Ficha de Catalogación

Datos Generales

Nombre del Proyecto:	NXT PC Remot Control
Título del Software:	NXT PC Remot Control
Autores:	Prof. Gonzalo José Hernández Prof. Jesús Insuasty Prof. Jairo Antonio Guerrero
Categoría del Software:	Aplicación para sistemas linux

Tecnología de Despliegue

Sistema Operativo:	Linux
Motor de Base de Datos:	Archivos simples de configuración
Dependencias.	Bluetooth (hardware & software)

Tecnología usada para el desarrollo

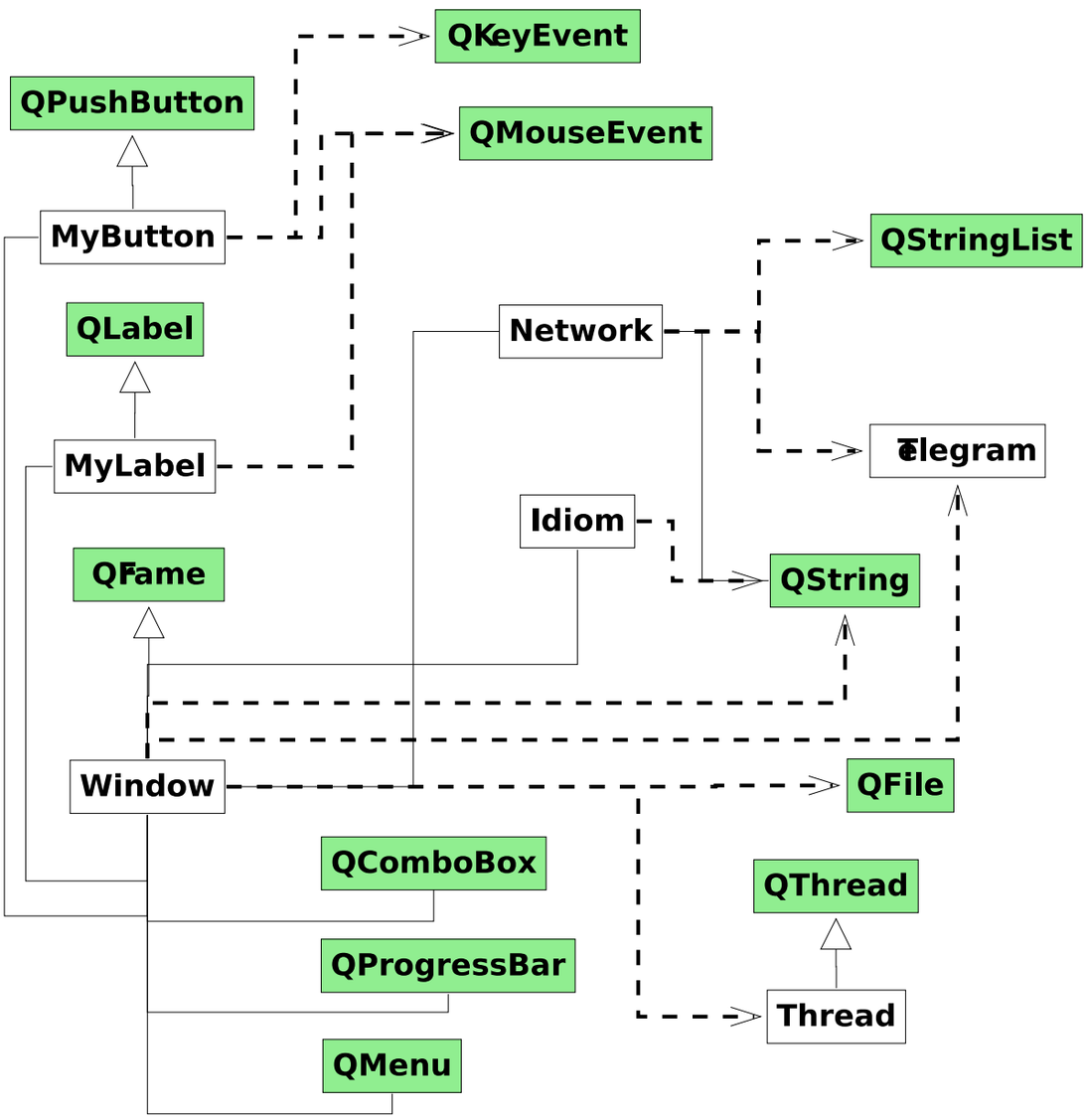
Sitema de Desarrollo:	QtCreator
Lenguaje de Programación:	C++

Documentación adjunta:	User Guide – Booklet System Manual – Booklet User Guide – Video Installer(Archlinux) – Video Installer(Debian distros) – Video
Robustez:	Las pruebas de efectividad pueden ser verificadas en el video *User Guide*
Extensibilidad:	Desarrollado como proyecto *Open Source* permitiendo libre acceso al código.
