Quick Start Guide

for

sysWORXX Pi-AM62x
CTR-6xx devices
CTR-8xx devices

Revision History

Version	Date	Changes
01	20.03.2025	Initial version
02	02.05.2025	Chapter 5: Example added

Contents

List of Tables

List of Figures

Reference documents

/1/ SYS TEC: Datasheet CTR-800. L-2933e
/2/ SYS TEC: Datasheet CTR-600. L-2941e
/3/ SYS TEC: Datasheet sysWORXX Pi-AM62x. L-2954e

1 Introduction

The sysWORXX Pi-AM62x is an extremely versatile mini-computer that can be used for a wide variety of applications. Whether as a smart controller, a testing platform for new technologies, or as a starting point for a new project – the possible uses are virtually unlimited. Thanks to its flexibility and the large Linux community, there are numerous guides, tutorials, and software solutions to make getting started easier.

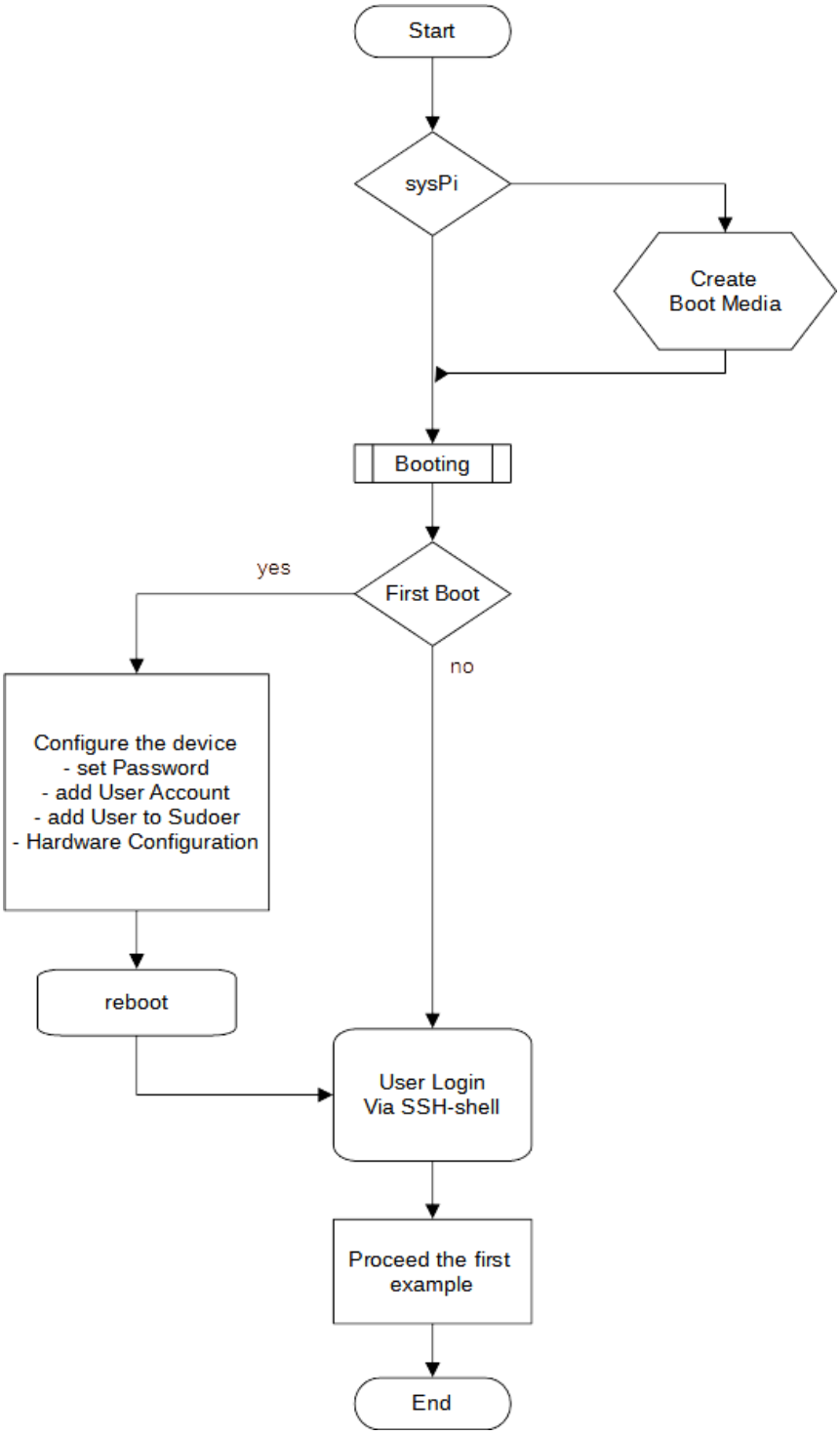
This tutorial focuses on the basic steps to get started with the sysWORXX Pi-AM62x. It covers the following points:
- Creating a boot medium: We'll show you how to prepare an SD card and install the operating system so your device is ready to go.
- Configuring the network interface: You'll learn how to set up the network connection to establish a stable internet or network connection.
- Setting up the IOs: We'll explain how to configure the Pi's inputs and outputs to connect and control sensors, switches, or actuators.
- Creating your first program: Finally, we'll show you how to write a small program using Node-RED to implement your first automation or control functions.

Please note that due to the variety of possible applications, it's not possible to describe all possible uses in detail. Instead, this introduction is intended to provide you with the basic skills you need to implement your own projects with the sysWORXX Pi-AM62x. If you have any questions or special requests, we'll be happy to help!

Figure 1: Top view of the sysWORXX Pi-AM62x

2 Workflow

The following workflow shows a suggested approach for the first attempts.



3 How We Need

This tutorial based on sysWORXX Pi-AM62x. The following figure shows the required interfaces and equipment.

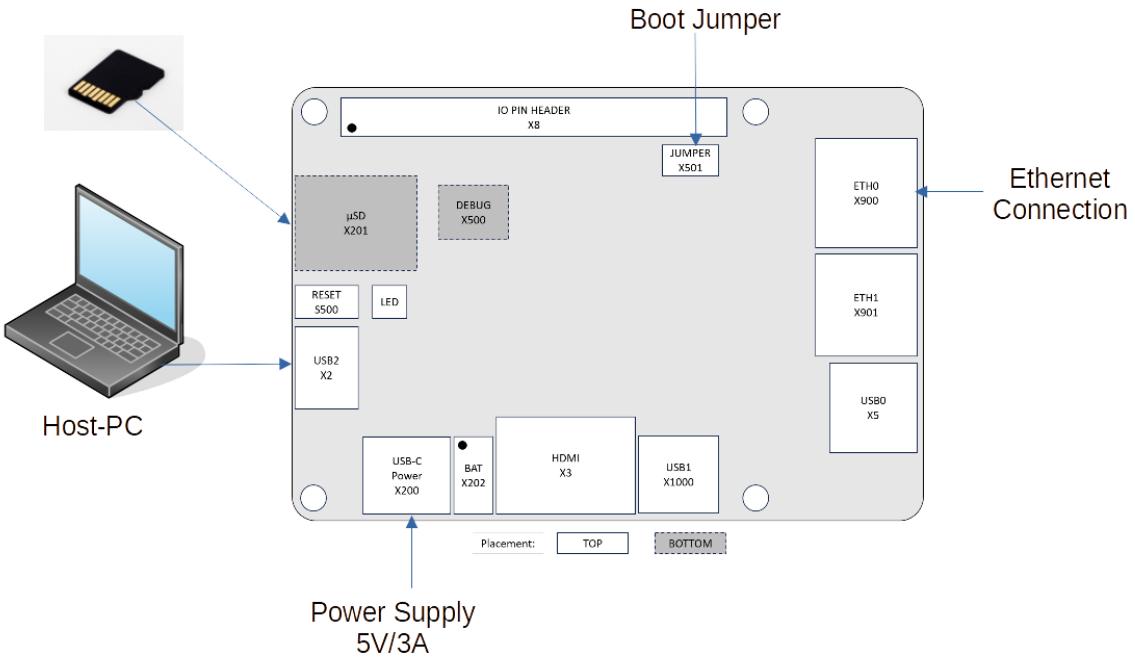


Figure 2: Tutorial setup, interface and connector assignment

Table 1: Description of used Terminals

Terminal	Description
X2	USB-Service interface: connects the Host-PC via virtual COM port
X8	40-pin IO-Header
X200	USB-Power
X201	microSD-Card Holder
X501	Boot Jumper: Jumper selects the boot device. Jumper is set: microSD-Card is Boot Device Jumper is not set: eMMC is Boot Device
X900	Ethernet interface 0 (eth0)
X901	Ethernet interface 1 (eth1)
S500	Reset button
LED	A RGB-LED

4 Getting Started

This chapter describes initial commissioning.

