2013•2014
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef

Ontwerpen van een schaalbaar en generiek testplatform voor managed netwerk switches

Promotor :
ing. Leo RUTTEN

Promotor :
ing. BART SWINNEN

Copromotor :
ing. HANS VANDERMAESEN

Stijn Sontrop

Proefschrift ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2013•2014

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterproef

Ontwerpen van een schaalbaar en generiek testplatform voor managed netwerk switches

Promotor :
ing. Leo RUTTEN

Promotor :
ing. BART SWINNEN

Copromotor :
ing. HANS VANDERMAESEN

Stijn Sontrop

Proefschrift ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Inhoudsopgave

Dankwoord

Dit dankwoord is gericht aan iedereen die mij geholpen heeft met de stage en het samenstellen van dit eindwerk. Toch wil ik enkele mensen in het bijzonder even extra in de schijnwerpers zetten.

Eerst en vooral de mensen van Luminex, die constant met raad en daad klaar stonden. Bart, Frans, Hans, Jan en Wouter: jullie ondersteuning was van onschatbare waarde!

Ook de ondersteuning van mijn interne promotor, Leo Rutten, mag zeker niet onderschat worden. Zijn kritische blik heeft zeer veel bijgedragen aan de verwezenlijking deze thesis.

Abstract (Nederlands)

Luminex Network Intelligence ontwikkelt toestellen voor de entertainment sector. Hun productgamma bevat onder andere “managed netwerk switches” en “netwerk naar DMX convertoren”. Van de toestellen wordt een hoge betrouwbaarheid geëist, dus is grondige controle onontbeerlijk. Ieder toestel dat de assemblagelijn verlaat, dient geprogrammeerd en getest te worden. In het verleden diende de werknemer aan de assemblagelijn hiervoor verschillende scripts op te roepen, en was er hierop praktisch geen foutcontrole.

Het doel van deze masterproef is het ontwikkelen van een generiek test- en programmeerplatform waarmee de bovenstaande acties geautomatiseerd kunnen worden, verborgen achter een gebruiksvriendelijke interface.

Door de gebruiker zo veel mogelijk te begeleiden in een grafische gebruikersinterface, wordt er gepoogd om fouten zo veel mogelijk te detecteren en te vermijden.

Het systeem dient flexibel opgebouwd te worden, zodat het eenvoudig mogelijk is om verschillende reeksen toestellen te ondersteunen.

Voor deze masterproef is er een programma ontwikkeld dat de benodigde testen uitvoert. Daarnaast zijn er tools ontwikkeld die helpen tijdens ontwikkeling en in het magazijn.

Abstract (Engels)

Luminex Network Intelligence develops devices for the entertainment sector. Their productrange contains managed network switches and Ethernet to DMX convertors. High reliability is their key point so thorough testing is essential. Each product that leaves the assembly line needs to be programmed and tested. In the past the factory worker needed to use different kind of scripts and procedures, and there was almost no fault control on this.

The purpose of this master's thesis is the development of a generic test- and programming platform, which will automate the actions above, hidden behind a user-friendly interface.

By guiding the user as much in a graphical interface, we try to detect and avoid as much errors as possible. The target is to build a flexible system, which is easy to port to various kinds of product ranges.

For this master's thesis, an application is developed that executes the necessary tests. Furthermore some tools that help during production and in the warehouse have been developed.

Lijst van gebruikte afkortingen en symbolen

CPU	Central Processing Unit
DMX-512A	Digital Multiplex, protocol voor de sturing van verlichting
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
IGMP	Internet Group Management Protocol
MOC	Meta-Object Compiler
PHY	Physical layer van het OSI model
PoE	Power over Ethernet
RDM	Remote Device Management
RLinX	Redundant links by Luminex
RSTP	Rapid Spanning Tree Protocol
UDP	User Datagram Protocol
VLAN	Virtual LAN
XML	Extensible Markup Language

Lijst van figuren

1. Introductie

1.1 Luminex

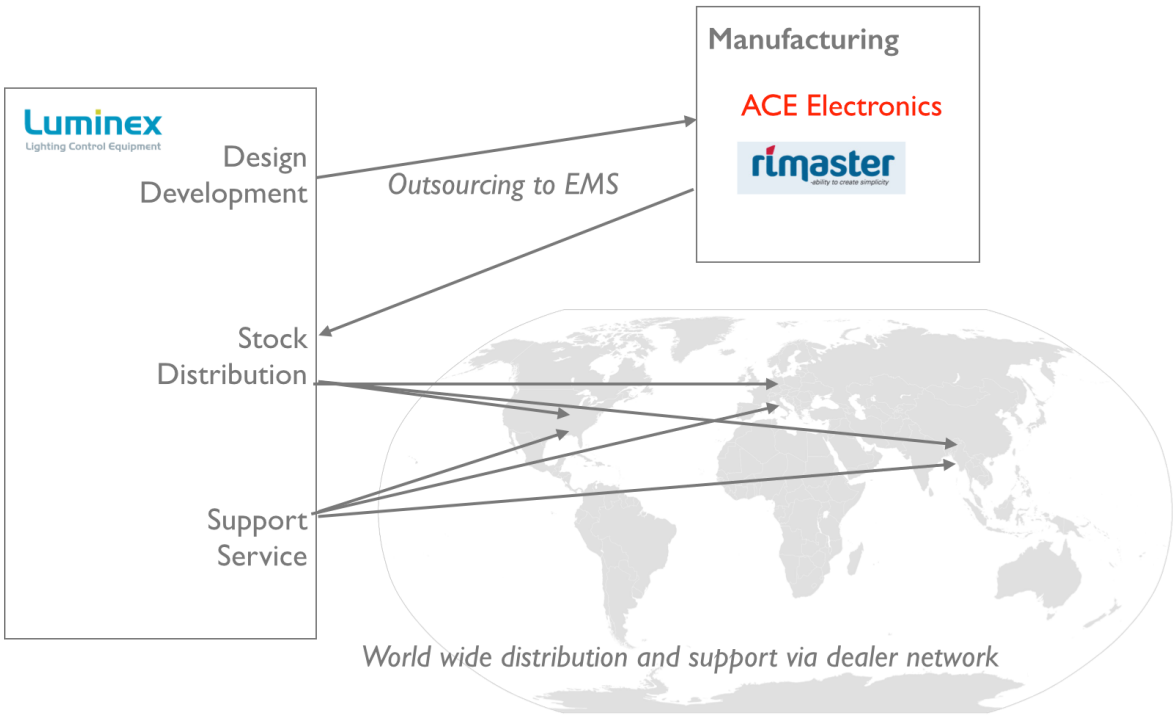
Luminex is een jong en ambitieus bedrijf gespecialiseerd in de ontwikkeling van netwerkapparatuur voor de entertainmentsector. Ze zijn gestart met de ontwikkeling van DMX512-A datadistributieapparatuur, maar doorheen de jaren werd duidelijk dat computernetwerken niet meer weg te denken zijn uit de moderne entertainmentsector. De opkomst van onder andere LED armaturen en videosystemen heeft gezorgd voor een grote nood aan betrouwbare performante verbindingen. De armaturen hebben steeds meer data nodig, waardoor het niet langer werkbaar is om dit allemaal aan te sluiten met enkel DMX bekabeling.

Ook in de pro-audiomarkt wordt de noodzaak voor ethernet netwerken groot, omdat steeds meer en meer audiosignalen digitaal getransporteerd worden.

Ze onderscheiden zich van andere fabrikanten door toestellen te ontwikkelen die ingewikkelde netwerkopstellingen mogelijk maken, zonder dat de gebruiker grondige kennis van netwerken dient te hebben. Het belangrijkste idee achter hun switches is “plug&play”: een gebruiker moet gewoon alle kabels insteken en kunnen verwachten dat het hele systeem werkt. Om dit te bekomen worden meerdere registers van de switch automatisch ingesteld, zoals VLAN’s, trunking, IGMP en RSTP.

De toestellen van Luminex worden ontworpen met het “on the road” gebruik in het achterhoofd: daarom zijn ze onder andere uitgerust met stevige connectoren, zoals onder andere Neutrik Ethercon in plaats van RJ45. Dit alles maakt dat je hun toestellen kan tegenkomen op kleine en grote shows wereldwijd: van Rock Werchter tot zelfs het Eurovisiesongfestival.

Luminex ontwikkelt zowel mechanica, elektronica als software in huis. De hele productie wordt uitbesteed. Vervolgens komen afgewerkte toestellen terug in voorraad. Deze toestellen worden dan uitgevoerd naar meer dan 40 landen wereldwijd. Via dit verdelernetwerk biedt Luminex support en service aan zijn eindgebruikers.



Afbeelding 1: Bedrijfsstructuur van Luminex

1.2 Doelen

Het doel van deze masterproef is het ontwikkelen van een generiek en gebruiksvriendelijk testsysteem voor de toestellen ontwikkeld door Luminex.

Er dient een systeem ontwikkeld te worden waarmee het mogelijk is om verschillende reeksen toestellen tijdens assemblage te programmeren en te testen.

Naast dit hoofddoel worden er nog enkele andere procedures beschreven die zeker nuttig bleken voor het stagebedrijf.

1.2.1 Generiek

Het systeem dient zodanig ontworpen te worden dat met minimale inspanningen ondersteuning kan toegevoegd worden voor nieuwe productreeksen.

1.2.2 Schaalbaar

De software dient zo opgezet te worden dat deze eenvoudig opgeschaald kan worden. Als er bij voorbeeld een test gebouwd wordt die een toestel test met 4 uitgangen, dient het eenvoudig mogelijk te zijn om dit uit te breiden naar een toestel van 8 uitgangen.

1.3 Uitgewerkte oplossingen

In deze thesis worden er verschillende oplossingen aangedragen. Als eerste wordt het gevraagde test- en programmeersysteem besproken. Daarnaast worden er enkele tools ontwikkeld welke nuttig zijn tijdens productie en ontwikkeling.

2. Assemblage test- en programmeerplatform

2.1 Inleiding

Toestellen die de assemblagelijn verlaten, dienen voorzien te worden van firmware en een functionele test te ondergaan. In het verleden werden hiervoor enkele command line scripts gebruikt, welke relatief veel aandacht vereisten van de gebruiker. Voor de test werd er veel verantwoordelijkheid gelegd bij de operator, en was hier weinig controle op.

Voor deze masterproef wordt er een generiek systeem ontwikkeld dat deze handelingen kan verbergen achter een gebruiksvriendelijke interface. Door een modulaire opbouw, is het mogelijk om het systeem eenvoudig uit te breiden naar andere productreeksen.

2.2 Opbouw

De applicatie is opgebouwd uit 1 hoofdprogramma met verschillende plugins.

Bij het opstarten van de applicatie, zal deze zoeken naar geschikte plugins, en deze proberen in te laden. Als alle plugins ingeladen zijn presenteert de software een lijst van alle ondersteunde productreeksen en de specifieke toestellen. Als de gebruiker een toestel selecteert, wordt er een nieuwe instantie aangemaakt van de geselecteerde plugin. Als alle instellingen ingevoerd zijn, wordt de plugin gestart in een nieuwe thread.

De host applicatie biedt alle GUI functionaliteit en interactie met de gebruiker aan. Het is niet toegestaan aan de plugins om zelf grafische interfaces te genereren. Hierdoor is het mogelijk om van een plugin meerdere instanties te starten om zo meerdere toestellen tegelijk te programmeren of te testen. De applicatie biedt de plugins de nodige mogelijkheden aan om te communiceren met de gebruiker, echter worden al deze dialogen afgevangen in een coherente omgeving. De gebruiker krijgt voor iedere productfamilie steeds de vertrouwde dialoogvensters te zien.

