ISSN 1230-3801

Zeszyt 158 nr 3-4/2021, str. 49- 63
*Volume 158 No. 3-4/2021, pp. 49-63*

# PLATFORMA SPRZĘTOWA DO OBLICZEŃ KWANTOWYCH

## *HARDWARE FOR QUANTUM COMPUTATIONS*

Tomasz KUCZERSKI
Wojskowy Instytut Techniczny Uzbrojenia, ul. Wyszyńskiego 7, 05-220 Zielonka
*Military Institute of Armament Technology, 7 Wyszynski St., 05-220 Zielonka, Poland*
*Author's e-mail address: kuczerskit@witu.mil.pl.;ORCID: 0000-0001-7595-726X*

Włodzimierz MIKKE

DOI 10.5604/01.3001.0015.6776

**Streszczenie:** Opracowanie zawiera przykładową realizację platformy sprzętowej przeznaczonej do symulacji wybranych elementarnych obliczeń kwantowych. Opisano podstawowe karty zestawu Sinara wraz z oprogramowaniem oraz platformą do eksperymentów z dziedziny fizyki kwantowej oraz obliczeń kwantowych. Przedstawiono przykładowe zastosowania platformy sprzętowej łącznie z instrukcją uruchomienia podstawowych algorytmów kwantowych.

Słowa kluczowe: obliczenia kwantowe, symulacja zjawisk fizycznych, kwantowa platforma sprzętowa

**Abstract:** An exemplary hardware platform for simulation of some elementary quantum computations is presented in the paper. Basic cards of Sinara system with software and platform for experiments in domain of quantum physics and computations were described. Some exemplary applications of the hardware platform and instruction for starting up basic quantum algorithms are presented.

Keywords: quantum computations, simulation of physical effects, quantum hardware platform

## 1. Wstęp

Sinara jest otwarto-źródłowym ekosystemem sprzętowym projektowanym do użycia w eksperymentach oraz w zastosowaniach fizyki kwantowej. Przewagami jakie oferuje to rozwiązanie jest kolaboracyjna praca ośrodków o wielu potrzebach, dostęp do wszystkich schematów oraz szerokie pole zastosowań. Płytki FPGA tego projektu są tworzone specjalnie pod potrzeby laboratoriów oraz ośrodków podejmujących pracę z tą platformą. Ostatnio cieszy się zastosowaniem w projektach z kompu-

## 1. Introduction

Sinara is an open-source hardware ecosystem designed for experiments and applications on quantum physics. Benefits offered by the solution concern collaboration of many centres with different demands, access to all diagrams and a wide range of applications. Boards FPGA of this project are created especially for demands of laboratories and centres working on this platform. It has been used recently for projects on both hyper-conductive [1]

terami kwantowymi zarówno nadprzewodnikowymi [1] jak i z pułapkami jonowymi. Wszelkie schematy, dane techniczne jak i dyskusje na temat rozwijania projektu oraz pytania techniczne znajdują się na repozytorium: https://github.com/sinara-hw.

## 2. System kontroli ARTIQ

Używanym powszechnie system do komunikacji z płytkami jest system kontroli ARTIQ (ang. Advanced Real-Time Infrastructure for Quantum physics), otwarto-źródłowy projekt rozpoczęty w NIST. Jest on językiem programowania wysokiego stopnia, opartym na języku Python, który jest kompilowany i wykonywany na dedykowanych płytkach FPGA z nanosekundową zdolnością rozdzielczą i krótszym niż mikrosekunda opóźnieniem. Główną przewagą używania tego systemu jest tzw. protokół DRTIO (Distributed Real Time IO), który dostarcza giga-bitową łączność i dystrybucję czasu przez kabel miedziany lub światłowód. Dzięki temu protokołowi możliwa jest synchronizacja zegarów na wszystkich urządzeniach, zapewniając deterministyczną relację faz oraz zapewnia nanosekundową rozdzielczość dla wydarzeń wejścia/wyjścia na wszystkich urządzeniach w eksperymencie.

## 3. Zestaw kart Sinara

Przedstawiamy opisy modułów znajdujących się na stanowisku pracy, które w dużej mierze opiera się o ww. repozytorium, gdzie odsyłamy po bardziej szczegółowe dane techniczne. Następnie prezentujemy skrócony opis instalacji potrzebnych komponentów oraz opis dysku USB z zainstalowanym systemem Ubuntu. Na końcu przedstawiamy podstawowe funkcje korzystania z zestawu.

Moduł Kasli może być uznany za najważniejszy w zestawieniu sprzętowym, ponieważ

and ion trap quantum computers. All diagrams, technical data, and discussions concerning development of the project, and technical questions are placed on repositorium: https://github.com/sinara-hw.

## 2. ARTIQ Control System

Control system ARTIQ (Advanced Real-Time Infrastructure for Quantum Physics) is commonly used to communicate with the boards as an open-source project started in NIST. It is a high level programming language based on Python language which is compiled and realised on dedicated boards FPGA with a nanosecond resolution and a delay below microsecond. The main advantage of using this system is the so called protocol DRTIO (Distributed Real Time IO) providing a gigabyte communication and distribution of time via a copper wire or an optical fibre. Due to this protocol the timing of clocks is possible for all devices to secure deterministic realisation of phases and nanosecond resolution of events on input/output for all devices of the system taking part in the experiment.