Every sysWORXX device can be used as a headless computer; in the case of the sysWORXX Pi-AM62x, it can also be used as a computer with a display and keyboard. The type of use is determined by the software (boot image). This quick start only describes use as a headless computer, which is certainly the typical application for industrial controllers.

With a headless computer, all settings and configuration work are usually performed using a remote console. For this, the device must be accessible via the network. Only a few initial settings must be performed in advance via a serial console. These include, for example, creating additional users and determining or setting the IP addresses for the network.

The sysWORXX devices are usually delivered with a pre-installed software image. The software images already contain numbers of standard libraries and application software. Users can add additional packages to these. This is most easily done when the device is connected to the Internet.

Only the sysWORXX Pi-AM62x does not have the image pre-installed. For initial setup, a software image for use with a microSD card can be downloaded from the website. The required microSD card should be at least 8 GB in size. A 5 V power supply with a USB-C connector and a USB cable with a USB-C connector for connecting to a host (Linux or Windows) are also required.

After downloading the boot image, it must be saved to the SD card. Various tools are available for this under Linux or Windows (see chapter 4.1).

The module supports two different storage devices as boot media: microSD card and eMMC. A microSD card must be used for initial startup (see chapter 4.2).

After booting, the image can be transferred to the eMMC (see chapter 4.2).

Further settings can then be made to allow network access (see chapter 4.3.3). A serial console such as PuTTY or TeraTerm is required for this (see chapter 4.3.1).

Finally, we will show how to write a small program using Node-RED to implement initial access to the controller hardware (see chapter 5).

4.1 Create a Boot Media (only for sysWORXX Pi-AM62x)

Default images prepared for the device can be loaded from the website. To write the image to an SD card, you will need a PC with an SD card interface or a corresponding USB adapter. To write an image to an SD-Card the following steps should be done, depending on used Host operating system.

The SD-Card Image is provided by SYS TEC as a packed xz-file. The following steps assume the file *"sysworxx-image-default-sysworxx.xz"* is used. Newer versions may have a slightly different file name. Follow the steps to install this file to an SD-Card.

4.1.1 Create a Boot Media under Linux

4.1.1.1 Unpack the Image

```
$ xz -dk sysworxx-image-default-sysworxx-rootfs.wic.xz
```

The result is a file named *sysworxx-image-default-sysworxx-rootfs.wic.*

4.1.1.2 Write the Image to SD card

The SD-Card interface is mounted as a block device. To get a list of available block devices run *lsblk*. Find the correct block device for SD card and then redirect the output of *lsblk* to the device as shown below.

```
$ lsblk
```

Note: For Device Name, use the name displayed for the entry with type = disk.

For example: mmcblk0 and not mmcblk0p1 (this is the first partition)

Then write the image to the SD-Card using the command *dd*:

```
$ sudo dd if=sysworxx-image-default-sysworxx-rootfs.wic \
of=/dev/mmcblk0 status=progress
```

Note: The device name (e.g. dev/mmcblk0) must be specified and not the partition.

Then

```
$ sudo sync
```

4.1.2 Create a boot media under Windows

4.1.2.1 Unpack the Image

Unpack the packed *.xz-file using the tool 7-zip.

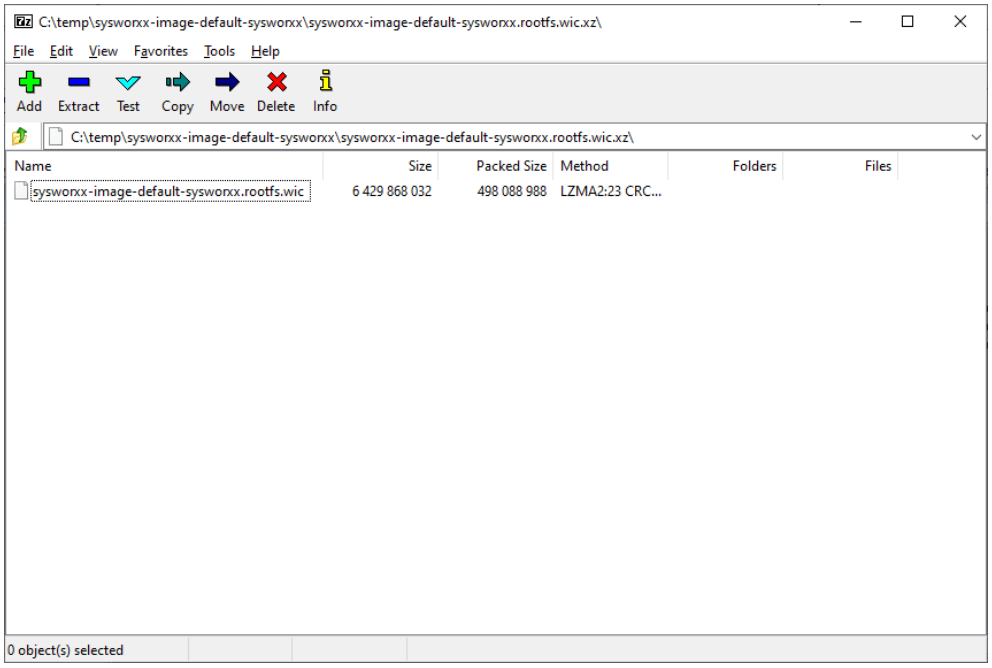


Figure 3: Unpack the Image under Windows using 7-zip

The result is the file *sysworxx-image-default-sysworxx.rootfs.wic*.

4.1.2.2 Prepare the SD-Card

Insert and prepare the SD-Card. The SD-Card must have one partition.

To remove multiple partitions on an SD card in Windows 10, you can use the built-in Disk Management. Here are step-by-step instructions:

Connect SD card: Insert your SD card into your computer's card reader.

Open Disk Management: Press Windows key + R, type *diskmgmt.msc* and press Enter. This opens Disk Management.

Find the correct SD card: In the list of disks, you can identify your SD card by its size and label. Make sure you select the correct disk to avoid losing data.

Delete partitions: Right-click the partition you want to remove and select "Delete Volume." Confirm the warning message. Repeat this step for all partitions you want to remove.

Important: Deleting partitions will delete all data on the SD card. Make sure you back up all important data beforehand.

4.1.2.3 Write the Image to SD-Card

Download an install the tool *"Win32 Disk Imager"*: https://sourceforge.net/projects/win32diskimager/

This tool will be used to copy the image file to the SD-Card image.

Run *"Win32 Disk Imager"*

1. Choose the uncompressed SD-Card image
2. Choose the drive letter of the SD-Card
3. Click on the "Write" button to write the image to the SD-Card

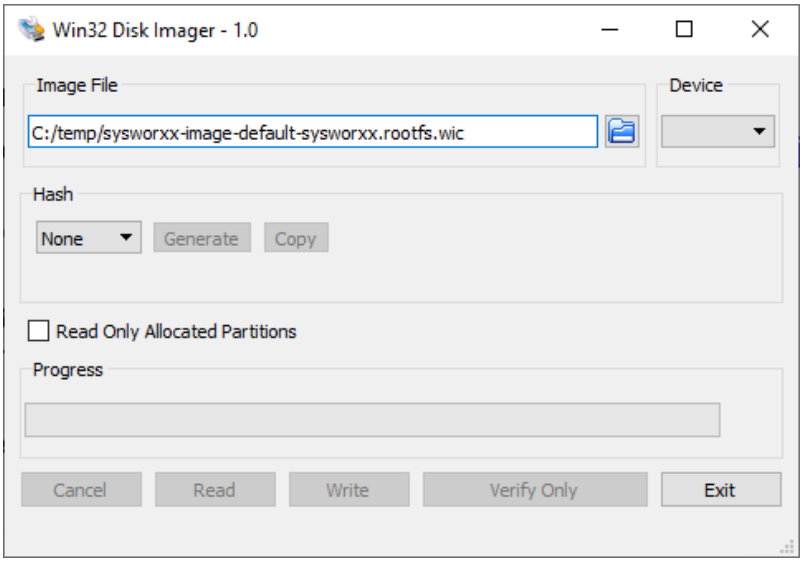


Figure 4: Write SD-Card Image using Win32 Disk Imager

When the image has been completely written to the SD card, eject the card using the context menu in Explorer.

