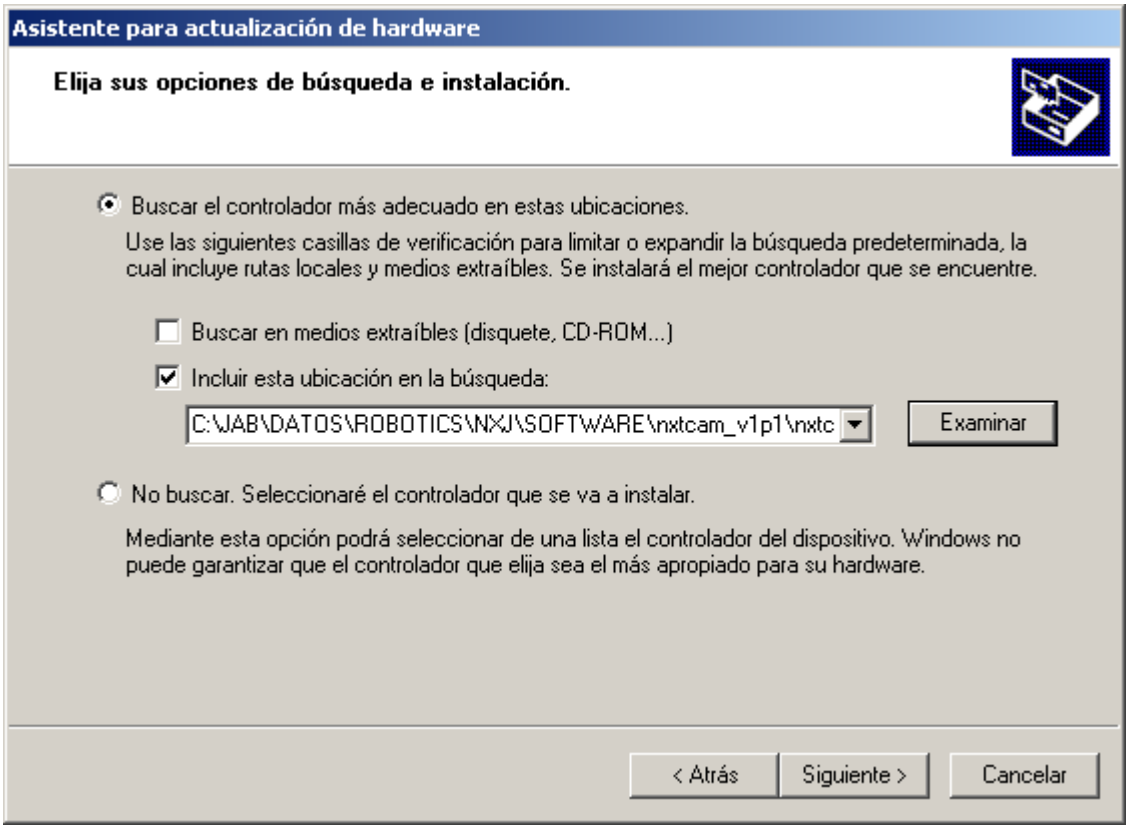
If we try to see Device Manager Window again you will see a new problem, it is necessary to install a driver for USB Serial Port, then we have to repeat the previous task twice to finish.



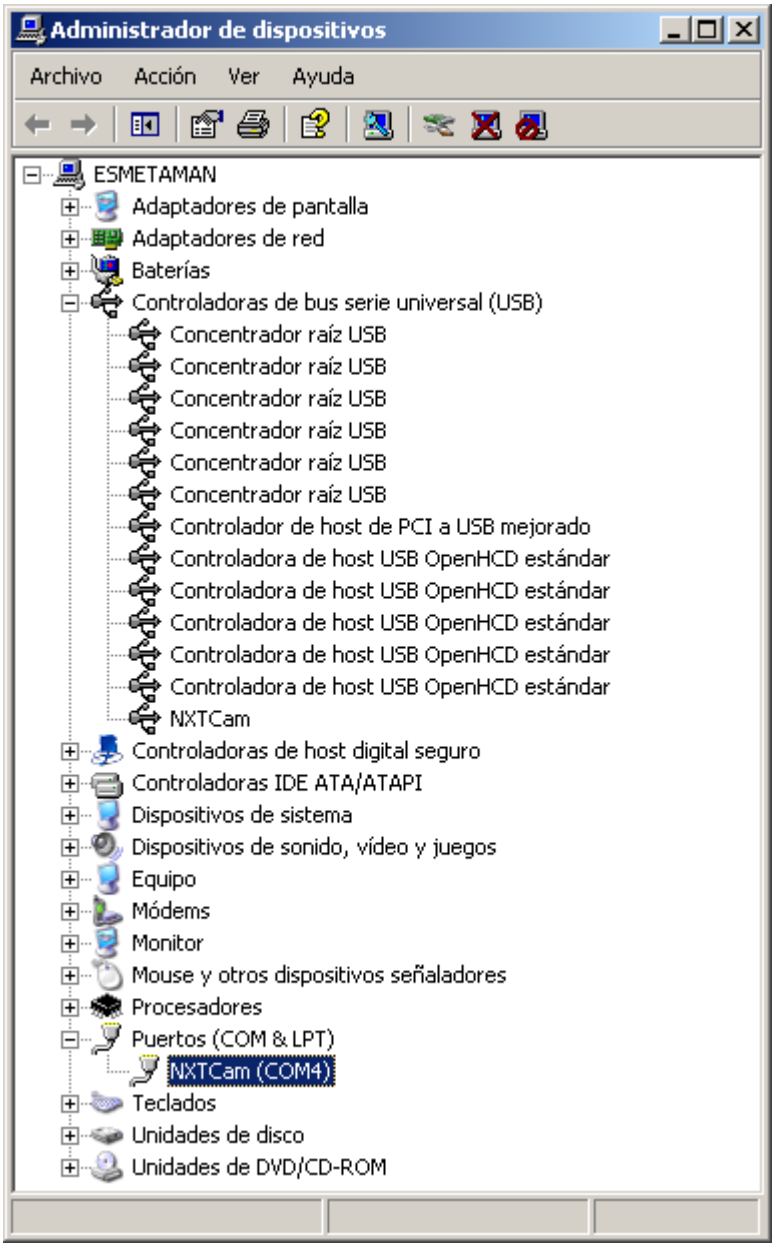
Select the node USB Serial Port in the tree menu and right-click to update the driver then follow the indications showed by the assistant:



Select the same path you use when you installed NXTCam Driver.