Desempeño:	El desempeño puede ser verificado en el video *User Guide*.
Usabilidad:	NXT PC Remote Control es requerido para ejecutar su funcionalidad en sistemas Linux.
Integridad:	El desarrollo modular, documentado y trabajado desde un repositorio de código (*googlecode*), verifica su *Integridad*.
Portabilidad:	NXT PC Remote Control puede ser utilizado en cualquier distribución de Linux. Se han creado instaladores para Archlinux, para Debian distros, y también la documentación para ser compilado otras distribuciones.

Mantenimiento:	Este proyecto esta alojado en *googlecode*. Seguirá siendo mantenido por su autor. A la fecha esta disponible el *Release 0.34.*

Modelado General

NXT PC Remote Control esta diseñado como se muestra en el siguiente diagrama de clases. En siguientes hojas se presenta el código fuente de todo el proyecto, (Versión hasta la fecha 0.r34)



idiom.h

```
#ifndef IDIOM_H
#define IDIOM_H

#include <QString>

enum idiomtype{
  ENG=0,SPA=1
};

/** ====================================================================
 * @brief The Idiom class is used to enable to "NXT PC Remote Contro" to
 * change systematically between two idioms: English and Spanish
 */
class Idiom {
public:
  idiomtype it;
  QString windowTitle[2];
  QString scanButtonLabel[2];
  QString connectButtonLabel[2];
  QString disconnectButtonLabel[2];
  QString menuRecentConnections[2];
  QString menuClearConnections[2];
  QString menuSelectIdiom[2];
  QString menuEnglish[2];
  QString menuSpanish[2];
  QString menuAbout[2];
  QString messageSearching[2];
  QString messageBluetoothDisabled[2];
  QString messageNearDivices[2];
  QString messageDeviceAvailable[2];
  QString imageInfo[2];
public:

  /** --------------------------------------------------------------------
   * @brief Idiom constructor launch the attributes values.  This class, has
   * all text needed to show in application interface.
   */
  Idiom() : it(ENG) {
    windowTitle[ENG] = "NXT PC Remote Control";
    windowTitle[SPA] = "Control Remoto de PC para NXT";

    scanButtonLabel[ENG] = "Scan";
    scanButtonLabel[SPA] = "Buscar";

    connectButtonLabel[ENG] = "Connect";
    connectButtonLabel[SPA] = "Conectar";

    disconnectButtonLabel[ENG] = "Disconnect";
    disconnectButtonLabel[SPA] = "Desconectar";

    menuRecentConnections[ENG] = "Recent connections";
    menuRecentConnections[SPA] = "Conexiones Recientes";

    menuClearConnections[ENG] = "Clean connections";
    menuClearConnections[SPA] = "Limpiar conexiones";

    menuSelectIdiom[ENG] = "Switch language";
    menuSelectIdiom[SPA] = "Cambiar idioma";

    menuEnglish[ENG] = "English";
    menuEnglish[SPA] = "Ingles";

    menuSpanish[ENG] = "Spanish";
    menuSpanish[SPA] = "Espanol";
```

```
    menuAbout[ENG] = "About (Ver.0-34)";
    menuAbout[SPA] = "Acerca de (Ver.0-34)";

    messageSearching[ENG] = "Searching to devices...";
    messageSearching[SPA] = "Buscando Dispositivos...";

    messageBluetoothDisabled[ENG] = "Bluetooth disabled";
    messageBluetoothDisabled[SPA] = "Bluetooth deshabilitado";

    messageNearDivices[ENG] = "There isn't near devices";
    messageNearDivices[SPA] = "No hay dispositivos cercanos";

    messageDeviceAvailable[ENG] = "Device isn't available";
    messageDeviceAvailable[SPA] = "El dispositivo ya no esta disponible";

    imageInfo[ENG] = ":/images/info-eng.png";
    imageInfo[SPA] = ":/images/info-spa.png";
  }

  /** ------------------------------------------------------------------------
   * @brief getText method return the current set idiom
   * @return first three letters of idom
   */
  QString getText() {
    switch (it) {
    case ENG: return "eng";
    case SPA: return "spa";
    }
  }

  /** ------------------------------------------------------------------------
   * @brief setIdiomType method, put a new selected idiom
   * @param new idiom (of idiomtype)
   */
  void setIdiomType(idiomtype newit)  {
    it = newit;
  }

  /** ------------------------------------------------------------------------
   * @brief the next methods return all attributes values, namely,
   * this methods return de messages in current idiom.