4.2 Booting the device (only for sysWORXX Pi-AM62x)

The sysWORXX device can boot from different sources, either from a microSD card or from the on-board eMMC. The boot medium can be selected via the jumper X501.
If the Jumper is set, the sysWORXX device boots from the on-board eMMC. If the jumper is not set, the sysWORXX device boots from a plugged in microSD card. For the first boot from a plugged in microSD card may not set the jumper.

4.2.1 Booting device from SD-Card

Insert the microSD-Card and remove the Boot Jumper X501.

4.2.2 Booting device from eMMC

To boot the software from an eMMC, the image must first be installed on the eMMC. To do so, follow these steps:

Preparation
Insert the microSD-Card and boot the sysWORXX device as described in chapter 4.2.1. Then, create a user as described in the chapter 4.3.2.

First copy the Installer from the host PC to the sysWORXX device. After the connection is established via scp-command, the sysWORXX device user password must be entered.

```
scp E004481-sysworxx-image-default-emmc-installer-* pi-user@sysworxx:/tmp
pi-user@192.168.15.115's password: *********
```

Next, run the following commands in the serial console on the sysWORXX device

1. Change to root user:

```
$  su root
Password: ********
```

2. Make the Installer executable:

```
root@sysworxx:/#chmod +x /tmp/E004481-sysworxx-image-default-emmc-installer-*
```

3. Run the Installer:

```
root@sysworxx:/tmp/E004481-sysworxx-image-default-emmc-installer-*
```

Unpacking and installing on the eMMC takes a few seconds.

4. Power off the sysWORXX device, remove the microSD-Card, set the Boot Jumper X501 and power-on the sysWORXX device.

4.3 Configure the Device

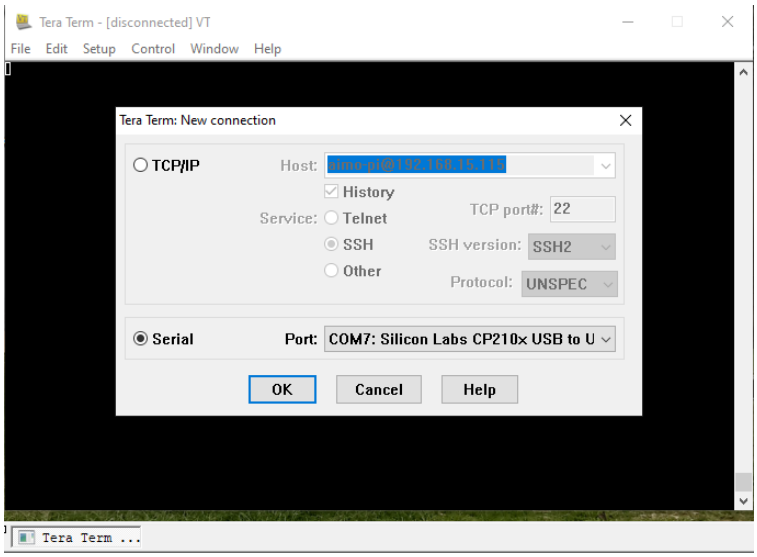
After first boot of the device, the initial settings can then be configured via the serial console. For network access, it will need to configure a user account and a network connection in the operating system.

4.3.1 Terminal Connection via Serial

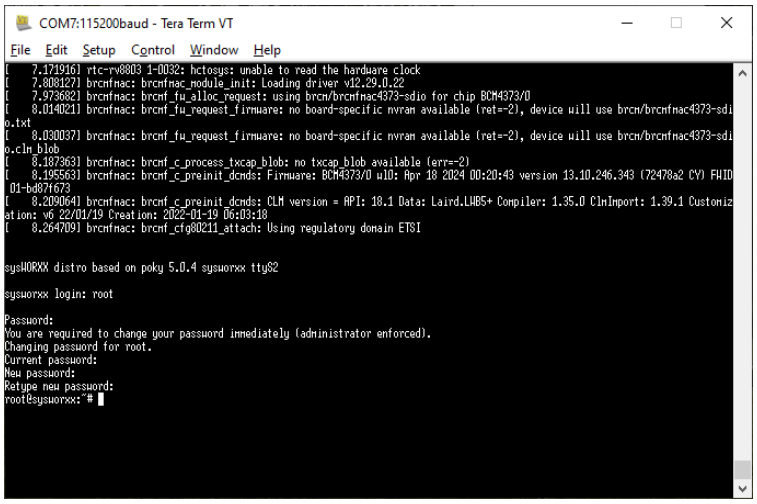
Various terminal programs are available. Two examples are presented here.

4.3.1.1 Option 1: Using Tera term

Start *Tera Term* and select for a new connection the serial COM port.



Bitrate should be 115200 Baud, 8N1.

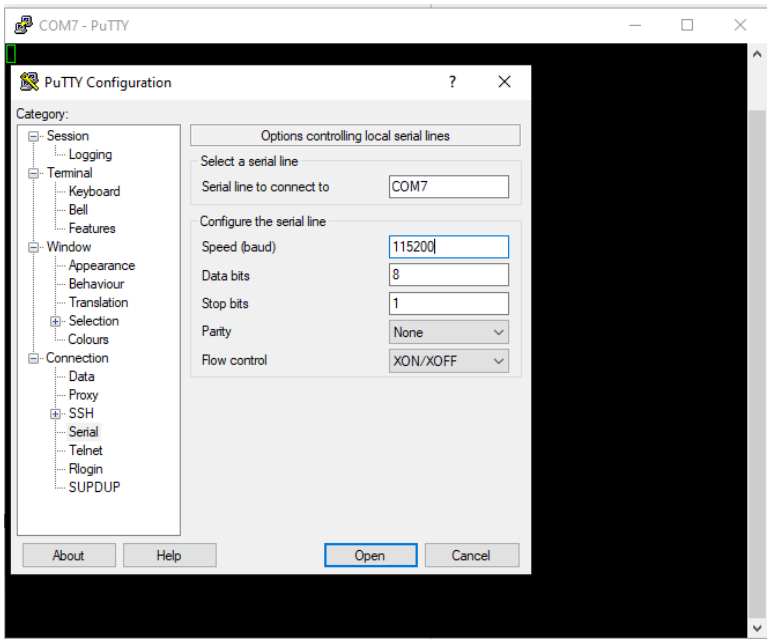


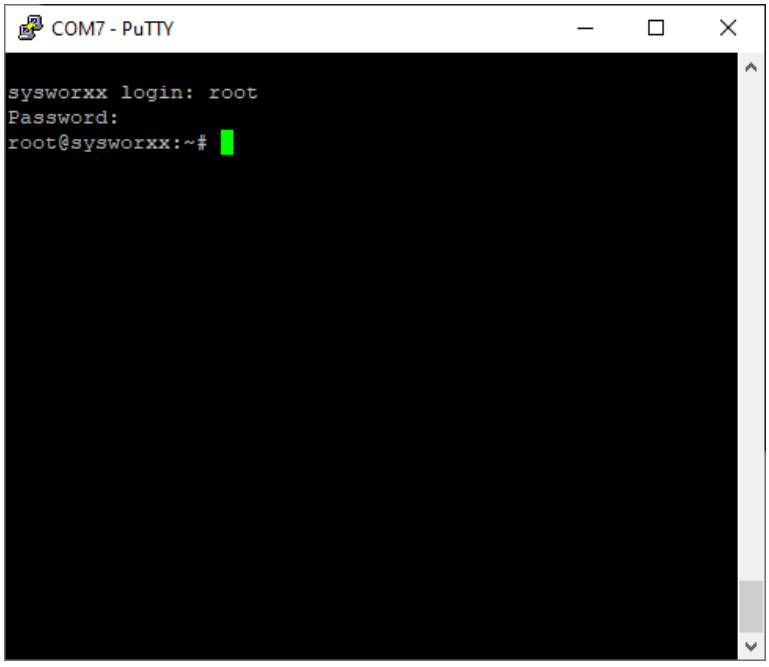
Note: The predefined user “root” is used for the administration (system configuration, user administration, software updates etc.). The default password is “root” and must be changed after first login.

```
sysworxx login: root
Password:
You are required to change your password immediately (administrator enforced).
Changing password for root.
Current password:
New password:
Retype new password:
root@sysworxx:~#
```

4.3.1.2 Option 2: Using PuTTY

A free, open-source program is "PuTTY": https://www.putty.org/





4.3.2 User Accounts

All user accounts listed in Table 2 are predefined upon delivery. Those allow for a login to the command shell.

Table 2: Predefined user accounts

User name	Password	Remark
root	root	Predefined root user for the administration (system configuration, user administration, software updates etc.) **Attention**: After the first login, it will be asked to change the password.