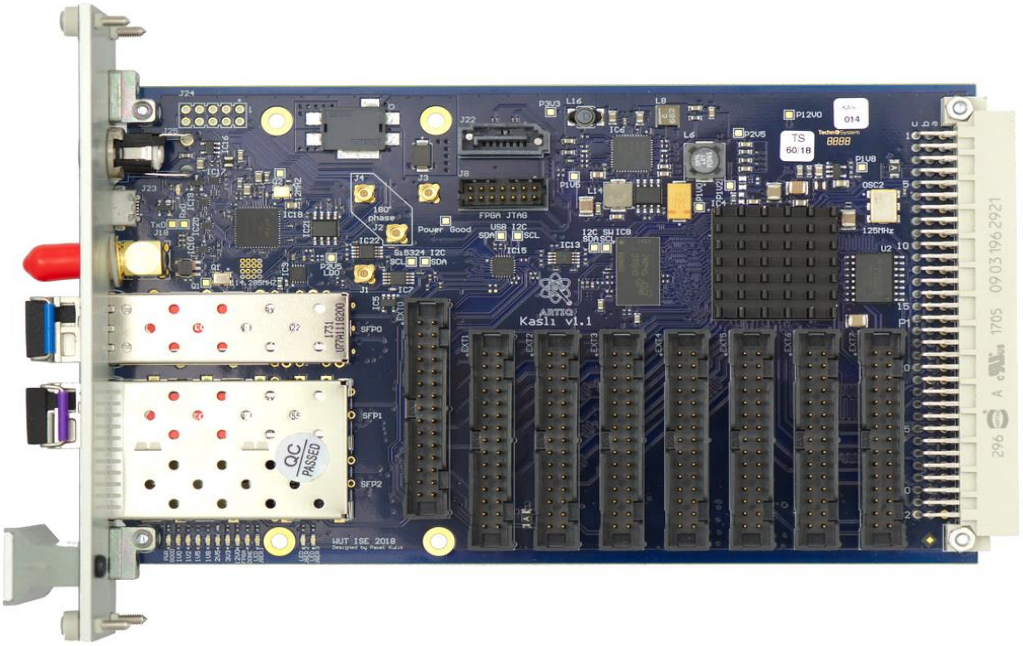
## 3. Set of Sinara Cards

Here descriptions of modules present in the workplace are included on the base of the above mentioned repositorium where more detailed technical data can be available. Next, a short description is presented for installation of needed components and for USB disc with installed Ubuntu system. At the end the main functions for using the system are presented.

The Kasli module may be considered a most important one in the hardware because all other modules will be connected

do niego będą podłączane wszystkie pozostałe moduły. W wersji 2. zawiera 12 wtyków mogących kontrolować moduły rozszerzenia (ang. Eurocard Extension Modules - EEM). Może funkcjonować samodzielnie lub w połączeniu z drugim modułem Kasli w roli "master" lub "satelite".

to it. Version 2 contains 12 ports for controlling the extension modules (Eurocard Extension Modules - EEM). It may work independently, or in connection with the second Kasli module as a "master" or "satellite".




**Fot. 1.1. Moduł Kasli v1.1**
***Photo. 1.1. Module Kasli v1.1***

Połączenia na przednim panelu:

- 3 wejścia SFP (jedno do Ethernetu, jeśli "master").
- 1 wejście micro USB, służące do wgrywania "frameworków",
- 1 wejście SMA do chipa pozyskiwania i dystrybucji zegara , używane jako odniesienie dla zegara RTIO

Połączenia EEM:

- 8 EEM (0-7) dostępne jako 30-pinowe wejścia IDC na Kasli,
- 4 EEM (8-11) dostępne na 96-pinowych wejściach DIN 41612.

Moduł DIO jest modułem rozszerzającym (EEM) dostarczającym 8 cyfrowych wejść Input-Output (IO) typu BNC na przednim pane-

Connections on the front panel:

- 3 inputs SFP (one to Ethernetu, if "master").
- 1 input for micro USB, to record "frameworks",
- 1 input SMA to chip of clock acquisition and distribution, used as a reference for RTIO clock.

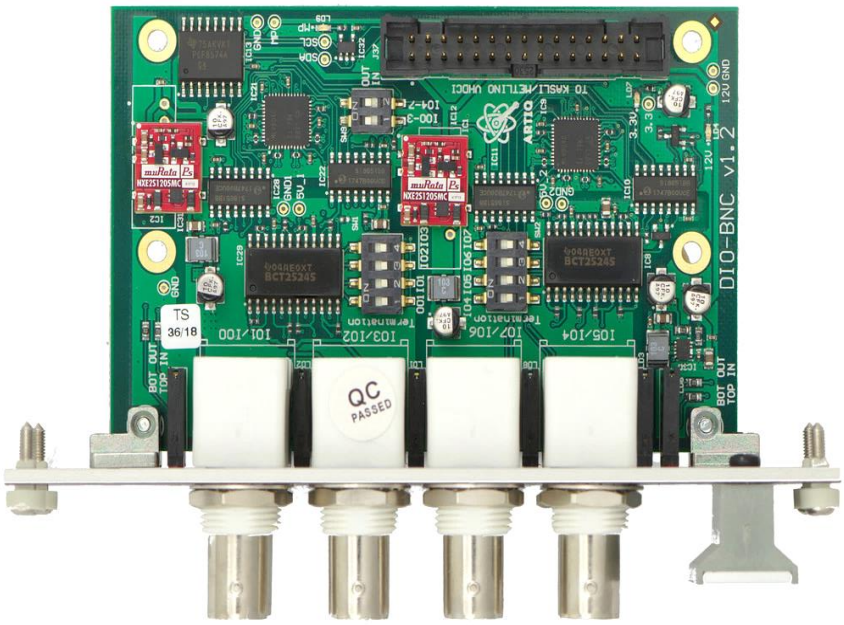
Connections EEM:

- 8 EEM (0-7) available as 30-pins inputs of IDC on Kasli,
- 4 EEM (8-11) available on 96-pin inputs DIN 41612.

