How to use Lego color sensor of Mindstorms NXT 2.0 in NXT++?



If you are advanced user of NXT++, and want to add code to your existing project just go to point (II).

(I) Quick start in Visual Studio 2010 for begginers

1) install microsoft visual studio 2010 with visual c++,
2) install drivers of NXT_32 or NXT_64 depending on your system,
3) if you wish to use bluetooth connection with NXT brick, install bluetooth drivers (in case of problem with bluetooth drivers, uninstall drivers provided by computer dealer and use standard windows drivers,
4) download library http://wmii.uwm.edu.pl/~artem/nxtpp0-6-2.zip (the library include color sensor support, the original one without color sensor support is available in http://sourceforge.net/projects/nxtpp/),
5) extract file nxtpp0-6-2.zip, and move the folder nxtpp0-6-2 directly to drive C:\ (because the solution is configured on this path, if you have different path just change links in your project,
6) open file: C:\nxtpp0-6-2\FINAL\examples\vs10\re4.sln
if you have different path to re4.sln file, open solution re4.sln from your path, and in re4 properties, particularly in:
- configuration properties/C/C++/general Additional Include Directories/
and
- configuration properties/Linker/General/Additional Library Direcories/
change the links on yours,
7) connect your NXT brick via cable or bluetooth and run the solution. In our example from file C:\nxtpp0-6-2\FINAL\examples\vs10\re4\re4\main.cpp we have fixed Solution Configurations as Release and Solution Platforms as Win32. And we made assumption that color sensor is connected to port IN_3, change if you use different port.

(II) Usage for advanced users:

Paste the following code into the file NXT++.h:

```
//! Sets a sensor in a specified port to a color sensor in the chosen mode.
/*! \param port The port that you wish to set as a color sensor. Can be the numbers 0-3 or
IN_1, IN_2, IN_3, or IN_4.*/
/*\param p is the parameter, which let you choose different mode of color sensor, there are
alternative modes: 'r' - set to read red color intensity, 'g' - set to read green color
intensity, 'b' - set to read blue color intensity and 'f' - set to read full color number,
where sensor can return numbers: 1=black, 2=blue, 3=green, 4=yellow, 5=red, 6=white*/;
void SetColor(Comm::NXTComm* comm, int port, char p);

//! Tells the NXT brick to disable color sensor
/*! \param port The port that contains the color sensor. Can be the numbers 0-3 or IN_1,
IN_2, IN_3, or IN_4.*/
void SetColorOff(Comm::NXTComm* comm, int port);
```

Paste the following code into the file nxt++.cpp:

```
void NXT::Sensor::SetColor(Comm::NXTComm* comm, int port, char p)
{
	if(p=='F' || p=='f')
	{
	 ViUInt8 directCommandBuffer[] = { 0x05, port, 0x0D, 0x00 };
	 comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
```

```
        }
        if(p=='R' || p=='r')
        {
         ViUInt8 directCommandBuffer[] = { 0x05, port, 0x0E, 0x00 };
         comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
        }
        if(p=='G' || p=='g')
        {
         ViUInt8 directCommandBuffer[] = { 0x05, port, 0x0F, 0x00 };
         comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
        }
        if(p=='B' || p=='b')
        {
         ViUInt8 directCommandBuffer[] = { 0x05, port, 0x10, 0x00 };
         comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
        }
        if(p=='N' || p=='n')
        {
         ViUInt8 directCommandBuffer[] = { 0x05, port, 0x11, 0x00 };
         comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
        }
        if(p=='E' || p=='e')
        {
         ViUInt8 directCommandBuffer[] = { 0x05, port, 0x12, 0x80 };
         comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
        }
        Wait(300);
}

void NXT::Sensor::SetColorOff(Comm::NXTComm* comm, int port)
{
         ViUInt8 directCommandBuffer[] = { 0x05, port, 0x12, 0x00 };
         comm->SendDirectCommand( false, reinterpret_cast< ViByte* >( directCommandBuffer ),
sizeof( directCommandBuffer ), NULL, 0);
}
```

If you use Visual Studio, open and rebuild solution NXT++.sln

And use nxt sensor in main.cpp of your project:

Exemplary usage:

```
/*the color sensor specified in port IN_3 with mode decColor*/
NXT::Sensor::SetColor(&comm, IN_3, decColor);

//parameter decColor could be:
//'r' - setting to read red color intensity
//'g' - setting to read green color intensity
//'b' - setting to read blue color intensity
//'n' - setting to read a light intensity
//'f' - setting to read full color number,

color=NXT::Sensor::GetValue(&comm, IN_3);
//variable color should be declared as int,
// depending on chosen mode in variable color we have: for 'r', 'g', 'b', color intensity,
// for mode 'f' one of the numers, 1,2,3,4,5,6, where 1=black, 2=blue,
// 3=green, 4=yellow, 5=red and 6=white

// Tells the NXT brick to disable color sensor
NXT::Sensor::SetColorOff(&comm, IN_3);
```

To see all functionalities of color sensor, you can put the following code to your solution main.cpp file:

```
//beggining of the file main.cpp
//we made assumption that LEGO color sensor is connected to port IN_3
//example prepared by artem, http://wmii.uwm.edu.pl/~artem
#include "NXT++.h"
#include <conio.h>

Comm::NXTComm comm;

int main()
{
 std::cout << "\naquiring signal ... ";
 int pointer=0;
 if(NXT::OpenBT(&comm))/*initialize the NXT via cable or bluetooth and continue if it
succeeds*/
  {
   pointer=1;
   std::cout<< "signal found";
   std::cout<<"\nbattery Level = "<<NXT::BatteryLevel(&comm)/100<<" percent";
   NXT::StartProgram(&comm,"ExemplaryProgram");
   //declaration of variables
   int color;
   char decColor;
   char temp_choice='o';
   do
       {//the beginning of loop
       std::cout<<"\n\n ----------------------------------------";
       std::cout<<"\n color sensor, commands:";
       std::cout<<"\n press r - to get a red color intensity";
       std::cout<<"\n press g - to get a green color intensity";
       std::cout<<"\n press b - to get a blue color intensity";
       std::cout<<"\n press n - to get a light intensity";
       std::cout<<"\n press f - to detect the full color, considering only:";
       std::cout<<"\n           black, blue, green, yellow, red and white)";
       std::cout<<"\n press q - to turn off the sensor\n";
    decColor=getch();
      /*The first initiation of sensor is delayed by 0.3s, the following condition let to
avoid delays in multiple reading of the same mode*/
       if(decColor!=temp_choice)
        {
         NXT::Sensor::SetColor(&comm, IN_3, decColor);
         temp_choice=decColor;
     }
     color=NXT::Sensor::GetValue(&comm, IN_3);
        if(decColor=='F' || decColor=='f' || decColor=='d')
          {
           std::cout<<"\n\nIt looks like ";
              if(color==1){std::cout<<"black\n";}
              if(color==2){std::cout<<"blue\n";}
              if(color==3){std::cout<<"green\n";}
              if(color==4){std::cout<<"yellow\n";}
              if(color==5){std::cout<<"red\n";}
              if(color==6){std::cout<<"white\n";}
      //NXT::Sensor::SetColor(&comm, IN_3, 'n');
          }
        if(decColor=='R' || decColor=='r'){std::cout<<"\nRed light value = "<<color;}
        if(decColor=='G' || decColor=='g'){std::cout<<"\nGreen light value = "<<color;}
        if(decColor=='B' || decColor=='b'){std::cout<<"\nBlue light value = "<<color;}
        if(decColor=='N' || decColor=='n'){std::cout<<"\nLight intensity = "<<color;}
        if(decColor=='Q' || decColor=='q'){NXT::Sensor::SetColorOff(&comm, IN_3);}
```

```
        }//the end of loop
     while(decColor!='q' && decColor!='Q');
        std::cout<<"\n\nprogram is shutting down";
        NXT::StopProgram(&comm);
   }
        NXT::Close(&comm); //close the NXT
        if(pointer==0){std::cout<<"\n\nconnection with NXT brick failed";Wait(3000);}
        return 0;
}

//the end of the file main.cpp

//the color sensor support for NXT++ deveoped by artem http://wmii.uwm.edu.pl/~artem
//the original library without our changes is available in:
//http://sourceforge.net/projects/nxtpp/
```