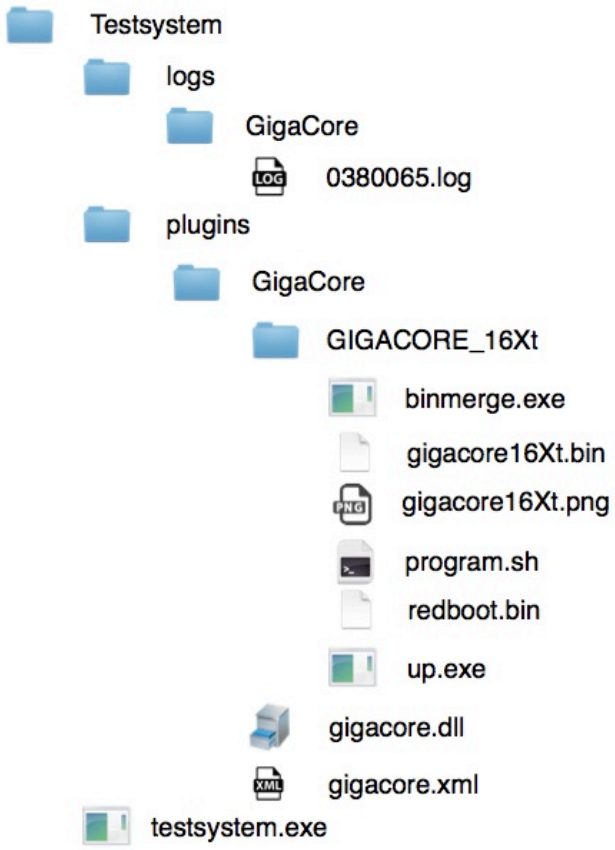
De hostapplicatie biedt ook de nodige voorzieningen om alle voortgang en foutmeldingen op te slaan in logbestanden. Voor ieder type toestel wordt er een map aangemaakt met daarin een logbestand voor ieder serienummer. Dit geeft een snel overzicht van de toestellen die afgehandeld zijn.

Doordat de hostapplicatie praktisch alle interactiviteit en logging aanbiedt, dient de plugin enkel de toestel-specifieke functionaliteit aan te bieden. Hierdoor is het eenvoudig om andere modellen te ondersteunen.

Alle instellingen van de applicatie en plugins worden opgeslagen in XML bestanden. Deze bieden het voordeel dat ze eenvoudig manueel aanpasbaar zijn. Dit staat toe dat componenten van het systeem vlot kunnen vervangen worden. Deze XML bestanden bevatten onder andere parameters voor iedere teststap (test geactiveerd, calibratiegegevens sensoren, IP adressen testborden), maar ook de bestandsnamen van de firmware bestanden, naam van de seriële poort… En dit voor ieder

ondersteund model. Om een nieuwe variant van een bestaand toestel toe te voegen, is het normaalgezien voldoende om wat kopieer- en plakwerk uit te voeren in het XML bestand van de plugin. Ieder XML bestand wordt door de applicatie gevalideerd met een XML schema (XSD bestand). Zie bijlage 1 voor het XML bestand van de GigaCore reeks.

Als de applicatie gecompileerd wordt voor Windows, wordt volgende mappenstructuur gegenereerd:

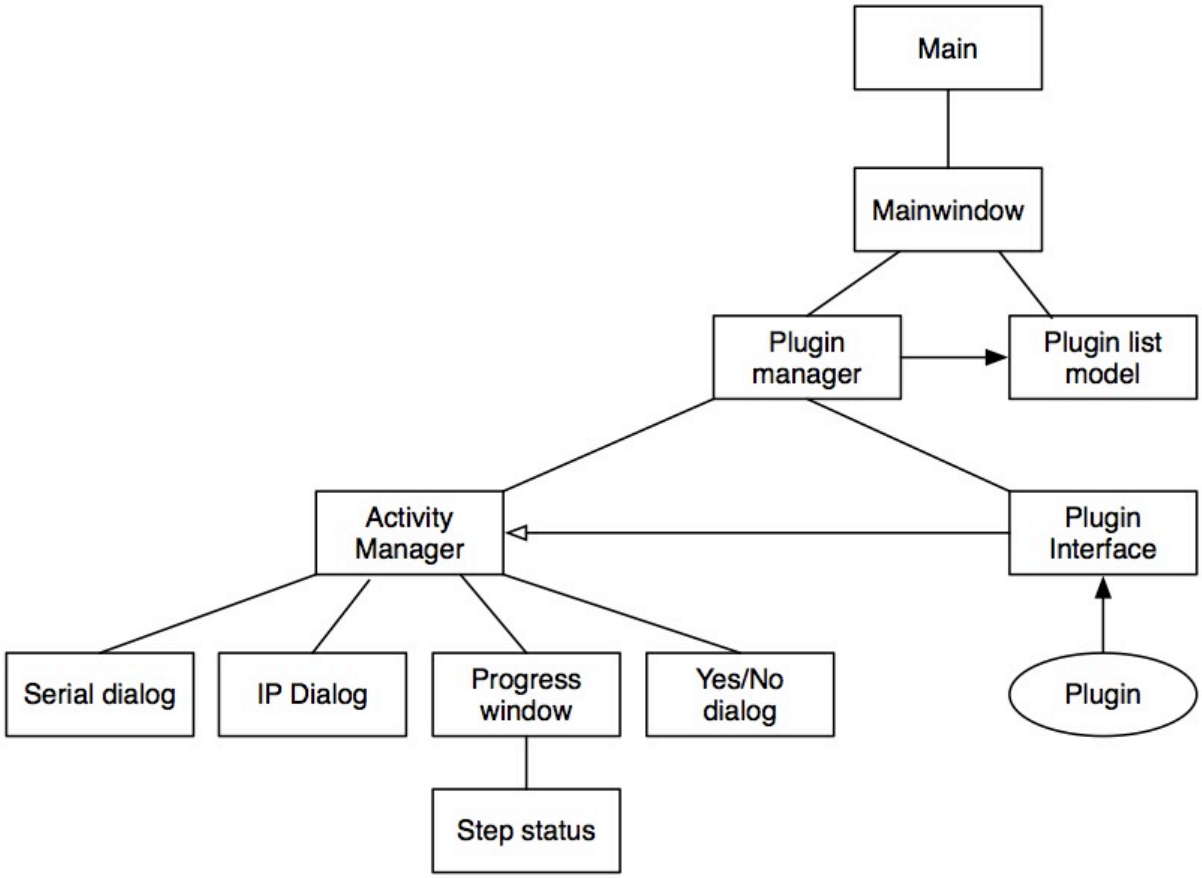


Afbeelding 2: Mappenstructuur van de applicatie

Iedere plugin heeft zijn eigen map onder de map plugins. In afbeelding 2 wordt enkel de GigaCore plugin getoond. De plugin bepaalt zelf de structuur binnen zijn eigen map. Om toch enigszins gestructureerd te werken, wordt er binnen de pluginmap voor ieder model een map gemaakt met de voor dit model specifieke bestanden. In afbeelding 2 ziet u enkel het GIGACORE_16Xt model. Voor een nieuw model te ondersteunen van een bestaande productreeks (plugin), dient men dus een map hiervoor aan te maken, de benodigde firmware bestanden hierin te plaatsen, en de verwijzingen hiernaar aan te maken in het XML bestand. Het XML bestand verwijst naar de bestanden die de plugin nodig heeft om te werken.

Het testsysteem maakt voor iedere plugin een map aan onder de logs map. Hierin wordt voor ieder behandeld toestel een logbestand aangemaakt met als bestandsnaam <serienummer>.log.

Volgend schema toont de structuur van de applicatie.



Afbeelding 3: Structuur van de applicatie

Het hoofdvenster van de applicatie start de plugin manager. Deze zal de lijst met gevonden plugins doorgeven aan het plugin list model, welke dit kan laten zien aan de gebruiker. Het hoofdvenster geeft dan aan de plugin manager opdracht om een nieuwe instantie van de geselecteerde plugin onder te brengen in een nieuwe activity manager. Deze activity manager koppelt alle signals en slots van de plugin aan, en start de gekozen taken.

2.2.1 Hardware opbouw

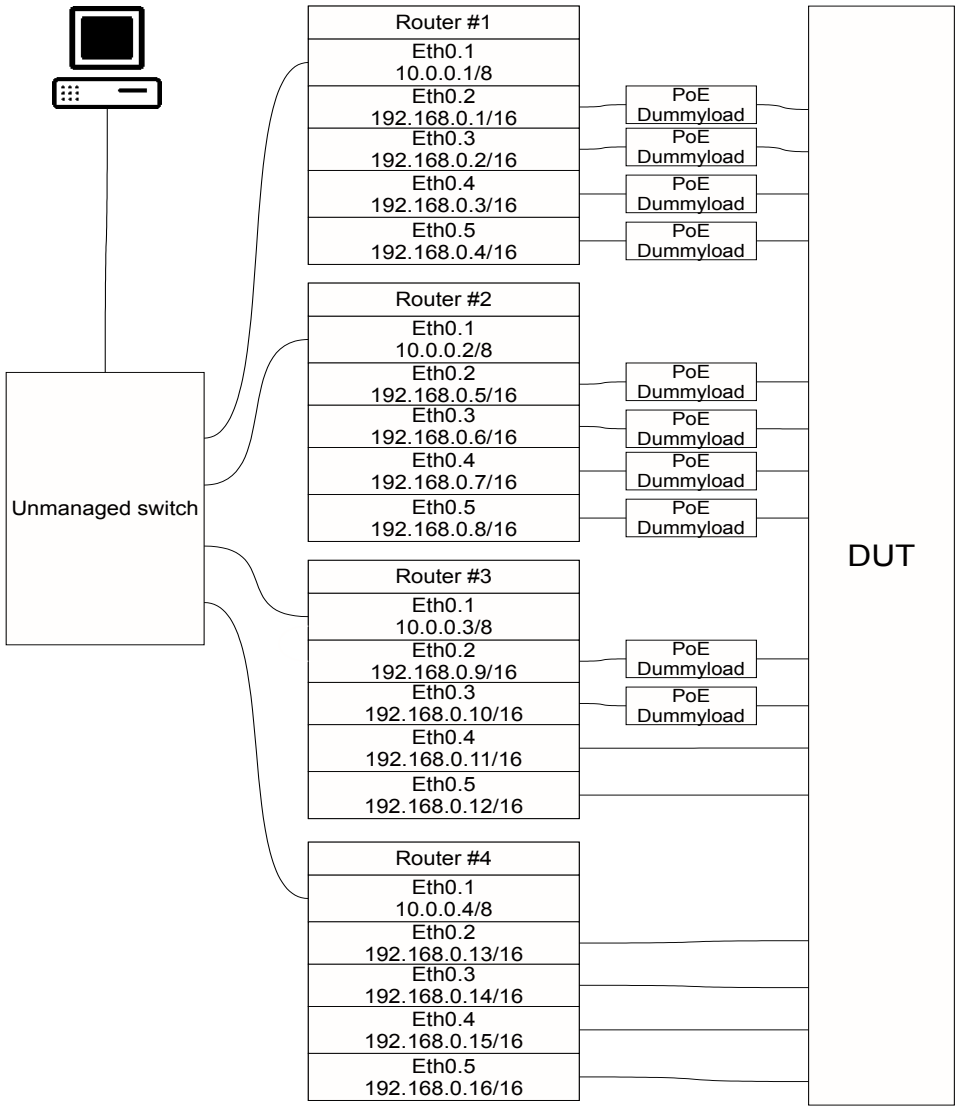
Om iedere poort van een netwerkswitch afzonderlijk te kunnen testen, is er voor iedere poort een netwerkinterface van een computersysteem nodig. Het uitrusten van een computer met 16 netwerkpoorten is echter erg duur. Daarom is er gekozen voor het gebruiken van goedkope TP-Link routers met OpenWRT firmware. Iedere router beschikt over 5 netwerkpoorten, intern verbonden met een managed switch. Door in deze switch iedere poort in te stellen als een access port in een apart VLAN en de CPU poort als trunk, bekomt men verschillende onafhankelijke netwerkinterfaces. Bij een van de routers is er een IPtables routing regel toegevoegd om communicatie tussen de computer en de te testen netwerkswitch toe te laten.