Module DIO is the extension module (EEM) providing 8 digital Input-Outputs (IO) of BNC type on the front panel. It has

lu. Ma dwa rzędy po 4 kanały, z których każdy indywidualnie ma izolację uziemiającą i możliwość zmiany kierunku z użyciem przełączników na płytce lub przez I2C.

two rows of 4 channels in each one, and each of them individually has a grounding insulation, and possibility for changing direction by using switches on the board, or by I2C.




**Fot. 1.2. Moduł DIO-BNC**

***Photo. 1.2. Module DIO-BNC***

Informacje techniczne:

- wyjścia mogą dostarczyć napięcie powyżej 2V na 50 Ω obciążenie,
- impedancja wyjściowa: 50 Ω,
- tolerancja zwarcie: nieskończona,
- minimalna szerokość sygnału: 3 ns,
- maksymalnie 150 MHz częstość przełączenia z 50% cyklem pracy,
- zmiana kierunku IO w grupach po 4 kanały.

Moduł Urukul jest 3 kanałowym syntezatorem częstotliwości opartym o DDS dla użycia jako EEM. Pozwala na rozdzielczość sygnału poniżej Hz, występuje w dwóch wariantach, z różnymi procesorami: AD9912 i AD9910. Specyfikacja tego modułu jest bardzo specyficzna, ponieważ posiada on wiele parametrów zależnych od użycia. Do precyzyjnego opisu odsyłamy do strony

Technical information:

- the outputs may provide voltages above 2V on 50 Ω termination,
- output impedance: 50 Ω tolerance of short-circuiting: unlimited,
- minimal width of signal: 3 ns,
- maximal 150 MHz switching rate at 50% operation cycle,
- change of IO direction in groups of 4 channels.

Module Urukul is a three-channel frequency synthetiser based on DDS to be used as EEM. It allows for resolution of signal below Hz and exists with different processors: AD9912 and AD9910. Specification of this module is greatly exceptional as it has many parameters depending on the application. The precise description is at https://github.com/sinara-hw/Urukul/wiki.

https://github.com/sinara-hw/Urukul/wiki.

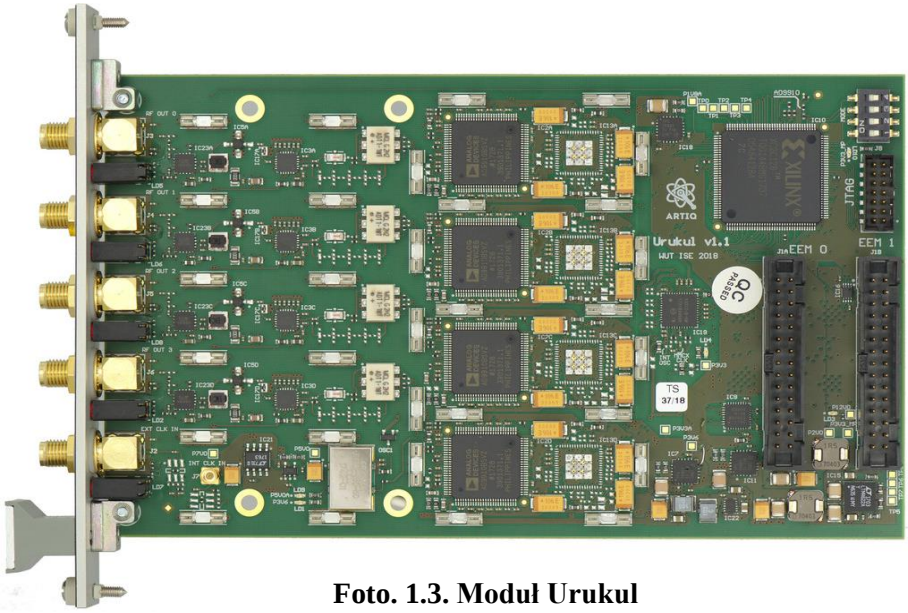



**Foto. 1.3. Moduł Urukul**

***Photo. 1.3. Module Urukul***

Informacje techniczne dla modułu z procesorem AD9910:

- częstotliwość wyjściowa (-3dB): 1Hz – 400MHz;
- rozdzielczość częstotliwości: 0.25 Hz;
- rozdzielczość offsetu fazy: Cyfrowa amplituda rozdzielczości (ASF): 14bit;
- konsumpcja mocy: 7 W.

Sinara 5432 DAC "Zotino" jest 32-kanałowym, 16-bitowym DAC EEM z częstotliwością aktualizacji 1MSPS (podzieloną na kanały). Został zaprojektowany z myślą o niskim szumie i dobrej stabilności.

Informacje techniczne:

- liczba kanałów: 32;
- rozdzielczość: 16-bit;
- częstotliwość aktualizacji: 1MSPS (może być podzielone na kanały);
- napięcie wyjściowe: +- 10V;
- impedancja wyjściowa: 470 Ω równolegle z 2.2 nF;
- konsumpcja mocy: 3 W bez obciążenia, 8,7 W z maksymalnym obciążeniem na wszystkich kanałach.

Technical information for module with AD9910 processor:

- output frequency (-3dB): 1Hz – 400MHz;
- frequency resolution: 0.25 Hz;
- phase offset resolution: digital resolution amplitude (ASF): 14bit;

power consumption: 7 W.

Sinara 5432 DAC "Zotino" is a 32-channel, 16-bits DAC EEM with actualisation rate 1MSPS (divided on channels). It was designed to have a low noise level and good stability.

Technical information:

- number of channels: 32;
- resolution: 16-bit;
- actualisation rate: 1MSPS (may be divided on channels);
- output voltage: +- 10V;
- output impedance: 470 Ω in parallel with 2.2 nF;
- power consumption: 3W without load, 8.7 W at maximal load in all channels.

