Proyecto Fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

# Implementation and Evaluation of an Interface between a virtual Open-Source Software Programmable Logic Controller and the Physical GPIO-Pins of the Host Machine

# Implementación y Evaluación de una Interfaz entre el Software Open-Source de un Controlador Lógico Programable Virtual y los Pines Entrada/Salida de Propósito General de la Máquina Anfitriona

Author: Ainoa Navarro Martínez

Tutors: Christian Horn and Miguel Ángel Ridao Carlini

**Institut für Werkzeugmaschinen und Fabrikbetrieb**
**Industrielle Automatisierungstechnik**
**Technische Universität Berlin**

**Dep. Ingeniería de Sistemas y Automática**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, 2017

Proyecto Fin de Grado en Ingeniería Electrónica, Robótica y Mecatrónica

# Implementation and Evaluation of an Interface between a virtual Open-Source Software Programmable Logic Controller and the Physical GPIO-Pins of the Host Machine

# Implementación y Evaluación de una Interfaz entre el Software Open Source de un Controlador Lógico Programable Virtual y los Pines Entrada/Salida de Propósito General de la Máquina Anfitriona


Author:

Ainoa Navarro Martínez

Tutors:

Christian Horn

Miguel Ángel Ridao Carlini

Institut für Werkzeugmaschinen und Fabrikbetrieb
Industrielle Automatisierungstechnik
Technische Universität Berlin

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla



Sevilla, 2017

Proyecto Fin de Carrera: Implementation and Evaluation of an Interface between virtual Open-Source Software Programmable Logic Controller and Physical GPIO-Pins of the Host Machine

Autor: Ainoa Navarro Martínez

Tutor: Miguel Ángel Ridao Carlini

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

# Gratitude

I am grateful for the opportunity offered by Christian Horn to carry out this project, as well as the support and patience he has always offered me.

I am grateful to the Technical University of Berlin for the space and necessary resources for the realization of the project, as well as all those people who have offered their help as Udo Templiner, who has always attended me when I requested help.

I thank Miguel Ángel Ridao for his advice and support to make this project abroad.

Finally, I would like to thank my family and friends, as well as José Luis Holgado Álvarez for the affection, support, patience and calm necessary to make this work possible.

*Ainoa Navarro Martínez*

*Sevilla, 2017*

# Agradecimientos

Agradezco la oportunidad brindada por Christian Horn para realizar este proyecto, así como el apoyo y la paciencia que me ha ofrecido en todo momento.

Agradezco a la Universidad Técnica de Berlín el espacio y los recursos necesarios para la realización del proyecto, así como aquellas todas personas que han ofrecido su ayuda como Udo Templiner, que siempre me ha atendido cuando he requerido su ayuda.

Agradezco a Miguel Ángel Ridao sus consejos y su apoyo para poder realizar este proyecto en el extranjero.

Por último, me gustaría agradecer a mi familia y amigos, así como a José Luis Holgado Álvarez el cariño, el apoyo, la paciencia y la calma necesarias para poder llevar acaba este trabajo.

*Ainoa Navarro Martínez*

*Sevilla, 2017*

# Abstract

The Bachelor Thesis focuses on developing an interface between virt-Soft-PLC and physical hardware GPIO-Pins of the host machine based on existing code for a different interface. For this purpose, an existing interface between virt-Soft-PLC (Awlsim) has been used as template for the new interface. As foundation of the work the mapping of the I/O between Host-Operating System (Hypervisor) and Guest-Operating System (Virtual Machine with Soft-PLC) has been configured.

The objective is the reading of a digital signal trough this port, the signal must be read in the virtual machine to be receive by the PLC software that works in this virtual machine.

To read the signal the host machine Kernel has been modified, because the project requires Real Time.

The I/O pins mapping between the Host and the Guest machine must be configured to exchange the signal between them.

# Resumen

El trabajo se centra en el desarrollo de una interfaz entre el software virtual del PLC y los pines de entrada/salida de propósito general de la máquina anfitriona, en concreto, el ordenador embebido ARK-1550, que cuenta con un puerto con pines de entrada/salida Digital. La interfaz está basada en el código de una interfaz existente llamada Awlsim.

El objetivo consiste en la lectura de una señal digital a través de dicho puerto, la señal debe ser leída en la máquina virtual para posteriormente ser recibida por el software del PLC que funciona en la misma.

Para la lectura de la señal ha sido necesario modificar el Kernel de la máquina anfitriona, puesto que el proyecto requiere el uso de tiempo real.

La configuración del mapeo de los pines de entrada y salida entre el sistema operativo de la máquina anfitriona (Hipervisor) y el sistema operativo de la máquina huésped (Máquina Virtual con el software del PLC) debe ser implementada para poder realizar el intercambio de una señal física entre ambas máquinas.

# Index

# Table Index

# Figures Index

# 1 Introduction

## 1.1 The Birth of the PLC

A PLC is a programmable logic controller, is a computer used in industrial automatization, it has unlimited applications since it can rule all the control processes in a factory or in an assembly line.

The first PLC was invented at the end of the seventies by Bedford Associates that developed Modicon (Modular Digital Controller), it purpose was substituting the old wired relay system that General Motors was using in its factories. Since that moment PLC started to spread all over the world because its multiple input output signal, which make them perfect to rule any kind of industrial process. PLCs can also rule remote process in Real Time.

The first language that PLCs uses was a list of instructions, but it was not the only developing language, ladder logic, Block diagrams or Structured text are examples of them. It also uses several types of communication protocols such as SCADA, Profibus, Ethernet, Modbus… depending on the application.

Its multipurpose, flexibility, ease, scalability and compatibility makes the PLC perfect for any kind of industrial automatization, those are the reasons for its success.

## 1.2 PLC`s Hardware

The hardware of a PLC its constituted by the power supply, the CPU, I/O Interface modules, the communications interface, the program and data memory module and the programming device.

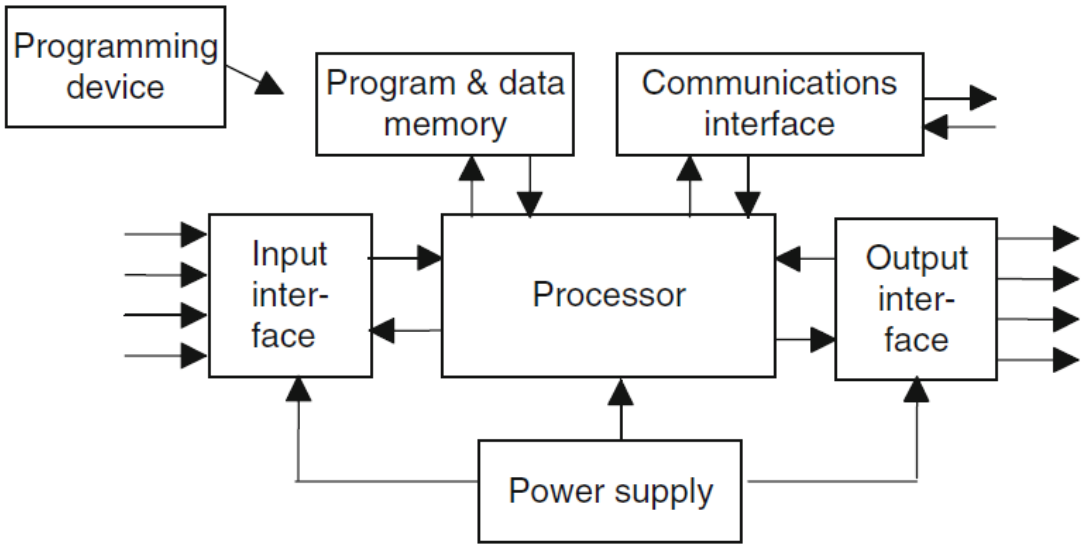


Figure 1 PLC Hardware

### Power Supply

The power supply is used to supply the required energy to the CPU as well as the rest of the I/O circuits attached to the PLC and the programming device.

### CPU

The CPU controls the RAM, counters, relays, timers … The CPU reads all the input and output signals involved in the process to run the application program.

### I/O Interface Modules

The Interface Modules allows the communication between the CPU and all the electromechanical components involved in the process using buses for that purpose, I/O Interfaces manage all the different tensions and currents of the circuits to connect all the actuators and sensors.

### Communication Interface

The communication interface sends and receives all the data the communication network that may use different communication protocols like Ethernet, Profibus …

### Program and Data Memory

The memory keeps all the information to run the control program like the process data and the control data.

A PLC has two mainly types of memories, volatile (RAM) and non-volatile memory (EPROM).

### Programming Device

The programming Device provides the interface between the PLC and the programmer, this device could be a regular computer with a dedicated software to program the PLC.

## 1.3 State of the art

The different manufacturers of each PLC develop, support and provide the required software to program and use PLCs.

Nowadays PLCs are widely spread, the applications and the electronic devices involved in the new control system are endless, so the pattern tries to coordinate all, developers need the freedom to run, distribute, copy, and modify the software without restrictions. The trend in last years is to create Open source PLC`s software, in the past PLC`s and its applications to manage them where integrated and sold by the manufacturers, also the operative system where they were running was predetermined. The operative systems required to run PLC`s software were not free, but these days the use of Linux to run SCADA or DCS system has several advantages like more security, stability as well as less costs.

There are lots of different projects like Open PLC, Awlsim, Small PLC, that are open source projects that try to develop Open Source PLC`s software as well as Hardware.

## 1.4 Motivation

PLCs are very used in the industry, to make them even more multipurpose, Open-Source and accessible to any kind of OS, computer, application or process that its already running were the motivation to develop this project.

A former project that substitutes the hardware of an old Siemens PLC to make the control application developed with a Siemens software running in the cloud, motivated me to try to send a signal to communicate a machine with an Open-Source software to program PLCs (Alwsim).

To accomplish this project a new interface that works in Real Time had to be developed, this interface is based in Awlsim. The mapping between the I/O port of the Host and the Guest machine was done to achieve the communication. The hardware as well as other resources for this project were provided by the University of Berlin.

# 2 Project Target

The target of the project consists in reading an incoming signal through a GPIO-pins port that the computer Ark-1550 has, this signal must be read by a Guest Virtual Machine running in the Host Machine installed in the computer.

To achieve the reading, a communication interface between the Host, the Guest and the GPIO-pins must be developed, for that the virt-Soft-PLC (Awlsim) interface is used as an example for developing the required interface.

As the signal needs to be read as quickly as possible, Real-time is a need, to use Real Time, the Linux kernel installed in the host and in the guest machine must be patched and modified to use Real Time.

After that the signal needs to be generated and read by a program, for that purpose user- space direct access to the pins is required to read the signal.

Once the signal is read by the host machine, the next step would be to virtualize the GPIO-pins in the guest machine to read the signal.

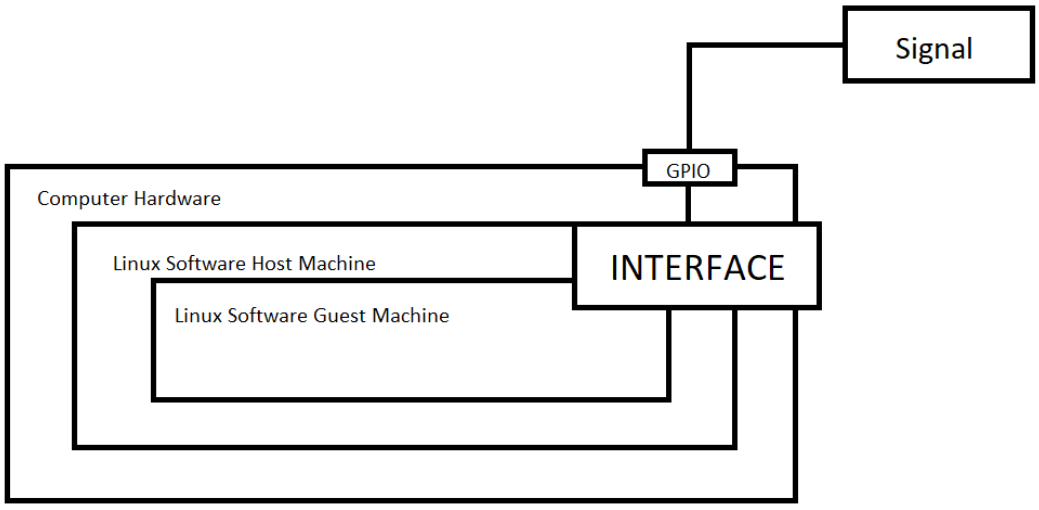


Figure 2 General Scheme

# 3 Hardware

## 3.1 The computer Ark-1550

The Ark-1550 is the computer which has been used to develop the project, this computer was chosen because of its features, specially designed for embedded applications and its Digital I/O.

The Ark-1550 is a Fanless embedded computer designed by Advantech, all the electronics are integrated in a compact and protected designed. The dimensions are 223 x 46.6 x 133.0 mm and is powered by 12V DC input, it has a single board and it offers:

- Intel® Dual Core Celeron 2980U 1.6 GHz / Core i5 4300U 1.9 GHz SoC
- One Hot Swappable 2.5" SATA HDD Bay and mSATA slot
- Triple Independent Displays by VGA + HDMI + LVDS (LVDS option)
- Optional VESA / DIN Rail Mounting kits
- Supports 2 x Intel GbE and 1 x GPIO
- Built-in 1 x full size MiniPCIe (i.e.: 3G module) and 1 x half size MiniPCIe slot (i.e.: WIFI module)
- Lockable DC jack design
- Supports iManager, SUSIAccess and Embedded Software APIs
- Support Intel Turbo Boost Technology 2.0 in Core i5 4300U

### 3.1.1 The Digital I/O

The most interesting feature for this project is the Digital I/O, it has an 8-bit DIO connector and one ground pin. Each bit can be set as a digital input or output independently. The signal will be sent through this port to communicate the hardware with the guest machine PLC software.

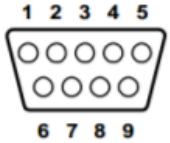


Figure 3 Digital Input/output

| Digital I/O Connector Pins | |
|---|---|
| Pin | Signal Name |
| 1 | DIO bit 4 |
| 2 | DIO bit 3 |
| 3 | DIO bit 2 |
| 4 | DIO bit 1 |
| 5 | DIO bit 0 |
| 6 | GND |
| 7 | DIO bit 5 |
| 8 | DIO bit 6 |
| 9 | DIO bit 7 |

Table 2 Digital I/O Connector Pins

The pin number six is the ground and the pins four and five will be used as an output and input respectively.

The signal must be sent through this port, the signal will be generated with a signal generator and will be sent to the pin number five, then program that was developed, will read this signal and will send it to the pin number four, so the signal can be read and showed in an oscilloscope using a test probe.

# 4 Software

## 4.1 Awlsim: S7 Compatible Soft-PLC

Awlsim is an Open-Source software to program Programmable Logic Controllers (PLCs) from Siemens which is compatible with AWL/STL Soft-PLC, it has been written in Python. Awlsim is the template to develop the new interface.

Awlsim can run on Windows as well as any other platforms that support Python, like Linux or Mac OS X.

Since Real Time is a must of this project, Awlsim can get several thousand to millions of AWL/STL instructions per second, this feature is interesting, even though the execution speed really depends on the machine where Awlsim it is being executed as well as the Python interpreter.

Awlsim installation is required because the developing of the new interface is based on it.

### 4.1.1 Installation of Awlsim

To use Awlsim-gui the Awlsim release 0.55 Source package must be downloaded from:

https://bues.ch/cms/automation/awlsim.html#download

Since the Host Machine is an Ark-1550 embedded computer from Advantech and its Operative System Ubuntu 16.04, a few additional tools were required to run Awlsim-gui, one of them is Building PySide on the system.

#### 4.1.1.1 Prerequisites

The installation of several dependencies is needed as well as the latest version of pip distribution to install Alwsim.

The first step will be the installation of the building dependencies:

```
$ sudo apt-get install build-essential git cmake libqt4-dev libphonon-dev python2.7-dev libxml2-dev libxslt1-dev qtmobility-dev libqtwebkit-dev
```

*qtmobilty-dev* package could be found in the following link:
https://packages.ubuntu.com/trusty/qtmobility-dev

The second step will be the installation of pip according to the python version used:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python2.7 get-pip.py
```

If the python version used is Python 3.2, exists an incompatibility that does not allow Awlsim-gui to run, to download the *get-pip.py* archive required for this Python version can be found here:
https://pip.pypa.io/en/stable/installing/

The third step will be installing the latest wheel distribution:

```
$ sudo pip2.7 install wheel
```

#### 4.1.1.2 Building PySyde Distribution

The first step will be downloading the PySide source distribution:

```
$ wget https://pypi.python.org/packages/source/P/PySide/PySide-1.2.4.tar.gz
```

The second, extract the source distribution and change to the distribution directory:

```
$ tar -xvzf PySide-1.2.4.tar.gz
```

```
$ cd PySide-1.2.4
```

The last step will be building the wheel binary distribution that it is already downloaded:

```
$ python2.7 setup.py bdist_wheel --qmake=/usr/bin/qmake-qt4
```

#### 4.1.1.3 Installing PySide distribution

While installing PySide the name of the PySide distribution file may change depending on your platform, since it may change it is recommended to look for the proper name, if not the installation will fail.