4 van de 5 netwerkpoorten worden verbonden met de te testen switch, de andere poort wordt verbonden met de computer waarop de testsoftware draait. Door

middel van GET en POST commando's kan de computer de routers commando's geven.

In de GigaCore testopstelling bevinden zich 4 TP-Link routers met OpenWRT firmware en 10 PoE dummyloads. Indien er in de toekomst andere toestellen ondersteund dienen te worden, moeten er simpelweg extra routers bijgeplaatst worden.

De opbouw van deze testopstelling wordt weergegeven in het volgende diagram:



Afbeelding 4: Schema testopstelling GigaCore

De computer linksboven draait de test- en programmeersoftware, het vierkant met label “DUT” is de netwerkswitch die getest wordt.

2.2.2 Modulaire opbouw

De modulaire opbouw heeft als belangrijkste kenmerk de mogelijkheid tot dynamisch inladen van plugins. Om nieuwe productreeksen te ondersteunen moet er niets veranderd worden aan de hoofdapplicatie: enkel het toevoegen van een module is voldoende.

Bij het opstarten van de applicatie, wordt de plugins map doorzocht naar geschikte pluginbestanden (*.so, *.dll, *.dylib). De applicatie controleert of er een geschikte headerdefinitie gevonden wordt in dit bestand. Alle geschikte plugins worden opgenomen in een lijst.

Iedere plugin stelt een productreeks voor, en bevat een of meerdere toestellen. Om dit duidelijk te maken aan de gebruiker biedt iedere plugin hiervoor foto’s van de reeks en afzonderlijke toestellen.

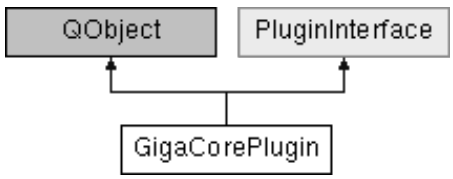
Als een nieuwe programmeer- of testprocedure gestart wordt, wordt hiervoor een ActivityManager object aangemaakt. Dit object creëert een nieuwe instantie van de desbetreffende plugin, en verbindt alle signals en slots. De ActivityManager zorgt er voor dat de plugin toegang krijgt tot dialoogvensters en logboeken, maar bewaakt ook de thread op vastlopers. Doordat iedere activiteit in een nieuwe thread gestart wordt, is het voor plugins niet toegestaan om rechtstreeks de user interface te benaderen: de plugin moet hiervoor steeds via de ActivityManager gaan.

Om een plugin aan te maken in Qt, is het voldoende om “`TEMPLATE = lib`” en “`CONFIG += plugin`” toe te voegen aan het project file.

Iedere plugin dient public over te erven van een abstracte interface klasse die enkel virtual null functies bevat:

```
virtual QIcon getPluginIcon(void) = 0;
```

Dit verplicht de programmeur van de plugin om minstens al deze functies te implementeren. De PluginInterface klasse is echter geen afstammeling van QObject, dus wordt het rechtstreeks connecteren van zijn signals en slots niet toegestaan door de Qt MOC. Men dient alvorens het connecten, de plugin te casten naar een QObject pointer:



Afbeelding 5: Overerving pluginsysteem

```
QObject *pluginobject = dynamic_cast<QObject*>(m_plugin);
connect(pluginobject, SIGNAL(showProgress(int,QString)),
m_progresswindow, SLOT(setProgress(int,QString)));
```

Hierdoor wordt de plugin herkend als een QObject, en kan er communicatie plaatsvinden via het signal/slot mechanisme.

Belangrijk bij het aanmaken van een plugin mechanisme in Qt, is het toevoegen van een versienummering aan de identifier string van de interface. Qt zal weigeren om

plugins te laden met een andere identifier string. Dit zorgt er voor dat de interface waarmee de plugin gecompileerd is steeds overeen komt met de interface in de applicatie.

In de interface:

```
#define PluginInterface_iid
"be.luminex.TestSystem.PluginInterface/0.6"
Q_DECLARE_INTERFACE(PluginInterface, PluginInterface_iid)
```

In de plugin:

```
Q_PLUGIN_METADATA(IID "be.luminex.TestSystem.PluginInterface/0.6"
FILE "LumProgPlugin.json")
```

Deze identifier wordt automatisch gecontroleerd door de qobject_cast routine. Een nadeel is dat alle plugins opnieuw gecompileerd dienen te worden als de interface wijzigt, zelfs diegenen waar functioneel niets aan veranderd is. Men zou zelf een systeem kunnen schrijven dat door middel van vlaggen controleert welke functies er aangeboden worden in een plugin, maar dat is buiten het bestek van deze applicatie. Er is namelijk gepoogd om de interface zodanig te ontwerpen dat de programmeur van de plugins meer dan voldoende vrijheid krijgt om functionaliteit te implementeren zonder tegen de limieten van de plugin interface aan te lopen.

Iedere plugin dient minimaal volgende functionaliteit te voorzien:

- Een beschrijving: naam, afbeelding, ondersteunde modellen, ondersteunde functies
- Registers voor het antwoord van een gebruiker op serienummer, IP, ja/nee en andere dialoogvensters. Qt werkt namelijk event-driven: de applicatie “hangt” niet voor een dialoogvenster, maar krijgt een signaal als dit venster gesloten wordt.
- Een initialiseerfunctie: Het Qt framework start niet automatisch een constructor in plugins. De programmeur dient zelf een dergelijke functie te implementeren en starten. Deze functie wordt gebruikt om variabelen te initialiseren, bestanden in te lezen…
- Een startfunctie: Deze geeft aan dat de plugin mag starten met de opgegeven acties. Deze acties worden doorgegeven als vlaggen.
- Een stopfunctie: Deze functie geeft aan aan de plugin dat de huidige actie onderbroken dient te worden. De plugin kan zelf bepalen of de huidige activiteit onderbroken wordt, of eerst afgewerkt. De hostapplicatie bezit echter ook de nodige functionaliteit om een vastgelopen plugin hard uit het geheugen te verwijderen.

Optioneel kan de plugin ook ondersteuning bieden voor het grafisch (met pictogrammen) tonen van de verschillende stappen.

2.2.3 Generieke opbouw

Het punt waardoor de generieke opbouw het duidelijkst wordt, is de parameter “<numports>12</numports>” in het beschrijvende XML bestand van een model.

Door simpelweg dit getal aan te passen, past de software zijn testprocedures aan zodat toestellen met een ander aantal poorten ondersteund kunnen worden.

Een doorgedreven gebruik van functies en klassen, zorgt er voor dat er zeer veel code herbruikt kan worden. Het was een belangrijk punt om niet zozeer uit te gaan van de huidige productreeksen, maar de toekomst steeds een stap voor te zijn. Het systeem moet eenvoudig kunnen opschalen met verschillende toekomstige productreeksen.

2.2.4 Software framework

Er zijn verschillende frameworks beschikbaar om een grafische applicatie te ontwikkelen. Onder andere WxWidgets, Qt, JUCE en GtkMM bieden allemaal uitgebreide functionaliteit om op verschillende platformen een grafische gebruikersinterface te maken. Er is gekozen voor Qt 5.2.1 omdat er in het stagebedrijf al de nodige ervaring met Qt aanwezig is. Hierdoor kan de applicatie achteraf nog steeds eenvoudig onderhouden worden.

Qt biedt naast de grafische ondersteuning ook de nodige functies om eenvoudig sockets op te zetten, bestanden en seriële poorten te benaderen en een zeer gebruiksvriendelijk signal/slot mechanisme. Dit houdt ook een grote valkuil in: het is erg moeilijk om te debuggen van waar een slot exact wordt opgeroepen. Misbruik van het signal/slot mechanisme leidt al snel tot spaghetticode.

Hoewel Qt ondersteuning biedt voor verschillende besturingssystemen, dient de programmeur rekening te houden met verschillende mappenstructuren op de verschillende besturingssystemen. Zo plaatst Mac OSX bij voorbeeld het programma in een map met extensie .app. Als men dus logs wilt opslaan in dezelfde map als het programma, moet men eerst een map omhoog gaan in de mappenstructuur.

Ook in het Makefile voor het systeem staan er verschillende regels voor Windows en Unix. De gebouwde plugins met hun configuratiebestanden dienen na het compileren gekopieerd te worden naar de plugins map van de applicatie. Echter zijn de commando's voor bestanden te kopiëren verschillend op Windows en Unix.

Bij voorbeeld voor de GigaCore module zijn volgende acties benodigd:

```
win32 {
    DESTDIR_WIN = $${DESTDIR}
    DESTDIR_WIN ~= s,/,\\,g
    PWD_WIN = $${PWD}
    PWD_WIN ~= s,/,\\,g
    for(FILE, OTHER_FILES){
        QMAKE_POST_LINK += $$quote(cmd /c copy /y
$${PWD_WIN}\\$${FILE} $${DESTDIR_WIN}$$escape_expand(\\n\\t))
    }
    QMAKE_POST_LINK += $$quote(cmd /c xcopy $${PWD_WIN}\\GIGACORE_12
$${DESTDIR_WIN}/GIGACORE_12 /q /e /y$$escape_expand(\\n\\t))
    QMAKE_POST_LINK += $$quote(cmd /c xcopy $${PWD_WIN}\\GIGACORE_14R
$${DESTDIR_WIN}/GIGACORE_14R /q /e /y$$escape_expand(\\n\\t))
    QMAKE_POST_LINK += $$quote(cmd /c xcopy
$${PWD_WIN}\\GIGACORE_16Xt $${DESTDIR_WIN}/GIGACORE_16Xt /q /e
/y$$escape_expand(\\n\\t))
```

```
}
unix {
    for(FILE, OTHER_FILES){
        QMAKE_POST_LINK += $$quote(cp $${PWD}/$${FILE}
$${DESTDIR}$$escape_expand(\\n\\t))
    }
    QMAKE_POST_LINK += $$quote(cp -r $${PWD}/GIGACORE_12
$${DESTDIR}$$escape_expand(\\n\\t))
    QMAKE_POST_LINK += $$quote(cp -r $${PWD}/GIGACORE_14R
$${DESTDIR}$$escape_expand(\\n\\t))
    QMAKE_POST_LINK += $$quote(cp -r $${PWD}/GIGACORE_16Xt
$${DESTDIR}$$escape_expand(\\n\\t))
}
```

Dit stuk code kopieert het pluginbestand samen met de bestanden voor instellingen, firmware van de toestellen, programmeerscripts… naar de plugins map van het testsysteem bij compilatie. Op Windows wordt hiervoor gebruik gemaakt van xcopy, en op Unix van cp.

In de gratis (open source) versie van Qt wordt er dynamisch gelinkt met de Qt bibliotheken. Hierdoor moet er een installatieprogramma gemaakt worden dat de gebruiker voorziet van deze bibliotheken.

Qt voorziet in 2 methodes om een grafische gebruikersinterface aan te maken: Qt Widgets en Qt Quick (QML). Qt Widgets is de klassieke manier, en is al jaren in ontwikkeling: op een standaard window worden componenten als textboxes, buttons, labels… geplaatst, en dit wordt door Qt in C++ code omgezet. De enige manier om hier interactiviteit mee te bereiken, is door er zelf C++ code achter te hangen.