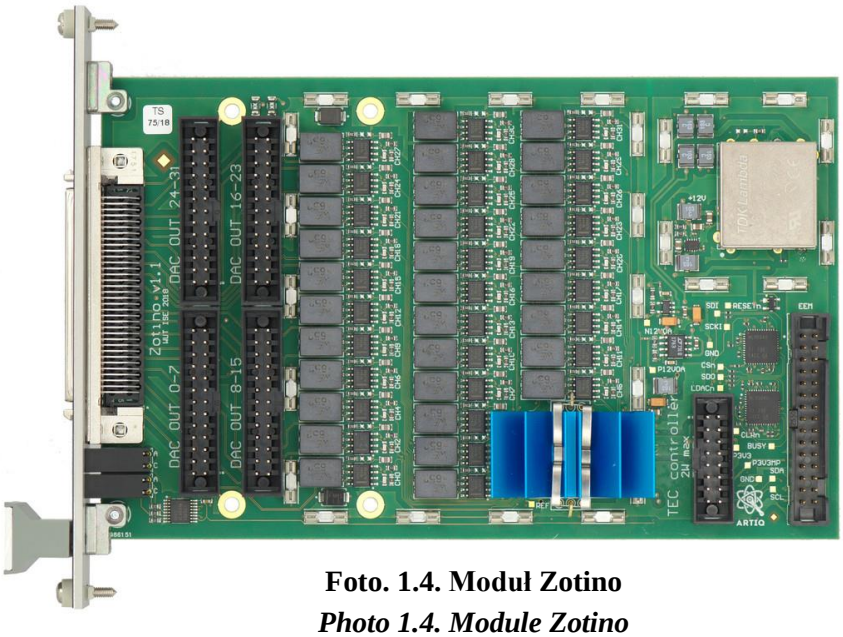



**Foto. 1.4. Moduł Zotino**
***Photo 1.4. Module Zotino***

Sinara 5108 Sampler to 8-kanałowy, 16-bitowy moduł EEM ADC z częstotliwością próbkowania 1.5MSPS (dla wszystkich kanałów jednocześnie). Wyposażony w niskoszumowy, różnicowy front-end z cyfrowo programowalnym wzmocnieniem, zapewniający zakres wejściowy od +/- 10mV (G=1000) do +/- 10V (G=1).

Sinara 5108 Sampler is an 8-channel, 16-bit EEM ADC module with the sampling rate 1.5MSPS (for all channels concurrently). It is equipped with a low-noise differential front-end with digitally programmed gain, securing the input range from +/- 10mV (G=1000) to +/- 10V (G=1).

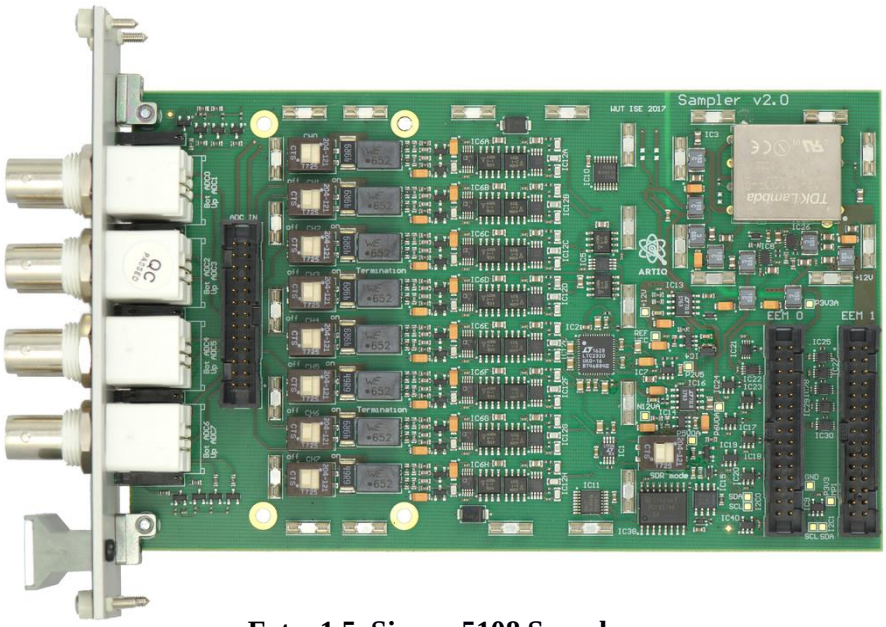



**Foto. 1.5. Sinara 5108 Sampler**
***Photo 1.5. Sinara 5108 Sampler***

Informacje techniczne:

- liczba kanałów: 8,
- rozdzielczość: 16-bitowa,
- częstotliwość próbkowania: do 1.5 MHz,
- pasmo: 200 kHz -6 dB dla G={1, 10, 100}, 90 kHz for G=1000.
- zakresy wejściowe: +-10V (G=1), +-1V (G=10), +-100 mV (G=100), +-10 mV (G=1000).

## 4. Stanowisko komputerowe

Do obsługi zestawów i przeprowadzania eksperymentów nie jest konieczne użycie komputerów o wygórowanych parametrach. Preferowane specyfikacje sprzętowe:

- współczesny system operacyjny:
  - Windows 8 lub nowszy/Mac OS 10.13+/aktualny system Linux,
- CPU x86 64-bitowy (Intel lub AMD),
- 4 GB pamięci RAM,
- 20 GB wolnej pamięci na dysku,
- wejście Ethernet lub adapter USB/Ethernet w przypadku urządzeń bez tego wejścia.

### 4.1. Oscyloskop

Do demonstracyjnej pracy, tj. bez rzeczywistego eksperymentu, wszystkie efekty pracy modułów będą obserwowane na oscyloskopie. Do pracy wystarczy oscyloskop dwukanałowy, wyposażony w generator funkcji np. Hantek DSO4084C.