Hint: To secure the device, users must change all predefined passwords (see 4.3.2.2). If users are not needed, they can be disabled or removed except the “root” user. Additionally, SSH can be configured for public key authentication for even better security. “Root” cannot be used as login via SSH, a new user must first be created via serial (see 4.3.2.1).

As a root user, you cannot connect to the device over the network. It's a good idea to restrict root access to a network connection because these privileges are very powerful. If someone gains unauthorized access to root privileges, they could take control of the entire system, leading to security risks. Restricting root privileges reduces the risk of malicious actions being performed if someone tries to tamper with or exploit the system. This keeps your system more secure and stable. To connect to the device via SSH, you must create a user account and a password for that user. To create a user account you must have root privileges.

4.3.2.1 Adding and Deleting User Accounts

Adding and deleting user accounts requires the login to the command shell via serial console.

Adding a new user account takes place via Linux command *"useradd"*. To create a new user, one can use the command *"useradd"* as follows:

```
useradd [options] [username]
```

To **delete** an existing user account from the device, Linux command *"userdel"* plus the respective user name must be used:

```
$ userdel -fr <UserName>
```

To get a full list of options for one of the commands, run it with the “*--help*” argument.

4.3.2.2 How to Change the Password for User Accounts

Changing the password for user accounts requires the remote login to the command shell.

To change the password for an existing user account, Linux command *"passwd"* plus the respective user name must be entered:

passwd <username>

The following example shows how to create and delete a user account for the user pi-user.

```
$ useradd pi-user

$ passwd pi-user
New password:********
Retype new password:********
passwd: password updated successfully
```

Now enter the user password. After that, you can connect over the network using SSH or change the user in the serial console.

```
$ su pi-user
```

To find out which user you are logged in under, you can use the following command.

```
$ whoami
pi-user
```

4.3.2.3 How to Add the User to sudoers group

By default, an added user does not have root privileges. However, certain commands require root privileges to execute (for example: to configure Device Tree Overlay). This can be temporarily granted using the sudo command. In the Yocto Linux used, sudo rights are controlled by default via membership in the sudoer group. To achieve this, proceed as follows.

```
root@sysworxx:~# echo "pi-user ALL=(ALL:ALL) ALL" > /etc/sudoers.d/00_pi-user
root@sysworxx:~# logout

sysworxx login: pi-user
Password: **********
```

4.3.2.4 Use an SSH key for any user login

For security reasons, root login via SSH with username and password is disabled (PermitRootLogin is set to "prohibited-password"). To still connect to a root user, a private key must be used. The public key must be stored on the sysWORXX device, and the private key must be stored on the host PC and then used when establishing the SSH connection.

To log in via SSH as any user, a public SSH key must be provided in the sysWORXX device user's home directory (~/.ssh). Log in as the desired user at a serial terminal and follow these steps to provide a key on the device:

On Host-PC generate an ssh-key using ssh-keygen (or use an already existing key if wanted).
(see also: https://www.man7.org/linux/man-pages/man1/ssh-keygen.1.html)

```
ssh-keygen
```

Open the *.pub file and add its contents to clipboard (copy).

On sysWORXX device create authorized-key files which contains identities (aka SSH identities) to login.

```
$  mkdir ~/.ssh && chmod 700 ~/.ssh
$  cat > ~/.ssh/authorized_keys
```

After executing the last line (cat ...), the terminal waits for input from stdin. Copy/Paste the public key to the terminal application and quit input with CTRL-D.

Restart the sysWORXX device:

```
$  reboot
```

To test the ssh-connection open Windows Power Shell and type:

```
$ ssh <user>@<IP-Adresse of sysWORXX device>
```

And answer the security question.

Example:

```
PS> ssh root@192.168.15.147
The authenticity of host '192.168.15.147 (192.168.15.147)' can't be established.
ED25519 key fingerprint is SHA256:….
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.15.147' (ED25519) to the list of known hosts.
root@sysworxx:~#
```

4.3.3 Network Configuration (Ethernet)

To connect to a network, the device needs an IP address in the respective network. The best way to configure network settings is with the NetworkManager tool. NetworkManager is a utility that facilitates the management of network connections on Linux systems. It was developed to simplify configuration and switching between different network types—such as WLAN, Ethernet, VPN, or mobile data. The Network Manager is available as a graphical tool if a desktop is available, but also as a command-line tool. We demonstrate how to use the command-line tool here.

NetworkManager automatically detects available networks and establishes connections without requiring you to manually edit configuration files. NetworkManager ensures a seamless connection when network conditions change (e.g., when switching from WLAN to Ethernet).

Establishing a connection:
You can set up new connections, edit existing ones, or disconnect them using the nmcli command-line tool.

Managing networks:
It allows you to save various network settings so that your system automatically connects to known networks.

Troubleshooting:
NetworkManager provides tools for diagnosing and troubleshooting network problems.

In short, NetworkManager is a central tool for simplifying network management under Linux, especially if you frequently switch between different networks or prefer simple operation. The next sections show how to configure an Ethernet interface.

4.3.3.1 Config IP Address via DHCP

DHCP stands for Dynamic Host Configuration Protocol. It is a network protocol that automatically assigns IP addresses, subnet masks, default gateways, and other important network settings, such as name servers, to devices on a network. This is extremely convenient because it eliminates the need to manually assign an IP address to each device. DHCP ensures that devices can connect to the network quickly and easily without conflicts or connection problems. DHCP is therefore included as the default setting in the provided software image for the devices.
A DHCP server must be present on the network. To test this, connect the device to the network and enter the nmcli command. If a DHCP server is present, the settings currently in effect for the selected interface will be listed. Here is an example using eth0.

```
root@sysworxx:~# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ab57b0b1-6081-3b23-86db-3d2845baeeea  ethernet  eth0
lo                  b61f03e0-ce93-467a-9ea4-b4254ff51314  loopback  lo
Wired connection 2  355a91e2-d7ca-3abd-bed5-7e01124dc492  ethernet  --
```

The IP address for the connection on the eth0 interface can be determined as follows:

```
root@sysworxx:~# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 44:d5:f2:a0:5b:f6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.15.115/24 brd 192.168.15.255 scope global dynamic
noprefixroute eth0
       valid_lft 691080sec preferred_lft 691080sec
    inet6 fe80::ee3d:a8d7:ac1d:f4d9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

The assigned IP (v4) address is 192.168.15.115.

4.3.3.2 Config IP Address Manually

Configuring static IP addresses, on the other hand, is necessary in situations where devices on the network require a fixed, unchanging IP address. This is the case, for example, with servers, printers, or other devices that need to be permanently accessible. This ensures that these devices are always found under the same address, which makes administration and connection easier. We'll show you how to do this in the following chapter for Ethernet interface eth1.

To perform the configuration with the nmcli you need a terminal connection (via serial or via eth0), and you need the connection name of the interface eth1. To get this, type *nmcli connection show*:

```
root@sysworxx:~# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ab57b0b1-6081-3b23-86db-3d2845baeeea  ethernet  eth0
lo                  b61f03e0-ce93-467a-9ea4-b4254ff51314  loopback  lo
Wired connection 2  355a91e2-d7ca-3abd-bed5-7e01124dc492  ethernet  --
```

The connection name can be found in the "Name" column, in our case "Wired connection 2".

Next, the various settings are made. The configured settings are stored immediately in the file 'Wired connection 2.nmconnection' under the path /etc/NetworkManager/system-connections/.

Set the static IP address and Subnet mask with CIDR notation:

```
$  nmcli connection modify "Wired connection 2" ipv4.addresses 192.168.1.100/24
```

Set the Standard Gateway:

```
$  nmcli connection modify "Wired connection 2" ipv4.gateway 192.168.1.1
```

Enables manual (static) IP configuration:

```
$  nmcli connection modify "Wired connection 2" ipv4.method manual
```

Set to automatic configuration after reboot and connection up:

```
$  nmcli connection modify "Wired connection 2" connection.autoconnect yes
```

Activate the connection:

```
$  nmcli connection up "Wired connection 2"
```

Check if the configuration will be done:

```
root@sysworxx:~# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ab57b0b1-6081-3b23-86db-3d2845baeeea  ethernet  eth0
Wired connection 2  355a91e2-d7ca-3abd-bed5-7e01124dc492  ethernet  eth1
lo                  6f7eb2e5-b161-4299-b093-081da6cfd5ef  loopback  lo
```

To switch back from static to DHCP these steps must be taken:

```
$  nmcli connection del "Wired connection 2"
$  reboot
```

4.3.4 Terminal Connection via SSH

The default setting in the image is such that a remote connection over the network cannot be established by a root user. To enable this, there are the following methods:

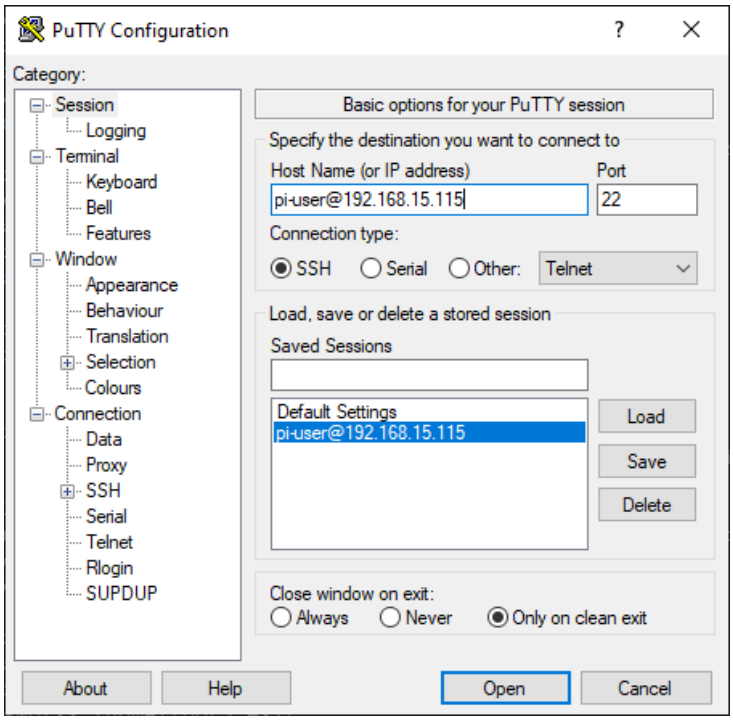
- Use a user without root privileges, then establish the remote connection using this user and switch to the root user using the su command.

- Use an SSH key to access the root user or any other user.

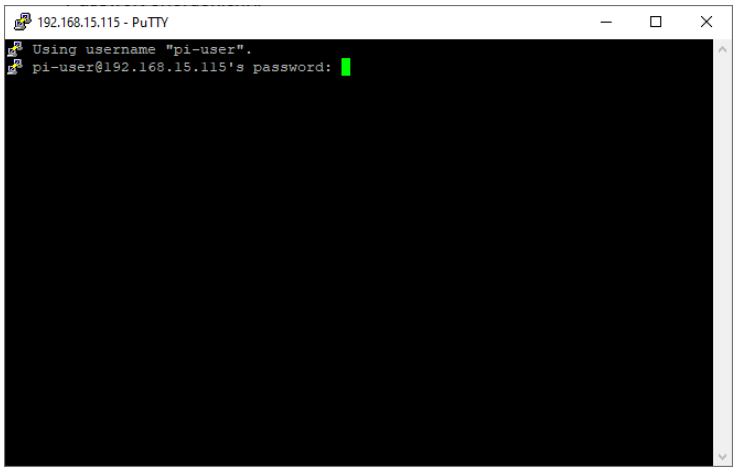
4.3.4.1 Use a user without root privileges

If no other user exists, please create one as described in chapter 4.3.2. SSH will then be used to connect to the device.

After starting the terminal program, you must first enter the IP address of the device in the "Host Name (or IP address)" field of the configuration dialog (see chapter 4.3.3 to get the IP address).



After clicking "Open", the connection is established. It will first need to log in with a password.



To log off the user, to close the terminal and end the connection, ‘exit’ must be entered.

If root privileges are required, use the su command to change to the root user.

```
pi-user@sysworxx:~$ su
Password:
root@sysworxx:/home/pi-user# cd ~
root@sysworxx:~#
```

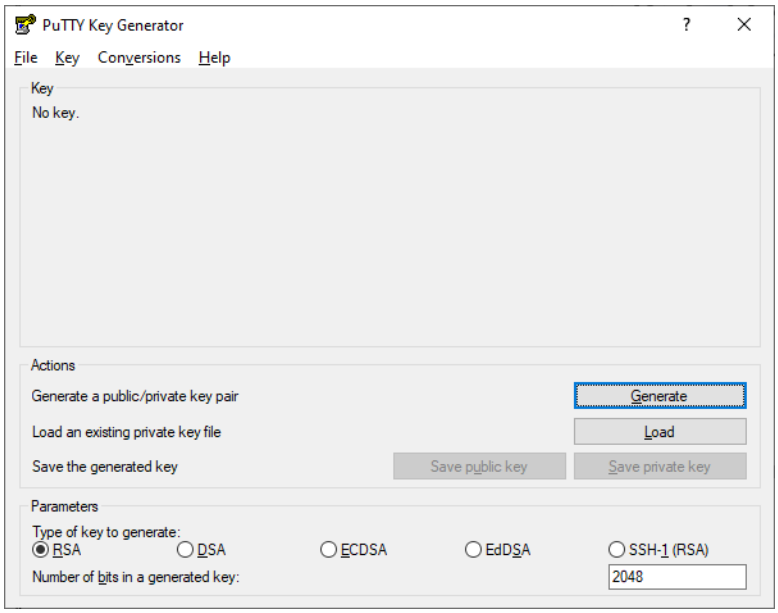
4.3.4.2 Use an SSH key for root user

For security reasons, root login via SSH with username and password is disabled (PermitRootLogin is set to "prohibited-password"). To still connect to a root user, a private key must be used. The public key must be stored on the sysWORXX device, and the private key must be stored on the host PC and then used when establishing the SSH connection.

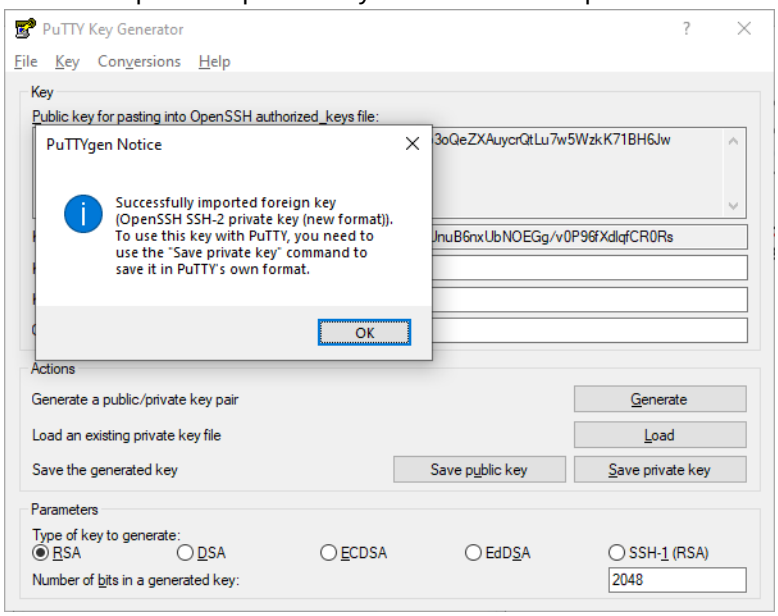
To be able to login via SSH as user root, proceed as descripted in chapter 4.3.2.4. Then convert the private key located at the Host to a PuTTY Private Key as descripted in the next section.

To establish an SSH connection using a private key, the key must be in the PuTTY Private Key (*.ppk) file format. The generated key uses the OpenSSH format. This format can be converted to *.ppk using the PuTTYgen tool. To do so, follow these steps:

Open PuTTYgen:



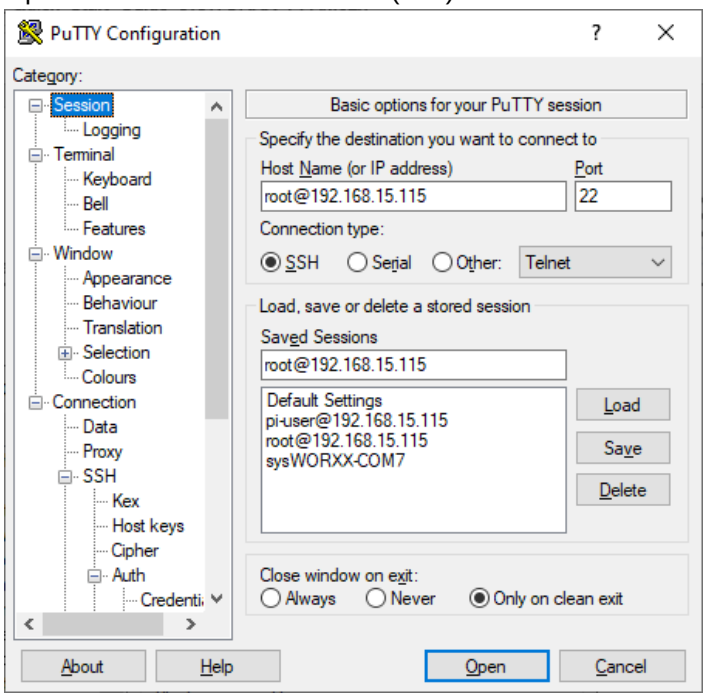
Load the OpenSSH private key file and enter the password



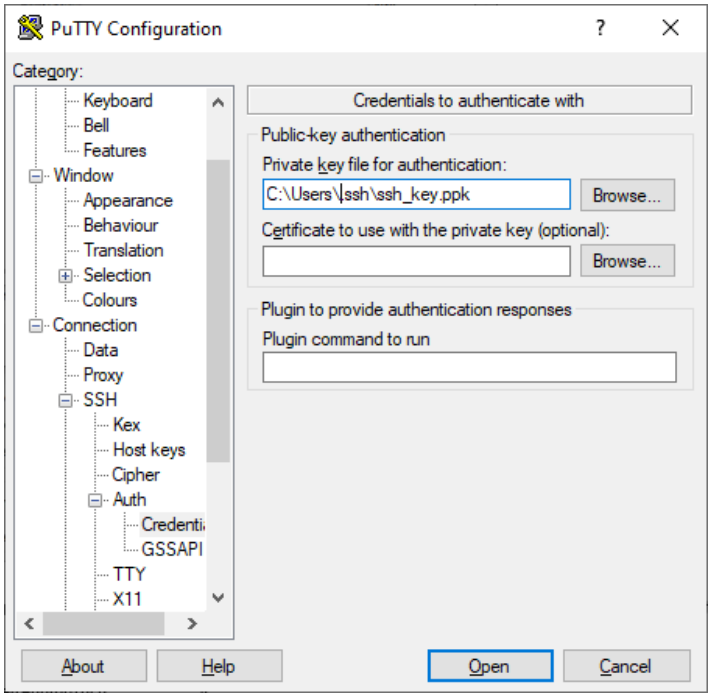
Press button “Save private key” and store the file as ssh_key.ppk. Close the PuTTYgen tool.

To set up the connection via PuTTY, proceed as follows:

Open PuTTY and enter the user (root) and IP address.



Under the SSH category, open the Credentials to authenticate dialog in the Auth entry. Select the private key file and click Open.



The connection is established, the password must be entered.

4.3.5 Hardware Configuration (sysWORXX Pi-AM62x)

This chapter describes the configuration of the on-board IO-Header. This step is not necessary for devices in the CTR family. The necessary settings are already implemented in the software.

The sysWORXX Pi-AM62x has a 40-pin IO-Connector. The pin assignment is described in chapter 6. The pins can be configured as either general-purpose input (DI), general-purpose output (DO) or as a special alternate function. The configuration will be done by loaded Device Tree. Some Device Trees are ready to use in general or for use with a specific HAT. Selecting and configuration of a special Device Tree is described in chapter 4.3.5.3.

A general-purpose output set to high carries an output voltage of 3.3V. A voltage of 3.3V at a general-purpose input is read as high, a voltage of 0V as low. See in /3/ for details of voltage specifications.

4.3.5.1 The Linux Device Tree Concept

Linux requires a device tree to control the hardware. A device tree in Linux is a data structure that provides the operating system with information about the system's hardware. Since Linux runs on many different hardware platforms, without this information it cannot know which devices are present and how to communicate with them. The Device Tree can only be configured as a root-user.
Device Tree Overlays (DTBO) are a practical way to flexibly customize the hardware configuration of a Linux system, especially on embedded systems like the sysWORXX Pi-AM62x, without permanently changing the main Device Tree file.
An overlay is a separate Device Tree file containing additional or modified hardware information. They are applied at boot time to enable or configure specific hardware components. Multiple overlays can be applied simultaneously, provided that no pins are used simultaneously by the overlays.

The sysWORXX Pi-AM62x by default only has no inputs and no DTBO is set. Via Device Tree Overlays more can be configured. The script *dtbo-setup* is provided to simplify DTBO configuration.

Applying Device Tree Overlays is performed by the *u-boot* bootloader. The DTBO configuration is saved in *u-boot environment*. Therefor a reboot is needed to apply the new Device Tree Overlay setting. The environment lives in the second partition (mmcblk#p2) on the respective boot device. This partition is mounted and the environment is available under: /boot/u-boot/uboot.env. Available DTBO configurations:

- sysWORXX Pi Generic (GPIOs, UART, SPI, I2C via IO-Pin Header)
- sysWORXX Pi HAT – Smart Metering
- sysWORXX Pi HAT – Industrial Communication

Attention: If the device tree overlay is set on the SD card and then booted from the eMMC, the device tree setting is no longer available. It is necessary to set the device tree overlay for the eMMC again.

4.3.5.2 List Available Device Tree Overlays

Some Device Tree Overlays are ready to use in general or for use with a specific HAT.