Qt Quick is de nieuwe methode, en maakt gebruik van een declaratieve opmaaktaal, genaamd QML. De QML code wordt tijdens het draaien van het programma geëvalueerd. Om interactiviteit te bekomen biedt Qt Quick ondersteuning voor zowel C++ als Javascript.

Voor dit project is er gekozen voor de klassieke Qt Widgets manier van aanpakken. Doordat dit al langer in ontwikkeling is, is dit een stabieler systeem waar de kinderziektes al grotendeels uit zijn.

2.3 Programmeerprocedure GigaCore

Er is niet voor iedere reeks toestellen een volledige programmeer- en testprocedure uitgewerkt. Dit is enkel gebeurd voor de GigaCore managed netwerk switches. Aangezien dit de meest gecompliceerde producten zijn, is het ondersteunen van de andere productreeksen in relatief weinig tijd te implementeren.

De applicatie genereert voor ieder te programmeren toestel een binair bestand waarin de firmware en parameters zoals MAC adres en serienummer verwerkt worden.

De programmeerapplicatie stuurt Asix Up aan, welke door middel van JTAG de firmware op het toestel plaatst.

2.4 Testprocedure GigaCore

De gebruiker dient de switch aan te sluiten op een testsysteem. Dit systeem bevat een aantal OpenWRT router boards en PoE dummyloads. Om het aansluiten te vereenvoudigen, zijn alle netwerkconnectoren met elkaar verbonden via plaatjes. Hierdoor is het mogelijk om in 1 vlotte beweging verschillende netwerkpoorten tegelijk aan te sluiten.

De testprocedure bestaat uit een aantal stappen. De stappen die interactie van de gebruiker vragen zijn samen gegroepeerd, zodat de gebruiker zo efficiënt mogelijk kan werken.

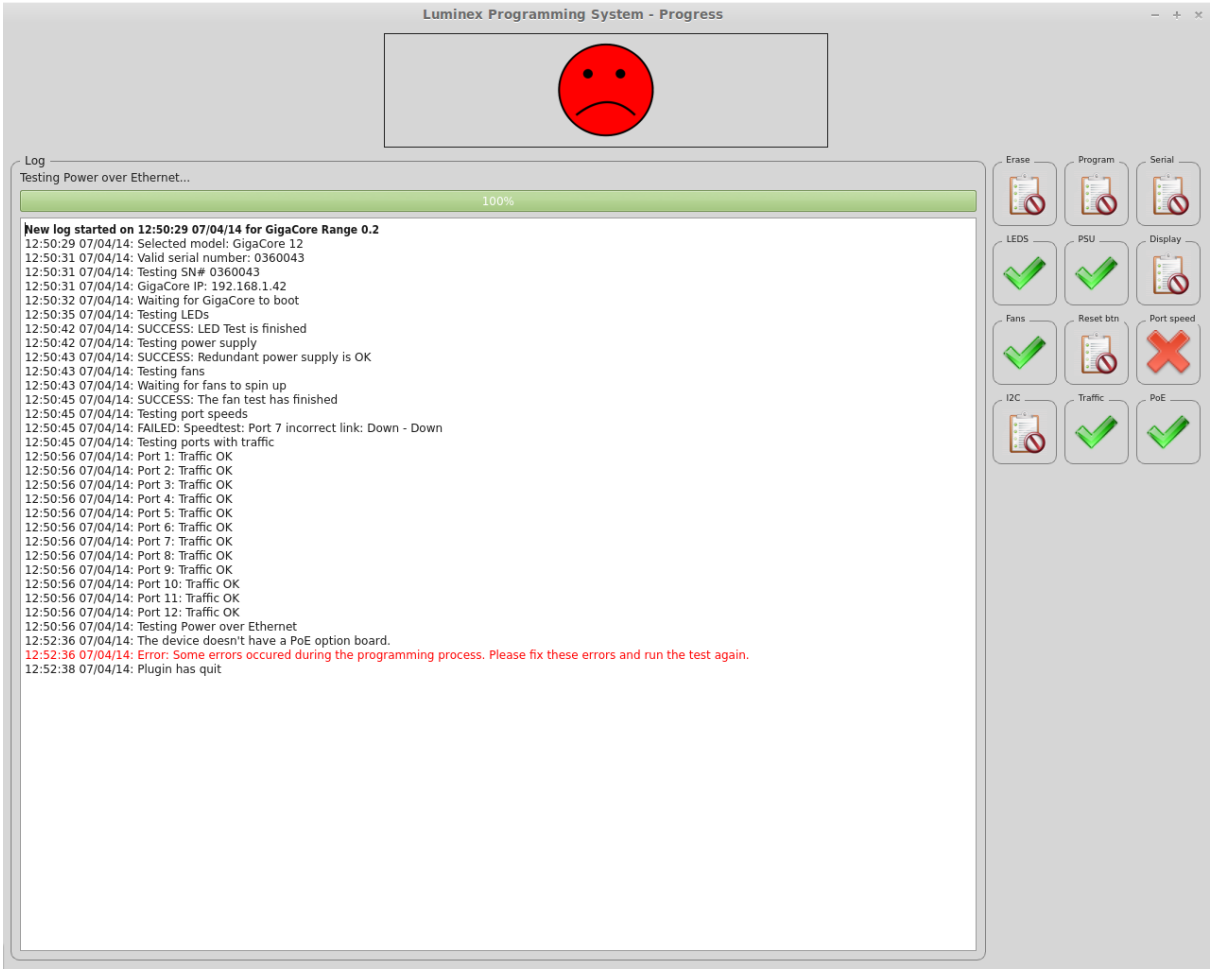
Volgende teststappen worden uitgevoerd:

- Leds
  In verschillende stappen worden groepen leds ingeschakeld. Op het computerscherm verschijnt een tekening met de oplichtende leds ingekleurd. De gebruiker moet nagaan of de juiste leds oplichten, en of de juiste kleuren leds gemonteerd zijn. Om deze leds aan te sturen is er een HTTP POST commando voorzien in de GigaCore.
  Deze test vangt defecte en foutief geplaatste leds op. Hoewel het mogelijk is om iedere LED afzonderlijk aan te sturen, wordt dit niet gedaan omdat het te veel tijd zou kosten. Er wordt telkens 1 groep van leds per keer ingeschakeld (bij voorbeeld alle blauwe leds).
- Redundante voeding
  Sommige modellen beschikken over een redundante voedingsaansluiting. Er wordt gevraagd aan de gebruiker om de hoofdvoeding los te koppelen en na te gaan of het toestel overschakelt op de backup voeding.
  Voor de toestellen die uitgerust zijn met PoE functionaliteit dient deze test uitgevoerd te worden met ingeschakelde PoE voeding.
- Display
  Er wordt gevraagd aan de gebruiker om te controleren of het display correct gemonteerd is, en of de draaiknop correct werkt. Een tekening geeft schematisch weer hoe dit uitgevoerd moet worden.
- Ventilatoren
  De ventilatoren worden gedurende korte tijd ingeschakeld, en daarna wordt het toerental uitgelezen. De test is succesvol als het toerental zich tussen 2 grenzen bevindt.
  Deze test vangt defecte en foutief aangesloten ventilatoren op.
- Reset knop
  De applicatie vraagt aan de gebruiker om de reset knop aan de achterzijde in te drukken. De applicatie leest de switch uit, en controleert of deze knop ingedrukt is.
  Deze test vangt defecte reset schakelaars op. Een op actief vasthangende

schakelaar wordt niet gedetecteerd door deze test, omdat de netwerk switch dan niet zal opstarten.

- Poort snelheden
  De applicatie vraagt de link speed van iedere poort op. De test slaagt als deze 1Gbit/sec full duplex is.
  Deze test vangt defecte PHY's en connectoren op. Als bij voorbeeld een aderpaar niet aangesloten is, zal de poort geen gigabit snelheid meer halen en faalt de test.
- Poort trafiek
  De OpenWRT bordjes versturen op hoge snelheid ping pakketten naar elkaar op verschillende netwerkinterfaces. De test slaagt als de packet loss 0% is.
  Een andere mogelijkheid is het gebruik van tools als Iperf. Met Iperf is het echter niet mogelijk om in te stellen vanaf welke netwerkinterface de pakketten het systeem verlaten als verschillende interfaces verbonden zijn met het zelfde subnet. Dit is bij ping wel mogelijk. Door de ping pakketten groter te maken, en de wachttijd tussen 2 pings te verkleinen wordt er een relatief grote bandbreedte ingenomen.
- I2C Bus
  Sommige modellen beschikken over een I2C aansluiting. Er wordt een commando verstuurd over deze bus om een LED op een IO-expander te doen oplichten. De gebruiker moet controleren of deze led brandt.
- Power over Ethernet
  Power over Ethernet wordt ingeschakeld op de poorten van de netwerk switch. Daarna wordt het vermogensverbruik gemeten en gecontroleerd of dit binnen een bereik bevindt.
  Een veel voorkomende fout bij het monteren van de PoE borden, is dat een of meerdere pinheaders niet verbonden worden doordat het bord schuin gemonteerd wordt. Sommige poorten krijgen dan geen PoE voeding. Deze test zal dan opmerken dat er geen energie verbruikt wordt op deze poort, en de test faalt.

De status van deze stappen wordt aangegeven door middel van pictogrammen. Zonder het uitgebreide log te moeten lezen, is in 1 oogopslag duidelijk wat er fout ging. Er zijn pictogrammen voor "wacht tot verwerking", "overgeslagen", "succesvol" en "gefaald".



Afbeelding 6: Statusvenster tijdens testen

De banner bovenaan het venster geeft instructies met betrekking tot de huidige stap. Bij de LED test wordt hier bij voorbeeld aangegeven welke leds er dienen op te lichten op het toestel. Aan het einde van de procedure wordt deze banner gebruikt om een duidelijke indicatie te geven of de procedure geslaagd is.

2.6 Testprocedure Ethernet-DMX nodes

De Ethernet-DMX node reeks van Luminex omvat toestellen die een omzetting uitvoeren tussen Ethernet en DMX signalen.

DMX is een stuursignaal dat in de entertainmentsector gebruikt wordt om verlichting te besturen. Het is een differentiële RS-485 bus die waar ieder verlichtingsarmatuur parallel op aangesloten is. Het is mogelijk om verschillende DMX “universes” over een standaard Ethernet netwerk te transporteren. Hiervoor bestaan verschillende protocollen zoals Art-Net en streaming ACN. Met een Ethernet-DMX node kan er een omzetting gebeuren tussen deze protocollen en een DMX signaal.

De hieronder beschreven testprocedure is enkel theoretisch uitgewerkt, de software implementatie dient nog te gebeuren.

Alle toestellen in deze reeks bezitten dezelfde functionaliteit, maar verschillen enkel in hoeveelheid DMX uitgangen en de aanwezigheid van een display.

Volgende tabel beschrijft de verschillende modellen in deze reeks:

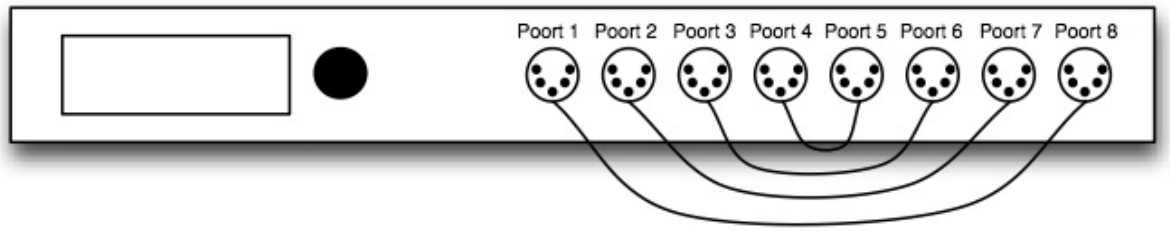
Model	DMX Poorten	Display
Node 2 MKII	2	Nee
Node 2 MKII Truss	2	Nee
Node 4 MKII	4	Ja
Node 4 MKII Truss	4	Ja
Node 8 MKII	8	Ja

Deze testprocedure beschrijft hoe deze toestellen volledig getest kunnen worden zonder specifieke hardware, enkel een computer is nodig.