   */

  idiomtype getIdiomType()              { return it; }

  QString getWindowTitle()              { return windowTitle[it]; }
  QString getScanButtonLabel()          { return scanButtonLabel[it]; }
  QString getConnectButtonLabel()       { return connectButtonLabel[it]; }
  QString getDisconnectButtonLabel()    { return disconnectButtonLabel[it]; }
  QString getMenuRecentConnections()    { return menuRecentConnections[it]; }
  QString getMenuClearConnections()     { return menuClearConnections[it]; }
  QString getMenuSelectIdiom()          { return menuSelectIdiom[it]; }
  QString getMenuEnglish()              { return menuEnglish[it]; }
  QString getMenuSpanish()              { return menuSpanish[it]; }
  QString getMenuAbout()                { return menuAbout[it]; }
  QString getMessageSearching()         { return messageSearching[it]; }
  QString getMessageBluetoothDisabled() { return messageBluetoothDisabled[it]; }
  QString getMessageNearDivices()       { return messageNearDivices[it]; }
  QString getMessageDeviceAvailable()   { return messageDeviceAvailable[it]; }
  QString getImageInfo()                { return imageInfo[it]; }
};

#endif // IDIOM_H
```

network.h

```
#ifndef NETWORK_H
#define NETWORK_H

typedef unsigned char byte;

#include <QStringList>
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>
#include <unistd.h>
#include <bluetooth/rfcomm.h>
#include <iostream>

/** ===========================================================================
 * @brief The Telegram class transport all information between Window class
 * and Network class.
 */
class Telegram {
private:
  byte* content;
  int   size;
public:

  /** -------------------------------------------------------------------------
   * @brief Telegram constructor launch its attributes and set de three first
   * bytes of telegram.  These bytes are mecanically
   */
  Telegram() : content(NULL) , size(3) {
    content = new byte[size];
    content[0] = size-2;
    content[1] = 0x00;
    content[2] = 0x80; // Direct Command whitout response
  }

  /** -------------------------------------------------------------------------
   * @brief Telegram constructor to copy process events.
   */
  Telegram(Telegram& source) {
    size = source.size;
    content = new byte[size];
    for (int i=0;i<size;i++) content[i] = source.content[i];
  }

  /** -------------------------------------------------------------------------
   * @brief append method, add new bytes to end of telegram.
   */
  void append(byte piece) {
    byte* aux = new byte[size+1];
    for (int i=0;i<size;i++) aux[i] = content[i];
    aux[size] = piece;
    delete content;
    content = aux;
    size++;
    content[0] = size-2;
  }

  /** -------------------------------------------------------------------------
   * @brief append method with more than one bytes... add to end of telegram
   * all bytes sended in "pieces" array.
   */
  void append(byte* pieces, int count) {
    for (int i=0; i<count; i++) {
      append(pieces[i]);
    }
  }
```

```
  /** ------------------------------------------------------------------------
   * @brief Telegram destructor is important to keep clear to computer
   * memory
   */
  ~Telegram() {
    delete content;
  }

  /** ------------------------------------------------------------------------
   * @brief send method put in socket communications the telegram.
   */
  bool send(int sock) {
    return write(sock, content, size);
  }
};

/** ========================================================================
 * @brief The Network class work as low level, allow send and recive
 * information of Bluetooth device connected.
 */
class Network {
private:
  QString macAddress;
  int     sock;
public:

  /** ------------------------------------------------------------------------
   * @brief scanDevices method... search bluetooth devices around of computer.
   * This method will inactive the application during 10 seconds approximately.