```
$ dtbo-setup ls

Available DTBO:
  k3-am625-systec-sysworxx-pi-hat-industrial-communication.dtbo
  k3-am625-systec-sysworxx-pi-hat-smart-metering.dtbo
  k3-am625-systec-sysworxx-pi-sysworxx-io-default.dtbo
```

4.3.5.3 Load Generic Device Tree Overlay

The generic sysWORXX Pi-AM62x Device Tree Overlay is required for the further application examples in this guide. Select the desired Device Tree Overlay. After that the device must be reboot.

```
$ dtbo-setup set k3-am625-systec-sysworxx-pi-sysworxx-io-default.dtbo

$ reboot
```

4.3.5.4 Reset Device Tree Overlays

Running the command dtbo-setup set without any selected Device Tree Overlay disables the Device Tree Overlay.

```
$ dtbo-setup set
$ reboot
```

5 First example using Node-RED

The previous chapters described the hardware and network configuration. This chapter shows how to use a dashboard to read signal states from the IO header and control output states. The example assumes that the generic Device Tree Overlay has been loaded. This is already preinstalled for the CTR-xxx devices; for the sysWORXX Pi-AM62x, it must be installed as described in the chapter 4.3.5.3. The application itself uses Node-RED.

Node-RED is an open-source, flow-based development environment specifically designed for visual programming of IoT and automation projects. Node-RED uses a browser-based interface where you can connect nodes using drag-and-drop to create flows. This makes it easy to understand, even for beginners.

The applications consist of individual, modular components (nodes) that exchange and process data. This simplifies the organization and maintenance of complex systems. There is a wide selection of pre-built nodes for various tasks, such as HTTP requests, MQTT, databases, email, WebSockets, and much more. You can also develop your own nodes.

Node-RED integrates easily with other systems and services, such as cloud services, smart home systems, or local servers. Thanks to a large community and a wide range of available plugins, Node-RED can be easily extended and adapted to your needs. As an open-source project, Node-RED is freely available and is actively developed.

It runs on many operating systems, including Linux, Windows, macOS, and embedded systems such as Raspberry Pi. The intuitive interface and simple configuration make it attractive even for non-programmers.

Node-RED documentations: https://nodered.org/docs/

5.1 Start the Node-RED Service

At first we start Node-Red as a service. Setting up Node-RED as a service means it runs automatically in the background at system startup, without requiring manual startup. This ensures that flows are always available, even after rebooting. It also makes the service easier to manage, such as stopping, starting, or restarting, simplifying maintenance and troubleshooting. Overall, this makes using Node-RED on your Linux system more stable and convenient.

```
$ systemctl start node-red
```

With the following commands ensure that the service will be stop.

```
$ systemctl stop node-red
$ reboot
```

The following command ensures that the service starts automatically every time you reboot.

```
$ systemctl enable node-red
$ reboot
```

You can use the following command to check whether the service is running.