When you finish the process, check again your Device Manager Window, you will not see any problem with NXTCam

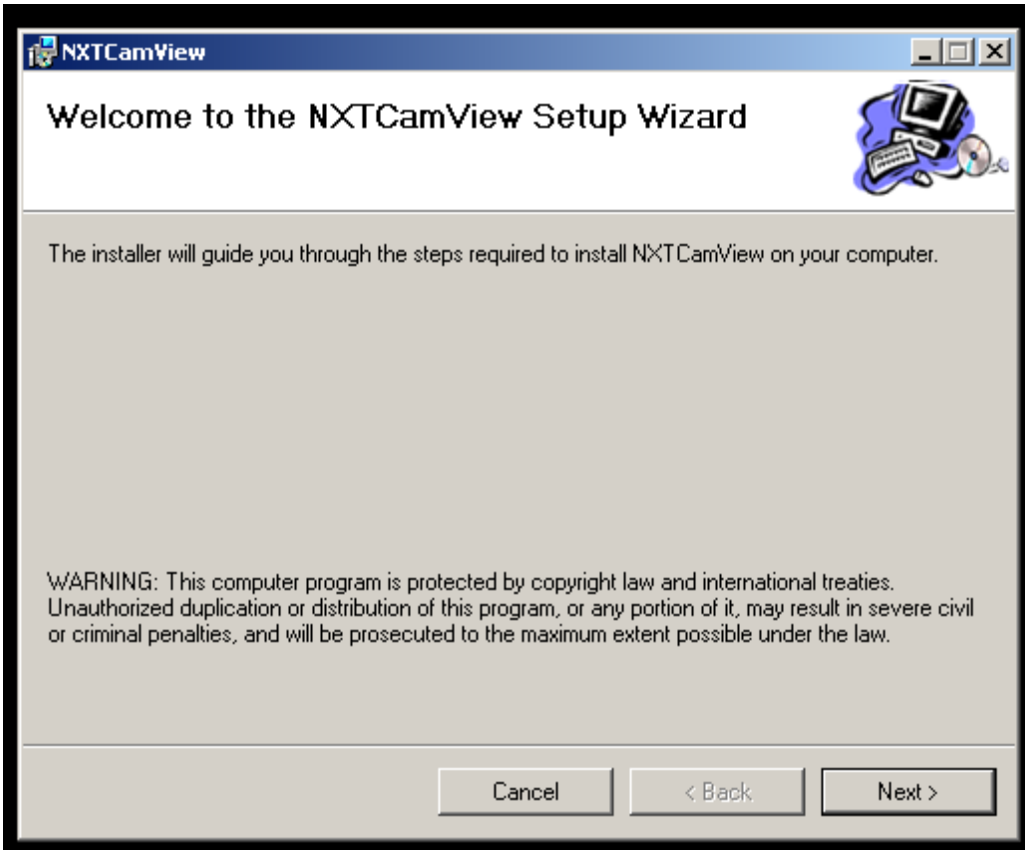


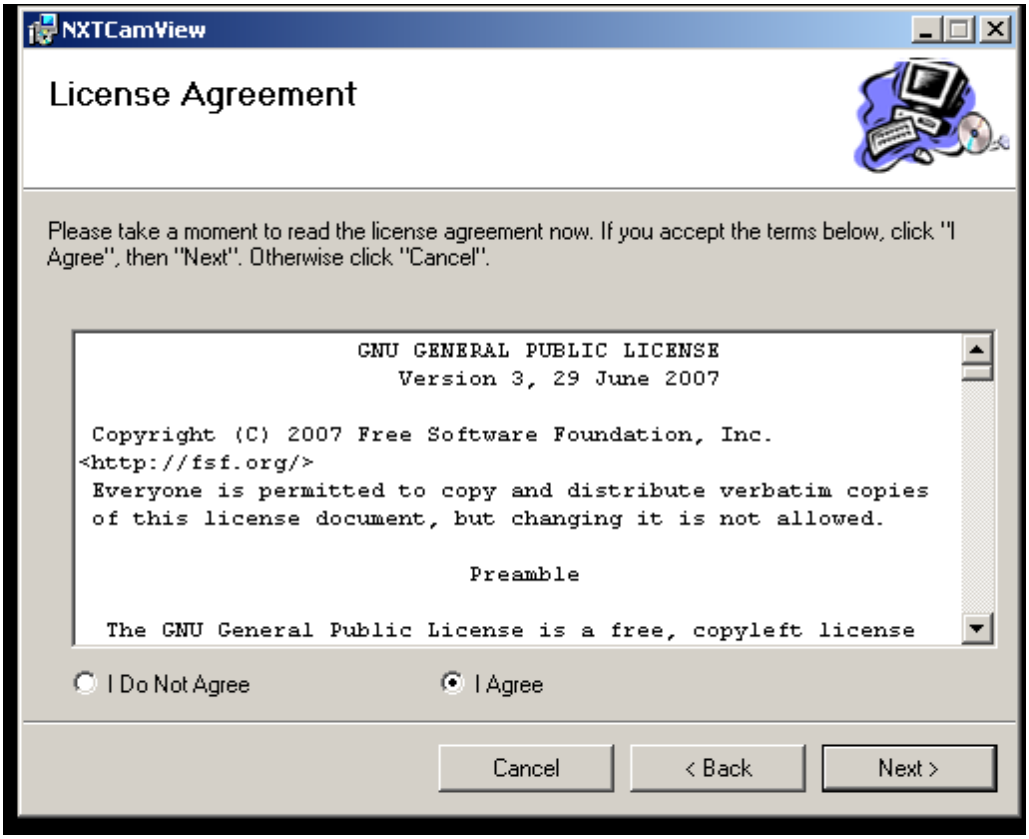
Install NXTCamView

NXTCamView is a .NET application developed by Mindsensors community and stored in the website www.sourceforge.net .Latest release of NXTCamView can be downloaded in the following URL: http://nxtcamview.sourceforge.net/