   */
  QStringList scanDevices() {
    QStringList devices;

    inquiry_info *ii = NULL;
    int max_rsp, num_rsp;
    int dev_id, sock, len, flags;
    int i;
    char addr[19] = { 0 };
    char name[248] = { 0 };

    dev_id = hci_get_route(NULL);
    sock = hci_open_dev( dev_id );
    if (dev_id < 0 || sock < 0) {
        perror("opening socket");
        throw(1);
    }

    len  = 8;
    max_rsp = 255;
    flags = IREQ_CACHE_FLUSH;
    ii = (inquiry_info*)malloc(max_rsp * sizeof(inquiry_info));

    num_rsp = hci_inquiry(dev_id, len, max_rsp, NULL, &ii, flags);
    if( num_rsp <= 0 ) {
      throw(2);
    }

    for (i = 0; i < num_rsp; i++) {
        ba2str(&(ii+i)->bdaddr, addr);
        memset(name, 0, sizeof(name));
        if (hci_read_remote_name(sock, &(ii+i)->bdaddr, sizeof(name), name, 0)
            < 0) {
          strcpy(name, "unknown");
        }
        char text[30];
        sprintf(text,"%s  [%s]",addr,name);
        devices.append(text);
        //printf("%s  %s\n", addr, name);
    }

    free( ii );
    close( sock );
```

```
    return devices;
  }

  /** ----------------------------------------------------------------------
   * @brief bind methoc... connect the applications with wanted device.
   */
  bool bind(QString address) {
    macAddress = address;
    struct sockaddr_rc addr; //  = { 0 };
    sock = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
    addr.rc_family = AF_BLUETOOTH;
    addr.rc_channel = (uint8_t) 1;
    str2ba( macAddress.toStdString().c_str(), &addr.rc_bdaddr );
    if ( connect(sock, (struct sockaddr *)&addr, sizeof(addr)) == 0) {
      return true;
    }
    else {
      return false;
    }
  }

  /** ----------------------------------------------------------------------
   * @brief unbind method... disconnect the applications
   */
  void unbind() {
    close(sock);
  }

  /** ----------------------------------------------------------------------
   * @brief directCommand... it's disposed to be a middle layer between
   * GUI interface and low layer "blueZ"
   */
  bool directCommand(Telegram t) {
    t.send(sock);
    return true;
  }

  /** ----------------------------------------------------------------------
   * @brief directCommand with bytes array... it's disposed to be a middle
   * layer between GUI interface and low layer "blueZ" sended a lot of bytes
   */  bool directCommand(byte* pieces, int count) {
    Telegram t;
    t.append(pieces, count);
    t.send(sock);
    return true;
  }

};

#endif // NETWORK_H
```

window.h

```
#ifndef WINDOW_H
#define WINDOW_H

#include <QFrame>
#include <QPushButton>
#include <QLabel>
#include <QComboBox>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QKeyEvent>
#include <QProgressBar>
#include <QMenu>
#include <QFile>
#include <QThread>

#include <network.h>
#include <idiom.h>

#define len(x) sizeof(x)/sizeof(byte)
#define non(x) (byte)-(x)

/** =====================================================================
 * @brief MyButton class overload to QPushButton due to, was necesary do
 * programming with especial events.
 */
class MyButton : public QPushButton {
  Q_OBJECT
public:

  /** -------------------------------------------------------------------
   * @brief MyButton constructor transport label to superclass
   */
  MyButton(QString label) : QPushButton(label) {
  }
signals:

  /** -------------------------------------------------------------------
   * @brief This is an abstract method to rightClick() programming event
   */
  void rightClick(QPoint);

protected:

  /** -------------------------------------------------------------------
   * @brief keyPressEvent create a keyboard press event
   */
  void keyPressEvent(QKeyEvent *ev) {
    ev->ignore();
  }

  /** -------------------------------------------------------------------
   * @brief mousePressEvent create a mouse click event
   */
  void mousePressEvent(QMouseEvent *ev) {
    if (ev->button() == Qt::RightButton) {
      emit rightClick(ev->pos());
    }
    else if (ev->button() == Qt::LeftButton) {
      emit clicked();
    }
  }
};

/** =====================================================================
 * @brief MyLabel class overload to QLabel due to, was necesary do
 * programming with especial events.
```

```
 */
class MyLabel : public QLabel {
  Q_OBJECT
signals:

  /** ----------------------------------------------------------------------
   * @brief This is an abstract method to rightClick() programming event
   */
  void rightClick(QPoint);

  /** ----------------------------------------------------------------------
   * @brief This is an abstract method to clicked() programming event
   */
  void clicked(bool);

protected:

  /** ----------------------------------------------------------------------
   * @brief mousePressEvent create a mouse click event
   */
  void mousePressEvent(QMouseEvent *ev) {
    if (ev->button() == Qt::RightButton) {
      emit rightClick(ev->pos());
    }
    else {
      emit clicked(false);
    }
  }
};

/** ========================================================================
 * @brief Thread class is implemented for fix disable GUI programmed in
 * another release of this applications.