Volgende onderdelen blijken het merendeel van de defecten te veroorzaken:

- Connectoren
- Leds / Lightpipes
- Display en rotary encoder
- RS-485 transceivers + beveiligingscircuit
- Defect RAM geheugen

De gebruiker wordt gevraagd om een netwerkverbinding tussen de node en de computer te maken, en verschillende poorten met elkaar te verbinden, zoals aangegeven in het volgende schema:



Afbeelding 7: Aansluitschema Ethernet-DMX node test

De computer stelt poorten 1 tot en met 4 in als uitgang en 5 tot en met 8 als ingang. De computer stuurt dan Art-Net data naar de node verstuurd. De computer vergelijkt de data die binnenkomt op poorten 5 tot en met 8 met de gegenereerde data. Als blijkt dat deze data correct terug ingelezen wordt, wordt deze test nogmaals herhaald, maar worden nu poorten 1 tot 4 ingang en 5 tot 8 uitgang.

Deze testmethode is perfect op te schalen, zolang de te testen toestellen beschikken over een even aantal poorten.

Bij toestellen die beschikken over een display, wordt aan de gebruiker gevraagd om het contrast af te regelen. Hierdoor is er een controle dat zowel het display als de rotary encoder werken. De gebruiker dient ook visueel te inspecteren dat alle LED's (power en Ethernet link) branden.

Het grote voordeel van deze testmethode is dat er buiten een computer en enkele DMX kabels geen extra hardware benodigd is. Het nadeel is dat als bij een poort bij voorbeeld enkel de positieve lijn van de differentiële bus goed aangesloten is (bij voorbeeld een defecte polyfuse), dit mogelijk nog steeds goedgekeurd wordt.

Voor een test die ook deze fouten kan detecteren, dient er hardware gebruikt te worden die de spanning op beide aders van elk differentieel paar meet.

2.7 Testprocedure DMX splitters

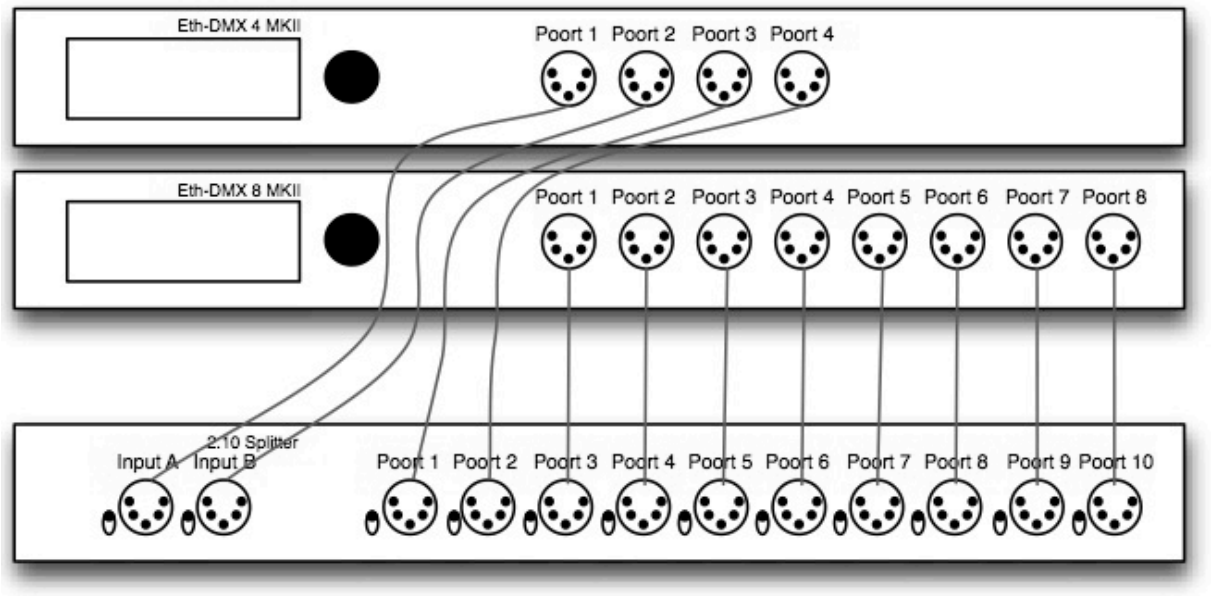
De Luminex DMX splitters zijn verdelers voor het RS-485 gebaseerde DMX protocol. Ze beschikken over 2 ingangen en 10 optisch geïsoleerde uitgangen. Bij iedere uitgang zit een schakelaar waarmee de gebruiker kan kiezen welk ingangssignaal er doorgegeven dient te worden. Om RDM over DMX te kunnen ondersteunen, werken alle poorten bidirectioneel: iedere uitgang kan ook data ontvangen en terugsturen naar zijn gekoppelde ingang.

De hieronder beschreven procedure is enkel theoretisch uitgewerkt, de software implementatie dient nog te gebeuren.

Volgende onderdelen blijken het merendeel van de defecten te veroorzaken:

- Connectoren
- Accu
- RS-485 transceivers + beveiligingscircuit
- Schakelaars
- Leds

Om deze DMX splitters te kunnen testen, is er externe hardware benodigd. De testopstelling bestaat uit een computer, een 2 Luminex Ethernet-DMX nodes en verschillende DMX kabels. De nodes worden hierbij gebruikt om DMX data te kunnen verzenden en ontvangen doorheen de splitter.



Afbeelding 8: Aansluitschema DMX splitter test

Met de schakelaars bij de ingangen (op schema: “Input A” en “Input B”), kan de gebruiker kiezen of de splitter unidirectioneel of bidirectioneel dient te werken. De schakelaars bij de uitgangen (op schema: “Poort 1” tot en met “Poort 10")) kiezen met welke ingang de uitgang verbonden wordt.

De test bestaat er in om 2 richtingen data te verzenden door iedere poort. De gebruiker wordt tijdens de testprocedure gevraagd om schakelaars te bedienen. Als het toestel ingesteld wordt op bidirectionele modus, wordt er getest of dit ook daadwerkelijk wordt toegestaan op iedere poort. Als dit uitgeschakeld is, wordt getest hoe het toestel reageert op inkomende data op een uitgang.

De data richting van de poorten van de 2 nodes kan via een netwerkprotocol worden ingesteld, en de data van iedere poort wordt tussen de node en de computer getransporteerd via het Art-Net protocol.

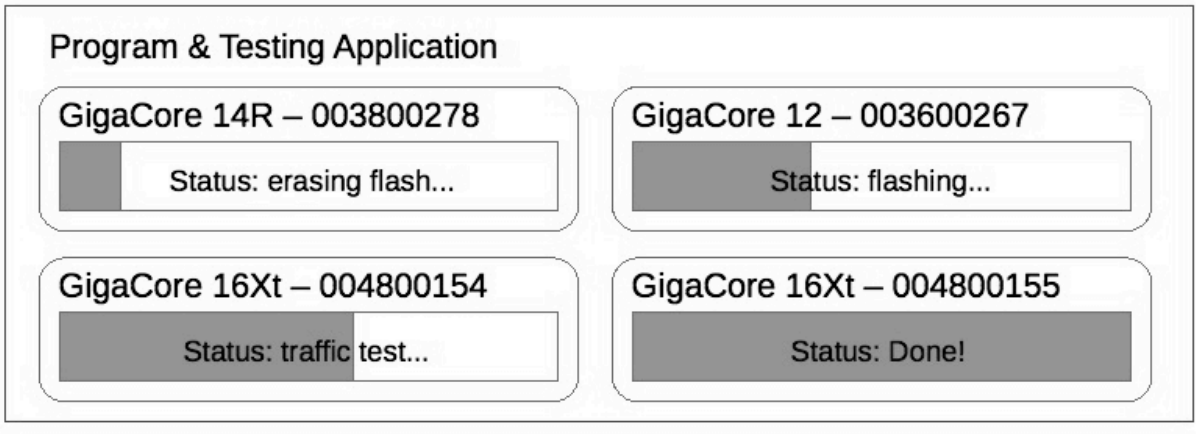
De gebruiker dient te controleren dat bij iedere poort de data-activiteitsled brandt.

Aan het einde van de testprocedure wordt gevraagd aan de gebruiker om de netspanning van het toestel los te koppelen. In normale omstandigheden heeft de ingebouwde accu voldoende tijd gehad om enige lading te verwerven. De gebruiker dient te controleren of de power LED oranje kleurt na het loskoppelen van de netspanning.

2.8 Toekomstige uitbreidingsmogelijkheden

Er dienen modules geschreven te worden om alle productreeksen van Luminex te ondersteunen. Omdat de GigaCore module de meest uitgebreide is, kan deze dienen als voorbeeld om andere productreeksen te programmeren.

Momenteel start de testapplicatie slechts 1 test of programmeerproces per keer. Een goede uitbreiding zou zijn om verschillende testprocessen te kunnen beheren vanuit 1 centrale interface. Volgende mockup geeft een voorbeeld van hoe een geparallelliseerde gebruikersinterface er uit zou kunnen zien:



Afbeelding 9: Mockup van gebruikersinterface

In plaats van een groot logvenster per proces, wordt ieder test- of programmeerproces een onderdeel van een overzichtsvenster.

Het is mogelijk om met 1 gecentraliseerd tegelijk verschillende toestellen te testen en te programmeren. Dit is een van de redenen om de trafiektest te laten uitvoeren door de routerbordjes: een computer kan makkelijk meerdere testopstellingen aansturen. Door meerdere processen tegelijk uit te voeren, zal de operator zijn tijd ook beter benut kunnen worden.

Het ondersteunen van verschillende software versies zou ook een goede aanvulling zijn. Tijdens ontwikkeling moeten er voor testen soms oudere firmware versies op een toestel geplaatst worden, met deze applicatie zou dat proces vereenvoudigd kunnen worden.

3. RSTP Karakterisatietest

3.1 Inleiding RSTP

Ethernet switches evalueren ieder inkomend pakket, en onthouden welke MAC adressen aanwezig zijn op iedere poort. Hierdoor weet een netwerk switch op welke poort hij een pakket voor een bepaald MAC adres moet versturen. Wanneer een MAC adres onbekend of een broadcast pakket is, zal het pakket "geflood" worden op alle poorten.

Als netwerkswitches in een lus zijn aangesloten, zal een enkel broadcast pakket steeds opnieuw blijven circuleren door het netwerk. Deze situatie zal uiteindelijk alle bandbreedte in de lus gebruiken waardoor het netwerk onbruikbaar wordt.

Het Rapid Spanning Tree Protocol (RSTP) lost deze loops op door verbindingen uit te schakelen tot een enkelvoudig netwerk over blijft.

Hoewel het protocol ooit uitgevonden is om loops te vermijden in slecht beheerde netwerken, is het ook bruikbaar om een vorm van redundantie toe te voegen aan het netwerk.

Het "RLinX" protocol van de Luminex netwerk switches is gebaseerd op RSTP. Ten opzichte van standaard RSTP biedt het een snellere recovery, en kunnen de switches aangeven welke poorten er een redundante verbinding vormen.