### 1.2. Oprogramowanie

Do korzystania z platformy jest konieczne posiadanie odpowiedniego oprogramowania. Na dysku USB znajduje się system operacyjny Ubuntu 20.04.2.0 z zainstalowanymi wszystkimi potrzebnymi komponentami. System będzie działał na większości komputerów po podłączeniu do gniazda USB.

Technical information:

- number of channels: 8,
- resolution: 16-bit,
- sampling rate: to 1.5 MHz,
- band: 200 kHz -6dB for G={1, 10, 100}, 90 kHz for G=1000,
- input ranges: +-10V (G=1), +-1 V (G=10), +-100 mV (G=100), +-10 mV (G=1000).

## 4. Computer Setup

The handling of sets and conducting the experiments can be provided by computers with not necessarily exceptional parameters. Preferred hardware specifications:

- present operational system:
  - Windows 8 or a newer one/Mac OS 10.13+/current system Linux,
- CPU x86 64-bit (Intel or AMD),
- 4 GB RAM,
- 20 GB of disc memory,
- Input Ethernet or adapter USB/Ethernet for devices without this input.

### 4.1. Oscilloscope

For demonstrative operation, i.e. without any real experiment, all effects of operations of modules will be observed on an oscilloscope. It is sufficient to use two-channel oscilloscope equipped with a generator of functions, for instance Hantek DSO4084C.

### 4.2. Software

Adequate software is needed to use the platform. The USB disc includes operational system Ubuntu 20.04.2.0 with all needed components. The system will be active in the most computers after joining to USB port. Detailed instruction of operation is included in chapter 3.

Szczegółowa instrukcja działania znajduje się w rozdziale 3.

W przypadku konieczności osobnej instalacji na innym komputerze, będzie konieczne połączenie z Internetem, a instrukcja instalacji na komputerze znajduje się w rozdziale 3.

### 1.3. Uruchomienie stanowiska komputerowego

Instrukcja uruchamiania oprogramowania z użyciem dołączonego dysku USB:

1) Najpierw wyłączyć komputer, na którym chcemy uruchomić system.
2) Podłączyć dołączony dysk USB.
3) Włączyć komputer i od razu przejść do ustawień systemu BIOS (przez klikanie odpowiedniego klawisza na klawiaturze, który w zależności od producenta komputera może być jednym z F2, F10, F12 lub innym).
4) W ustawieniach BIOS znaleźć zakładkę BOOT i wybrać opcję uruchamiania z USB jako pierwszą.
5) Po zapisaniu zmian, komputer powinien uruchomić się ponownie i powinniśmy zobaczyć menu wyboru systemu, należy wybrać pierwszą opcję "Ubuntu" przy pomocy przycisku Enter.
6) W tym momencie sprawdzana jest poprawność plików i po krótkim czasie powinien pokazać się pulpit systemu Ubuntu.

Uruchomiona wyżej dystrybucja jest dość minimalna i zawiera: podstawowe programy systemu Ubuntu, edytor tekstowy Atom, który może być używany do czytania dołączonych kodów jak i tworzenia nowych, menadżer pakietów Nix, służący do uruchamiania wszystkich programów w języku Artiq, pełną bibliotekę języka Artiq oraz kilka pakietów pomocniczych języka Python.

Po uruchomieniu systemu można przejść do narzędzia terminal (widoczny po lewej stronie) i wpisać tam komendę $ nix-shell ar-

In the case when a separate installation in other computer is needed, then connection with interned will be also needed, and instruction of installation in the computer is in chapter 3.

### 4.3. Starting up the Computer Setup

Instruction for activation of the software by using attached USB disc:

1) First the computer where the system is to be started has to be turned off.
2) The attached USB disc has to be connected.
3) The computer has to be on, and straight away the settings of BIOS system have to be selected (by clicking a suitable key on the keyboard, which depending on computer's manufacturer may be one of F2, F10, F12 or another).
4) In settings of BIOS the folder BOOT has to be selected, and the option of starting up with the USB has to be selected as the first one.
5) After recording the changes the computer has to be restarted, and when a menu for system selection is presented the first option "Ubuntu" has to be approved by Enter.
6) In this moment the correctness of files is examined and after a while a pulpit of Ubuntu has to be presented.

The distribution activated above is relevantly short and includes: basic programs of Ubuntu system, text editor Atom, which may be used for both reading out the attached codes and creating the new ones; manager of files Nix, used for activation of all programs in Artiq language, complete library of Artiq language, and a few assisting packages of Python language.

After activation of the system it is possible to pass to the tool terminal (visible on the left side) and put there a command $ nix-

tiq_env.nix, aby zainicjować tzw. shell. Język Nix umożliwia łatwe wpisywanie pakietów języka python potrzebnych do projektu poprzez pliki z zakończeniem *.nix*. Do zmiany pożądanych pakietów można otworzyć plik artiq_env.nix w programie Atom i odkomentować (usunąć znak # przed linią) potrzebne pakiety. Następnie należy podłączyć kablem Ethernet komputer z modułem Kasli w roli master, i aby sprawdzić poprawne działanie można wykorzystać komendę $ ping 192.168.1.70, ponieważ pod tym adresem powinien znajdować się moduł Kasli. W przypadku niewykrycia przez komputer modułu należy użyć zaawansowanych ustawień połączenia i ręcznie ustawić adres IP, Maskę oraz Bramę domyślną w następujący sposób:

Adres IP: 192.168.1.7 (lub inna liczba końcowa w zależności czy działa wiele komputerów w sieci):

Maska: 255.255.255.0

Brama domyślna: 192.168.1.1

## 2. Instalacja systemu ARTIQ

Poniżej przedstawiono procedurę instalacji wszystkich koniecznych komponentów na komputerze z połączeniem internetowym. Preferowaną metodą instalacji jest przez menedżer pakietów Nix, która może zostać wykonana wyłącznie na systemach Linux.