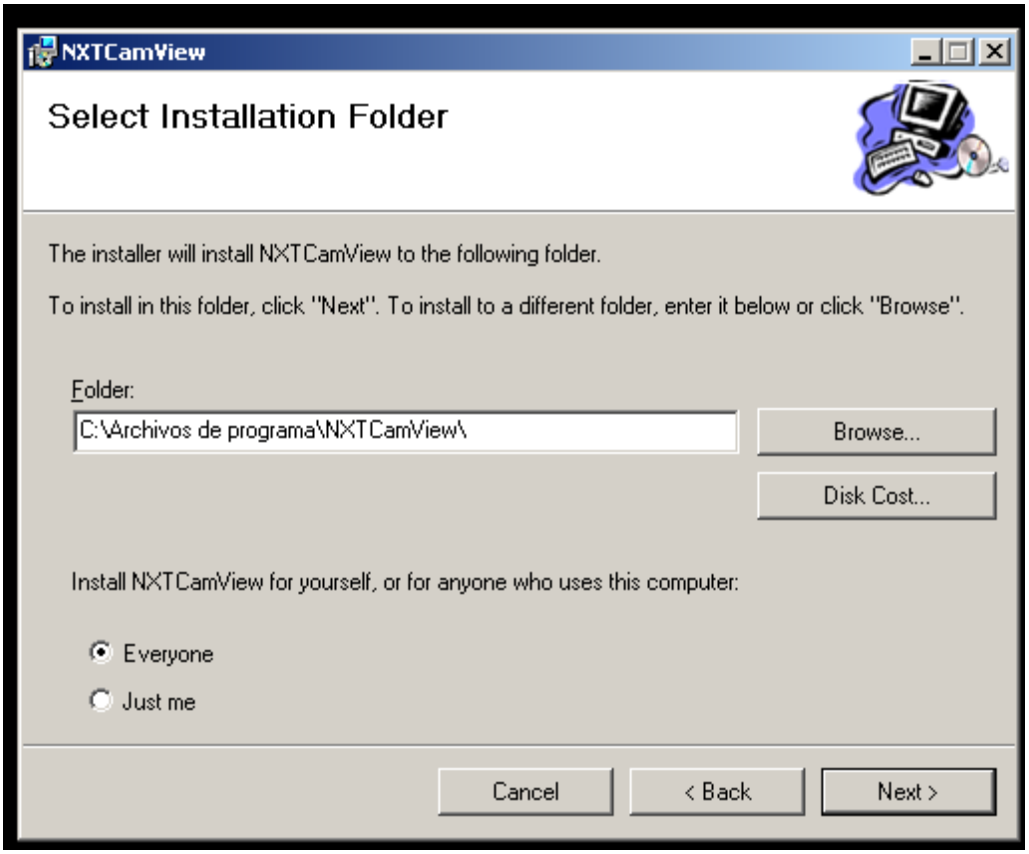
When you download the software and init the installation, NXTCamView offer you an assistant in the installation's process.

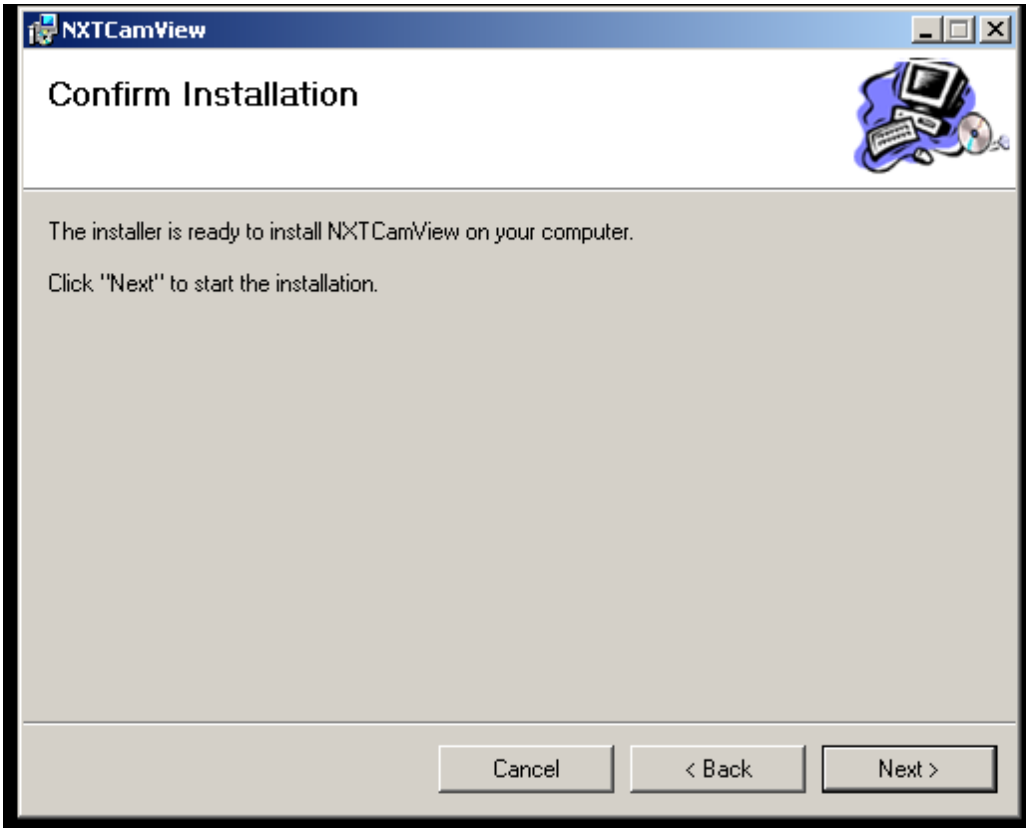
Step1: Accept licence:


NXTCamView
Welcome to the NXTCamView Setup Wizard
The installer will guide you through the steps required to install NXTCamView on your computer.
WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.
Cancel
< Back
Next >