 */
class Thread : public QThread {
  Q_OBJECT
private:
  QComboBox* devices;
  Network* net;
  int operation;

signals:

  /** ----------------------------------------------------------------------
   * @brief Creating events to after running.
   */
  void scanPerformed(int);
  void connectPerformed(bool);

public:
  /** ----------------------------------------------------------------------
   * @brief Thread constructor receive device combo and network references
   * to allow work with them.
   */
  Thread(QComboBox* combo,Network* n,int op)
    : devices(combo), net(n), operation(op)  {
  }

  /** ----------------------------------------------------------------------
   * @brief run method has actions to exec in thread.
   */
  void run() {
    switch (operation) {
    case 1:
      try {
        QStringList s = net->scanDevices();
        devices->clear();
        devices->addItems(s);
        emit scanPerformed(0);
      }
      catch(int e) {
        devices->clear();
```

```
        emit scanPerformed(e);
      }
      break;
    case 2:
      bool state = net->bind(devices->currentText().left(17));
      emit connectPerformed(state);
      break;
    }
  }
};

/** ============================================================================
 * @brief Window class is the main class en this project, because has the
 * GUI functions to intercommunicate two actors.  NXT PC Remote Control and
 * LEGO gameer.
 */
class Window : public QFrame {
  Q_OBJECT
private:
  byte          power;
  bool          lowswitch;
  byte          powerlow;
  MyButton      *scan,*bind;
  QComboBox     *devices;
  MyLabel       *info;
  Network       *net;
  QProgressBar  *lowspeed,*highspeed;
  QMenu         *menu;
  QMenu         *recents,*selectidiom;
  Idiom         idiom;
  Thread        *t;

  /** ------------------------------------------------------------------------
   * @brief loadSettings method set de initial profiles to NXT PC Remote
   * Control using source file ".nxt-pc-remote-control.cfg"
   */
  void loadSettings() {
    QFile f(".nxt-pc-remote-control.cfg");
    f.open(QIODevice::ReadOnly);
    if (!f.isOpen()) {
       idiom.setIdiomType(ENG);
       return;
    }
    QString data = f.readLine().data();
    idiom.setIdiomType(data=="eng\n"?ENG:SPA);
    QString data2 = f.readLine();
    int recentsCount = data2.toInt();
    for (int i=0; i<recentsCount; i++) {
      data = f.readLine().data();
      recents->addAction(data);
    }
    refreshIdiom();
    f.close();
  }

  /** ------------------------------------------------------------------------
   * @brief addRecent method admin de cache of connections worked
   */
  void addRecent(QString data) {
    data+="\n";
    foreach (QAction* action, recents->actions()) {
      if (action->text() == data)return;
    }
    recents->addAction(data);
  }

  /** ------------------------------------------------------------------------
   * @brief saveSettings save current application settings into a file
   */
  void saveSettings() {
    QFile f(".nxt-pc-remote-control.cfg");
    f.open(QIODevice::WriteOnly);
    if (!f.isOpen()) return;
```

```
    f.write( idiom.getIdiomType()==ENG?"eng\n":"spa\n" );
    char size[4];
    sprintf( size,"%d\n", recents->actions().length() );
    f.write( size );
    foreach (QAction* action, recents->actions()) {
      f.write( action->text().toStdString().c_str() );
    }
    f.close();
  }

  /** ------------------------------------------------------------------------
   * @brief refreshIdiom method update idiom of application.
   */
  void refreshIdiom() {
    setWindowTitle(idiom.getWindowTitle());
    scan->setText(idiom.getScanButtonLabel());
    scan->isEnabled() ? bind->setText(idiom.getConnectButtonLabel()) :
                        bind->setText(idiom.getDisconnectButtonLabel());
    info->setPixmap(QPixmap(idiom.getImageInfo()));
    selectidiom->actions().at(0)->setText(idiom.getMenuEnglish());
    selectidiom->actions().at(1)->setText(idiom.getMenuSpanish());
    menu->actions().at(0)->setText(idiom.getMenuRecentConnections());
    menu->actions().at(1)->setText(idiom.getMenuClearConnections());
    menu->actions().at(3)->setText(idiom.getMenuSelectIdiom());
    menu->actions().at(5)->setText(idiom.getMenuAbout());
  }

public:

  /** ------------------------------------------------------------------------
   * @brief Window constructor launch application saving information in its
   * attribute, additionally, update GUI presentation.