### 5.1. Instalacja przez Anaconda (Windows/ Linux)

Windows/LinuxInstalacja anacondy:

1) pobierz plik instalacyjny ze strony: https://www.anaconda.com/products/individual#Downloads;
2) kliknij podwójnie, aby uruchomić instalator;
3) postępować zgodnie z kolejnymi krokami instalacji;
4) wybrać folder instalacyjny. Rekomendowany folder znajduje się bezpośrednio na dysku C i zawiera nazwę z 7-bitowymi

shell artiq_env.nix, to initiate the so called shell. Language Nix facilitates the writing of packages (blocks) in python language needed for the project via the files with ending *.nix*. The needed packages can be changed by opening the file artiq_env.nix in program Atom and uncomment (remove mark # before the line) the needed packages. Next, the computer has to be connected with the Kasli module via Ethernet cable, and the command $ ping 192.168.1.70 can be used to check proper operation because module Kasli has to be at this address. In the case when the computer fails to detect the module the advanced settings of connection have to be used to set by hand the IP address, Mask and facultative Gate in the following way:

Address IP: 192.168.1.7 (or another end number depending if there are other computers working in the net):

Mask: 255.255.255.0

Facultative Gate: 192.168.1.1

## 5. Installation of ARTIQ System

Below, a procedure is presented for installation of all necessary components on the computer with the internet connection. A preferable method of installation employs the manager of packages Nix which may be executed exclusively on Linux systems.

### 5.1. Installation by Anaconda (Windows/ Linux)

Windows/Linux Installation of Anaconda:

1) Take installation file from the site: https://www.anaconda.com/products/individual#Downloads;
2) Make a double click to start installer;
3) Follow consecutive steps of installation;
4) Select an installation folder. The recommended folder is just in disc C and includes the name with 7-bit marks of

znakami ASCII (ie. C:\anaconda), ale domyślny, czyli C:\Users\<nazwa-użytkownika>\Anaconda3 również jest poprawny;

5) po instalacji należy wpisać komendę w terminalu $ conda do sprawdzenia działania Anacondy.

Ustawienie kanału Conda i instalacja ARTIQa w nowym środowisku:

```
$ conda config --prepend channels https://conda.m-labs.hk/artiq-beta
$ conda config --append channels conda-forge
$ conda create -n artiq artiq
```

Aktywacja środowiska:

```
$ conda activate artiq
```

### 5.2. Instalacja przez Nix

W przypadku braku obecności menadżera pakietów nix należy najpierw go zainstalować przez komendę:

```
$ curl -L https://nixos.org/nix/install | sh
```

Kiedy Nix jest już zainstalowany należy dodać kanał pakietów:

```
$ nix-channel --add https://nixbld.m-labs.hk/channel/custom/artiq/full-beta/artiq-full
```

Ponieważ domyślna instalacja używa deweloperskiej wersji nixpkgs, należy zmienić na inne wydanie:

```
$ nix-channel --remove nixpkgs
$ nix-channel --add https://nixos.org/channels /nixos-20.09 nixpkgs
```

Ostatecznie stosujemy wszystkie zmiany:

```
$ nix-channel --update
```

Nix nie zainstaluje pakietów bez sprawdzania kryptograficznego podpisu. Należy dodać klucz publiczny M-Labs tworząc plik ~/.config/nix/nix.conf z poniższą zawarto-

ASCII (i.e. C:\anaconda), but a facultative one, i.e. C:\Users\<name of the user >\Anaconda3 is also a correct one;

5) After installation the command has to be entered into the terminal $ conda to check the operation of Anaconda.

Setting the Conda channel and installation ARTIQ in the new environment:

```
$ conda config --prepend channels https://conda.m-labs.hk/artiq-beta
$ conda config --append channels conda-forge
$ conda create -n artiq artiq
```

Activation of the environment:

```
$ conda activate artiq
```

### 2.2. Installation through Nix

In the case when the manager is missing it has to be installed first by command:

```
$ curl -L https://nixos.org/nix/install | sh
```

When the Nix is already installed then the channel of packages has to be added:

```
$ nix-channel --add https://nixbld.m-labs.hk/channel/custom/artiq/full-beta/artiq-full
```

As the facultative installation uses a developer's version of nixpkgs, it has to be changed into another edition:

```
$ nix-channel --remove nixpkgs
$ nix-channel --add https://nixos.org/channels /nixos-20.09 nixpkgs
```

Finally we use all changes:

```
$ nix-channel --update
```

Nix cannot install the packages without checking the cryptographic signature. The public key M-Labs has to be added by creating the file ~/.config/nix/nix.conf with

ścią:

```
    substituters = https://cache.nixos.org https://nixbld.m-labs.hk
    trusted-public-keys = cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY= nixbld.m-labs.hk-1:5aSRVA5b320xbNvu30tqxVPXpld73bhtOeH6uAjRyHc=
```

Do instalacji ARTIQ należy stworzyć w oddzielnym folderze bądź folderze domowym plik o nazwie np. my-artiq-env.nix z zawartością:

```
let
  # pkgs zawiera zbiór pakietów NixOS i ARTIQ zależy od niektórych z nich, więc
  # dodatkowe pakiety mogą być potrzebne stamtąd.
  pkgs = import <nixpkgs> {};
  artiq-full = import <artiq-full> { inherit pkgs; };
in
  pkgs.mkShell {
    buildInputs = [
      (pkgs.python3.withPackages(ps: [
        # Lista potrzebnych bibliotek Pythona idzie tutaj.
        # Te będą nam oczywiście potrzebne.
        artiq-full.artiq
        artiq-full.artiq-comtools
        # Kolekcja NixOS zawiera inne pakiety, które mogą okazać się przydatne:
        #ps.pandas
        #ps.numpy
        #ps.scipy
        #ps.numba
        #(ps.matplotlib.override { enableQt = true; })
        #ps.bokeh
        #ps.cirq
```

the following content:

```
    substituters = https://cache.nixos.org https://nixbld.m-labs.hk
    trusted-public-keys = cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY= nixbld.m-labs.hk-1:5aSRVA5b320xbNvu30tqxVPXpld73bhtOeH6uAjRyHc=
```