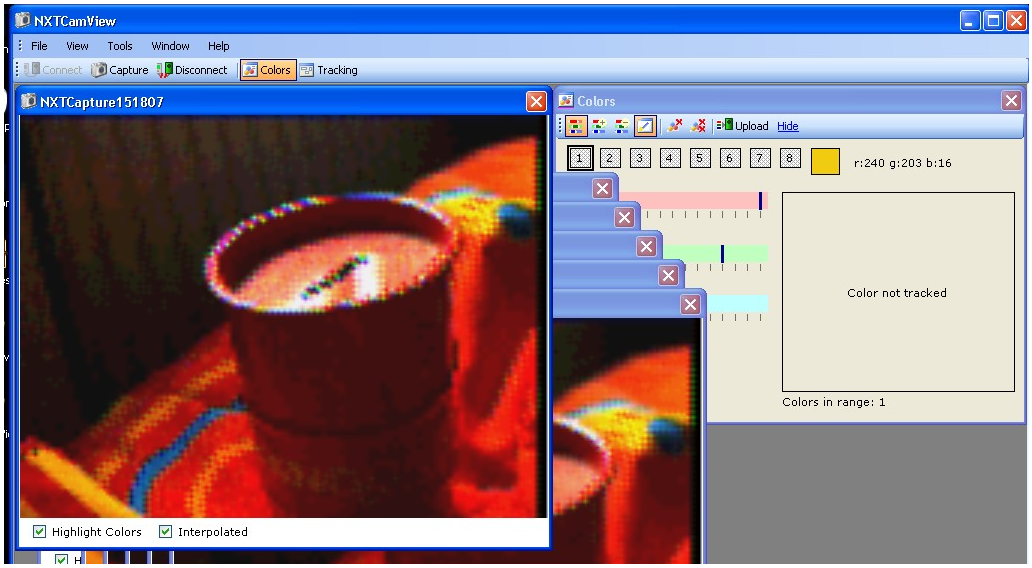
Step2: installation path


NXTCamView
Select Installation Folder
The installer will install NXTCamView to the following folder.
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".
Folder:
C:\Archivos de programa\NXTCamView\
Browse...
Disk Cost...
Install NXTCamView for yourself, or for anyone who uses this computer:
Everyone
Just me
Cancel
< Back
Next >



Calibratin lens

If you want to use NXTCam, it is necessary to callibrate the lens. It is necessary to make several captures until you are sure that the physical calibration process is right for your purpose.



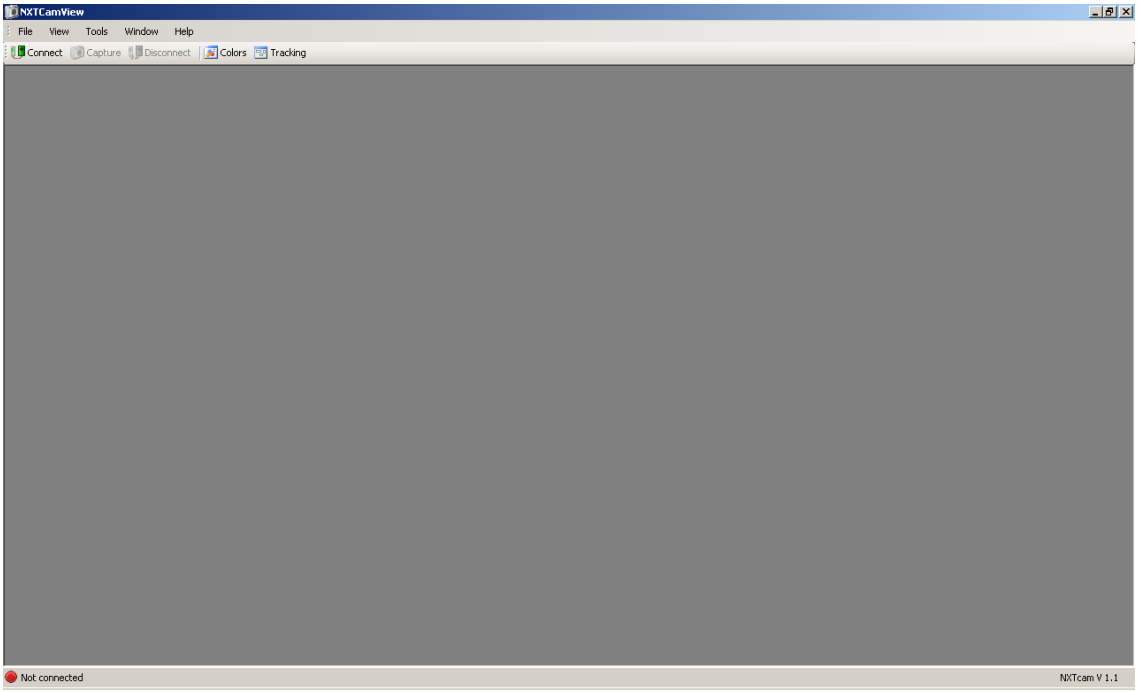
Training NXTCam with a colour pattern
NXTCam is a sensor that manages Color patterns then to establish when you use the tool NXTCamView.
In the following URL, http://nxtcamview.sourceforge.net/DemoScreenCam.htm, there are videos to establish color patterns.

In this paper, we learn to detect the following objects:

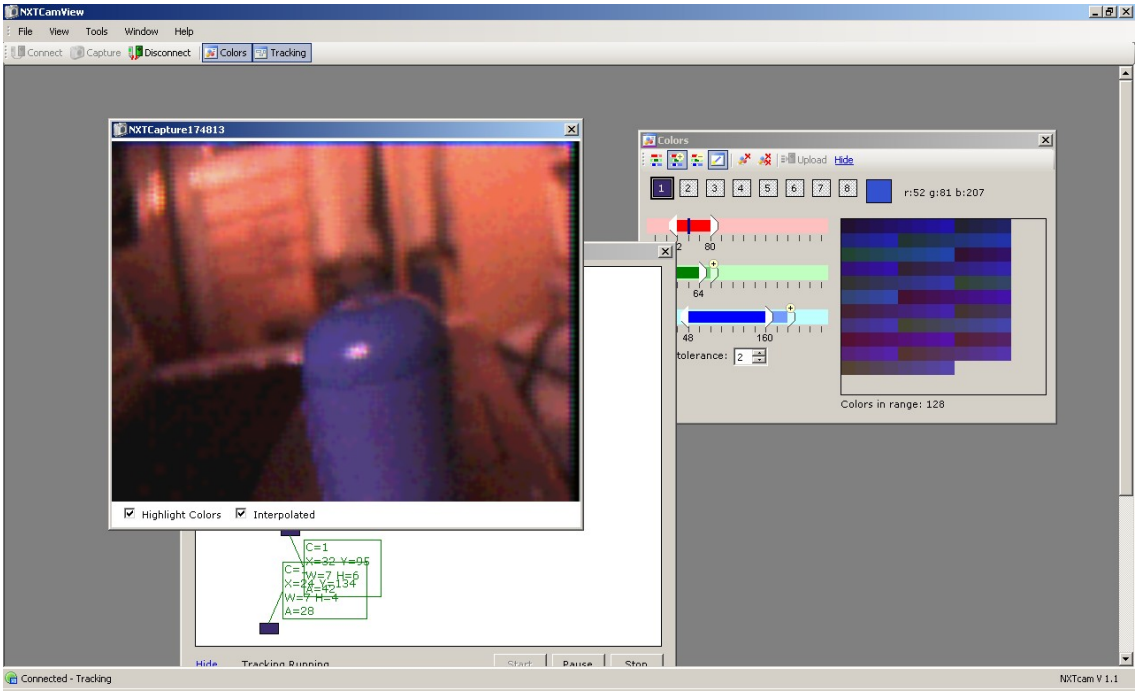
1. Lego Air Tank
2. Fluorescent Text liner