3.2 RSTP Recovery tijd

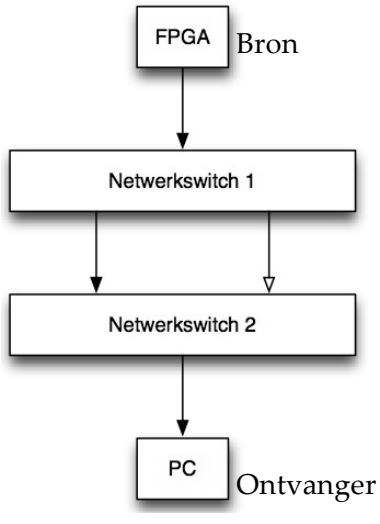
Als de actieve verbinding verbroken wordt, dient RSTP er voor te zorgen dat een van de uitgeschakelde verbindingen terug ingeschakeld wordt. Ondertussen is er een korte periode waarin er geen trafiek door de verbinding kan. Deze tijd is onder andere afhankelijk van de netwerktopologie (ring, mesh) en het aantal switches.

3.3 Meetopstelling

Er is een methode ontwikkeld om deze onderbreking nauwkeurig te kunnen meten. De meetopstelling bestaat uit een zender en een ontvanger.

De zender verstuurt met een vaste snelheid pakketten met daarin een teller die bij ieder verzonden pakket verhoogd wordt. De eerste optie was om hiervoor de OpenWRT bordjes van het testsysteem te gebruiken, maar al snel bleek dat deze niet in staat zijn om op een stabiele snelheid data te versturen. Ook het volledig benutten van een gigabitverbinding bleek een te hoge eis voor de processor op deze bordjes.

Om een vaste timing te verkrijgen, wordt er gebruik

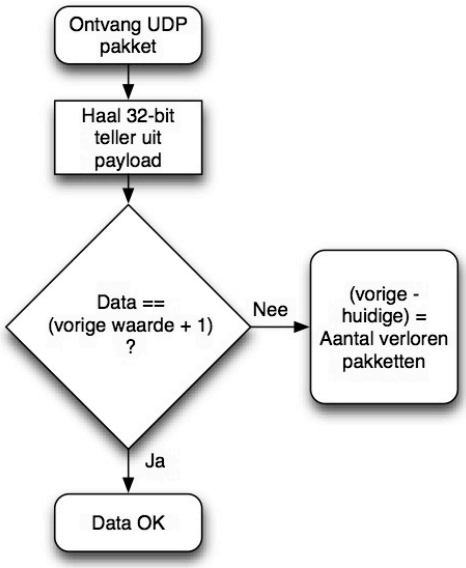


Afbeelding 10: RSTP recovery meetopstelling

gemaakt van een FPGA. Een klokdeler zorgt er voor dat om de 20 microseconden een UDP pakket verstuurd wordt over het netwerk.

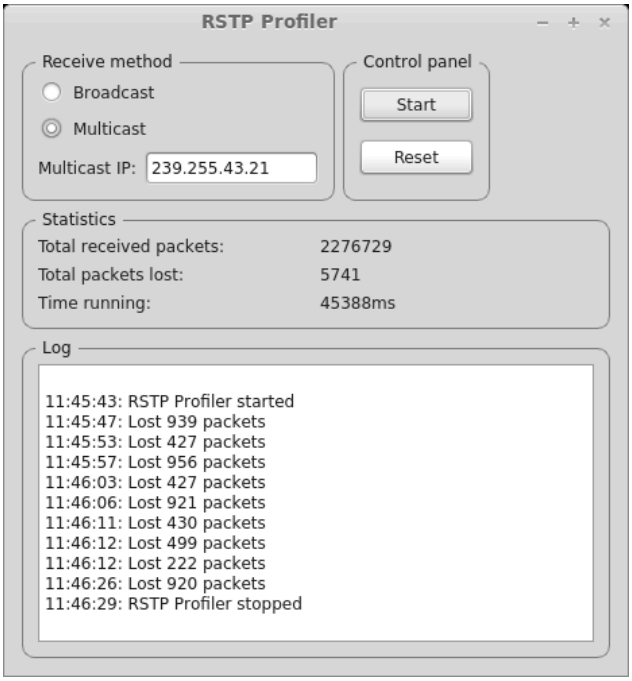
Er is gekozen voor een implementatie in een FPGA, zodat de gevraagde timing steeds behaald kan worden. Ook is er geen ondersteuning voor flow control in de FPGA: pakketten die niet verwerkt kunnen worden door de switch moeten gedropt worden. Er worden UDP pakketten gebruikt omdat deze geen retransmissions kennen, en door het netwerk gedropt mogen worden als ze niet afgeleverd kunnen worden.

Als ontvanger is er een simpele computerapplicatie die deze pakketten ontvangt en controleert welke er ontbreken. Door het aantal ontbrekende pakketten te vermenigvuldigen met 20 microseconden, weet men de duur van de onderbreking. De computer heeft geen nauwkeurige klok nodig, want de tijd tussen 2 pakketten is hard bepaald door de zender. De ontvangstroutine wordt in een aparte thread met hoge prioriteit geplaatst, zodat de kans dat pakketten verloren geraken in de computer verkleind wordt. Het is aan te raden om geen andere applicaties te draaien als er metingen uitgevoerd worden. Ook deze applicatie is geschreven in C++ met het Qt 5.2.1 framework.



Afbeelding 11: Flowchart RSTP recovery test

Omdat de netwerk switches anders om gaan met multicast data dan met broadcast data, is het met het testprogramma ook mogelijk om multicast data te versturen. IGMP Snooping zorgt er dan voor dat de multicast data enkel verstuurd wordt over de verbindingen waarop er expliciet gevraagd is voor de multicast data. Hiervoor vraagt 1 switch (de “querier”) aan alle aangesloten toestellen welke multicast data overal benodigd is. Andere switches “snoopen” deze informatie en zullen enkel de gevraagde data over verbindingen doorsturen.



Afbeelding 12: Screenshot RSTP recovery tijd meetprogramma

Een andere methode om deze onderbreking vast te stellen, is het gebruik van de netwerksniffer Wireshark. Wireshark kan een grafiek tekenen van de hoeveelheid ontvangen pakketten per tijdseenheid. Als er een constante datastroom verstuurd wordt, kan de onderbreking van de netwerkverbinding duidelijk gezien worden in deze grafiek. De datastroom kan bij voorbeeld opgewekt worden met een tool als LumiNet Monitor die een grote hoeveelheid ArtNet universes kan genereren. Deze methode is minder nauwkeurig, maar vereist geen specifieke hardware.

3.4 Opbouw zender in FPGA

De software voor de FPGA is gebaseerd op een demoproject. (Zie bronnen) Hieruit is de code om UDP frames te verzenden overgenomen. Een klokdeler deelt de 100MHz basisklok op, en genereert zo een startsignaal voor een statemachine. De pakketten worden opgewekt door “hardware”, niet door een softcore in de FPGA. De volledige controle gebeurt door de statemachine.

Deze statemachine genereert een UDP header, plaatst er de data in en verstuurt deze met de Marvel 88E1111-RCJ1 PHY die aanwezig is op het Digilent Atlys bord. Er is gekozen voor dit Atlys bord, omdat dit ter beschikking gesteld kon worden door UHasselt, en ondersteuning biedt voor gigabit Ethernet. Dit bord bevat een Spartan 6 FPGA van Xilinx.

Met de schakelaars aanwezig op het ontwikkelbord is het mogelijk om extra bytes toe te voegen aan de pakketten. Als een pakket groter gemaakt wordt, voegt de FPGA extra nullen toe aan het einde van het pakket.

De FPGA wordt niet bestuurd door de computer. Eenmaal geactiveerd blijft deze continu pakketten uitzenden.

De IP- en MAC adressen voor broadcast en multicast zijn hard gecodeerd in de FPGA. Men dient manueel het MAC adres te berekenen dat bij een multicast IP adres hoort.

3.5 Resultaten

Er zijn metingen uitgevoerd met steeds 2 GigaCore 12 toestellen. Poorten 1 en 2 waren in gebruik als interswitch links, en op poort 10 was bij de ene switch de FPGA als bron en bij de andere de computer als ontvanger aangesloten. Voor zowel multicast als broadcast data zijn er testen uitgevoerd op de huidige release firmware (v1.1.2) en de momenteel in ontwikkeling zijnde firmware (v2.0.0_RC1). Er is gemeten hoe lang de onderbreking is bij het uittrekken van de actieve link, en bij het terugplaatsen van deze link. Bij de multicast meting wordt er na het uittrekken gewacht tot men zeker is dat de IGMP registratie op deze link verwijderd is uit het geheugen van de switch. Alle testen zijn uitgevoerd met UDP pakketten van 96 bytes.

De resultaten zijn het gemiddelde van 10 metingen. Rechts van ieder gemiddelde staat de standaard afwijking.

	Disconnect (ms)	STD Dev (ms)	Reconnect (ms)	STD Dev (ms)
V1.1.2 Broadcast	7,80	0,08	5,60	1,21
V1.1.2 Multicast	9,97	0,06	9,55	1,55
V2.0.0_RC1 Broadcast	10,75	0,12	10,93	6,37
V2.0.0_RC1 Multicast	13,51	0,40	17,61	5,88
TP-Link Broadcast	1145,26	231,92	304,79	73,03
TP-Link Multicast	1220,43	270,84	395,09	85,51

Het valt op dat op de huidige firmware, de verbinding minder lang verbroken wordt dan bij de release candidate. De kleinere standaardafwijking geeft ook aan dat de resultaten hier stabieler zijn. Uit deze gegevens kan besloten worden dat de prestaties van het toestel met de nieuwe V2.0.0_RC1 firmware achteruit zijn gegaan.

Om een vergelijking te hebben, is dezelfde test ook uitgevoerd met 2 TP-Link TL-SG3216 managed switches. Hieruit blijkt dat de verbeteringen die Luminex met “RLinX” heeft toegevoegd aan RSTP zeer nuttig zijn: de Luminex GigaCore switches presteren maar liefst 100 keer beter dan de TP-Link switches op gebied van RSTP recovery.

Het verschil in afhandeling tussen broadcast en multicast data is ook duidelijk zichtbaar in de resultaten. Na het uitvoeren van deze metingen op V2.0.0_RC1 zijn er nog meerdere updates geweest aan de GigaCore firmware, maar was het Atlys FPGA bord niet meer beschikbaar om opnieuw te kunnen meten.

In bijlage 2 vindt u de volledige meetresultaten.

3.6 Toekomstige uitbreidingsmogelijkheden

Door deze applicatie te draaien op een FPGA met 2 netwerkaansluitingen, zou de resolutie van de test verbeterd kunnen worden. Niet enkel kan dan gemeten worden hoeveel pakketten er verloren gaan, maar ook de verwerkingstijd van pakketten kan dan nauwkeurig gemeten worden.

De huidige ontvang applicatie op de computer kan niet overweg met out-of-order pakketten, omdat het loggen van ieder ontvangen pakket veel geheugen zou vragen. Out-of-order pakketten worden nu door de applicatie aangemerkt als verloren. Een mogelijke uitbreiding is het opslaan van een beperkt aantal pakketten, zodat deze out-of-order ontvangen kunnen worden.

4. Geautomatiseerde updateprocedure

4.1 Inleiding

Vooraleer een toestel uit het magazijn naar een klant verzonden wordt, wordt de firmware er van bijgewerkt naar de laatste versie. Deze update gebeurt door middel van een upload naar een webpagina ingebouwd in het toestel.

De magazijnmedewerker heeft hier relatief veel tijd voor nodig: toestel aansluiten op de computer, surfen naar het IP adres, firmware uploaden, update verifiëren… Met een kleine applicatie kan dit werk vereenvoudigd worden.