To install ARTIQ a file has to be created in a separate folder, or in a home folder, named by e.g. my-artiq-env.nix containing:

```
let
  # pkgs contains a set of packages NixOS and ARTIQ depends on some of them, and then
  # additional packages may be needed from there.
  pkgs = import <nixpkgs> {};
  artiq-full = import <artiq-full> { inherit pkgs; };
in
  pkgs.mkShell {
    buildInputs = [
      (pkgs.python3.withPackages(ps: [
        # List of needed Python libraries goes here.
 # following ones will be needed, of course.
        artiq-full.artiq
        artiq-full.artiq-comtools
        # Collection NixOS includes other packages which could be useful:
        #ps.pandas
        #ps.numpy
        #ps.scipy
        #ps.numba
        #(ps.matplotlib.override { enableQt = true; })
        #ps.bokeh
        #ps.cirq
```

```
      #ps.qiskit
    ]))
    # Lista pakietów nie-Pythonowych powinna się znaleźć poniżej
    # Inne całkiem przydatne pakiety:
    #pkgs.gtkwave
    #pkgs.spyder
    #pkgs.R
    #pkgs.julia
  ];
}
```

Ostatecznie, aby uruchomić należy rozpocząć shell nixa komendą:

```
$ nix-shell my-artiq-env.nix
```

## 6. Eksperymenty

### 6.1. Eksperyment z LED

Wszystkie eksperymenty będą wykonywane przy pomocy egzekucji plików w języku Python. Do bliższego poznania korzyści płynących z użycia systemu Artiq jak RTIO odsyłamy do https://m-labs.hk/artiq/manual/rtio.html.

Wszystkie programy pisane będą na podobieństwo klasy. Jako pierwszy przykład rozważmy poniższy program:

```
from artiq.experiment import *

class LED(EnvExperiment):
    def build(self):
        self.setattr_device("core")
        self.setattr_device("led")

    @kernel
    def run(self):
        self.core.reset()
        self.led.on()
```

Należy utworzyć plik o nazwie led.py z powyższą zawartością.

Na początku importujemy wszystkie ele-

```
      #ps.qiskit
    ]))
    # A list of non-Python packages has to be found below
    # Other quite useful packages:
    #pkgs.gtkwave
    #pkgs.spyder
    #pkgs.R
    #pkgs.julia
  ];
}
```

Finally, in order to start it up, the shell nix has to be initiated by command:

```
$ nix-shell my-artiq-env.nix
```

## 6. Experiments

### 6.1. Experiment with LED

All experiments will be carried out by execution of files in Python language. To find out more about benefits from system Artiq as RTIO contact to https://m-labs.hk/artiq/manual/rtio.html.

All programs will be written in resemblance of class. Firstly, the following program will be considered:

```
from artiq.experiment import *

class LED(EnvExperiment):
    def build(self):
        self.setattr_device("core")
        self.setattr_device("led")

    @kernel
    def run(self):
        self.core.reset()
        self.led.on()
```

A file has to be created with the above content.

In the beginning we import all elements

menty z biblioteki artiq.experiment, a następnie przechodzimy do definicji klasy eksperymentu, która zawsze będzie przyjmowała argument EnvExperiment. Definiujemy funkcję build, która oznacza jakie urządzenia będą używane w eksperymencie. Drugą funkcją klasy jest run, które zawsze będzie przyjmowało obiekt. Warto zwrócić uwagę na dekorator @kernel, który mówi systemowi, że ta funkcja ma być kompilowana i egzekwowana na urządzeniu bazowym (w naszym przypadku jest nim moduł Kasli).

Przy dostawie sprzętu powinien zostać dany nośnik danych z plikiem device_db.py, który musi się znajdować w tym samym folderze co plik z kodem.

Do uruchomienia należy użyć komendy $ artiq_run led.py w terminalu z aktywowanym shellem Nix.

### 6.2. Eksperyment DRTIO

Drugi eksperyment obrazuje działanie komunikacji dwóch połączonych światłowodem modułów Kasli z podłączonymi urządzeniami w następujący sposób:

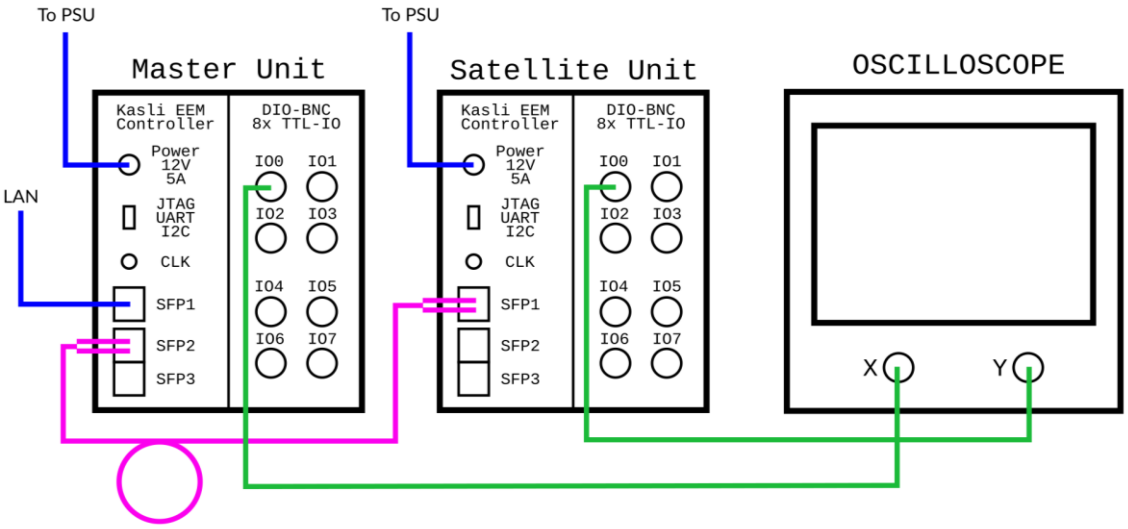
from the library artiq.experiment and next we go over to definition of the class of the experiment which always takes the argument EnvExperiment. We define function build, which says what devices will be used in the experiment. The second function of the class is run, which will always accept the object. It is worth to pay the attention to decorator @kernel, which says to the system that this function has to be compiled and executed on the base device (in our case it is Kasli module).