Lego air tank
When you use Pneumatic Pieces, one typical piece is a Lego Air Tank:

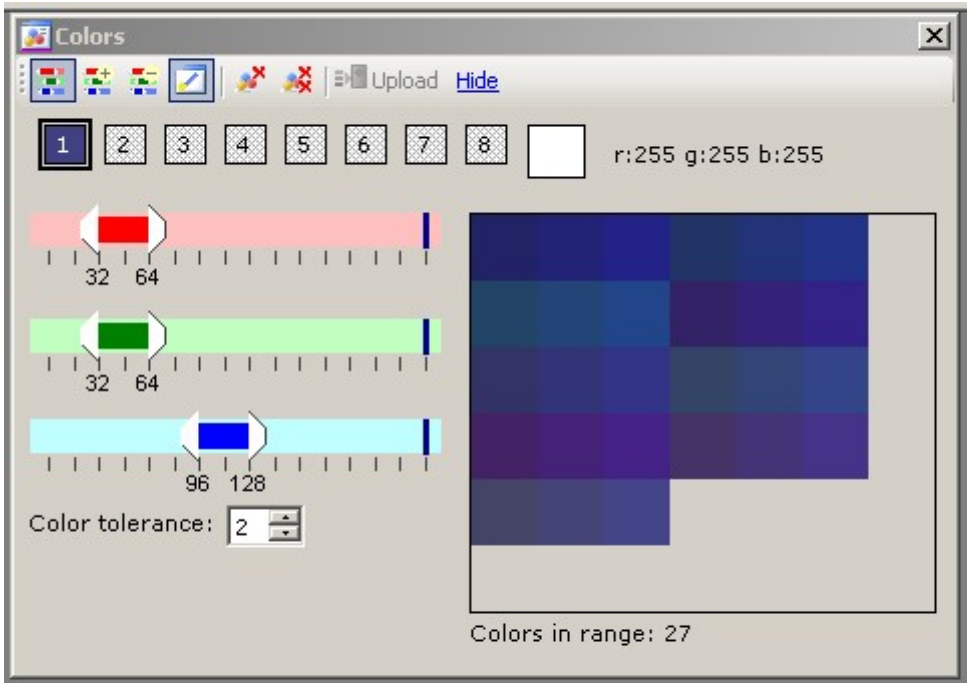
If we want to add a color map to detect Lego Air Tank Pieces then execute NXTCamView and connect NXTCam in your computer.



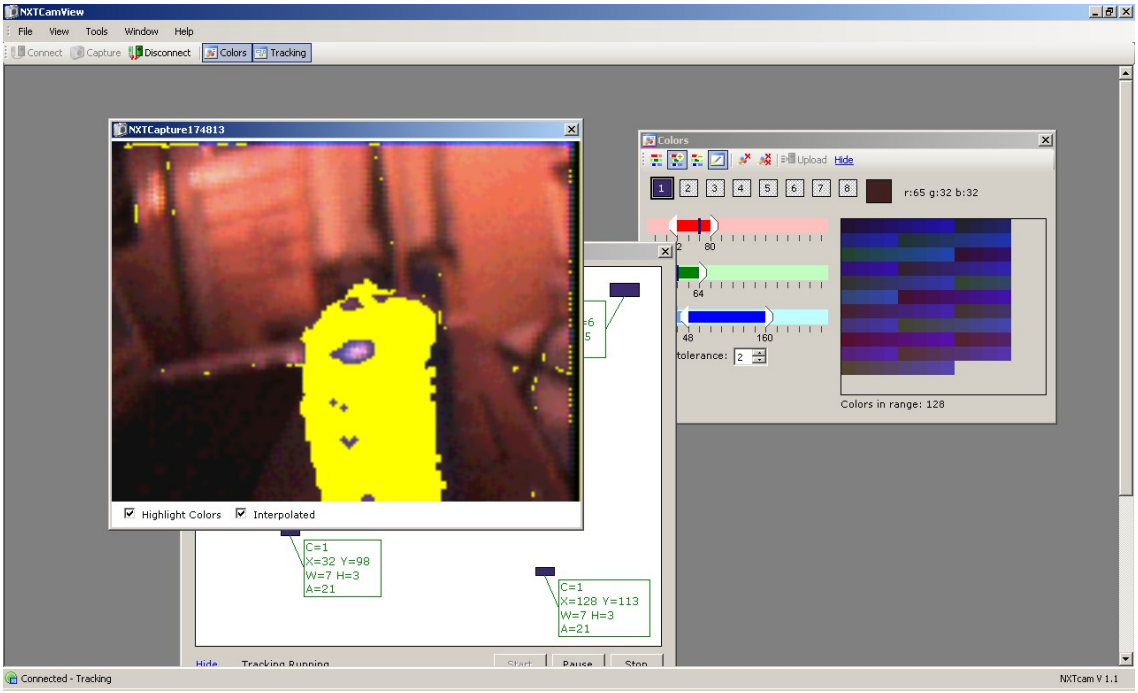
Once you have connected NXTCam, make an image capture:



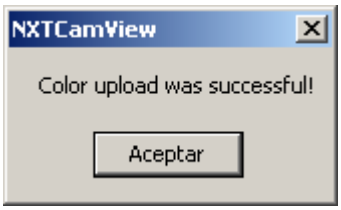
When you have an image that you have taken with NXTCam, create a color pattern using this tool:



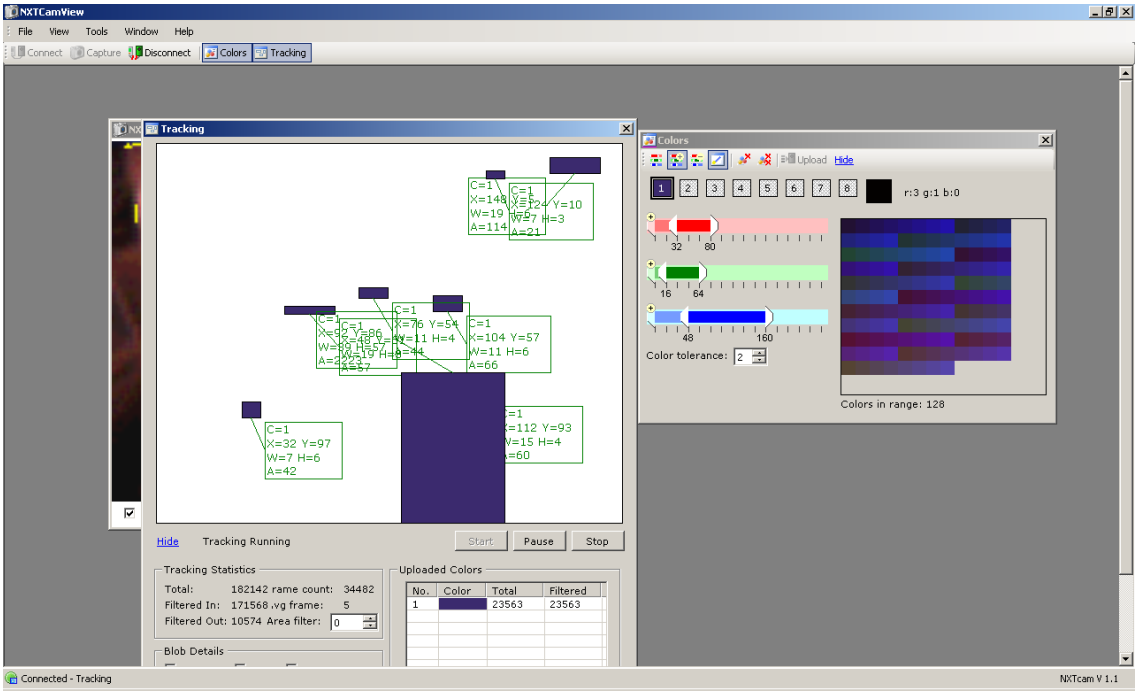
At the end of the process, you have a color pattern as the following:



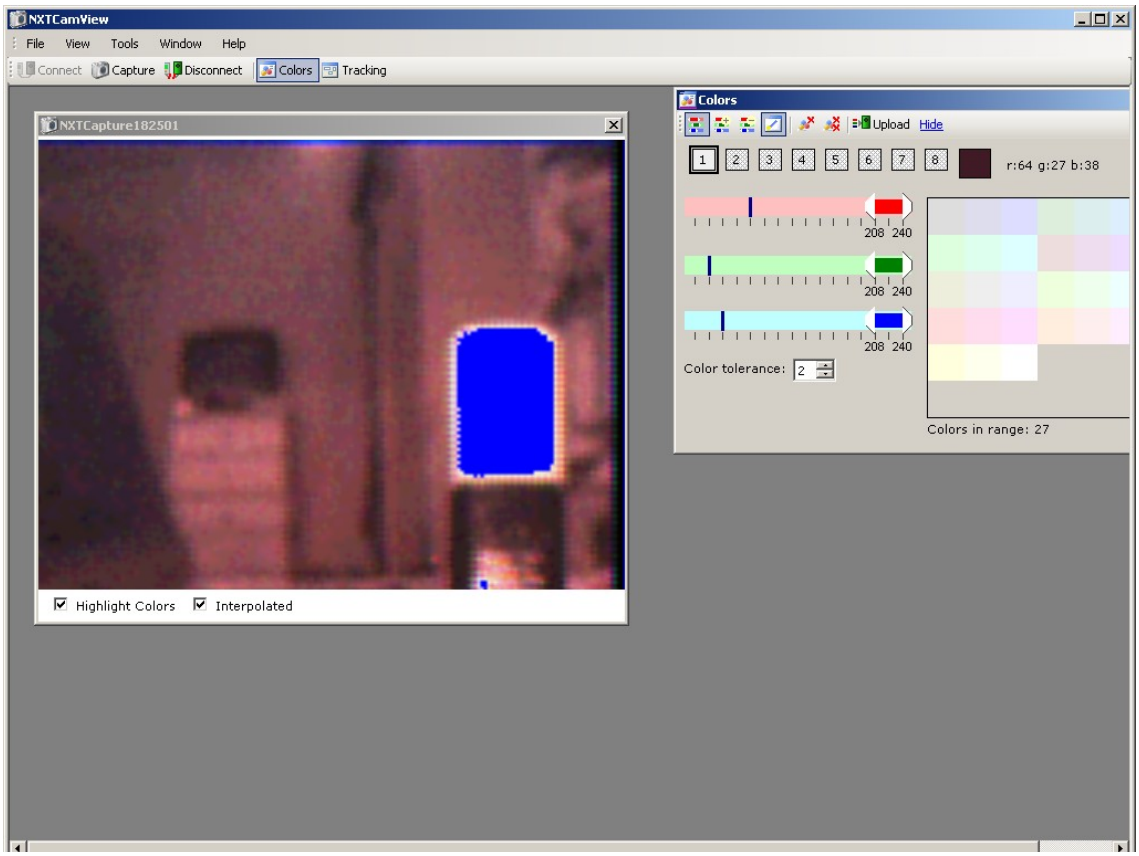
Then you have to upload the color pattern to the NXTCam using the button **update**. When the process goes well, you will see the following alert:



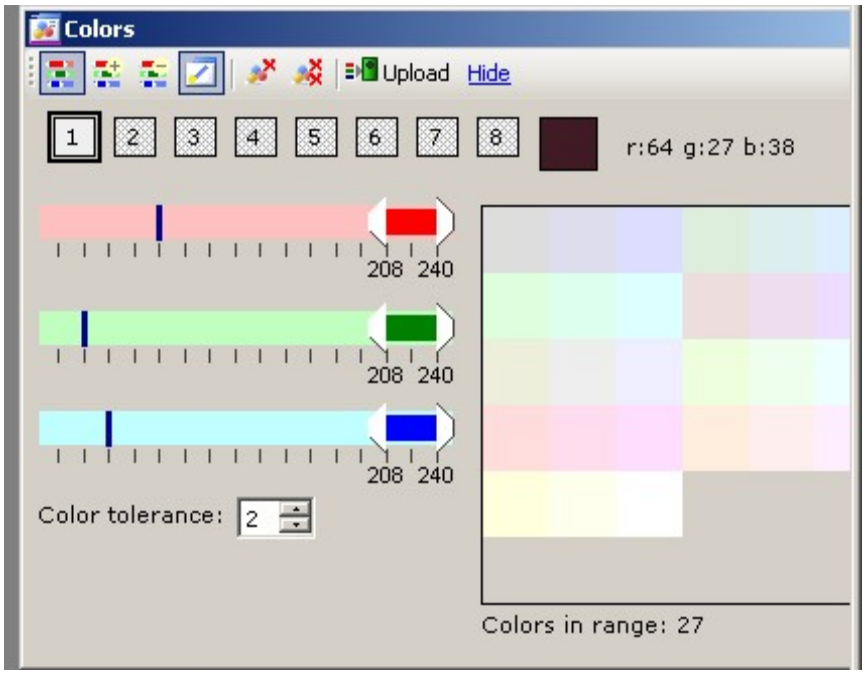
Test your color map pattern using the option tracking:



Once you have connected NXTCam, make an image capture:

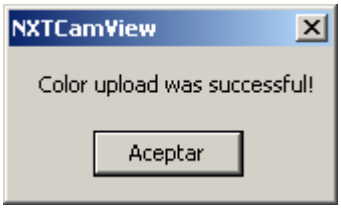


When you have an image that you have taken with NXTCam, create a color pattern using this tool:

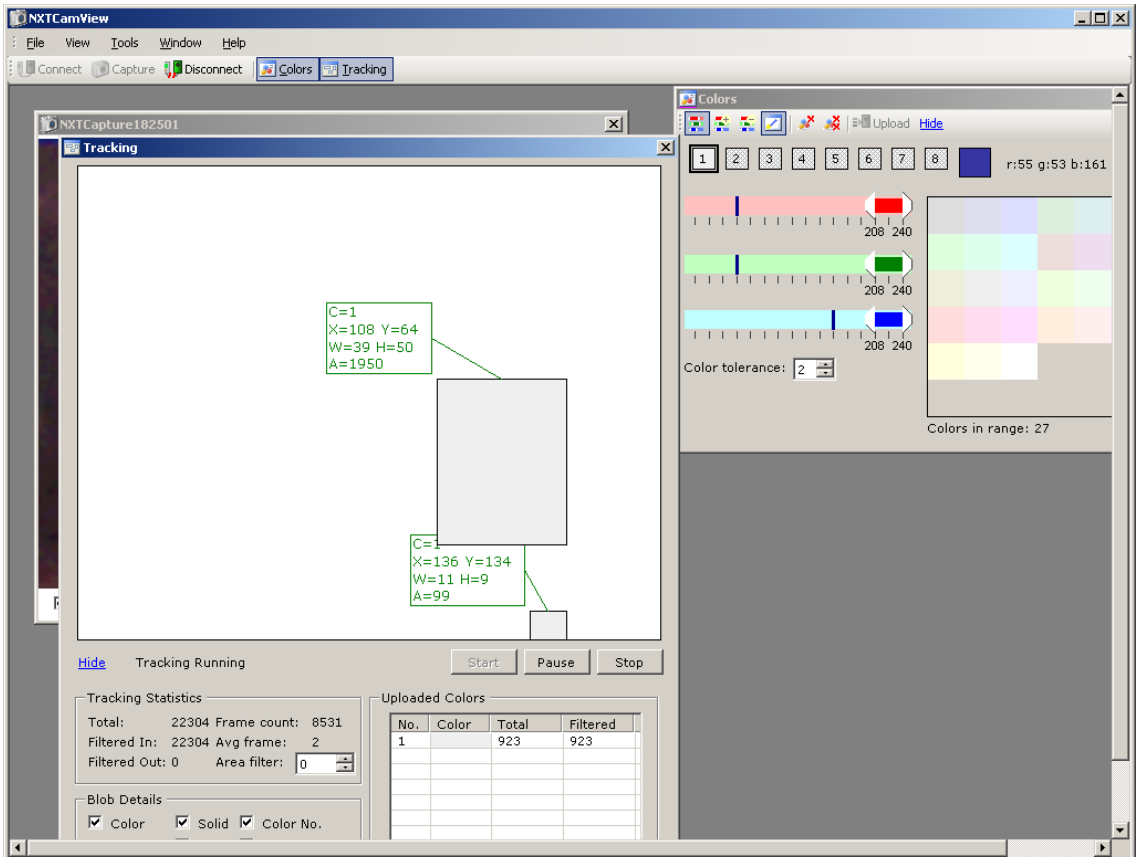


At the end of the process, you have a color.

Then you have to upload the color pattern to the NXTCam using the button **update**. When the process goes well, you will see the following alert:



Test your color map pattern using the option tracking:



Notes:

1. It is better if the object to track is different in relation to the environment
2. If you object is totally different, NXTCam will not detect false objects (Noise)

NXTCam API

NXJ allows managing NXTCam using the class NXTCam the current public methods are the following:

Method Summary	
int	**getNumberOfObjects**() Get the number of objects being tracked
int	**getObjectColor**(int id) Get the color number for a tracked object
Rectangle	**getRectangle**(int id) Get the rectangle containing a tracked object
void	**sendCommand**(char cmd) Send a single byte command represented by a letter

- **GetNumberOfObjects:** NXTCam is able to track a maximum of 8 objects in real time. This method returns the number of objects tracked by the sensor.
- **GetObjectColor:** When you track an object, it is possible to know the color detected in that object.
- **GetRectangle:** when you track an object, it is possible to return a rectangle Object to know the position and size.
- **SendCommand:** NXTCam allow doing several operations using I2C. Using this method you can interact with NXTCAM's I2C registers.