```
root@sysworxx:~# systemctl status node-red

* node-red.service - Node-RED
     Loaded: loaded (/usr/lib/systemd/system/node-red.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-04-25 06:51:04 UTC; 24s ago
   Main PID: 474 (su)
      Tasks: 0 (limit: 2060)
     Memory: 1.8M (peak: 1.8M)
        CPU: 56ms
     CGroup: /system.slice/node-red.service
             > 474 /bin/su root -c node-red

Apr 25 06:51:04 sysworxx systemd[1]: Started Node-RED.
Apr 25 06:51:05 sysworxx su[474]: Successful su for root by root
Apr 25 06:51:05 sysworxx su[474]: + ??? root:root
Apr 25 06:51:05 sysworxx su[474]: pam_unix(su:session): session opened for user root(uid=0) by
(uid=0)

root@sysworxx:~#
```

5.2 Start the Node-RED User Interface

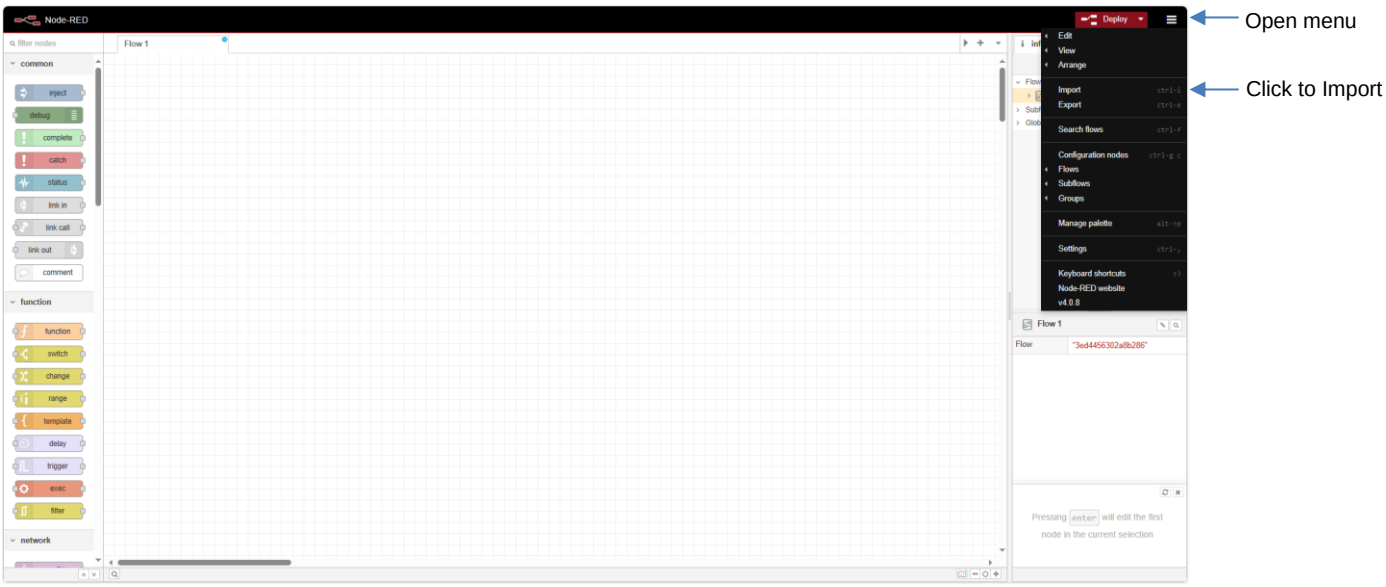
In Node-RED, the interface is often referred to as the "user interface" or "UI." It consists of various components, such as the editor, where you can create and edit flows, and the dashboard, which is used for visualizing and interacting with data. The interface is designed to be intuitive, making it easy to manage automations and data flows. Node-RED requires no tools on the host system. The automations and data flows can be fully deployed and tested on the target system. To connect to the target system, the following URL must be used.

```
Node-RED URL:  http://<Target-IP>:1880
```

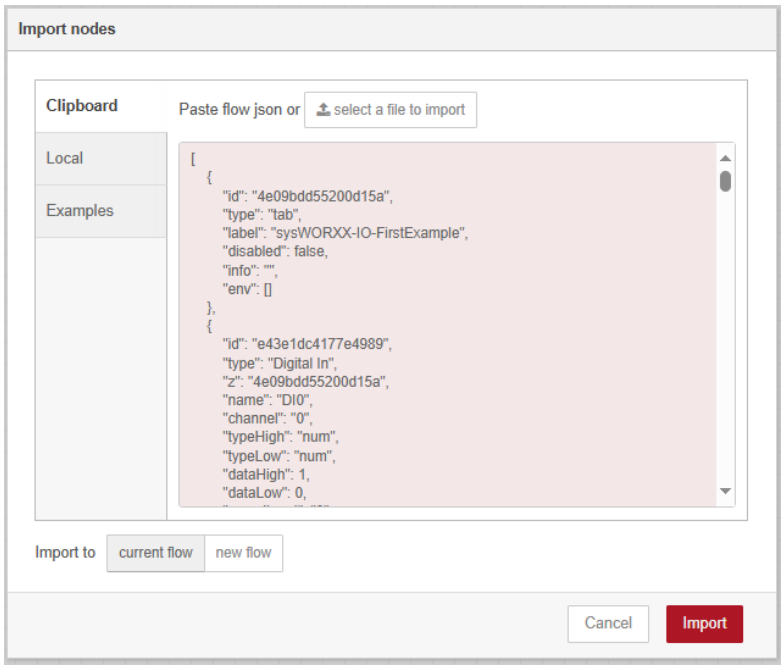
Figure 5: Node-RED User Interface (Editor)

5.3 Downloading Flows

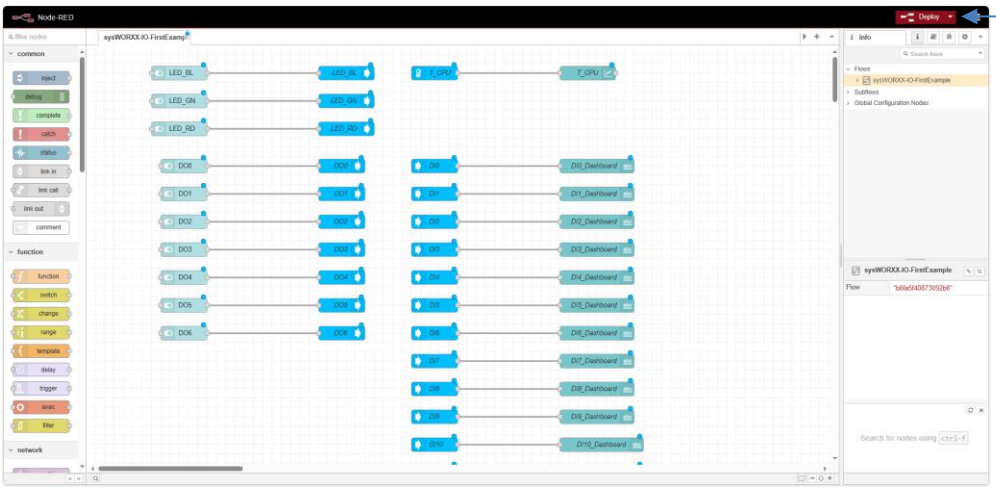
Flows can be exported and imported. The device specific flows for this example can be loaded from the web page and imported to the device.



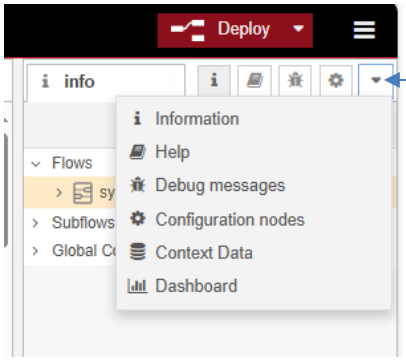
Select the Example flow and import. If you ask when press import copy.

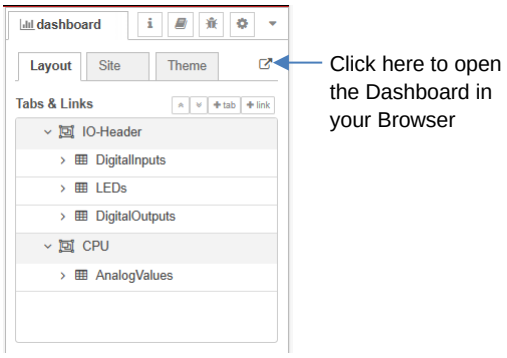


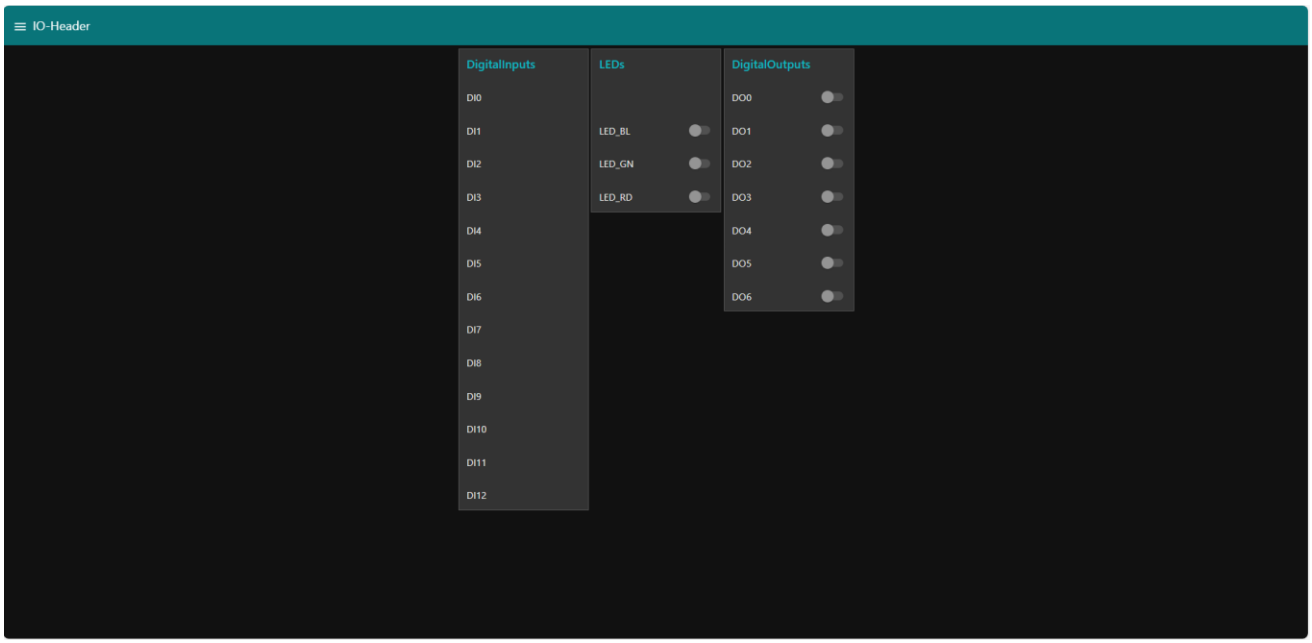
5.4 Runing Flows



To view the Dashboard open the menu Dashboard:







5.5 Securing Node-RED

In the initial steps, we used an unencrypted connection without a user account for the connection between the browser and the Node-RED server. However, this can result in any node within the network being able to access the flows. To prevent this, the use of user accounts is strongly recommended. However, this alone is not enough to prevent access to the Node-RED server if the connection is made via the unencrypted http protocol. In this case, the username and password are transmitted unencrypted. Therefore, it is equally important to encrypt the connection. This makes the application and the data significantly more secure.

By creating a user, you can control access to your Node-RED system and only allow authorized persons to make changes or view data. This protects against unauthorized access and potential misuse.

The use of TLS (Transport Layer Security) ensures that the data transmitted between your browser and the Node-RED server is encrypted. This means that third parties who might attempt to eavesdrop on the connection cannot read the transmitted information. This is especially important when sensitive data or control commands are being transmitted.

The process is described in detail in the Node-RED documentation (https://nodered.org/docs/user-guide/runtime/securing-node-red). The necessary steps are summarized here.

5.5.1 Connecting via TSL

Create a directory for storing the certificate and key files, change to this directory:

```
$  mkdir ~/.ssl
$  cd .ssl
```

Generate a self-signed certificate with a private key:

```
root@sysworxx:~/.ssh$ openssl req -new -days 3650 -newkey rsa:4096bits -sha512
-x509 -nodes -out nodeRED.crt -keyout nodeRED.key -subj "/CN=localhost"
```

```
root@sysworxx:~/.ssl$ ls
nodeRED.crt  nodeRED.key
```

Change Node-RED config file settings.js:

Open the settings.js with nano and browse to the https-section:

```
    /** Option 1: static object */
    //https: {
    //  key: require("fs").readFileSync('privkey.pem'),
    //  cert: require("fs").readFileSync('cert.pem')
    //},
```

Uncomment the section and add the files:

```
    /** Option 1: static object */
    https: {
      key: require("fs").readFileSync('/home/root/.ssl/nodeRED.key'),
      cert: require("fs").readFileSync('/home/root/.ssl/nodeRED.crt')
    },
```

Restart Node-RED service:

```
$ systemctl restart node-red
```

You can now access the Node-RED editor via the https protocol (ex: https://192.168.15.115:1880). The http protocol should no longer be used.

5.5.2 Set up user account and password

Generating the password hash for the user admin and save the hash:

```
$  node-red admin hash-pw
```

Uncomment the section adminAuth in the file settings.js:

```
   /** To password protect the Node-RED editor and admin API, the following
     * property can be used. See https://nodered.org/docs/security.html for
details.
     */
    //adminAuth: {
    //    type: "credentials",
    //    users: [{
    //        username: "admin",
    //        password: "<password-hash>",
    //        permissions: "*"
    //    }]
    //},
```

Add the password hash and a default user without login name. The admin user has full access, the pi-user and the default user has only read permission.

```
/** To password protect the Node-RED editor and admin API, the following
  * property can be used. See https://nodered.org/docs/security.html for
    details.
*/
    adminAuth: {
        type: "credentials",
        users: [{
            username: "root",
            password: "<password-hash>",
            permissions: "*"
        },
        {
            username: "pi-user",
            password: "<password-hash>",
            permissions: "read"
        }],
        default:
        {
            permissions: "read"
        }
    },
```

After restarting Node-RED, an input mask will appear in the browser when establishing the connection, in which you must enter your user name and password.

To create a password for the “pi-user”, please note the following. Node-RED runs as a process. It does matter under which user the service is started. The process inherits the authorisations and environment variables of the user who starts it. Note: For security reasons, it is recommended that Node-RED is not started as root-user, but as a user with restricted rights. To generate a password hash for a selected user, execute the command

```
pi-user@sysworxx:/$ sudo -u pi-user node-red admin hash-pw
```

as pi-user and enter the specific password. The hash generated in this way must then be entered in settings.js section for the user “pi-user”.

6 Appendix: sysWORXX Pi-AM62x Peripherals

Important: For all pins of the IO-Pin Header the applied voltage must not be higher than 3.3V. A general-purpose output set to high carries an output voltage of 3.3V. A voltage of 3.3V at a general-purpose input is read as high, a voltage of 0V as low. See in /3/ for details of voltage specifications.

Table 3: IO-Pin Header (X8) with AM62x signal names and default IO configuration

Default State[1]	Function	Pin name	Pin	Pin	Pin name	Function	Default State[1]
-	**3V3 power**	**3V3 power**	**1**	**2**	**5V power**	**5V power**	**-**
-	I2C3_SDA	GPIO1_23	**3**	**4**	**5V power**	**5V power**	**-**
-	I2C3_SCL	GPIO1_22	**5**	**6**	**Ground**	**Ground**	**-**
PD	DI_0	GPIO0_0	**7**	**8**	GPIO0_72	UART4_TX	High
-	**Ground**	**Ground**	**9**	**10**	GPIO0_71	UART4_RX	PU
low	DO_0	GPIO0_1	**11**	**12**	GPIO0_35	DO_1	-
PD	DI_1	GPIO0_2	**13**	**14**	**Ground**	**Ground**	**-**
low	DO_2	GPIO0_3	**15**	**16**	MCU_ GPIO0_13	DO_3	Low
-	**3V3 power**	**3V3 power**	**17**	**18**	MCU_ GPIO0_14	DI_2	PD
-	SPI1_MOSI	GPIO0_9	**19**	**20**	**Ground**	**Ground**	**-**
-	SPI1_MISO	GPIO0_10	**21**	**22**	GPIO0_40	DO_4	Low
-	SPI1_CLK	GPIO0_8	**23**	**24**	GPIO0_7	SPI1_CS0	-
-	**Ground**	**Ground**	**25**	**26**	GPIO0_13	DI_3	PD
PD	DI_4	GPIO0_44	**27**	**28**	GPIO0_43	DI_5	PD
PD	DI_6	GPIO1_24	**29**	**30**	**Ground**	**Ground**	**-**
PD	DI_7	GPIO1_25	**31**	**32**	GPIO1_9	DI_8	PD
PD	DI_9	GPIO1_28	**33**	**34**	**Ground**	**Ground**	**-**
PD	DI_10	GPIO0_37	**35**	**36**	GPIO0_41	DO_5	Low
low	DO_6	GPIO0_4	**37**	**38**	GPIO0_34	DI_11	PD
-	**Ground**	**Ground**	**39**	**40**	GPIO0_33	DI_12	PD

Pin name: The Pin Name column contains the name of the alternative function that is set by the CPU when the sysWORXX Pi-AM62x is started.

Function: The Function column contains the alternative function that is configured by the generic Device Tree Overlay.

[1] The default Pull State is configured by the Device Tree.
Low: the output is driven low
High: the output is driven high
PD: internal resistor pulls GPIO down (low)
PU: internal resistor pulls GPIO up (high)

Index