At the delivery of hardware a data medium has to be provided with the file device_db.py which has to be found in the same folder as the file containing the code.

To start it up the command $ artiq_run led.py has to be used in terminal with the activated shell Nix.

### 6.2. Experiment DRTIO

Second experiment illustrates communication of two Kasli modules connected by optical fibre with other connected devices in the following way:




**Rys. 4.1. Podłączenie sprzętu do eksperymentu RTIO**

***Fig. 4.1. Connections between devices in RTIO experiment***

Postępując podobnie do poprzedniego eksperymentu należy utworzyć plik z programem o nazwie np. drtio.py i uruchomić przy pomocy komendy $ artiq_run drtio.py z poniższą treścią:

```
from artiq.experiment import *

class DrtioDemo(EnvExperiment): # Definicja klasy eksperymentu

    def build(self):
    # This device will be used as core in all kernels that do not specify core explicitly
    self.setattr_device("core")
    # ttl0-ttl7 Kasli Master
    self.setattr_device("ttl0")
    # ttl8-ttl15 Kasli Satelita
    self.setattr_device("ttl8")

    @kernel
    def run(self):
    #https://m-labs.hk/artiq/manual-beta/core_drivers_reference.html?highlight=reset#artiq.coredevice.core.Core.reset
    self.core.reset()

    while True:
        # ttl8 znajduje się na urządzeniu satelicie, więc jest deterministyczne,
        # stałe opóźnienie między master i satelitą, które jest sumą:
        #  - opóźnienie komunikacji (kodowanie, dekodowanie, FIFO etc.)
        #  - długość światłowodu
        self.ttl8.on()
        # 15m światłowód
        # delay(344*ns)
        # 1m światłowód
        delay(273*ns)
        self.ttl0.on()

        delay(1000*ns)

        # Na spadającym brzegu można zauważyć nieskompensowane opóźnienie, kiedy
        # kanały są wyłączone w związku z pojedynczym licznikiem RTIO
        self.ttl8.off()
```

Acting similarly as at the former experiment a file has to be created with the program named for instance as drtio.py which has to be started up by command $ artiq_run drtio.py with the following text:

```
from artiq.experiment import *

class DrtioDemo(EnvExperiment): # Definition of experiment's class

    def build(self):
    # This device will be used as core in all kernels that do not specify core explicitly
    self.setattr_device("core")
    # ttl0-ttl7 Kasli Master
    self.setattr_device("ttl0")
    # ttl8-ttl15 Kasli Satelita
    self.setattr_device("ttl8")

    @kernel
    def run(self):
    #https://m-labs.hk/artiq/manual-beta/core_drivers_reference.html?highlight=reset#artiq.coredevice.core.Core.reset
    self.core.reset()

    while True:
        # ttl8 is in the satellite device and then it is deterministic one,
        # constant delay between master and satellite which is a sum:
        #  - delay of communication (coding, decoding, FIFO, etc.)
        #  - length of fibre optic cable
        self.ttl8.on()
        # 15m fibre optic cable
        # delay(344*ns)
        # 1m of fibre optic cable
        delay(273*ns)
        self.ttl0.on()

        delay(1000*ns)

        # On the falling down edge an uncompensated delay can be noted when
        # the channels are off in connection with singular counter RTIO
```

```
        self.ttl0.off()

        delay(1000*ns)
```

Ten przykład demonstruje w jaki sposób wykonywane są operacje na zdalnym urządzeniu oraz jak ustawić urządzenia, aby włączały sygnał jednocześnie.

Więcej przykładów wraz z objaśnieniami znajduje się pod linkiem https://m-labs.hk/artiq /manual/getting_started_core.html, a inne przykłady można znaleźć w folderze examples, gdzie zainstalowany został ARTIQ (wykonując komendę w terminalu: python3 -c "import artiq; print(artiq.__path__[0])" ).

```
        self.ttl8.off()
        self.ttl0.off()

        delay(1000*ns)
```

This example demonstrates the way in which the operations are executed on a remote device, and how to set the devices for simultaneous commutation of the signal.

More examples with comments are placed under the link https://m-labs.hk/artiq /manual/getting_started_core.html and other cases may be found in folder examples, where ARTIQ was installed (executing the command in terminal: python3 -c "import artiq; print(artiq.__path__[0])" ).

## Literatura / *Literature*

[1] Colm A. Ryan, Blake R. Johnson, Diego Ristè, Brian Donovan, and Thomas A. Ohki , "*Hardware for dynamic quantum computing*", Review of Scientific Instruments 88, 104703 (2017) https://doi.org/10.1063/1.5006525.

[2] Brkić B., Taylor S., Ralph J., and France N: High-*fidelity simulations of ion trajectories in miniature ion traps using the boundary-element method"*, Phys. Rev. A 73, 012326 – Published 20 January 2006.

[3] https://m-labs.hk/artiq/manual/

[4] https://technosystem.pl/produkty/

[5] https://github.com/sinara-hw/sinara