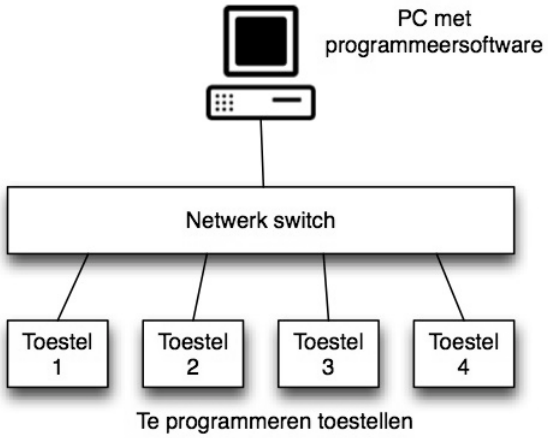
Ik merkte gedurende de stage op dat dit erg omslachtig was, en stelde voor om dit te verbeteren. Omdat ik voor de testprocedure veel ervaring heb opgedaan met de GigaCore reeks, heb ik in eerste instantie dit programma ook ontwikkeld voor de GigaCore toestellen. Door de modulaire opbouw van het testprogramma was het mogelijk om grote delen code te hergebruiken.

4.2 Uitwerking

Het programma vraagt van de gebruiker een lijst met serienummers van de te updaten toestellen. Doordat het serienummer ook als barcode aanwezig is op ieder toestel, is het ook mogelijk om de barcodes in te scannen.

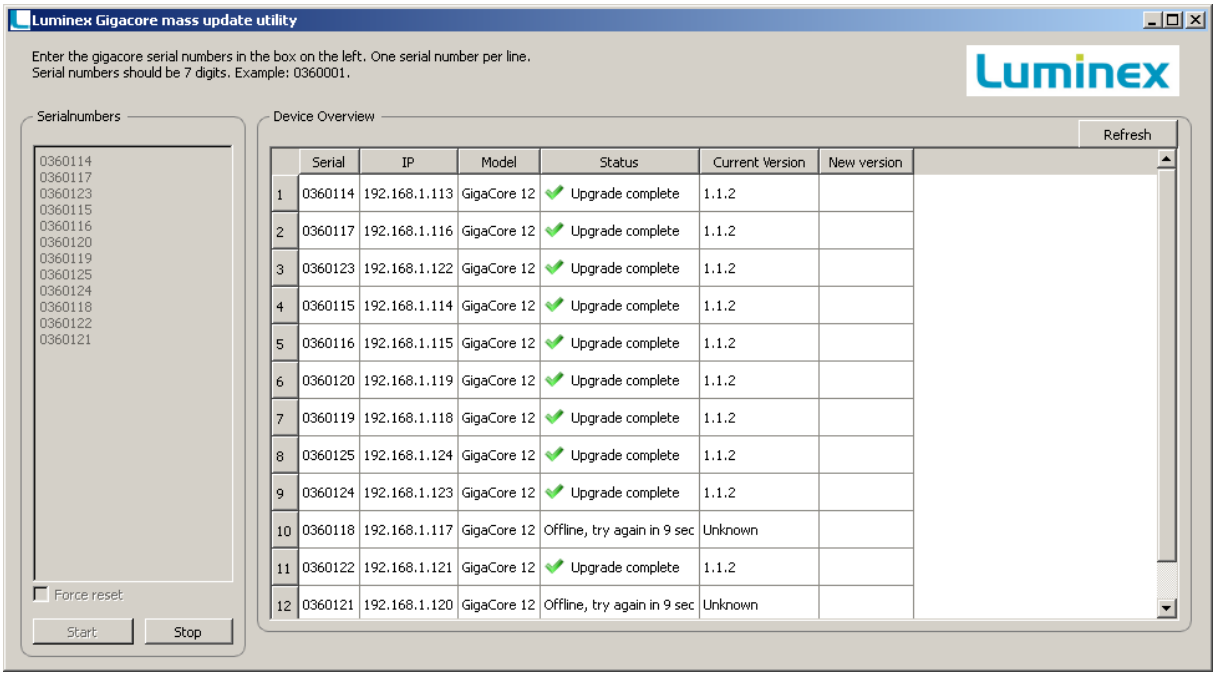
Uit deze serienummers wordt berekend om welk type toestel het gaat en welk IP adres ze hebben. Deze berekening is typerend voor iedere productreeks. Hiervoor worden modules herbruikt uit het test- en programmeersysteem dat reeds besproken is. De applicatie test of een toestel-specifieke module een bepaald serienummer herkent. Als de module het serienummer niet herkent, wordt het serienummer geprobeerd bij de andere modules. Als geen enkele module het serienummer herkent, wordt dit aan de gebruiker gerapporteerd.

De gebruiker sluit alle toestellen aan op een netwerkswitch, voorziet ze van stroom, en het programma blijft het netwerk afzoeken naar de opgegeven serienummers. Van zodra een toestel online is, zal de updateprocedure beginnen.



Afbeelding 13: Aansluitschema firmware update

Hoewel bij voorbeeld de GigaCore switches beschikken over meerdere netwerkpoorten, is het niet toegestaan om deze te upgraden als ze met elkaar zijn doorgelust. Als de upgradeprocedure start, wordt de werking van de netwerkswitch onderbroken en zou de procedure voor andere toestellen kunnen mislukken. De applicatie zal wel verschillende pogingen ondernemen om een toestel te upgraden.

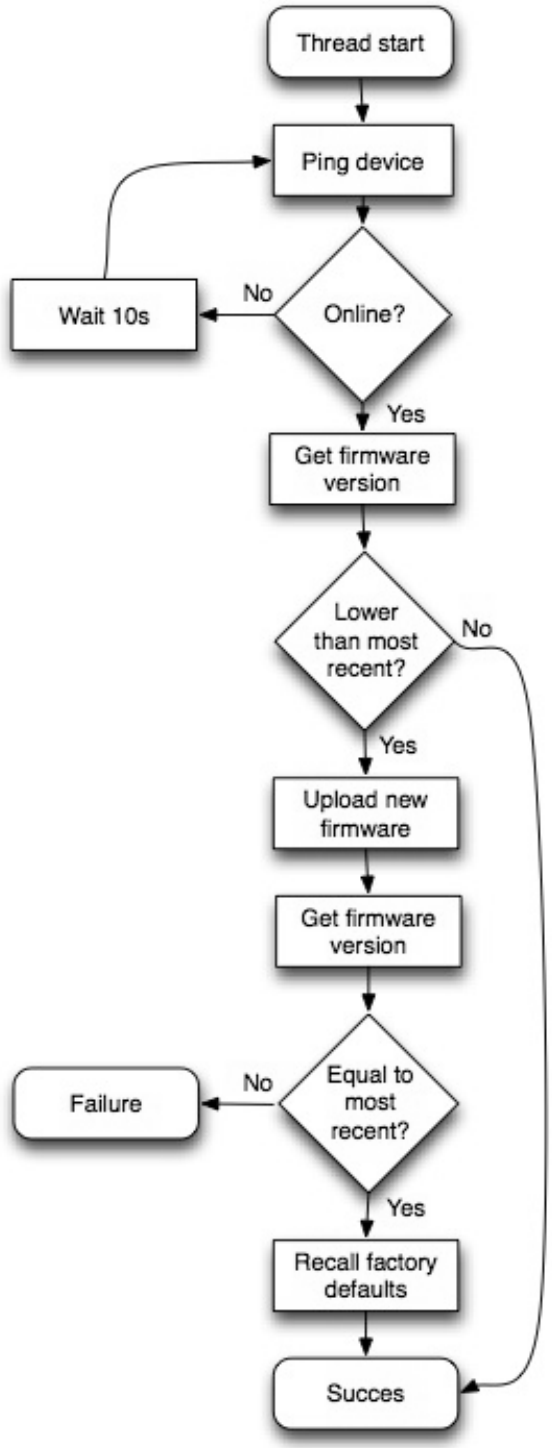


Afbeelding 14: Screenshot firmware update tool

Ook dit programma is net zoals het testsysteem modulair opgebouwd, en gebruikt dezelfde modules. Daardoor is ook dit eenvoudig uit te breiden naar andere productseries.

Deze applicatie voorziet ook in de nodige logging waardoor het mogelijk is om te controleren welke toestellen er geupgrade zijn, en te verifiëren dat er geen fouten zijn opgetreden.

Voor ieder ingevoerd toestel wordt er een instantie gestart van volgende procedure:



Afbeelding 15: Flowchart firmware update tool

4.3 Toekomstige uitbreidingsmogelijkheden

Dit programma kan in de toekomst uitgebreid worden om alle productreeksen van Luminex te ondersteunen, en automatisch firmware updates van hun website te downloaden. Dit biedt klanten dan een eenvoudige manier om al hun toestellen te updaten. In het ideale geval zou het programma het netwerk moeten scannen om toestellen te ontdekken, zodat het invoeren van serienummers niet meer nodig is. De

applicatie zou ook ondersteuning kunnen bieden voor verschillende firmware versies, zodat het bij voorbeeld mogelijk is om terug te rollen naar een oudere versie.

De huidige applicatie verwacht een lijst met serienummers, en als men op start klikt wordt deze lijst geblokkeerd. In een toekomstige versie zou deze lijst dynamischer moeten werken: de gebruiker moet eenvoudig toestellen kunnen toevoegen aan de wachtrij, zelfs als de applicatie bezig is met updaten.

Omdat de gehele applicatie ontworpen is in een QWidget, zou deze ook opgenomen kunnen worden in het bestaande LumiNetMonitor programma.

5. Besluit

5.1 Gerealiseerde doelen

5.1.1 Generiek systeem

Er is een systeem uitgewerkt dat door middel van een flexibele pluginstructuur eenvoudig nieuwe productreeksen kan ondersteunen. Er dienen geen wijzigingen te gebeuren aan de hostapplicatie, enkel een plugin dient toegevoegd te worden.

5.1.2 Schaalbaar systeem

Mede door de generieke opbouw en de nodige oplettendheid bij het ontwerpen, is het systeem opgebouwd om zo schaalbaar mogelijk te zijn. Het toevoegen van enkele regels in het XML configuratiebestand is al voldoende om verschillende componenten op te schalen. De plugins worden ontworpen om ondersteuning te bieden aan verschillende modellen uit 1 productreeks. Deze modellen hebben normaalgezien de nodige gemeenschappelijke eigenschappen (zoals het aantal poorten) zodat de test- en programmeerprocedures eenvoudig schaalbaar opgesteld kunnen worden.

5.1.3 Testsysteem

Er is een gebruiksvriendelijk systeem ontwikkeld dat ondersteuning bied om verschillende types toestellen te testen- en programmeren in productie. Door het flexibele pluginsysteem kan dit eenvoudig uitgebreid worden naar andere productreeksen.
De applicatie maakt gebruik van een grafische gebruikersinterface, en is zo opgebouwd om bruikbaar te zijn zonder intensieve training.

5.1.4 RSTP Karakterisatie

Er is een testprocedure uitgewerkt om te meten hoe goed de RSTP implementatie van een netwerkswitch presteert. Deze test is gebaseerd op een constante datastroom vanuit een FPGA. Het blijkt dat de prestaties van de geteste beta firmware een beetje slechter zijn dan de huidige release versie. De resultaten van deze test zijn gerapporteerd aan het ontwikkelaarsteam.

5.1.5 Geautomatiseerde upgradeprocedure

Er is een geautomatiseerd systeem uitgewerkt dat eenvoudig parallel de firmware van verschillende Luminex toestellen kan upgraden. De applicatie automatiseert de acties van een gebruiker, en kan daardoor heel wat werk (en tijd) besparen.

Bronnen

Lippman, S., Lajoie, J., Moo, B. (1998) C++ Primer. Addison-Wesley

Williams, J. (1 juni 2014). Digilent Atlys Resources: UDP Ethernet example. http://www.joelw.id.au/FPGA/DigilentAtlysResources

AQW Network Services. Multicast IP - MAC Address Converter. http://nettools.aqwnet.com/ipmaccalc/

Digia plc (2013). Qt Reference guide. http://qt-project.org/doc/qt-5/index.html

Pustylnik, M., Zafirovic-Vukotic, M., Moore, R., RuggedCom Inc. Performance of the Rapid Spanning Tree Protocol in Ring Network Topology.