```
$ ls dist
```

```
$ sudo pip2.7 install dist/PySide…
```

Once PySide is built and installed Awlsim-gui can be run without problems.

## 4.2 Kernel and Patch

The Kernel is the most fundamental part of an operative system, is a software that interfaces with the hardware components using drivers, the Kernel controls the running processes and it decides how much time and the resources that a process can use and which hardware is involved in this process.

Working in Real-Time is necessary in this project, the signal must be read and written through the pins as fast as possible to have the minimum delay. To achieve that, modifications in the Linux Kernel must be done. The CONFIG_PREEMPT_RT patch allows the Linux Kernel to be fully preemptible without interrupts, system calls, the CPU don't have to run user-mode code nor control other processes to pre-empt a new one.

The patch uses spinlocks and it implements priority inheritance for them as well as for the semaphores, it allows some critical sections that were protected before to be preemptible now, it converts the interrupt handlers into preemptible Kernel threads and it separates the Linux timer API to achieve a higher timer resolution. So, it allows the system to run Real Time Processes.

### 4.2.1 Kernel and Patch Installation

Before installing any version of the Kernel, the Patch that matches the Kernel should be found. In the case of this project the Kernel and the patch is 4.1.5 version.

The Kernel version as well as the patch can be downloaded in the following links:

https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.1.5.tar.xz

https://www.kernel.org/pub/linux/kernel/projects/rt/4.1/older/patches-4.1.5-rt5.tar.xz

The archives must be unpacked, and the Kernel must be patched:

```
$ xz –cd linux-4.1.5.tar.xz | tar xvf –
$ cd linux-4.1.5
$ xzcat ../patch-4.1.5-rt5.patchxz | patch –p1
```

### 4.2.2 Kernel Configuration and building

To change the Kernel configuration, the next command must be run:

```
$ make menuconfig
```

Then several configurations must be selected in the menu:

- Select Processor type and features→ Pre-emption Mode→ Activate Fully Preemptible Kernel (RT)

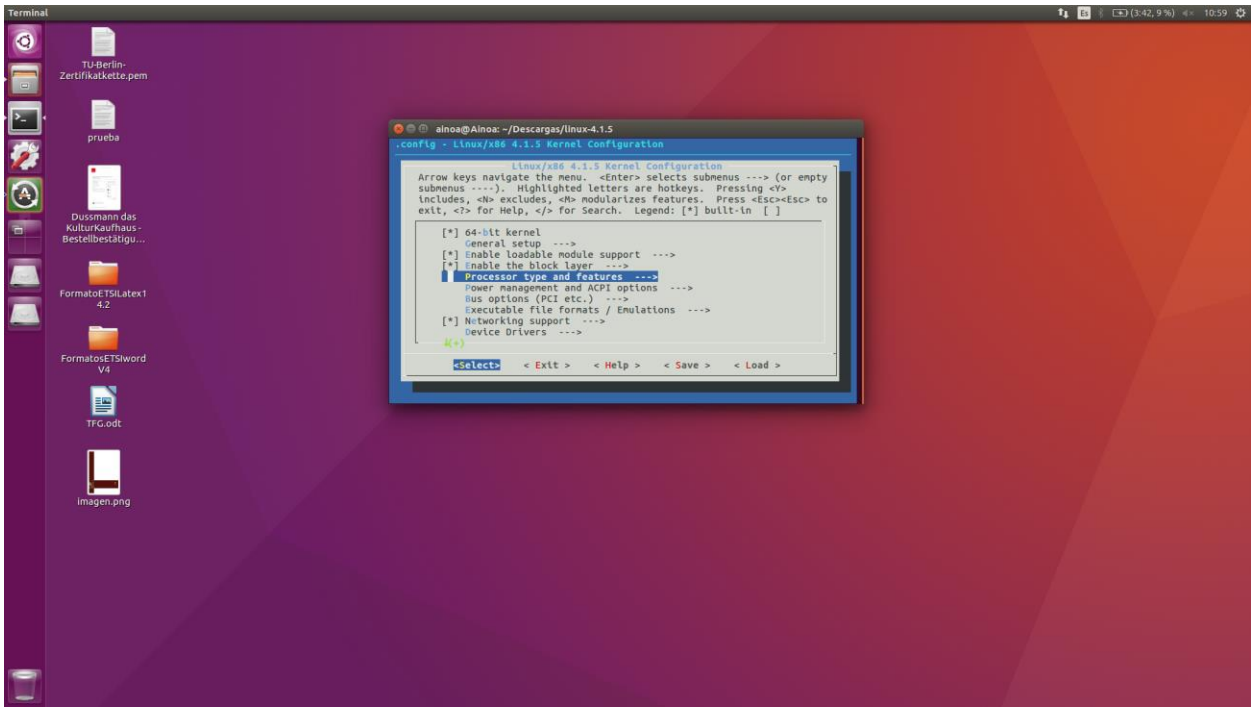


Figure 4 Selecting Kernel configurations

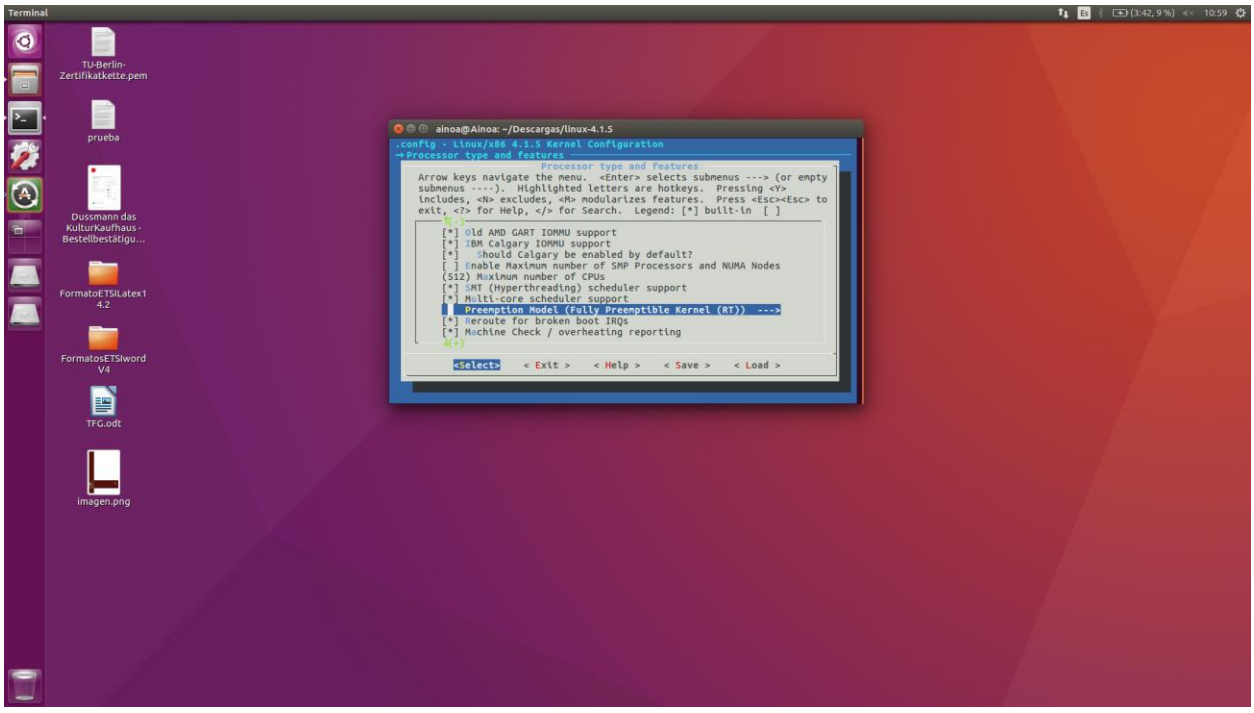


Figure 5 Selecting Kernel configurations

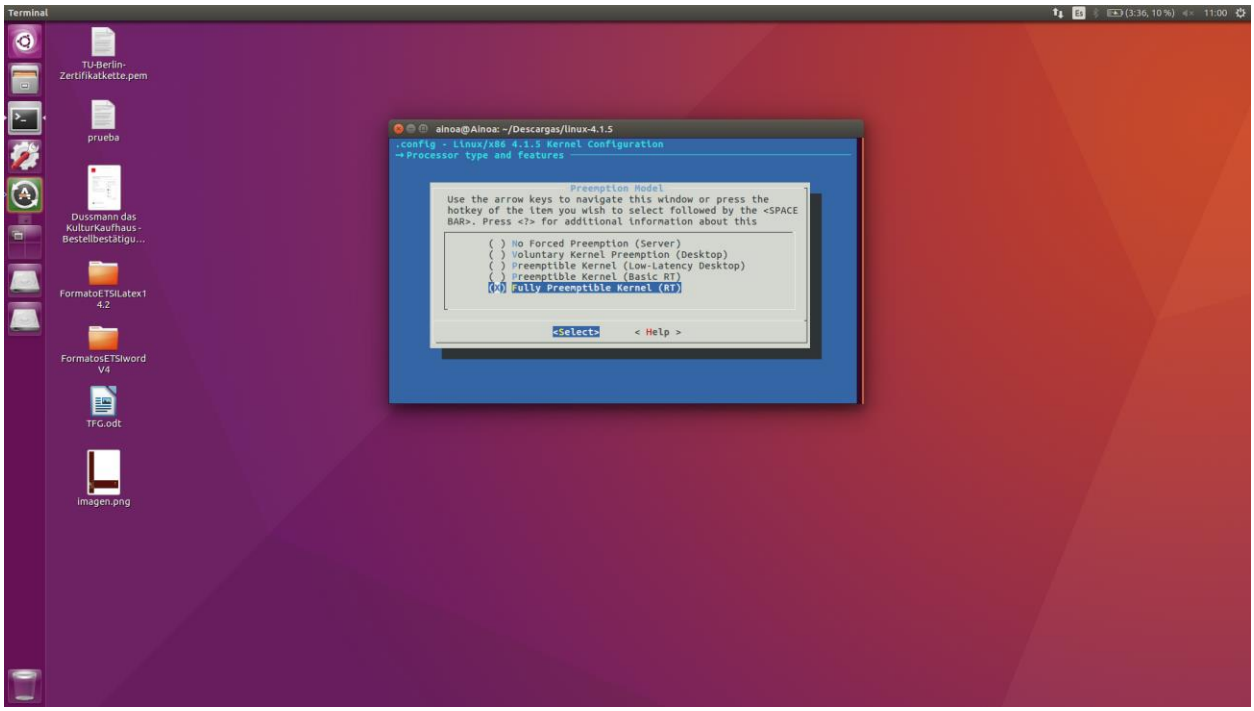


Figure 6 Selecting Kernel configurations

- Select General Setup→ Timer subsystem→ Activate High-Resolution-Timer Support

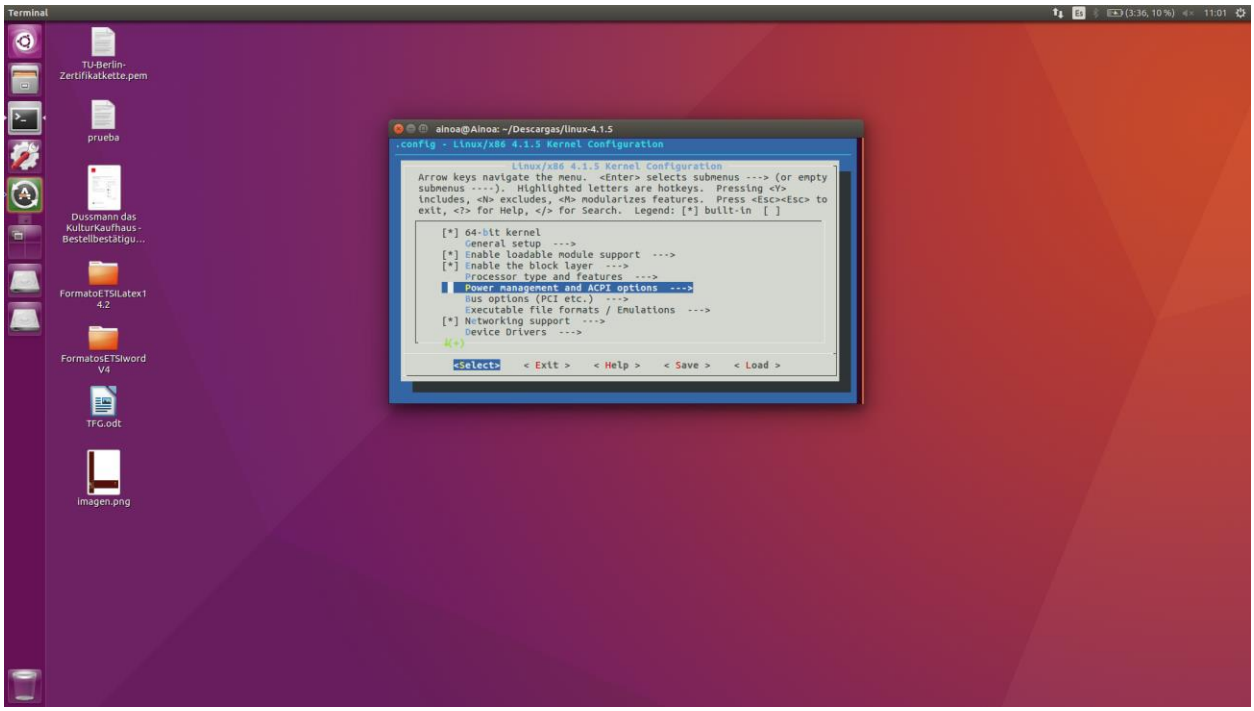


Figure 7 Selecting Kernel configurations

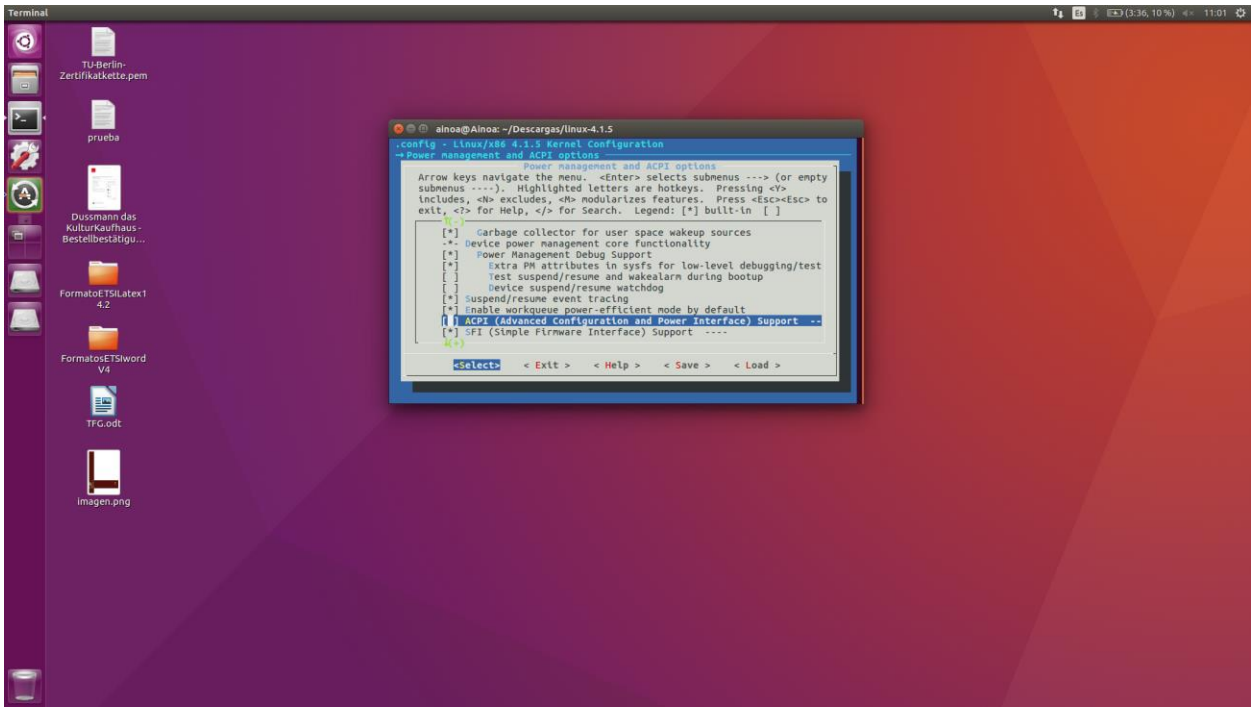


Figure 8 Selecting Kernel configurations

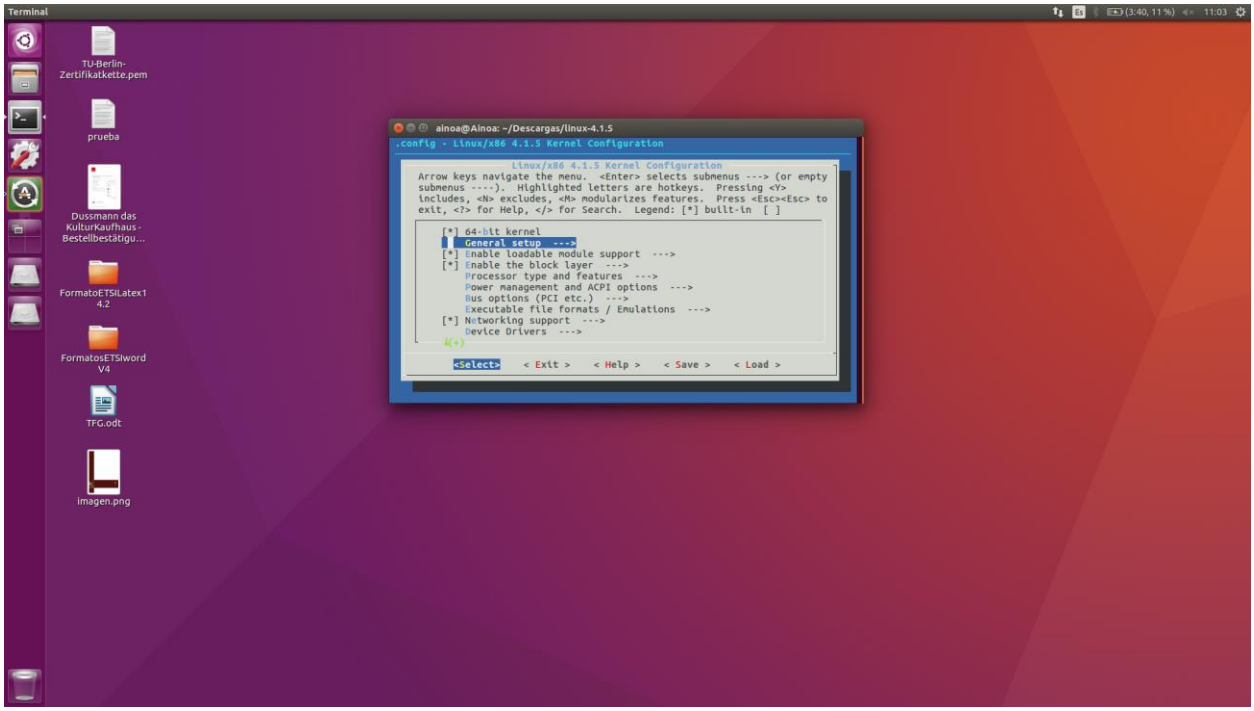


Figure 9 Selecting Kernel configurations

- Power management and ACPI options→ Disable ACPI (Advanced Configuration and Power Interface) Support

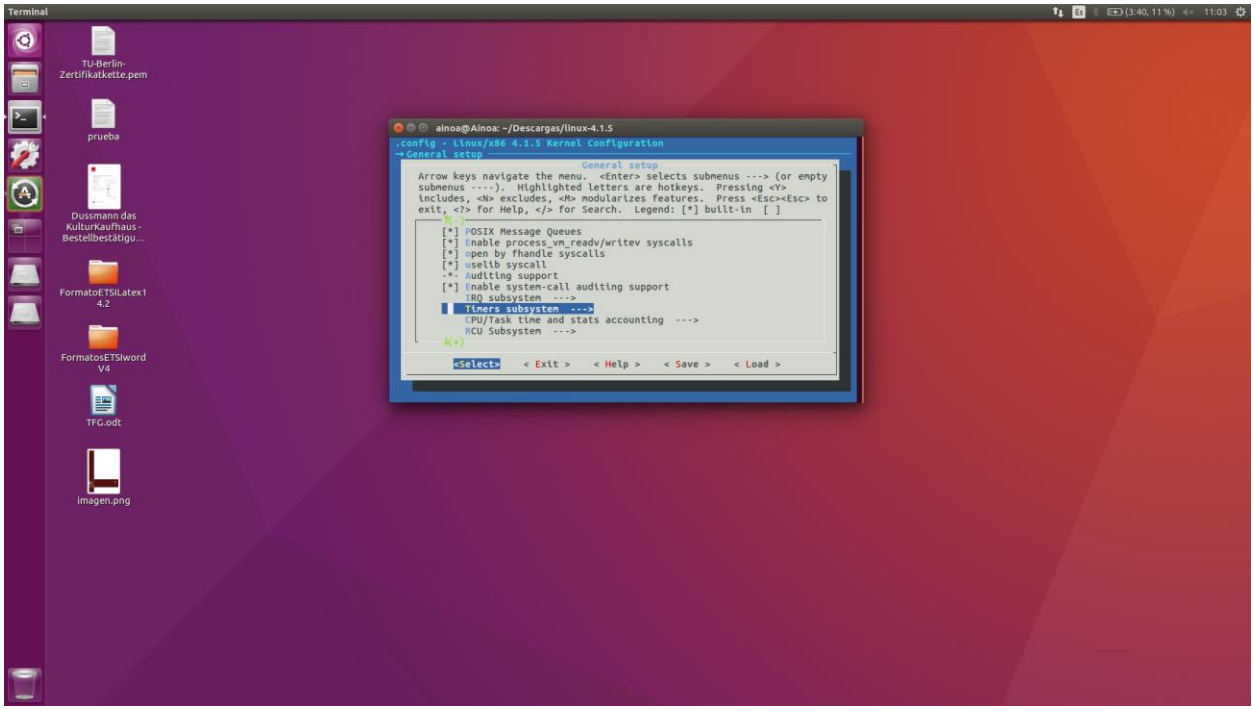


Figure 10 Selecting Kernel configurations

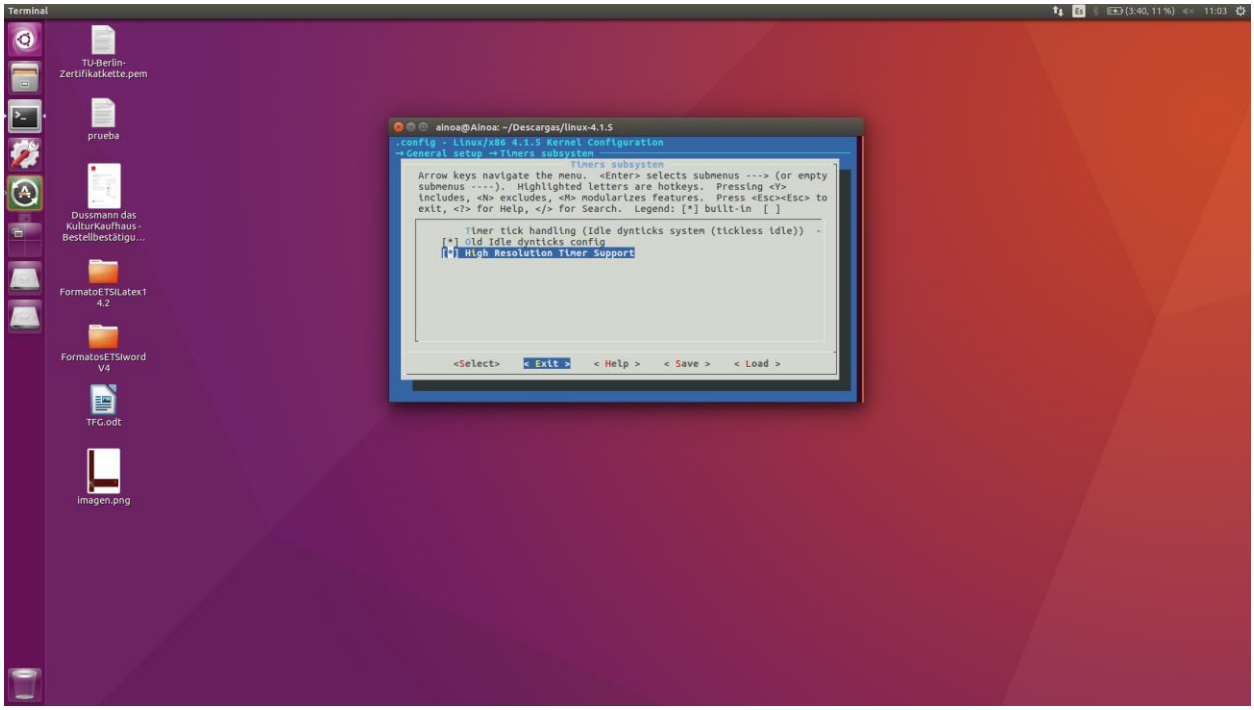


Figure 11 Selecting Kernel configurations

Select exit and run:

```
$ make
$ sudo make modules_install
$ sudo make install
```

Depending on the computer this process may take a while.

After the grub must be updated:

```
$ sudo update-grub
```

Reboot the system and select the new Kernel in the grub.

## 4.3 KVM (Kernel-Based Virtual Machine)

KVM is a solution to implement a full virtualization with Linux. It is formed by a Kernel module and other tools in the user space. KVM is an Open-Source software developed by Qumranet.

To use KVM a x86 or x86_64 processor with virtualization support is needed. KVM can run Linux/Unix/Windows guests of 32 o 64 bits.

KVM allows the execution of several Virtual Machines using ISO images with different OS (operative system). Each virtual machine has its own hardware which is virtualized, such as a Hard Disk Drive, a Network Interface Card or a Graphics Card …

KVM has been used to fully virtualize the Guest machine inside the Host machine.

### 4.3.1 Installation of KVM

Before installing KVM the CPU must support hardware virtualization, for that this command must be run:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

If the result is one or more the CPU supports hardware virtualization, the next step will be enabling the virtualization in the BIOS:

```
$ kvm-ok
```

If the result is KVM acceleration can be used it means the installation can be made, to start, several packages are required, such as:

```
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder
bridge-utils
```

- qemu-kvm is the backend
- libvirt-bin provides libvirtd to administer qemu and kvm instances using libvirt
- ubuntu-vm-builder for building virtual machines
- bridge-utils to set up a network for a hosted virtual machine

After the username must be added to the group libvirtd:

```
$ sudo adduser `id -un` libvirtd

Adding user '<username>' to group 'libvirtd' ...
```

To verify the installation run:

```
$ virsh list –all
```

To relog in or restart the Kernel modules:

```
$ rmmod kvm
```

```
$ modprobe -a kvm
```

## 4.4 Virt-manager

To create and manage the virtual machine (guest machine), virt-manager will be used.

virt-Manager is a GUI tool to manage virtual machines through libvirt. It manages KVM, XEN and LCX virtual machines. It represents the CPU usage as well as the Host CPU usage and the Memory usage of each virtual machine.

As KVM is required, virt-manager is a comfortable and easy GUI tool to manage the virtual machine.

### 4.4.1 Installation of Virt-manager (graphical user interface)

The installation of virt-manager also requires several packages like *libvirt-bin* and *bridge-utils*, which are already installed, as well as the installation of KVM.

```
$ sudo apt-get install virt-manager
```

Then, run virt-manager:

```
$ sudo virt-manager
```

Before creating a virtual machine, a new connection to local QEMU instance from file must be set →Add Connection menu. Select QEMU/KVM in the virtual machine list.

### 4.4.2 Installation of a Virtual Machine with Ubuntu 16.04

Using the virt-manager graphical user interface select → Create a new virtual machine in the tool bar.

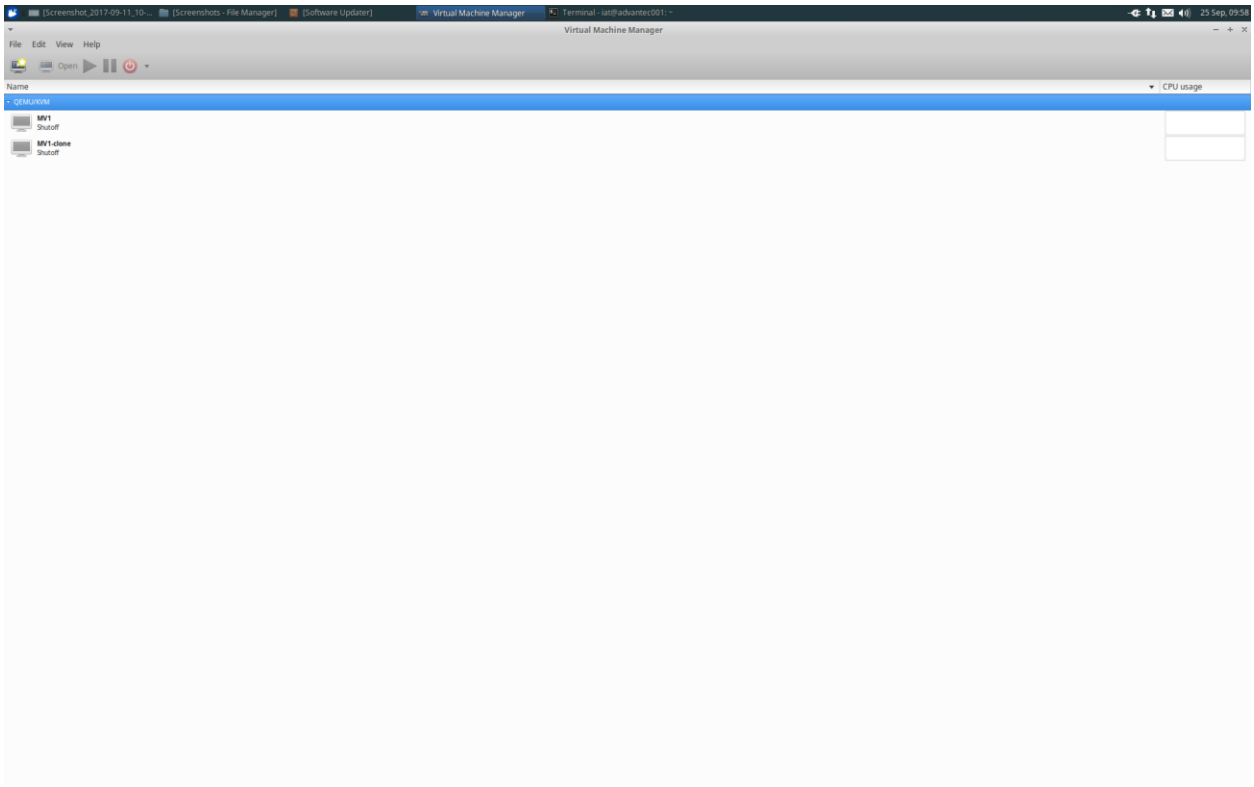


Figure 12 Step 0 Virtual Machine Installation

Then write the name of your virtual machine, in this case the virtual machine was called *MV1*, and then select how to install the operating system.

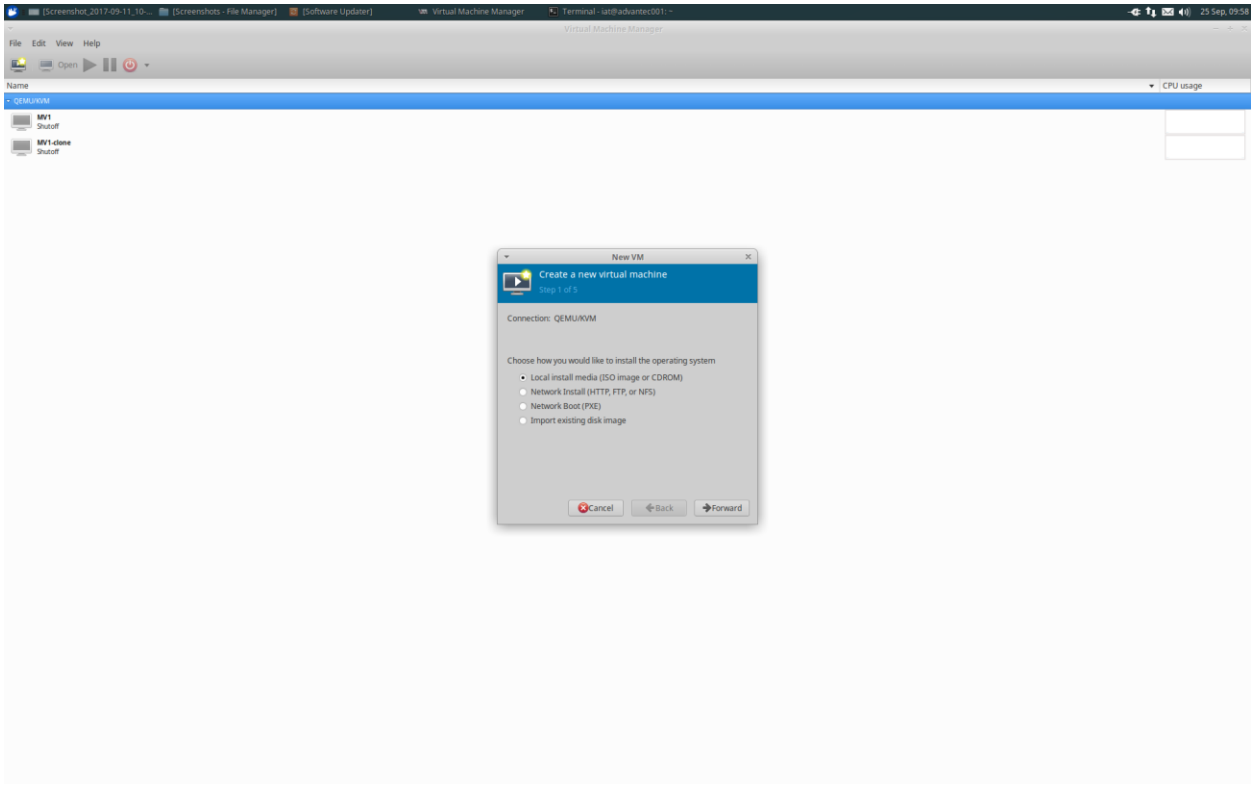


Figure 13 Step 1 Virtual Machine Installation

This virtual machine was installed using a ISO image of Ubuntu 16.04, which can be downloaded from the source page:

https://www.ubuntu.com/download/desktop

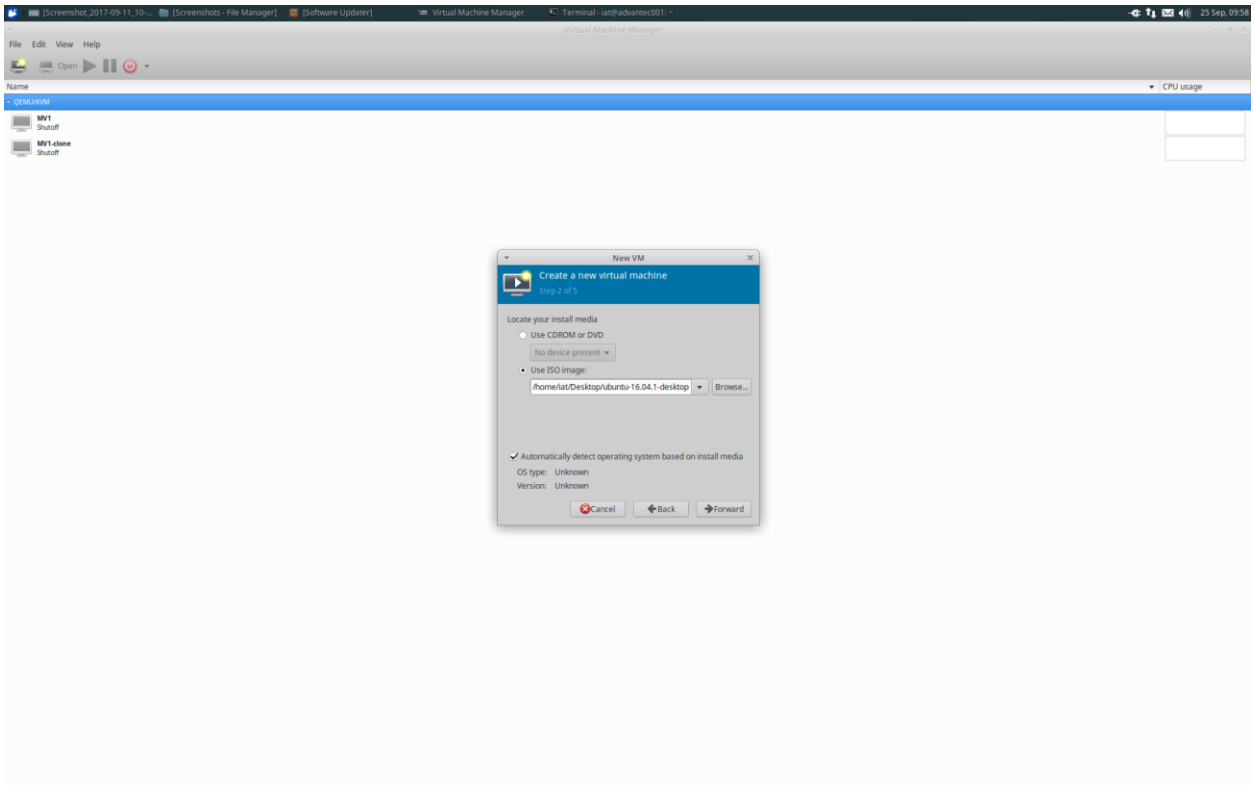


Figure 14 Step 2 Virtual Machine Installation

The third step will be choosing the Memory and CPU setting, 1024 MB for the RAM and 1 CPU, these values may change depending on the needs.

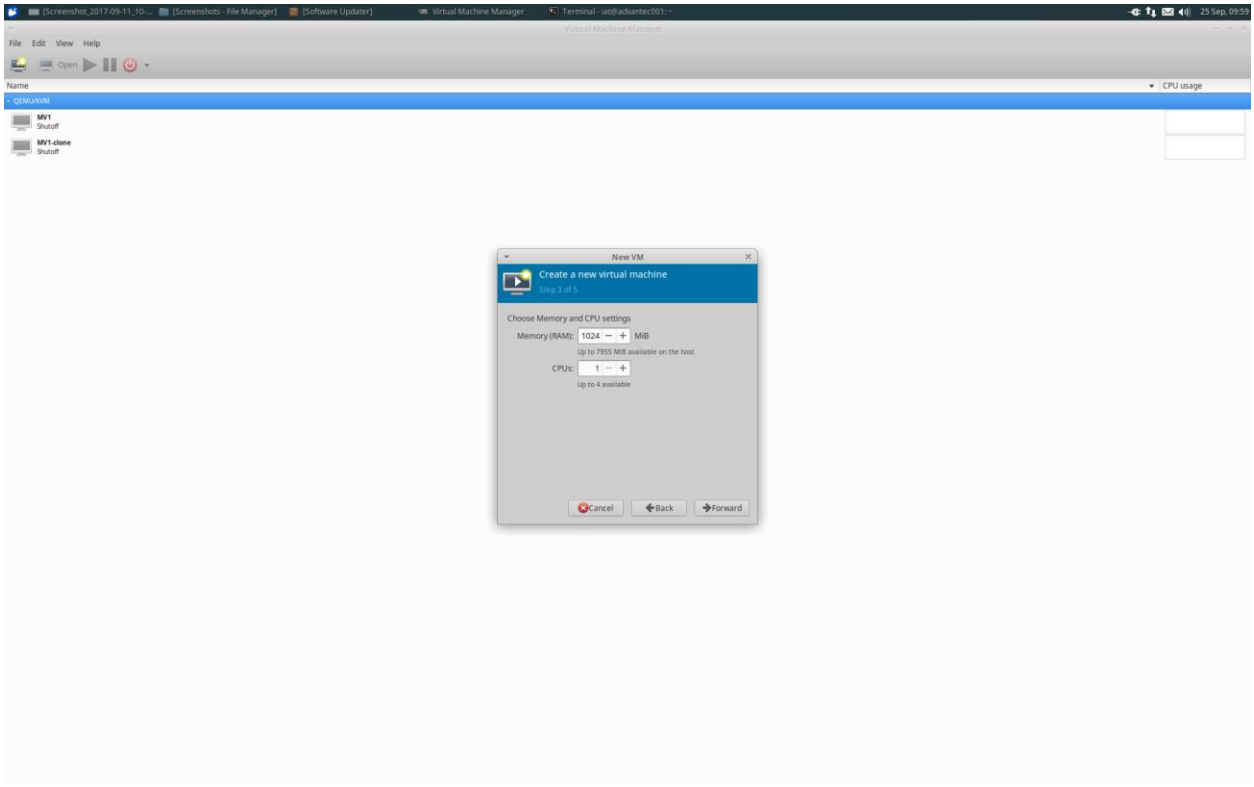


Figure 15 Step 3 Virtual Machine Installation

The fourth step will be the storage configuration that will be assigned to the virtual machine.

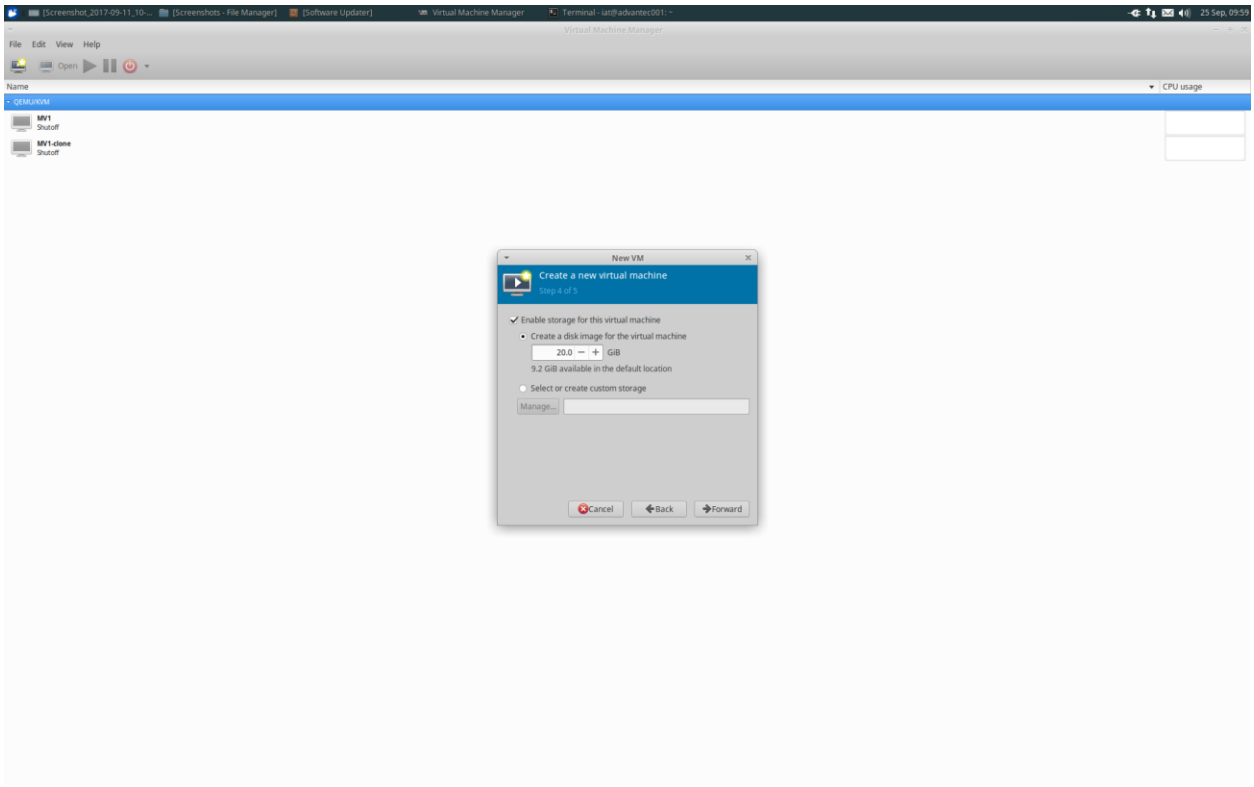


Figure 16 Step 4 Virtual Machine Installation

The last step, verify the settings and modify several advanced options like networking settings (select default), the virtualization type (KVM) and the architecture (x86_64). Then press Finish.

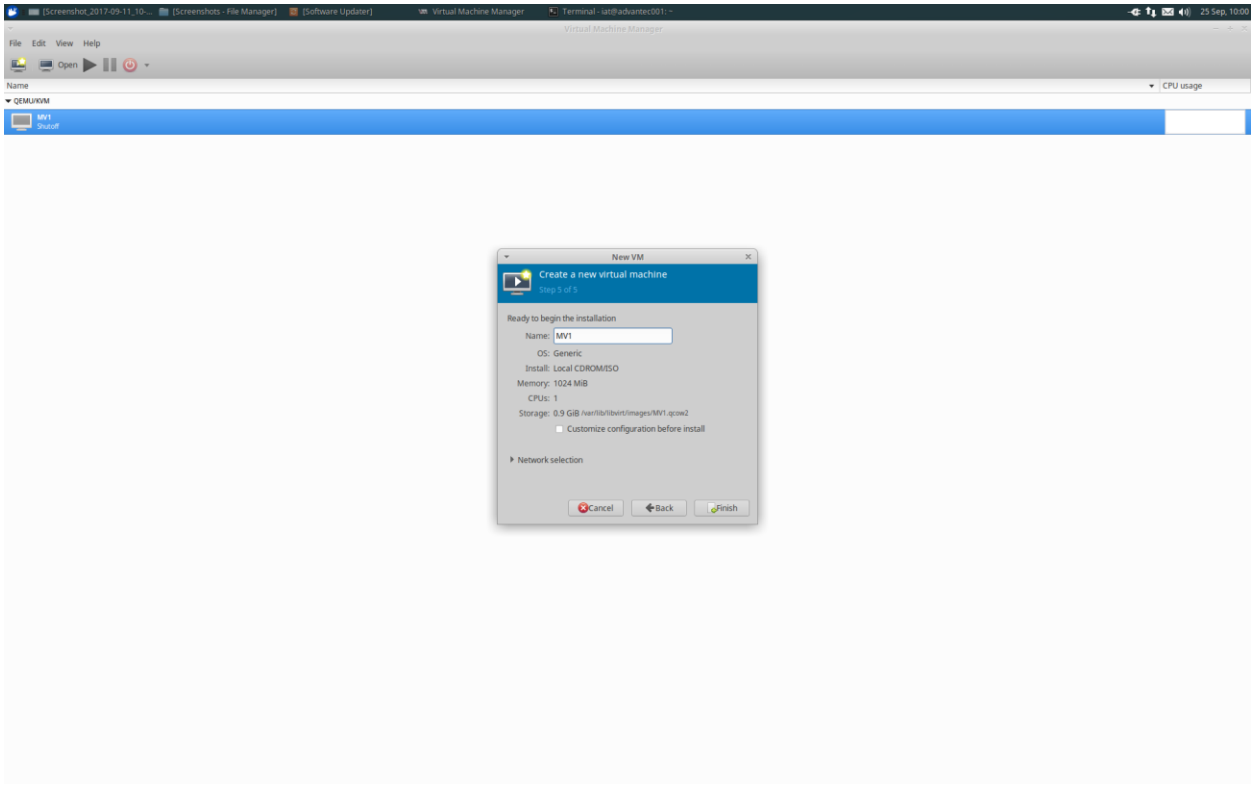


Figure 17 Step 5 Virtual Machine Installation

When the virtual machine runs for the first time the installation of Ubuntu will start, after the installation the virtual machine can be used.

The installation of the Kernel, as well as the installation of Awlsim must be done in the virtual machine too, following the same steps that were used in the host machine.

## 4.5 DKMS (dynamic Kernel Module Support)

As the compilation of the driver module is required to use the port, DKMS is needed for this purpose.

DKMS package helps to compile and install supplementary versions of Kernel modules into the Kernel tree.

It was developed by the Company Dell, one advantage of DKMS resides in the addition of new Kernel modules without installing a new one, as well as adding new driver on a system, that's why DKMS is very useful for this project.

### 4.5.1 DKMS Installation and Use

The package can be installed running the command:

```
sudo apt-get install build-essential dkms
```

After installing, a dkms.conf archive must be created and written in order to compile and install the required module, in this case *imanager-core*.