   */
  Window(): power(0x55), lowswitch(false), powerlow(0x3E) {
    setWindowTitle(idiom.getWindowTitle());
    resize(250,100);

    scan      = new MyButton(idiom.getScanButtonLabel());
    devices   = new QComboBox();
    bind      = new MyButton(idiom.getConnectButtonLabel());
    info      = new MyLabel();
    lowspeed  = new QProgressBar();
    highspeed = new QProgressBar();
    menu      = new QMenu();

    QHBoxLayout* buttons = new QHBoxLayout();
    QVBoxLayout* layout = new QVBoxLayout();

    setLayout(layout);
    buttons->addWidget(scan);
    buttons->addWidget(bind);
    layout->addLayout(buttons);
    layout->addWidget(devices);
    layout->addWidget(info);
    layout->addWidget(highspeed);
    layout->addWidget(lowspeed);

    devices->setEnabled(false);
    bind->setEnabled(false);
    info->setPixmap(QPixmap(idiom.getImageInfo()));
    setStyleSheet("QFrame{background-color:white}");
    lowspeed->setMinimum(0x32);
    lowspeed->setMaximum(0x64);
    highspeed->setMinimum(0x32);
    highspeed->setMaximum(0x64);
    lowspeed->setValue(powerlow);
    highspeed->setValue(power);

    setFixedSize(278,438);
    recents = new QMenu(idiom.getMenuRecentConnections());
    selectidiom = new QMenu(idiom.getMenuSelectIdiom());
    menu->addMenu(recents);
```

```
    menu->addAction(idiom.getMenuClearConnections());
    menu->addSeparator();
    menu->addMenu(selectidiom);
    menu->addSeparator();
    menu->addAction(idiom.getMenuAbout());
    selectidiom->addAction(idiom.getMenuEnglish());
    selectidiom->addAction(idiom.getMenuSpanish());

    loadSettings();
    net = new Network();

    connect(scan,SIGNAL(clicked()),this,SLOT(scanDevices()));
    connect(bind,SIGNAL(clicked()),this,SLOT(connectDevice()));
    connect(info,SIGNAL(rightClick(QPoint)),this,SLOT(popMenu(QPoint)));
    connect(menu,SIGNAL(triggered(QAction*)),this,SLOT(menuOption(QAction*)));

connect(recents,SIGNAL(triggered(QAction*)),this,SLOT(recentSelection(QAction*)));

connect(selectidiom,SIGNAL(triggered(QAction*)),this,SLOT(changeIdiom(QAction*)));
    connect(info,SIGNAL(clicked(bool)),this,SLOT(showAbout(bool)));
  }

  /** ----------------------------------------------------------------------
   * @brief Window destructor to exec saving settings and clear memory
   */
  ~Window() {
    saveSettings();
    delete net;
  }

protected:

  /** ----------------------------------------------------------------------
   * @brief keyPressEvent exec commands when user is play with NXT PC
   * Remote Control.