Bijlagen

Bijlage 1: XML instellingenbestand van de GigaCore plugin

Dit is het XML bestand dat de instellingen aanbiedt voor de GigaCore plugin. Om hier geen ellenlange code te moeten plaatsen, zijn alle modellen buiten de GigaCore 12 en GigaCore 16Xt verwijderd uit het bestand.

```
<?xml version="1.0" encoding="UTF-8" ?>
<data version="1.1.1">
	<model name="GigaCore 12" sysname="GIGACORE_12">
		<description>Managed switch GigaCore 12</description>
		<image>images/GigaCore12.png</image>
		<numports>12</numports>
		<serialstart>036</serialstart>
		<programming enabled="true" default="true">
			<param name="upfile">gigacore.ppr</param>
			<param name="params">params_gc12.lst</param>
			<param name="firmware">GigaCore12v1_1_1.gz</param>
			<param name="redboot">redboot.bin</param>
			<param name="tmpfile">gigacore.bin</param>
			<os name="win" enabled="true" />
			<os name="mac" enabled="false" />
			<os name="linux" enabled="false" />
		</programming>
		<testing enabled="true" default="true">
			<step name="serial" enabled="false" />
			<step name="leds" enabled="true" />
			<step name="psu" enabled="false" />
			<step name="display" enabled="false" />
			<step name="fan" enabled="true" />
			<stepparam name="fan1_speed" value="0" />
			<stepparam name="fan1_minreadvalue" value="0" />
			<stepparam name="fan1_maxreadvalue" value="0" />
			<stepparam name="fan2_speed" value="60" />
			<stepparam name="fan2_minreadvalue" value="1000" />
			<stepparam name="fan2_maxreadvalue" value="5000" />
			<step name="reset" enabled="true" />
```

```
                <step name="speeds" enabled="true" />
                <step name="I2C" enabled="false" />
                <step name="port" enabled="true" />
                <step name="poe" enabled="true" />
                <stepparam name="poe_minpower" value="5" />
                <stepparam name="poe_maxpower" value="9" />
                <os name="win" enabled="true" />
                <os name="mac" enabled="true" />
                <os name="linux" enabled="true" />
                <testboard ip="10.0.0.1" />
                <testboard ip="10.0.0.2" />
                <testboard ip="10.0.0.3" />
                <testboard ip="10.0.0.4" />
                <testboarddev ip="192.168.0.1" />
                <testboarddev ip="192.168.0.5" />
                <testboarddev ip="192.168.0.2" />
                <testboarddev ip="192.168.0.6" />
                <testboarddev ip="192.168.0.3" />
                <testboarddev ip="192.168.0.7" />
                <testboarddev ip="192.168.0.4" />
                <testboarddev ip="192.168.0.8" />
                <testboarddev ip="192.168.0.10" />
                <testboarddev ip="192.168.0.13" />
                <testboarddev ip="192.168.0.9" />
                <testboarddev ip="192.168.0.14" />
                <testboarddev ip="192.168.0.11" />
                <testboarddev ip="192.168.0.15" />
                <testboarddev ip="192.168.0.12" />
                <testboarddev ip="192.168.0.16" />
                <serialport port="tyyUSB0" baud="115200" />
                <login username="admin" password="" />
        </testing>
        <protocoltesting enabled="false" default="false">

        </protocoltesting>
</model>
```

```
<model name="GigaCore 16Xt" sysname="GIGACORE_16Xt">
        <description>Managed switch GigaCore 16Xt</description>
        <image>images/GigaCore16Xt.png</image>
        <numports>16</numports>
        <serialstart>048</serialstart>
        <programming enabled="true" default="true">
                <param name="upfile">gigacore.ppr</param>
                <param name="params">params_gc16Xt.lst</param>
                <param name="firmware">GigaCore16Xtv1_1_1.gz</param>
                <param name="redboot">redboot.bin</param>
                <param name="tmpfile">gigacore.bin</param>
                <os name="win" enabled="true" />
                <os name="mac" enabled="false" />
                <os name="linux" enabled="false" />
        </programming>
        <testing enabled="true" default="true">
                <step name="serial" enabled="true" />
                <step name="leds" enabled="true" />
                <step name="psu" enabled="true" />
                <step name="display" enabled="true" />
                <step name="fan" enabled="true" />
                <stepparam name="fan1_speed" value="60" />
                <stepparam name="fan1_minreadvalue" value="1000" />
                <stepparam name="fan1_maxreadvalue" value="5000" />
                <stepparam name="fan2_speed" value="60" />
                <stepparam name="fan2_minreadvalue" value="1000" />
                <stepparam name="fan2_maxreadvalue" value="5000" />
                <step name="reset" enabled="true" />
                <step name="speeds" enabled="true" />
                <step name="I2C" enabled="true" />
                <step name="port" enabled="true" />
                <step name="poe" enabled="true" />
                <stepparam name="poe_minpower" value="5" />
                <stepparam name="poe_maxpower" value="9" />
                <os name="win" enabled="true" />
                <os name="mac" enabled="true" />
```

```
                    <os name="linux" enabled="true" />
                    <testboard ip="10.0.0.1" />
                    <testboard ip="10.0.0.2" />
                    <testboard ip="10.0.0.3" />
                    <testboard ip="10.0.0.4" />
                    <testboarddev ip="192.168.0.1" />
                    <testboarddev ip="192.168.0.5" />
                    <testboarddev ip="192.168.0.2" />
                    <testboarddev ip="192.168.0.6" />
                    <testboarddev ip="192.168.0.3" />
                    <testboarddev ip="192.168.0.7" />
                    <testboarddev ip="192.168.0.4" />
                    <testboarddev ip="192.168.0.8" />
                    <testboarddev ip="192.168.0.10" />
                    <testboarddev ip="192.168.0.13" />
                    <testboarddev ip="192.168.0.9" />
                    <testboarddev ip="192.168.0.14" />
                    <testboarddev ip="192.168.0.11" />
                    <testboarddev ip="192.168.0.15" />
                    <testboarddev ip="192.168.0.12" />
                    <testboarddev ip="192.168.0.16" />
                    <serialport port="tyyUSB0" baud="115200" />
                    <login username="admin" password="" />
                </testing>
                <protocoltesting enabled="false" default="false">
                </protocoltesting>
        </model>
</data>
```

Bijlage 2: RSTP karakterisatie meetresulaten

Dit zijn de resultaten van de metingen die uitgevoerd zijn om de RSTP recovery tijd te meten.

Alle metingen zijn uitgevoerd met een UDP pakketgrootte van 96 bytes.

2 GigaCores, V1.1.2, broadcast traffic

Attempt	#PKTs Lost Disconnect	#PKTs Lost Reconnect	Disconnect drop [ms]	Reconnect drop [ms]	PKT Time [ms]
1	389	237	7,78	4,74	0,02
2	390	241	7,8	4,82	0,02
3	389	242	7,78	4,84	0,02
4	385	372	7,7	7,44	0,02
5	392	242	7,84	4,84	0,02
6	386	372	7,72	7,44	0,02
7	389	373	7,78	7,46	0,02
8	388	242	7,76	4,84	0,02
9	401	242	8,02	4,84	0,02
10	389	236	7,78	4,72	0,02
		Average:	**7,796**	**5,598**	
		STD Dev:	0,083330667	1,210931873	

2 GigaCores, V1.1.2, multicast traffic

Attempt	#PKTs Lost Disconnect	#PKTs Lost Reconnect	Disconnect drop [ms]	Reconnect drop [ms]	PKT Time [ms]
1	497	606	9,94	12,12	0,02
2	495	428	9,9	8,56	0,02
3	500	426	10	8,52	0,02
4	504	426	10,08	8,52	0,02
5	500	429	10	8,58	0,02
6	500	426	10	8,52	0,02
7	499	590	9,98	11,8	0,02
8	494	426	9,88	8,52	0,02
9	496	427	9,92	8,54	0,02
10	502	590	10,04	11,8	0,02
		Average:	**9,974**	**9,548**	
		STD Dev:	0,060033324	1,546433316	

2 GigaCores, V2.0.0_RC1, broadcast traffic

Attempt	#PKTs Lost Disconnect	#PKTs Lost Reconnect	Disconnect drop [ms]	Reconnect drop [ms]	PKT Time [ms]
1	534	235	10,68	4,7	0,02
2	536	235	10,72	4,7	0,02
3	535	676	10,7	13,52	0,02
4	535	239	10,7	4,78	0,02
5	536	644	10,72	12,88	0,02
6	535	142	10,7	2,84	0,02
7	532	875	10,64	17,5	0,02
8	540	846	10,8	16,92	0,02
9	533	140	10,66	2,8	0,02
10	540	1186	10,8	23,72	0,02
11	540	680	10,8	13,6	0,02
12	555	144	11,1	2,88	0,02
13	532	736	10,64	14,72	0,02
14	536	740	10,72	14,8	0,02
15	547	678	10,94	13,56	0,02
		Average:	**10,75466667**	**10,928**	
		STD Dev:	0,119212229	6,368266326	

2 GigaCores, V2.0.0_RC1, multicast traffic

Attempt	#PKTs Lost Disconnect	#PKTs Lost Reconnect	Disconnect drop [ms]	Reconnect drop [ms]	PKT Time [ms]
1	669	806	13,38	16,12	0,02
2	673	803	13,46	16,06	0,02
3	675	1461	13,5	29,22	0,02
4	667	804	13,34	16,08	0,02
5	670	850	13,4	17	0,02
6	665	847	13,3	16,94	0,02
7	734	846	14,68	16,92	0,02
8	666	297	13,32	5,94	0,02
9	666	1283	13,32	25,66	0,02
10	668	807	13,36	16,14	0,02
		Average:	**13,506**	**17,608**	
		STD Dev:	0,395984848	5,883918422	

2x TP-Link TL-SG3216, broadcast traffic

Attempt	#PKTs Lost Disconnect	#PKTs Lost Reconnect	Disconnect drop [ms]	Reconnect drop [ms]	PKT Time [ms]
1	58344	17774	1166,88	355,48	0,02
2	37509	13593	750,18	271,86	0,02
3	69650	13585	1393	271,7	0,02
4	60914	14435	1218,28	288,7	0,02
5	58782	22799	1175,64	455,98	0,02
6	43422	13605	868,44	272,1	0,02
7	60073	13836	1201,46	276,72	0,02
8	72688	10494	1453,76	209,88	0,02
9	68690	11959	1373,8	239,18	0,02
10	42560	20313	851,2	406,26	0,02
		Average:	**1145,264**	**304,786**	
		STD Dev:	231,9153769	73,02836109	

2x TP-Link TL-SG3216, multicast traffic

Attempt	#PKTs Lost Disconnect	#PKTs Lost Reconnect	Disconnect drop [ms]	Reconnect drop [ms]	PKT Time [ms]
1	59612	27735	1192,24	554,7	0,02
2	77806	21498	1556,12	429,96	0,02
3	38202	16398	764,04	327,96	0,02
4	56532	16399	1130,64	327,98	0,02
5	62059	18323	1241,18	366,46	0,02
6	83317	17376	1666,34	347,52	0,02
7	46375	18323	927,5	366,46	0,02
8	57418	13115	1148,36	262,3	0,02
9	53040	23758	1060,8	475,16	0,02
10	75852	24619	1517,04	492,38	0,02
		Average:	**1220,426**	**395,088**	
		STD Dev:	270,8438882	85,50640313	

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Ontwerpen van een schaalbaar en generiek testplatform voor managed netwerk switches

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2014**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Sontrop, Stijn

Datum: **6/06/2014**