```
# cd /home/iat/driver/iManager
# touch dkms.conf #create dkms.conf file
# gedit dkms.conf
```

In the dkms.conf archive the following lines are required:

```
MAKE="make -C KERNELDIR=/lib/modules/${kernelver}/build"
CLEAN="make -C clean"
BUILT_MODULE_NAME=iManager
BUILT_MODULE_LOCATION=.
PACKAGE_NAME=iManager
PACKAGE_VERSION=1
REMAKE_INITRD=yes
```

Then installing the required module into DKMS, the module installation files must be copied into the Kernel source tree:

```
# ls
    README dkms.conf lib src
# sudo cp -R . /usr/src/iManager-1
# sudo dkms add -m iManager –v 1
```

The required module has been added to the list of modules to build, to install the module into the current Kernel the following commands must be run:

```
# sudo dkms build -m iManager -v 1
# sudo dkms install -m iManager -v 1
```

Some errors may appear but even though the compilation and installation of the required module work

# 5 Sending the Signal to the Host Machine

## 5.1 Advantech iManager Linux Driver Set

Advantech iManager Linux Driver Set is a set of platform drivers which contains the needed support to use the 8-bit Digital I/O. To manage the GPIOs the Kernel config option CONFIG_GPIO_SYSFS needs to be enabled.

### 5.1.2 Loading the modules

If the module building and installation using DKMS is well done, the module should be installed, but not loaded, to load the module:

```
$ sudo modprobe imanager-core
```

### 5.1.3 Using GPIO from User-Space

To try the GPIOs this program was implemented, it writes the value of the signal that comes into the pin 504, the device's fourth pin, then its value is written into the pin 505 which is the device's fifth pin:

```
$ sudo su
$ cd /sys/class/gpio
$ echo 504 > export
$ echo in > gpio504/direction
$ echo 505 > export
$ echo out > gpio505/direction
$cd $HOME
#!/bin/sh
#modprobe imanager-core
#echo $1> /sys/class/gpio/export
#echo out> /sys/class//gpio/$1/direction
while busybox echo hey > /dev/null
do
value=`cat /sys/class/gpio/gpio505/value`
echo "$value" > /sys/class/gpio/gpio504/value
done
```

The aim of this program is to show that the communication between the Digital I/O of the host machine and the Linux OS is possible.

A push-pull signal of 10 Hz was generated with a function generator as an input through the pin number five, then this signal is sent through the pin number four, the received signal is represented with the help of an oscilloscope, later both signals will be compared.

The pin number six is the ground (black cable) and the pins four (test probe) and five (red cable) are used as an output and input respectively.

The connection between the signal generator, the computer Digital I/O and the oscilloscope is shown in the next photos:

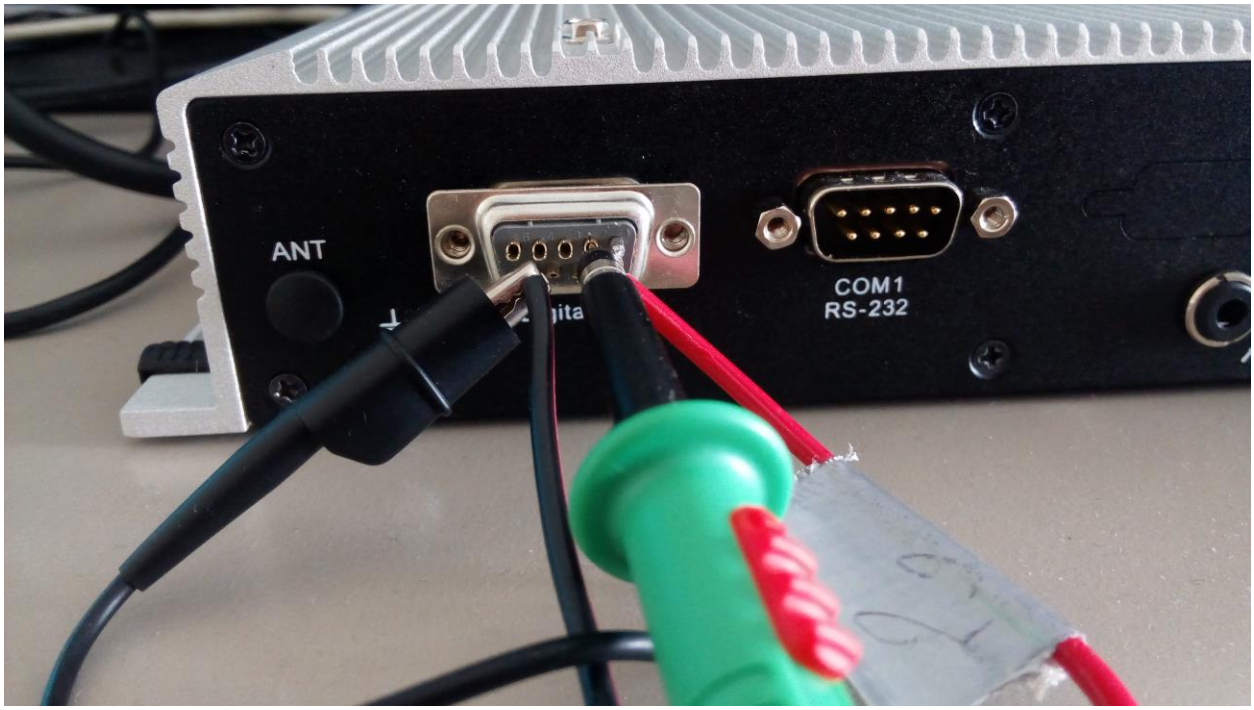


Figure 18 Digital I/O, oscilloscope and signal generator connection

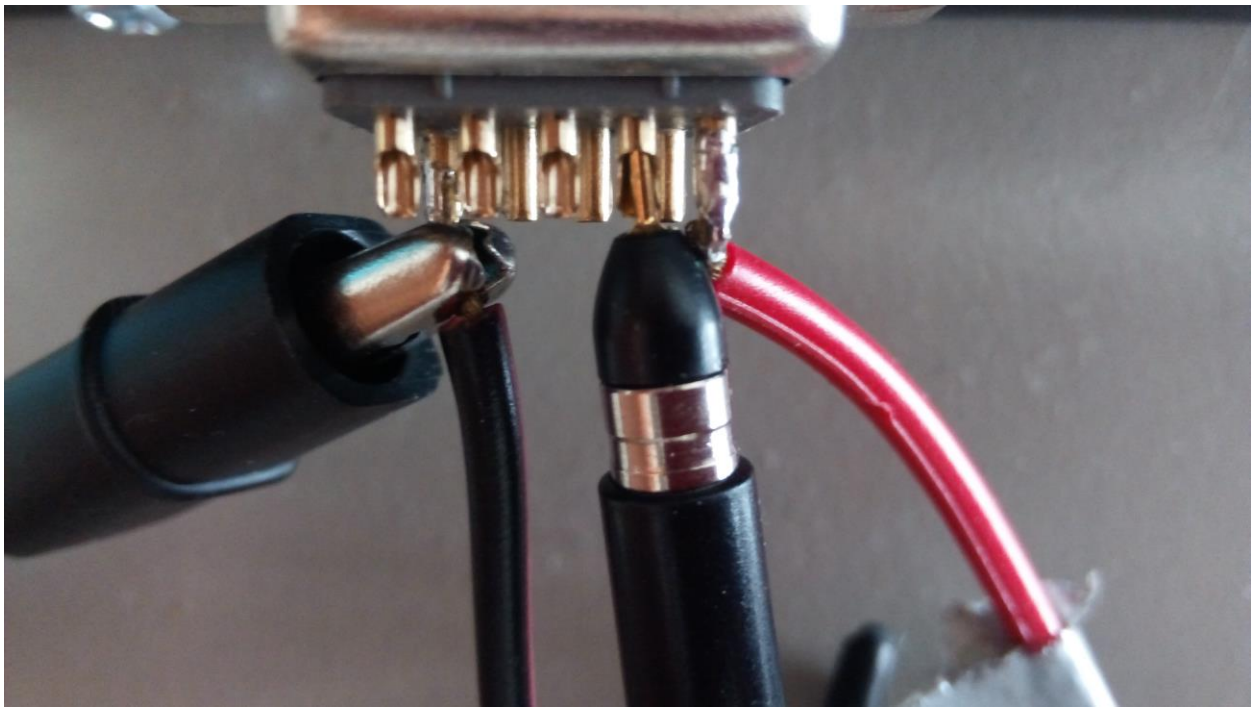

Figure 19 Digital I/O, oscilloscope and signal generator connection

The communication between the Linux System and the port works as it is shown in the next image:

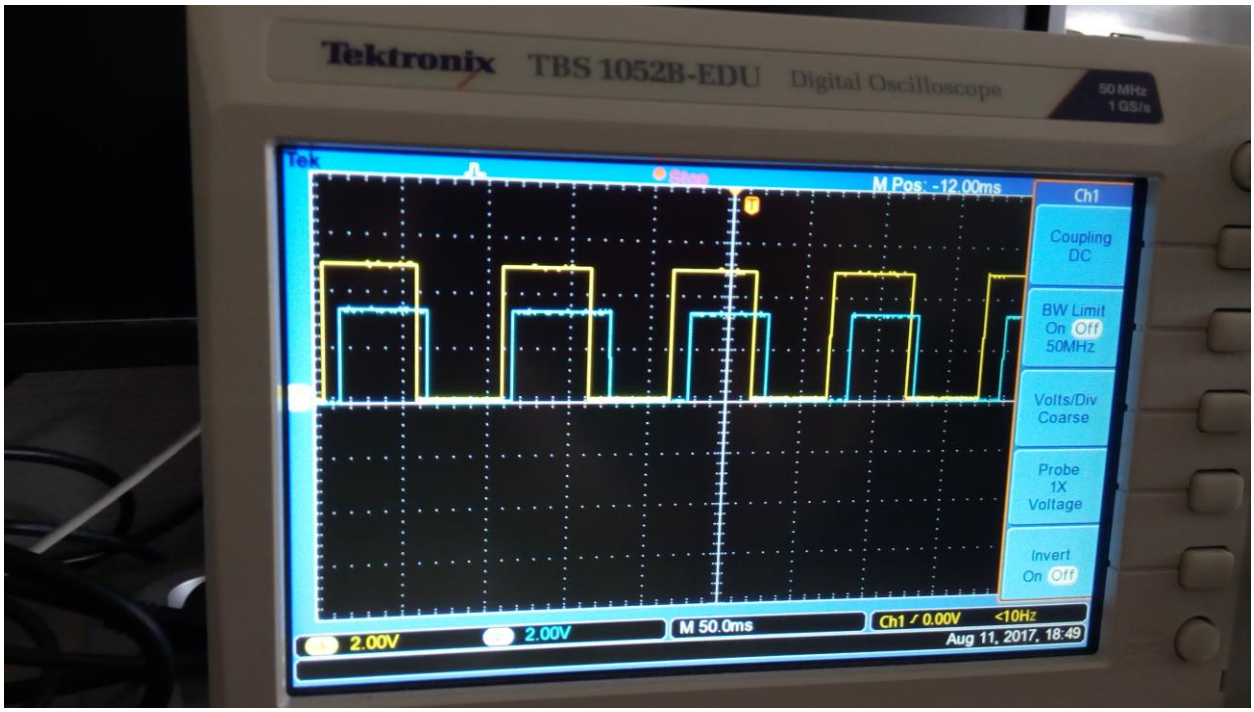


Figure 20 Comparison between the received (blue) and the desired (yellow) signal

The yellow line represents the signal that is sent to the device trough the pin number five (generated desired signal); the blue line represents the signal that is coming out from the pin four, a delay between the yellow and the blue signals is noticeable and measurable. Even though this image shows that the signal can be sent and read using the explained configuration.

## 5.2 Real Time Scheduling Priority

Change to a real time scheduling priority of the process that runs the program which sends and reads the signal may eliminate or decrease the delay.

The scheduling priority could be changed using *chrt* command.

There are three common different scheduling priorities:

1. SCHED_OTHER: the default scheduling priority
2. SCHED_FIFO: the scheduling priority for real time applications.
3. SCHED_BATCH: the scheduling for batch processes

By default, as the program to read and send the signal was running, the process has the default scheduling priority, so it was changed to the first input first output one to improve the performance time to reduce the delay:

1. Check the PID number of the process:

   ```
   $ ps ax
   ```

2. Check the scheduling priority:

   ```
   $ chrt -p [pid number]
   ```

The scheduling policy was *SCHED_OTHER*, so it must be changed

3. Set the SHED_FIFO scheduling policy:

   ```
   $ chrt -f -p [1...99] [pid number]
   ```

The first parameter to choose is a number between 1 and 99 to set maximum valid priority for the scheduling algorithm that is used, so 99 will be the maximum priority.

After changing the scheduling priority any change in the delay value was noticeable, the delay persists.

Changing the priority of the running process using the renice command was tried too:

```
$ renice 20 [pid number]
```

The maximum priority for a running process is 20, even though the results were the same, any difference in the delay could be appreciated.

At this point the reduction of the delay was put aside because it was not important for the following develop of the project, so this inconvenience will be solved in the future because it does not alter the way the software works but its time of performance.

Two different Kernel versions were tested, the first was the 4.7 and the second was 4.1.5 version, the pictures show the delay in both cases:

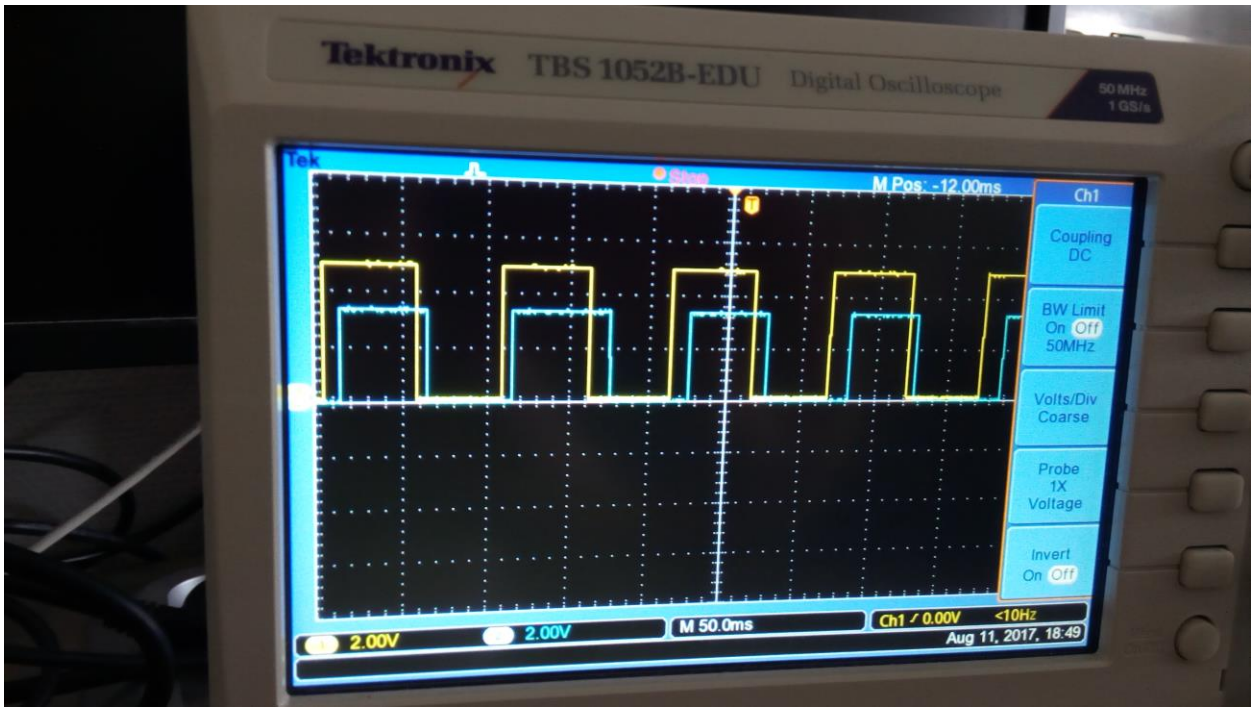


Figure 21 Delay using Kernel 4.7

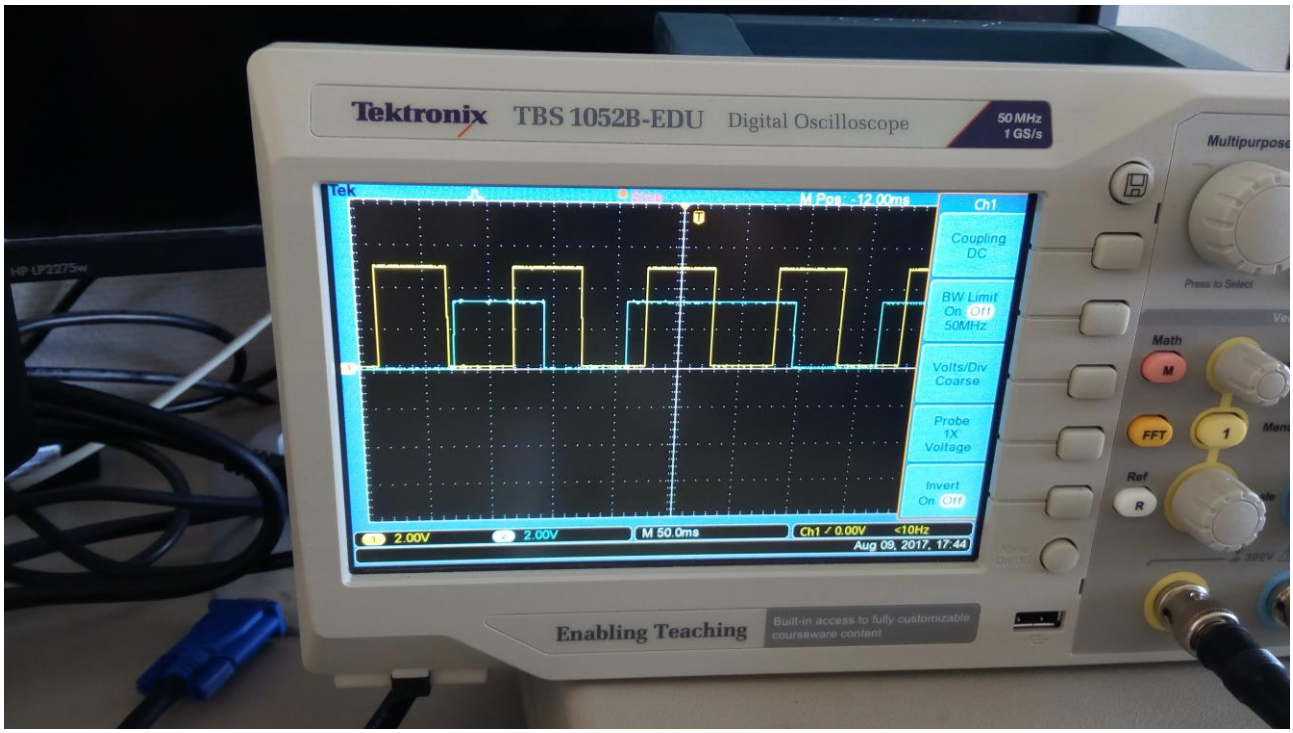


Figure 22 Delay using Kernel 4.1.5

As a conclusion, the Kernel configuration used may increase or reduce the delay, a modification of the Kernel options may reduce the delay to get the desired performance.

# 6 Sending the Signal to the Guest Machine

## 6.1 Vt-d Virtualization

To send the signal to the guest machine where the Soft-PLC is running, Vt-d virtualization is required.

The first step was checking if Vt-d virtualization was supported by the hardware that is used:

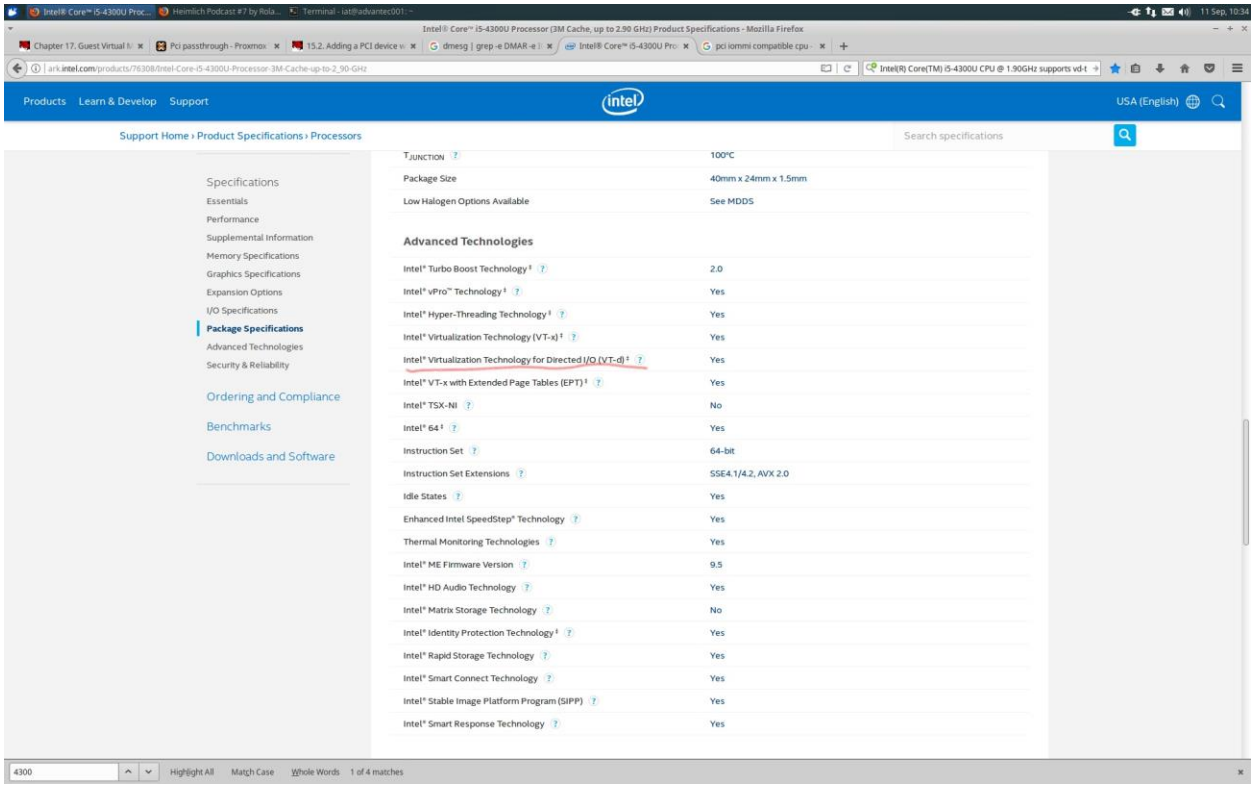


Figure 23 Checking the availability of Vt-d virtualization

The Vt-d virtualization must be enabled in the BIOS as well in the archive *grub.cfg*, to do that the following line must be modified.

From:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
```

To:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on"
```

## 6.2 Kernel 3.0.101 Installation

To use Vt-d several Kernel configurations must be enabled. After researching, those Kernel configurations are experimental ones and they were removed after 3.x kernel releases, so a new Kernel compilation and installation is required as the Kernel version that is installed is the 4.1.5 version.

Those configurations are in 2.6.28–2.6.39 and 3.0 kernel versions, so the 3.0.101 Kernel version was the chosen one.

Before installing any version of the Kernel, the Patch that matches the Kernel should be found because the RT-patch is also required. The Kernel version as well as the patch can be downloaded in the following links:

https://www.kernel.org/pub/linux/kernel/v3.0/linux-3.0.101.tar.xz

https://www.kernel.org/pub/linux/kernel/projects/rt/3.0/patch-3.0.101-rt130.patch.xz

The archives must be unpacked and the Kernel must be patched:

```
$ xz –cd linux-3.0.101.tar.xz | tar xvf –
$ cd linux-3.0.101
$ xzcat ../patch-3.0.101-rt130.patch.xz | patch –p1
```

## 6.3 Kernel Configuration and building

To change the Kernel configuration, the next command must be run:

```
$ make menuconfig
```

Then several configurations must be selected in the menu:

- Select Processor type and futures→ Preemption Model (Fully Preemptible Kernel (RT)) → Enable Fully Preemptible Kernel (RT)

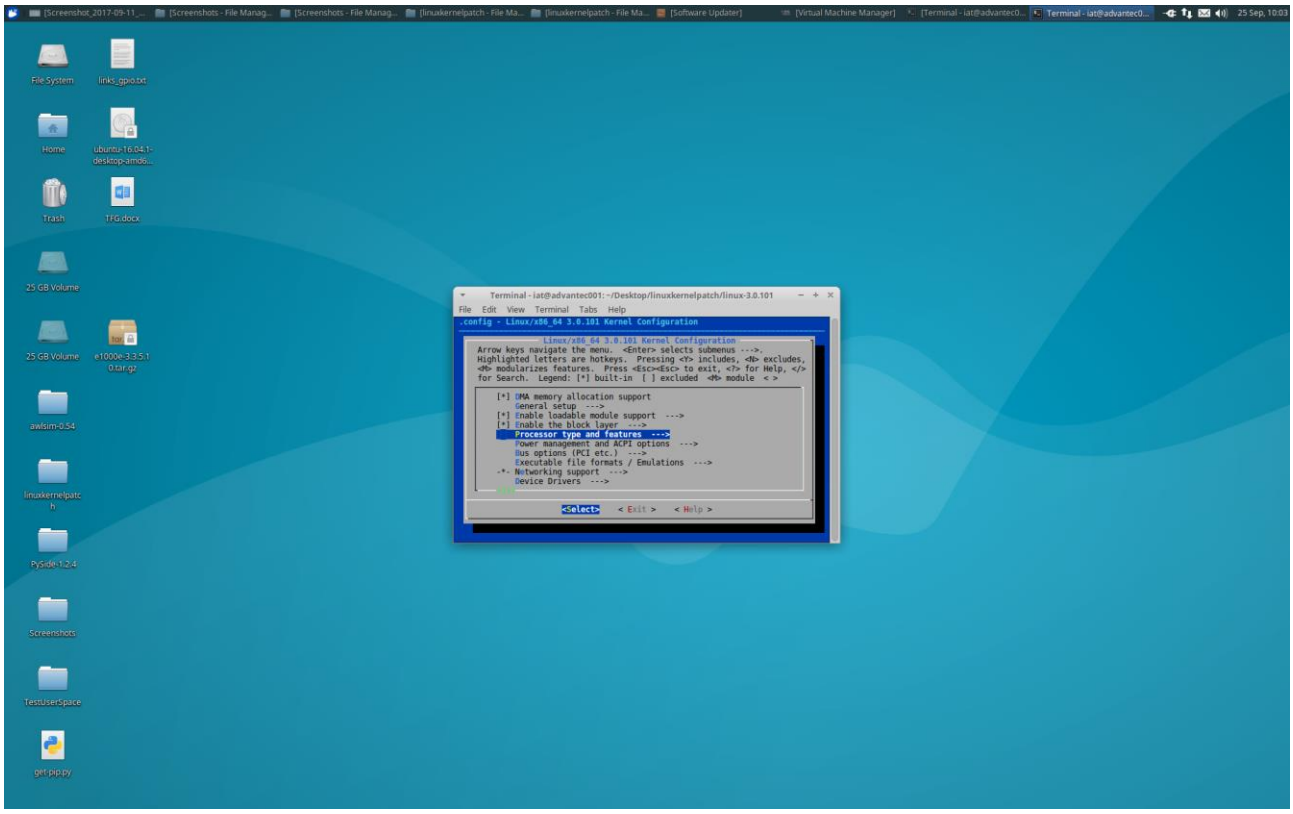