   */
  void keyPressEvent(QKeyEvent *event) {
    if (bind->text() == idiom.connectButtonLabel) return;
    if (!event->isAutoRepeat()) {
      switch (event->key()) {

        case Qt::Key_Alt: {
          lowswitch = true;
          break;
        }

        case Qt::Key_B: {
          byte bytes[] = { 0x03, 0x0B, 0x02, 0xF4, 0x01 };
          Telegram t;
          t.append(bytes, len(bytes));
          net->directCommand(t);
          break;
        }

        case Qt::Key_Up : {
          byte bytes3[] = { 0x04, 0x01, lowswitch?powerlow:power,
                            0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
          net->directCommand(bytes3, len(bytes3));

          byte bytes4[] = { 0x04, 0x02, lowswitch?powerlow:power,
                            0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
          net->directCommand(bytes4, len(bytes4));
          break;
        }

        case Qt::Key_Down : {
          byte bytes3[] = { 0x04, 0x01, lowswitch?non(powerlow):non(power),
                            0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
          net->directCommand(bytes3, len(bytes3));

          byte bytes4[] = { 0x04, 0x02, lowswitch?non(powerlow):non(power),
```

```
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes4, len(bytes4));
      break;
    }

    case Qt::Key_Left : {
      byte bytes3[] = { 0x04, 0x01, lowswitch?non(powerlow):non(power),
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes3, len(bytes3));

      byte bytes4[] = { 0x04, 0x02, lowswitch?powerlow:power,
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes4, len(bytes4));
      break;
    }

    case Qt::Key_Right : {
      byte bytes3[] = { 0x04, 0x01, lowswitch?powerlow:power,
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes3, len(bytes3));

      byte bytes4[] = { 0x04, 0x02, lowswitch?non(powerlow):non(power),
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes4, len(bytes4));
      break;
    }

    case Qt::Key_N : {
      byte bytes3[] = { 0x04, 0x00, lowswitch?powerlow:power,
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes3, len(bytes3));
      break;
    }

    case Qt::Key_M : {
      byte bytes3[] = { 0x04, 0x00, lowswitch?non(powerlow):non(power),
                        0x01, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00 };
      net->directCommand(bytes3, len(bytes3));
      break;
    }

    case Qt::Key_Minus : {
      if (lowswitch) {
        if (powerlow>0x32) powerlow--;
        lowspeed->setValue(powerlow);
      }
      else {
        if (power>0x32) power--;
        highspeed->setValue(power);
      }
      break;
    }

    case Qt::Key_Plus : {
      if (lowswitch) {
        if (powerlow<0x64) powerlow++;
        lowspeed->setValue(powerlow);
      }
      else {
        if (power<0x64) power++;
        highspeed->setValue(power);
      }
      break;
    }

  }
}
else {
  switch (event->key()) {
    case Qt::Key_Minus : {
      if (lowswitch) {
        if (powerlow>0x32) powerlow--;
```

```
            lowspeed->setValue(powerlow);
          }
          else
          {
            if (power>0x32) power--;
            highspeed->setValue(power);
          }
          break;
        }

        case Qt::Key_Plus : {
          if (lowswitch) {
            if (powerlow<0x64) powerlow++;
            lowspeed->setValue(powerlow);
          }
          else {
            if (power<0x64) power++;
            highspeed->setValue(power);
          }
          break;
        }
      }
    }
  }

  /** -------------------------------------------------------------------------
   * @brief keyReleaseEvent leave commands when user is play with NXT PC
   * Remote Control.
   */
  void keyReleaseEvent(QKeyEvent *event) {
    if (bind->text() == idiom.getConnectButtonLabel()) return;
    if (!event->isAutoRepeat()) {
      switch (event->key()) {
        case Qt::Key_Up:
        case Qt::Key_Down:
        case Qt::Key_Left:
        case Qt::Key_Right:
        case Qt::Key_N:
        case Qt::Key_M: {
          byte bytes0[] = { 0x04, 0x00, power, 0x02, 0x01, 0x00, 0x20, 0x00,
                            0x00, 0x00, 0x00 };
          net->directCommand(bytes0, len(bytes0));

          byte bytes1[] = { 0x04, 0x01, power, 0x02, 0x01, 0x00, 0x20, 0x00,
                            0x00, 0x00, 0x00 };
          net->directCommand(bytes1, len(bytes1));

          byte bytes2[] = { 0x04, 0x02, power, 0x02, 0x01, 0x00, 0x20, 0x00,
                            0x00, 0x00, 0x00 };
          net->directCommand(bytes2, len(bytes2));
          break;
        }

        case Qt::Key_Alt: {
          lowswitch = false;
          break;
        }

      }
    }
  }

public slots:

  /** -------------------------------------------------------------------------
   * @brief scanDevices method search all devices with bluetooth
   * functionality.