Configuration's Registers

Commands		Action
ASCII	Hex	
A	0x41	Sort tracked objects by size
B	0x42	Select Object tracking mode
C	0x43	Write to Camera Registers Use extreme CAUTION when using command C since this can stop your camera working properly. In case this happens, please power off your NXTCam and power it on again.
D	0x 44	Disable Tracking
E	0x45	Enable Tracking
G	0x47	Get the Color map from Camera Engine
H	0x48	Read data from the Camera Registers
I	0x49	Illumination on (Future)
L	0x 4C	Select Line tracking mode
N	0x4E	Set ADPA mode ON (setting stored in NVRAM)
O	0x4F	Set ADPA mode Off (default) (setting stored in NVRAM)
P	0x50	Ping camera Engine
R	0x52	Reset Camera Engine
S	0x53	Send the color map to camera Engine
T	0x54	Illumination Off
U	0x55	Sort tracked objects by color
V	0x56	Get camera Engine firmware version No
X	0x58	Do not Sort tracked objects

Tracking's Registers

Register	Read	Write	Comments
0x00-0x07	Software version - *(V1.10)*	-	
0x08-0x0f	Vendor Id - *mndsnsrs*	-	
0x10-0x17	Device ID - *NXTCam*	-	
0x41	-	Command	This register is command register. A command written here will be executed.
0x42	Number of objects detected	-	Shows how many objects are being tracked. Zero indicates that there are no objects being tracked.
0x43	1st object color	-	This is the first object color as per the sorting method selected.
0x44[1]	1st object - X upper left		Upper left X coordinate of first object
0x45	1st object - Y upper left		Upper left Y coordinate of first object
0x46	1st object - X lower right		Lower right X coordinate of first object
0x47[2]	1st object - Y lower right		Lower right Y coordinate of first object
0x48	2nd object color		
0x49-0x4C	2nd object co-ordinates		
0x4D	3rd object color		
0x4E-0x51	3rd object co-ordinates		
0x52	4th object color		
0x53-0x56	4th object co-ordinates		
0x57	5th object color		
0x58-0x5B	5th object co-ordinates		

Register	Read	Write	Comments
0x5C	6th object color		
0x5D-0x60	6th object co-ordinates		
0x61	7th object color		
0x62-0x65	7th object co-ordinates		
0x66	8th object color		
0x67-0x6A	8th object co-ordinates		
0x6B	No. of registers to Read	No. of registers to Write	This is the number of registers you need to read or write from camera image sensor
0x6C	1st Camera register Address	1st Camera register Address	
0x6D[3]	1st Camera register Data	1st Camera register Data	1st register Data read from image sensor or written to image sensor
.....	.....	.....	
0x7A	8th Camera register Address	8th Camera register Address	
0x7B	8th Camera register Data	8th Camera register Data	

Colormap's registers

0x80[4]	Color map data Red 0	Color map data Red 0	0x80 - 0xAF - These registers are used for Colormap data reading and writing

Register	Read	Write	Comments
0x80	Color map data Red 0	Color map data Red 0	
0x81	Color map data Red 1	Color map data Red 1	
0x82	Color map data Red 2	Color map data Red 2	
0x83	Color map data Red 3	Color map data Red 3	
0x84	Color map data Red 4	Color map data Red 4	
0x85	Color map data Red 5	Color map data Red 5	
0x86	Color map data Red 6	Color map data Red 6	
0x87	Color map data Red 7	Color map data Red 7	
.....	.....	.....	
0x8F	Color map data Red 15	Color map data Red 15	
0x90	Color map data Green 0	Color map data Green 0	
0x91	Color map data Green 1	Color map data Green 1	
.....	.....	.....	
0x9F	Color map data Green 15	Color map data Green 15	
0xA0	Color map data Blue 0	Color map data Blue 0	
0xA1	Color map data Blue 1	Color map data Blue 1	
.....	.....	.....	
0xAF	Color map data Blue 15	Color map data Blue 15	

Proprioceptors sensors in NXT

Introduction

Once you have discovered how to use NXT Sensors to get data from the world, it is necessary to discover other kind of sensors used to measure internal state of your own robot.

Memory

NXT brick is a small embebbed system with a low amount of memory to store programs and data so it is very interesting to monitor the amount of memory that you have to store information.

How to measure the memory in your system?

The way to get memory information about your NXT brick is:

```
System.getRuntime().freeMemory();
System.getRuntime().totalMemory();
```

Battery

How to measure battery life in your system?

The way to get battery information about your NXT brick is:

```
Battery.getVoltage();
Battery.getVoltageMilliVolt();
Battery.isRechargeable();
```

NXT Motor sensor features

NXT Motor has an internal encoder to know in what state is the motor.

Illustration 11: NXT Motor

How to get state of the motor?

```
import lejos.nxt.*;

public class InertiaTest
{
   public static void main(String[] args)
   {
        LCD.drawString("Inertia Test", 0, 0);
        Button.waitForPress();
        rotate720();
        Motor.A.resetTachoCount();
        Motor.A.setSpeed(800);
        rotate720();
   }
```

```
   public static void rotate720()
   {
        Motor.A.forward();
        int count = 0;
        while( count < 720 )count = Motor.A.getTachoCount();
        Motor.A.stop();
        LCD.drawInt(count , 0, 1);
        while(Motor.A.getActualSpeed()>0);
        LCD.drawInt(Motor.A.getTachoCount(), 7,1 );
        Button.waitForPress();
        LCD.clear();
   }
}
```

Summary

In this chapter you learn how to use many sensors which you have with your Lego Mindstorms NXT Kit or adquired with your favorite provider.

Exercises

Interesting Links

Sensors

http://en.wikipedia.org/wiki/Sensor

http://i.cmpnet.com/eetimes/news/09/09/1567chart_pg31.gif

http://www.seattlerobotics.org/encoder/jul97/basics.html

http://en.wikipedia.org/wiki/Autonomous_robot

http://en.wikipedia.org/wiki/Global_Positioning_System

LeJOS API

http://lejos.sourceforge.net/nxt/nxj/api/index.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/UltrasonicSensor.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/LightSensor.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/TouchSensor.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/SoundSensor.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/addon/CompassSensor.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/Battery.html

http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/NXTMotor.html