Figure 24 Selecting Kernel configurations

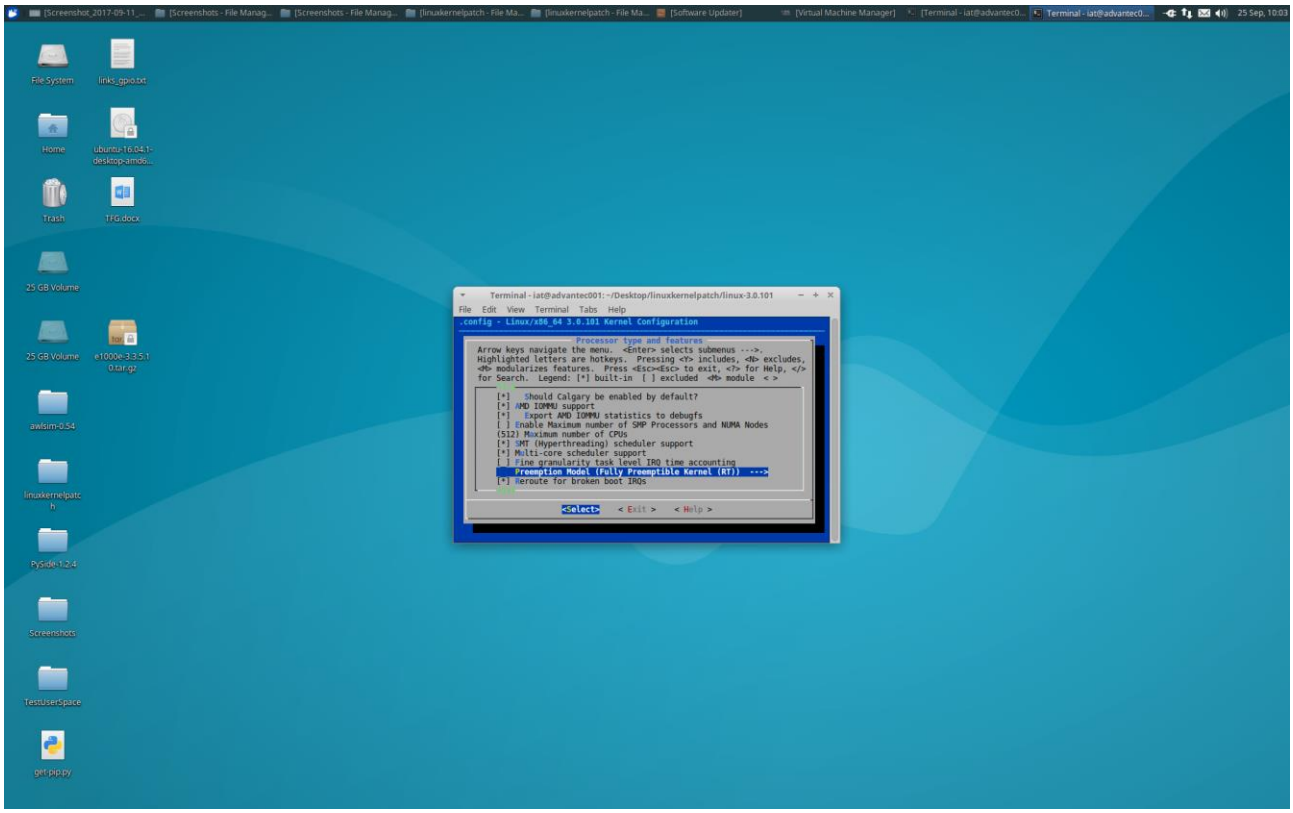


Figure 25 Selecting Kernel configurations

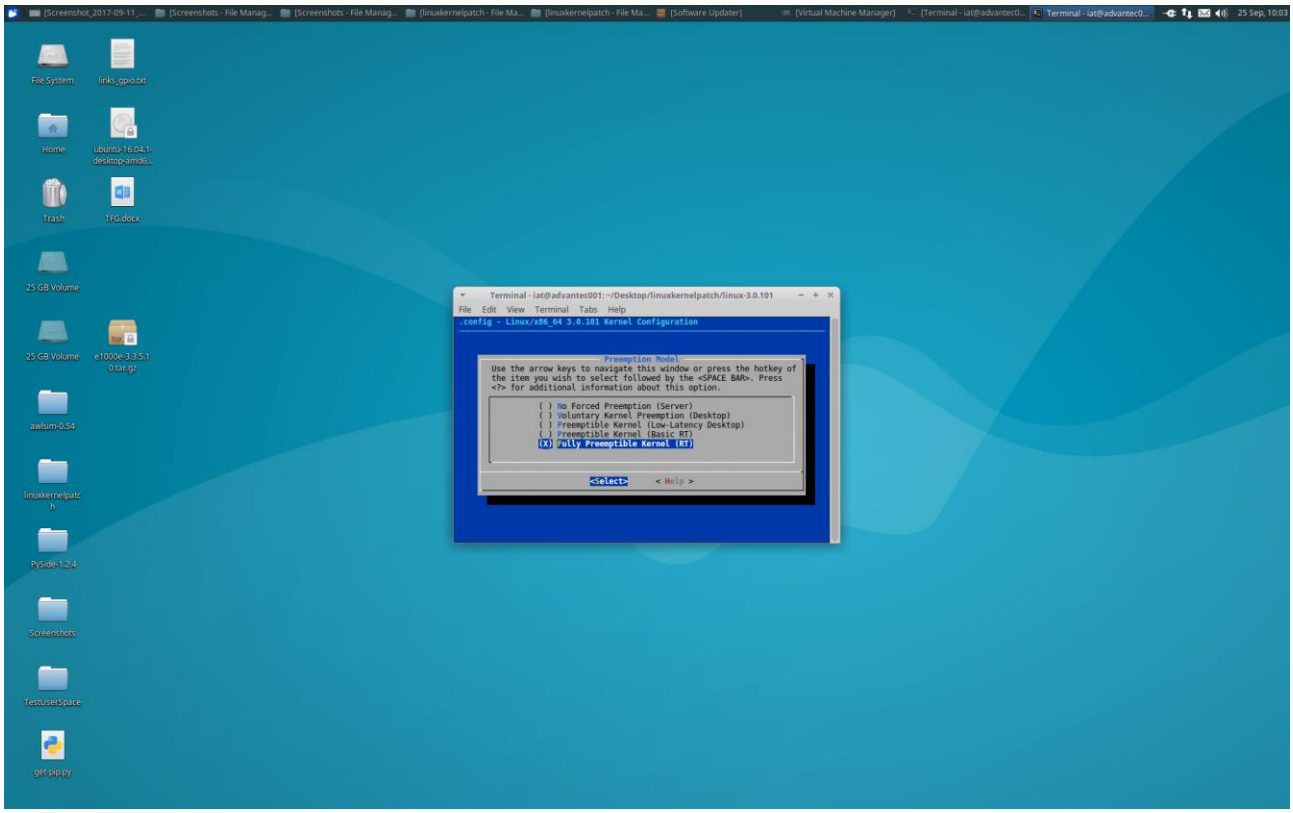


Figure 26 Selecting Kernel configurations

- Select General Setup→ Enable Prompt for development and/or incomplete code/drivers

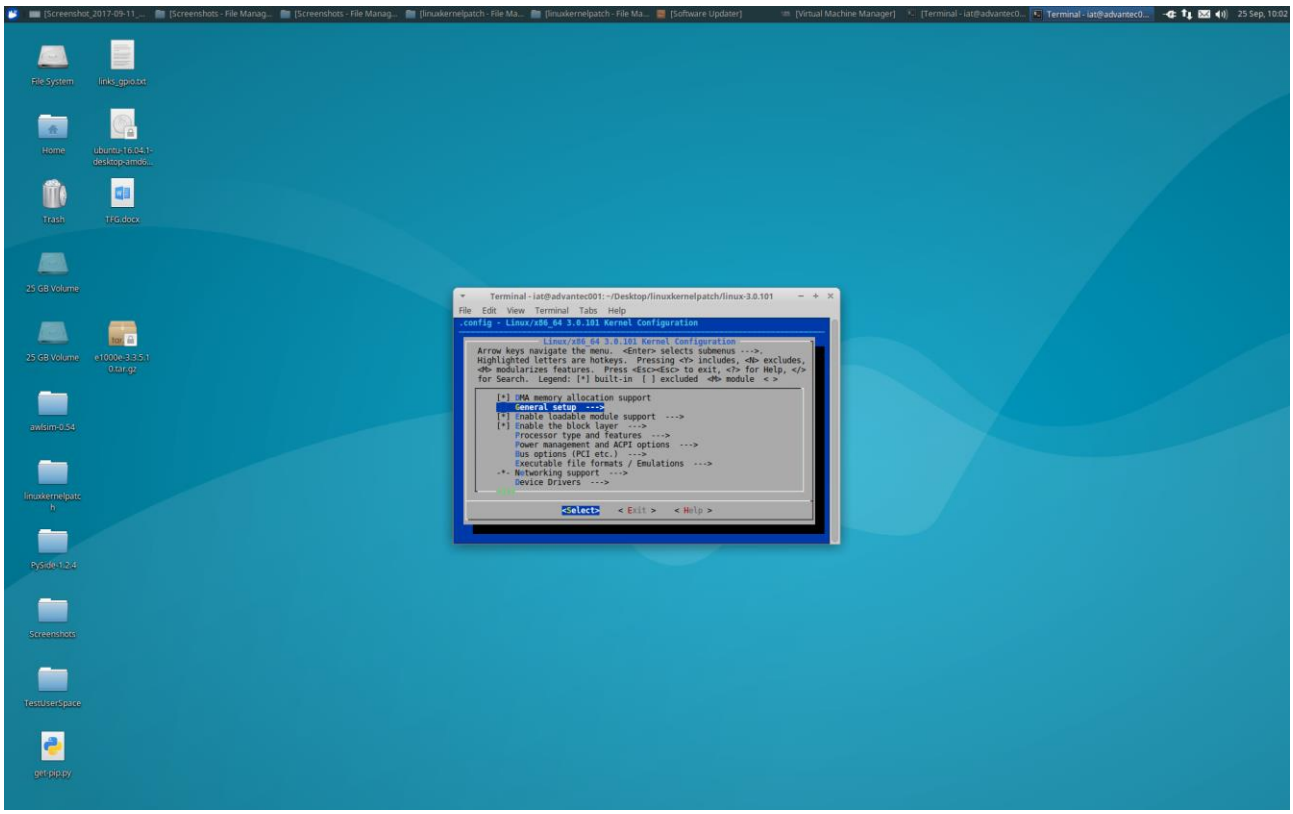


Figure 27 Selecting Kernel configurations

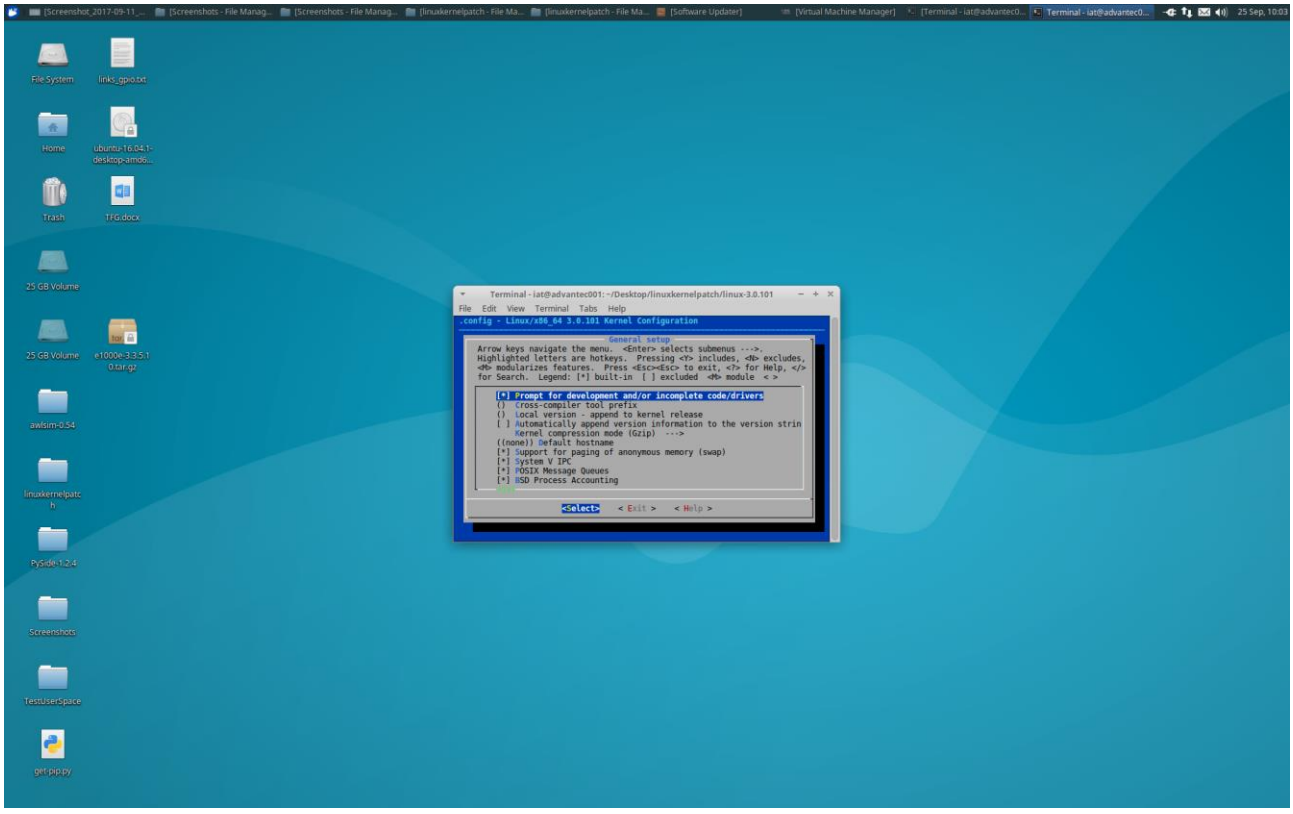


Figure 28 Selecting Kernel configurations

- Select Bus Options→ Enable Support for DMA Remapping Devices (Experimental)

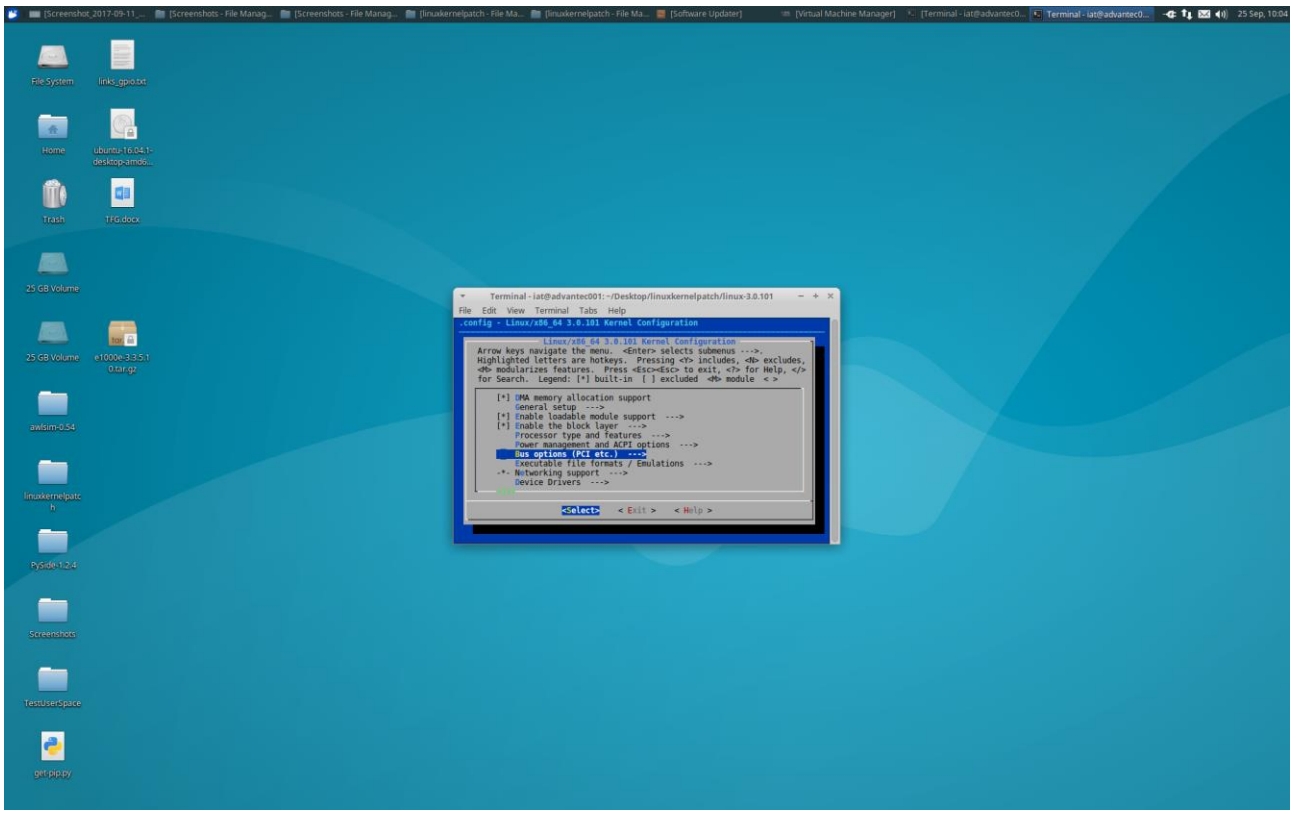


Figure 29 Selecting Kernel configurations

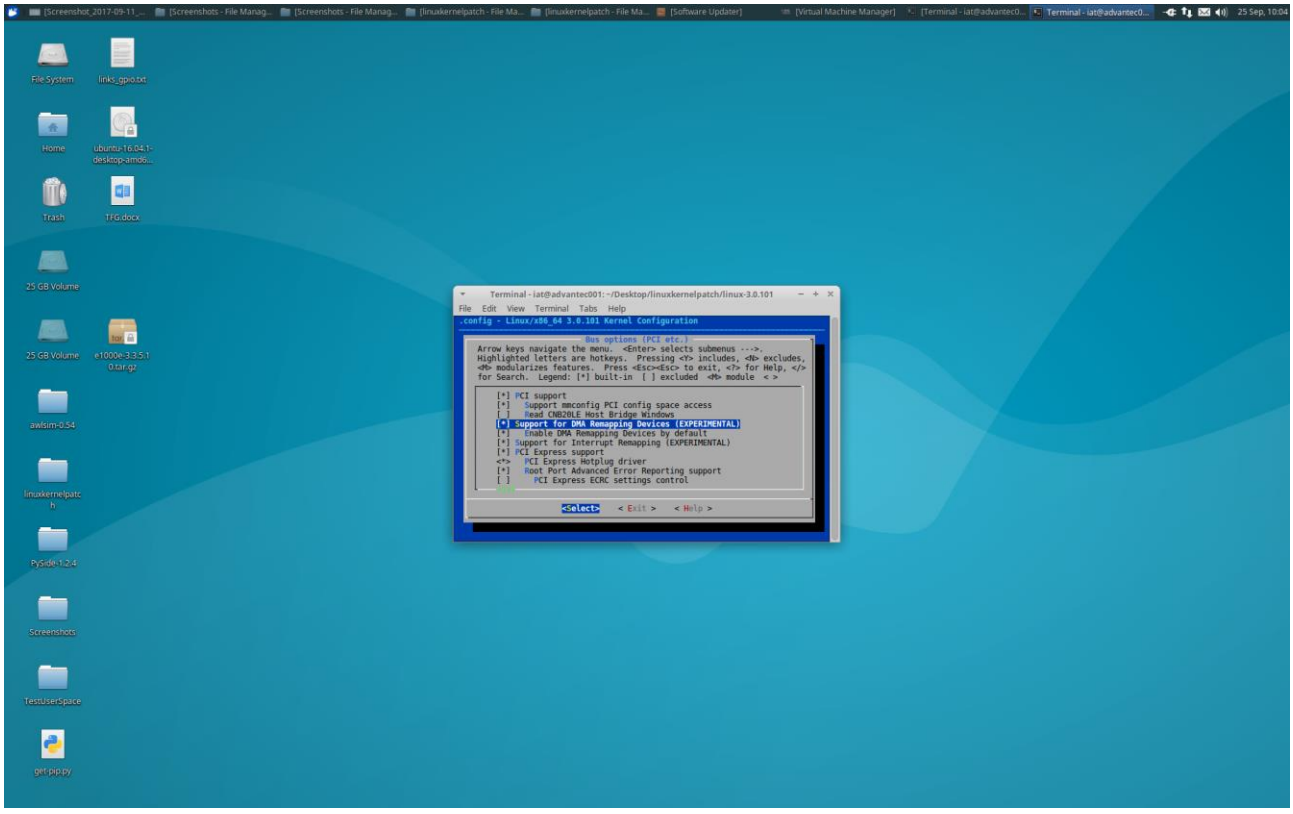


Figure 30 Selecting Kernel configurations

- Select Bus Options→ Enable DMA Remapping Devices by default

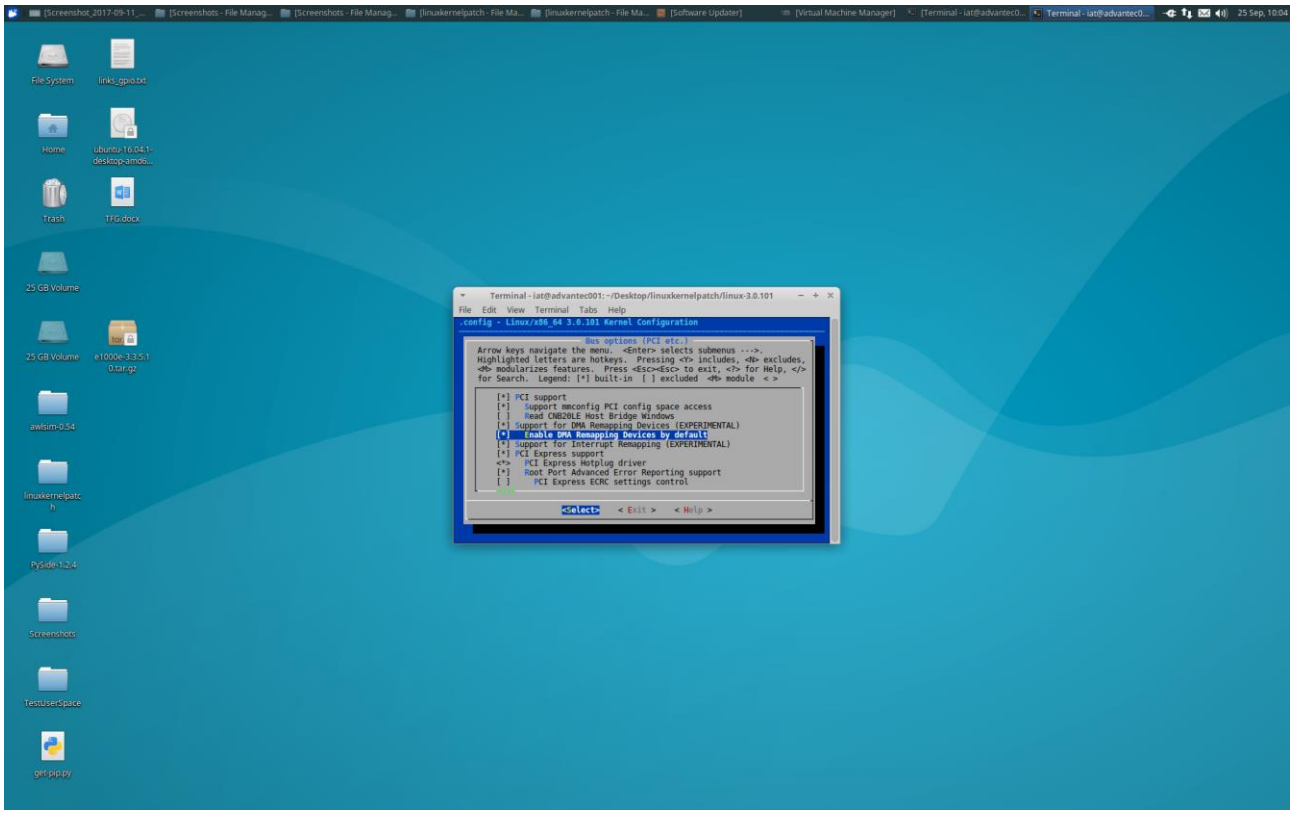


Figure 31 Selecting Kernel configurations

- Select Bus Options→ Enable Support for Interrupt Remapping (EXPERIMENTAL)

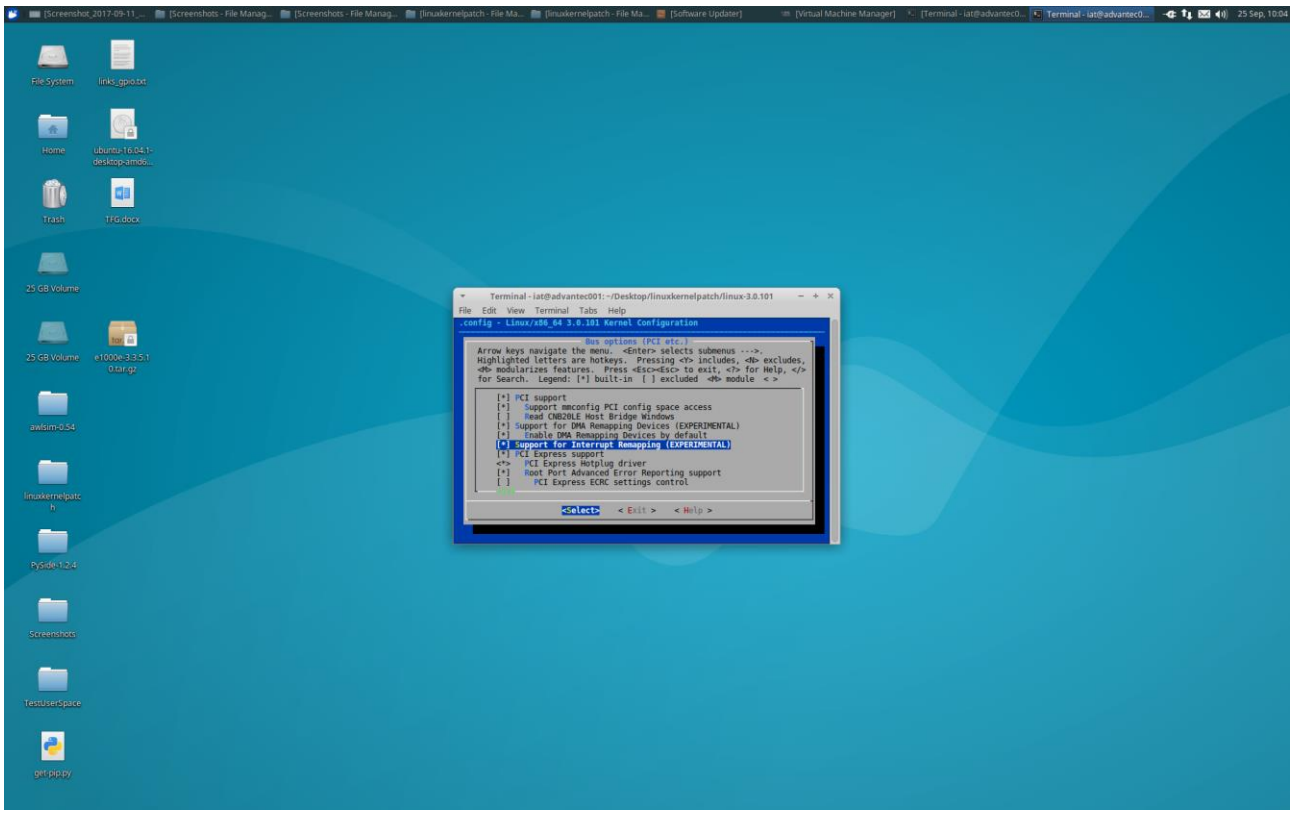


Figure 32 Selecting Kernel configurations

- Select Bus Options→ Enable PCI Stub driver

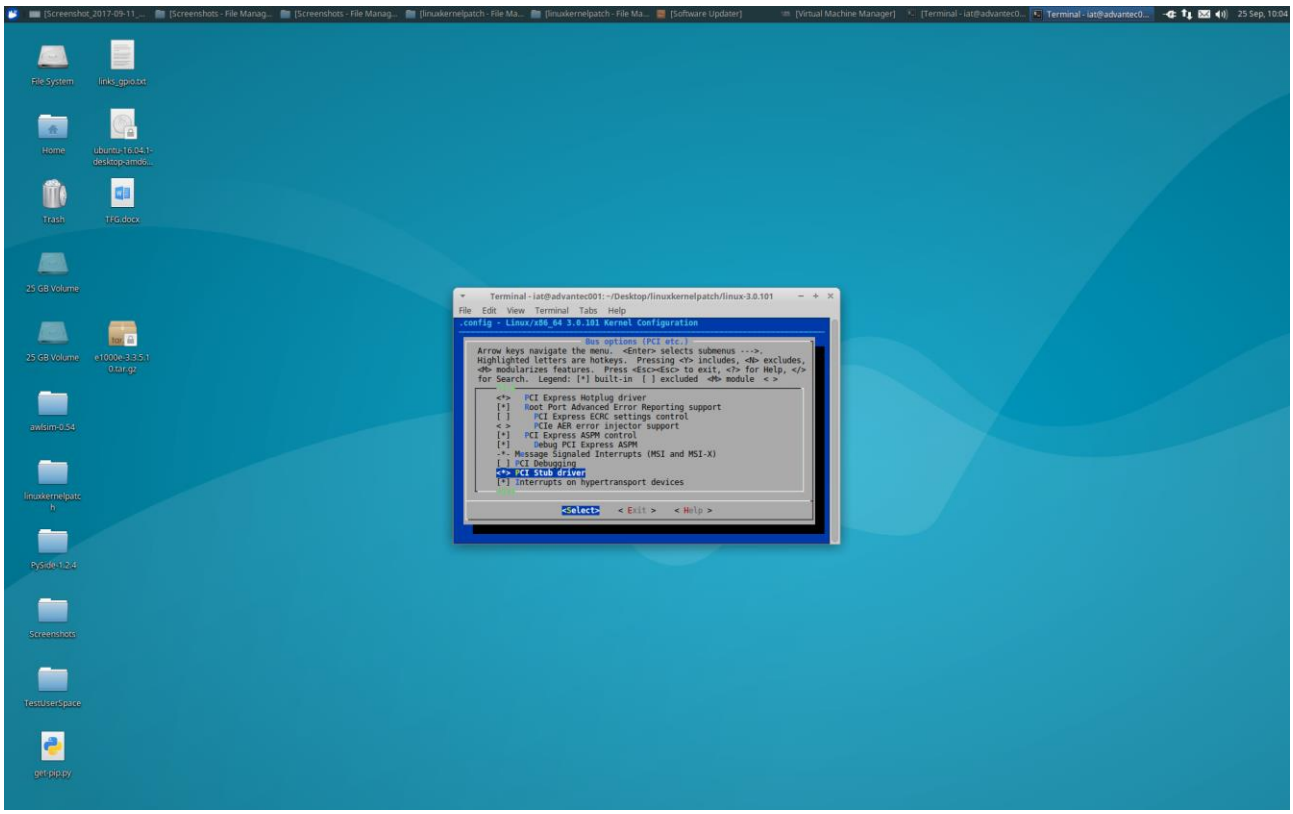


Figure 33 Selecting Kernel configurations

Those configurations have been modified in to use a Fully Preemptible Kernel to work in Real Time and to assign devices with Vt-d in KVM Vt-d Support to communicate the Digital I/O with the guest machine.

Select exit and run:

```
$ make CC=gcc-4.9
$ sudo make modules_install
$ sudo make install
```

The first command is very important, as a 3.0 Kernel version was required, the Kernel must be compiled with a gcc version under gcc-5 one.

Depending on the computer this process may take a while.

After the grub must be updated:

```
$ sudo update-grub
```

Reboot the system and select the new Kernel in the grub.

# Conclusion and Future Investigations

After rebooting the system and using the new Kernel version that was required to virtualize the Digital I/O to read the signal from the guest machine, the system was not working properly, the performance of any kind of program was slow, also several drivers were missing, like the Ethernet one. This problem was solved, even the drivers that were missing were installed, the system was not still working properly.

As it was impossible continue working with this Kernel version that allows the virtualization of the port, the continuation of the project was not possible because the communication between the guest machine and the digital I/O is not achievable. A possible cause may be that the driver set to manage the Digital I/O was not developed by the hardware developer so it may not work as it is expected. Also, the required experimental Kernel options were removed from the new Kernel releases which are not supported anymore.

Even though the communication between the Digital I/O and the Linux system works without problems as it was shown with the oscilloscope images that represents the desired signal and the received signal that Linux gets.

To continue and improve the project the delay must minimized using another Kernel configuration that should be modified.

About the virtualization of the Digital I/O that requires Vt-d, the installation and compilation of a Kernel version under the 3.0 and try if the experimental configurations required work.

Other possibility may include other hardware or the use of other OS to try the port virtualization in the guest machine.

# Bibliography


[1] Controlador Lógico Programable. (10 Oct 2017). *Wikipedia.org.* 23 Oct 2017 from https://es.wikipedia.org/wiki/Controlador_l%C3%B3gico_programable

[2] C.Ramírez Villarreal. (28 Sep 2005). Controladores Lógicos Programables. *Mailxmail.com.* 23 Oct 2017 from http://www.mailxmail.com/curso-controladores-logicos-programables/conceptos-generales-programacion

[3] Autómatas Programables. (Dec 2001). *Sc.ehu.es.* 23 Oct 2017 from http://www.sc.ehu.es/sbweb/webcentro/automatica/WebCQMH1/PAGINA%20PRINCIPAL/index.htm

[4] Ark-1550. (1983-2017). *Advantech.com.* 23 Oct 2017 from http://www.advantech.com/products/1-2jkbyz/ark-1550/mod_780d96df-217e-46ea-ab74-543125943316

[5] Advantech-ARK-1550-Datasheet. *Ecauk.com.* 23 Oct 2017 from https://ecauk.com/files/2014/02/Advantech-ARK-1550-Datasheet.pdf

[6] User Manual ARK-1550. (2014) *Prosoft.ru.* 23 Oct 2017 from https://www.prosoft.ru/cms/f/455727.pdf

[7] A. Maria. (20 Jun 2017). HOWTO setup Linux with PREEMPT_RT properly. *Wiki.linuxfundation.org.* 23 Oct 2017 from https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/preemptrt_setup

[8] Building a rt-preempt kernel for Debian Jessie - x86-amd64. *machinekit.io.* 23 Oct 2017 from http://www.machinekit.io/docs/developing/building-rt-preempt-kernel/

[9] Index of /pub/linux/kernel/projects/rt/. (2017). *kernel.org.* 23 Oct 2017 from https://www.kernel.org/pub/linux/kernel/projects/rt/

[10] RT PREEMPT HOWTO. (17 Jul 2017). *Rt.wiki.kernel.org.* 23 Oct 2017 from https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO#Patching_the_Kernel

[11] S Rostedt & D.V. Hart. (Jun 2007). Internals of the RT Patch. *Kernel.org*.
23 Oct 2017 from https://www.kernel.org/doc/ols/2007/ols2007v2-pages-161-172.pdf

[12] PySide Team. (2014). Building PySide on a Linux System (Ubuntu 12.04 - 14.04). *pyside.readthedocs.io.* 23 Oct 2017 from http://pyside.readthedocs.io/en/latest/building/linux.html

[13] Daniel P.Berrangé (2009-2017). Manage virtual machines with virt-manager. *Virt-manager.org*.
23 Oct 2017 from https://virt-manager.org/

[14] Leersharp. (10 Apr 2016). KVM/Virt Manager. *Help.ubuntu.com.*
23 Oct 2017 from https://help.ubuntu.com/community/KVM/VirtManager

[15] Ubuntu for desktop. (2017). *Ubuntu.com.* 23 Oct 2017 from https://www.ubuntu.com/desktop

[16] P.Bonzini. (10 Sep 2015). Realtime KVM. *Lwn.net.* 23 Oct 2017 from https://lwn.net/Articles/656807/

[17] Rik van Riel. (2015). Real-time KVM from the ground up. *Events.linuxfoundation.org*
23 Oct 2017 from http://events.linuxfoundation.org/sites/events/files/slides/kvmforum2015-realtimekvm.pdf

[18] H.Schwietering. (6 May 2017). KVM *wiki.ubuntuusers.de.*
23 Oct 2017 from https://wiki.ubuntuusers.de/KVM/

[19] GRUB. (20 Jun 2011). *Guia-ubuntu.com.* 23 Oct 2017 from
https://www.guia-ubuntu.com/index.php/GRUB

[20] M.Büsch. (30 Aug 2017). *Bues.ch.* 23 Oct 2017 from https://bues.ch/cms/automation/awlsim.html

[21] D.Kirkland. (2010). dkms - Dynamic Kernel Module Support. Manpages.ubuntu.com.
23 Oct 2017 from http://manpages.ubuntu.com/manpages/zesty/man8/dkms.8.html

[22] Linardv. (18 Mar 2017). DKMS. *Help.ubuntu.com.*
23 Oct 2017 from https://help.ubuntu.com/community/DKMS

[23] Dynamic Kernel Module Support. *Wiki.archlinux.org.*
23 Oct 2017 from https://wiki.archlinux.org/index.php/Dynamic_Kernel_Module_Support

[24] GPIO. (28 Jul 2017). *Linux-sunxi.org.* 23 Oct 2017 from http://linux-sunxi.org/GPIO

[25] GPIO. (27 Oct 2011). *Elinux.org.* 23 Oct 2017 from https://elinux.org/GPIO

[26] GPIO Sysfs Interface for Userspace. *Kernel.org.*

23 Oct 2017 from https://www.kernel.org/doc/Documentation/gpio/sysfs.txt

[27] Priority Scheduling. (2017). *Techopedia.com.*

23 Oct 2017 from https://www.techopedia.com/definition/21478/priority-scheduling

[28] Chrt command: Set / Manipulate Real Time Attributes of a Linux Process. (6 Mar 2008). *Cybercity.biz.*

23 Oct 2017 from https://www.cyberciti.biz/faq/howto-set-real-time-scheduling-priority-process/

[29] Renice command: Change the Priority of an Already Running Process. (25 Feb 2008). *Cybercity.biz.*

23 Oct 2017 from https://www.cyberciti.biz/faq/howto-change-unix-linux-process-priority/

[30] Linux nice command: Run Process with Modified Scheduling Priority (nicenesses). (16 Nov 2007). *Cybercity.biz.* 23 Oct 2017 from https://www.cyberciti.biz/faq/change-the-nice-value-of-a-process/

[31] R.Vidal-Dorsch. (19 Sep 2017). iManager. *Guthub.com.* 23 Oct 2017 from https://github.com/rvido/iManager

[32] TW.Burguer. (5 Mar 2012). Intel® Virtualization Technology for Directed I/O (VT-d): Enhancing Intel platforms for efficient virtualization of I/O devices. *Software.intel.com.* 23 Oct 2017 from https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices

[33] D. Ott. (1 Jan 2015). Understanding VT-d: Intel Virtualization Technology for Directed I/O. *Software.intel.com.*

23 Oct 2017 from https://software.intel.com/en-us/blogs/2009/06/25/understanding-vt-d-intel-virtualization-technology-for-directed-io

[34] KVM contributors. (25 Jul 2016). How to assign devices with VT-d in KVM. *Linux-kvm.org.*

23 Oct 2017 from https://www.linux-kvm.org/page/How_to_assign_devices_with_VT-d_in_KVM

[35] Creating Guests with vir-manager. (2017). *Access.redhat.com.* 23 Oct 2017 from https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/chap-Virtualization_Host_Configuration_and_Guest_Installation_Guide-Guest_Installation_Virt_Manager-Creating_guests_with_virt_manager.html

[36] CONFIG_DMAR: Support for DMA Remapping Devices (EXPERIMENTAL). (2017). *Catee.net.*

23 Oct 2017 from https://cateee.net/lkddb/web-lkddb/DMAR.html

[37] CONFIG_DMAR_DEFAULT_ON: Enable DMA Remapping Devices by default. (2017). *Catee.net.* 23 Oct 2017 from https://cateee.net/lkddb/web-lkddb/DMAR_DEFAULT_ON.html

[38] CONFIG_PCI_STUB: PCI Stub driver. (2017). *Catee.net.* 23 Oct 2017 from https://cateee.net/lkddb/web-lkddb/PCI_STUB.html

[39] The Linux Kernel Archives. (2017). *Kernel.org.* 23 Oct 2017 from https://www.kernel.org/

[40] S. AL Farisi. (Aug 2017). Explanation about PLC. Figure 1 PLC Hardware. *Sttemit.com.* 27 Oct 2017 from https://www.google.de/search?q=plc+structure&rlz=1C1JZAP_esES718ES718&source=lnms&tbm=isch&sa=X&ved=0ahUKEwikrd614NbWAhVREVAKHZpdBJQ4ChD8BQgKKAE&biw=1280&bih=628#imgrc=nZ_EVwFEtv7vcM:

[41] Intel® Core™ i5-4300U Processor 3M Cache, up to 2.90 GHz. (2017). Figure 22 Checking the availability of Vt-d virtualization. *Ark.intel.com.* 23 Oct 2017 from https://ark.intel.com/products/76308/Intel-Core-i5-4300U-Processor-3M-Cache-up-to-2_90-GHz

# Annex

## User Manual 1 (Quick Guide)

User manual to achieve the communication between the Digital I/O and the Linux system in the Host machine

1. Check if the hardware has a Digital I/O.
2. Check if the hardware can support KVM virtualization as well as Vt-d virtualization
3. Download a Kernel version 2.6.28–2.6.39 or 3.0
4. Download a RT-patch version according to the Kernel version that was previously download
5. Patch, install and compile the Kernel with the required options
6. Install Awlsim
7. Install Virt-manager
8. Create a virtual machine to use it as a guest
9. Install the same Kernel version and patch in the guest machine
10. Enable Vt-d virtualization in the BIOS
11. Install the DKMS to load the drivers
12. Install iManager Driver Set using DKMS
13. Write a program using direct access to GPIO to read directly from the Digital I/O pins
14. Generate a push-pull signal of 10 Hz with a signal generator
15. Connect this signal to an oscilloscope and the pin in the device to read it as well as the ground
16. Start the program that was previously implemented
17. See and Compare the results