   */
  void scanDevices() {
    devices->clear();
    devices->setEnabled(false);
    bind->setEnabled(false);
```

```
  scan->setEnabled(false);
  devices->addItem(idiom.getMessageSearching());
  info->setPixmap(QPixmap(":/images/clock.png"));
  info->setEnabled(false);

  t = new Thread(devices,net,1);
  connect(t,SIGNAL(scanPerformed(int)),this,SLOT(scanPerformed(int)));
  t->start();
}

/** ------------------------------------------------------------------------
 * @brief scanPerformed is run after net scan->
 */
void scanPerformed(int throwstate) {
  info->setPixmap(QPixmap(idiom.getImageInfo()));
  info->setEnabled(true);
  devices->setEnabled(true);
  scan->setEnabled(true);
  info->setPixmap(QPixmap(idiom.getImageInfo()));
  if (throwstate==0) {
    bind->setEnabled(true);
  }
  else {
    bind->setEnabled(false);
    switch(throwstate) {
      case 1: {
        devices->addItem(idiom.getMessageBluetoothDisabled());
        break;
      }
      case 2: {
        devices->addItem(idiom.getMessageNearDivices());
        break;
      }
    }
  }
  delete t;
}

/** ------------------------------------------------------------------------
 * @brief connectDevice bind NXT PC Remote Control with a wanted device
 */
void connectDevice() {
  if (bind->text() == idiom.getConnectButtonLabel()) {
    info->setPixmap(QPixmap(":/images/clock.png"));
    info->setEnabled(false);
    bind->setEnabled(false);
    scan->setEnabled(false);
    devices->setEnabled(false);
    t = new Thread(devices,net,2);
    connect(t,SIGNAL(connectPerformed(bool)),this,SLOT(connectPerformed(bool)));
    t->start();
  }
  else {
    net->unbind();
    scan->setEnabled(true);
    devices->setEnabled(true);
    bind->setText(idiom.getConnectButtonLabel());
    recents->setEnabled(true);
    for (int i=0; i<4;i++) menu->actions().at(i)->setEnabled(true);
  }
}

/** ------------------------------------------------------------------------
 * @brief connectPerfomred is run after net bind function.
 */
void connectPerformed(bool ok) {
  info->setPixmap(QPixmap(idiom.getImageInfo()));
  info->setEnabled(true);
  if (ok) {
    bind->setText(idiom.getDisconnectButtonLabel());
    addRecent(devices->currentText());
    for (int i=0; i<4;i++) menu->actions().at(i)->setEnabled(false);
  }
```

```
  else {
    scan->setEnabled(true);
    devices->clear();
    devices->addItem(idiom.getMessageDeviceAvailable());
    devices->setEnabled(true);
  }
  bind->setEnabled(true);
  delete t;
}

/** ------------------------------------------------------------------------
 * @brief popMenu method show a flotating menu with some additional
 * options.
 */
void popMenu(QPoint point) {
  menu->exec(info->mapToGlobal(point));
}

/** ------------------------------------------------------------------------
 * @brief recentSelection method short way, in the connection event.  It
 * method is exec when an user use the cache connections.
 */
void recentSelection(QAction* action) {
  info->setPixmap(QPixmap(":/images/clock.png"));
  info->setEnabled(false);
  bind->setEnabled(false);
  scan->setEnabled(false);
  devices->clear();
  devices->addItem(action->text().left(action->text().size()-1));
  devices->setEnabled(false);
  t = new Thread(devices,net,2);
  connect(t,SIGNAL(connectPerformed(bool)),this,SLOT(connectPerformed(bool)));
  t->start();
}

/** ------------------------------------------------------------------------
 * @brief changeIdiom method switch de interface language between Englis
 * and Spanish
 */
void changeIdiom(QAction *action) {
   if (action->text().left(3)==idiom.getMenuEnglish().left(3) &&
       idiom.getIdiomType()!=ENG) {
     idiom.setIdiomType(ENG);
     refreshIdiom();
   }
   else if (action->text().left(3)==idiom.getMenuSpanish().left(3) &&
            idiom.getIdiomType()!=SPA) {
     idiom.setIdiomType(SPA);
     refreshIdiom();
   }
}

/** ------------------------------------------------------------------------
 * @brief menuOption method exec two of all additional options: Show about,
 * and Clear cache connections
 */
void menuOption(QAction* action) {
  if (action->text()==idiom.getMenuAbout()) {
    showAbout(true);
  }
  else if (action->text()==idiom.getMenuClearConnections()) {
    recents->clear();
  }
}

/** ------------------------------------------------------------------------
 * @brief showAbout method, show the author information.
 */
void showAbout(bool state) {
  if (state==true) {
    info->setPixmap(QPixmap(":/images/about.png"));
  }
  else {
```

```
      info->setPixmap(QPixmap(idiom.getImageInfo()));
    }
  }

};

#endif // WINDOW_H
```

main.cpp

```
#include <QApplication>
#include <window.h>

/** ===========================================================================
 * @brief This es the starting point of NXT PC Remote Control
 */
int main(int argCount,char* argValues[]) {
  QApplication app(argCount,argValues);
  Window w;
  w.show();
  return app.exec();
}
```