# **RASPBERRY PI 4 NETWORKMANAGER CONFIGURATION TOOL**

The aim was to write a Qt application for the Raspberry Pi 4 with which you can switch between different WLAN access points and save the associated credentials. As a starting point, I used a raspbian-buster-lite image and a Qt installation as described in [Qt on the Raspberry Pi 4](https://www.interelectronix.com/qt-raspberry-pi-4.html). I also installed the Network Manager, which allows you to use the shell command (nmcli...) to create, configure and manage network connections. You can find information on this at <https://wiki.debian.org/de/NetworkManager> or at <https://developer.gnome.org/NetworkManager/stable/nmcli.html>. The Network Manager offers a command which you can use to start a monitoring process that then comments on changes to the various interfaces (wlan0 or eth0) (e.g., unavailable, disconnected, connecting, connected...). I wanted to use this monitoring to display the various statuses of the network points in the GUI. Two problems emerged:

* When several nmcli commands were issued in quick succession, the feedback about the various statuses was delayed and could not be displayed live in the GUI.
  + The feedback of the nmcli commands was sent in different slots and could therefore not be coordinated well.

MONITORING FUNCTION

First we need the monitoring function (monitorDevices) and the public slot function, which intercepts and evaluates the monitoring output and, for example, then sends the status messages to the GUI. In the "monitorDevices" function, which launches automatically when the application is started, the nmcli command is launched with sudo: sudo nmcli monitor The instruction "setProcessChannelMode(QProcess::MergedChannels)" states that "Standard-Output" and "Standard-Error" are displayed together in one channel and can thus be evaluated with one function. By using the line "connect (device\_monitoring\_process, SIGNAL(readyReadStandardOutput()), this, SLOT(processOutput()))", this means that whenever a message has been displayed (readyReadStandardOutput), it is sent to the slot "processOutput".

-------------------------------------------------------------------------------------

void CmdLauncher::monitorDevices() {

QStringList device\_monitoring\_on = {"nmcli", "monitor"};

device\_monitoring\_process = new QProcess(this);

device\_monitoring\_process->setProcessChannelMode(QProcess::MergedChannels);

device\_monitoring\_process->start("sudo", device\_monitoring\_on);

connect(device\_monitoring\_process, SIGNAL(readyReadStandardOutput()), this, SLOT(processOutput()));

}

-------------------------------------------------------------------------------------

## Evaluation function The evaluation of the messages is then carried out in the "processOutput" function. Here, the QProcess senderProcess intercepts all messages from all nmcli commands – not just those from the monitoring command.

-------------------------------------------------------------------------------------

void CmdLauncher::processOutput() {

QProcess\* senderProcess = qobject\_cast(sender());

senderProcess->setReadChannel(QProcess::StandardOutput);

QString message = QString::fromLocal8Bit(senderProcess->readAllStandardOutput());

qDebug()

-------------------------------------------------------------------------------------

## Launching another nmcli process If you now launch another nmcli command with the function below, the output will be intercepted by the above monitoring function and can then be evaluated and further processed in the "outputProcess". This function switches over to an existing network connection and launches it. What is important here is not to incorporate, for example, a "set\_wifi\_process->waitForReadyRead()" or a "set\_wifi\_process->waitForFinished()", because then the output of all messages will be blocked until this process is complete.

-------------------------------------------------------------------------------------

void CmdLauncher::setWifi(QString ssid) {

error\_messages = "";

emit getErrorMessagesChanged();

QString uuid = "";

for (int i = 0 ; i setProcessChannelMode(QProcess::MergedChannels);

set\_wifi\_process->start("sudo", args\_wifi\_exist);

connect(set\_wifi\_process, SIGNAL(readyReadStandardOutput()), this, SLOT(processOutput()));

}

-------------------------------------------------------------------------------------

The line "connect(set\_wifi\_process, SIGNAL(readyReadStandardOutput()), this, SLOT(processOutput()))" directs the output messages of this process back into the "processOutput" and can be evaluated again there at a central point.