
byteWIKI

Release 1.0

May 06, 2020

1	About the company	3
2	Unboxing byteDEVKIT STM32MP1	5
3	First start byteDEVKIT STM32MP1	9
4	Bring-up byteDEVKIT STM32MP1	13
5	Software Development	23
6	Hardware Development	35

Welcome to the

»byteWIKI«

About the company



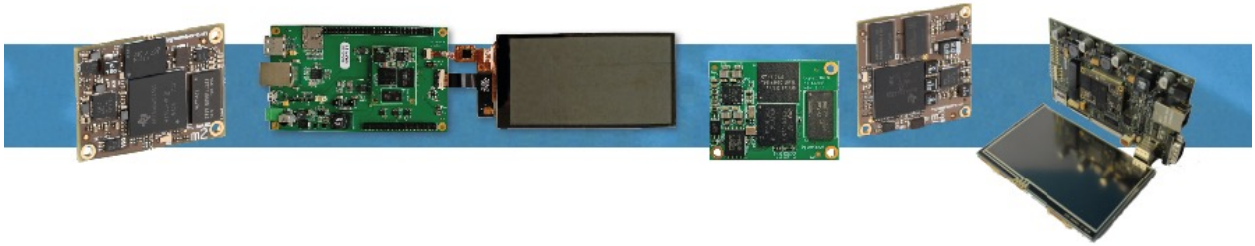
bytes at work is a modern Swiss Technology company specialized in industrial computing. Our focus lies on the development of hardware and embedded software, as well as customizing Linux systems. The entire development life cycle takes place in-house with transparent project management and customer involvement. This significantly reduces both development time and development costs.

We have years of experience in developing coordinated hardware and software solutions – from the prototype to the final product. We make your system usable end-to-end for your needs.

1.1 Our philosophy

Hardware and software for industrial computers have to fulfill an immense range of demanding challenges. They are used in completely different areas of industries and they have to be able to adapt unique and specific tasks. Our employees pay particular attention to each and every customer. That is why our products and services meet and even exceed our customers expectations.

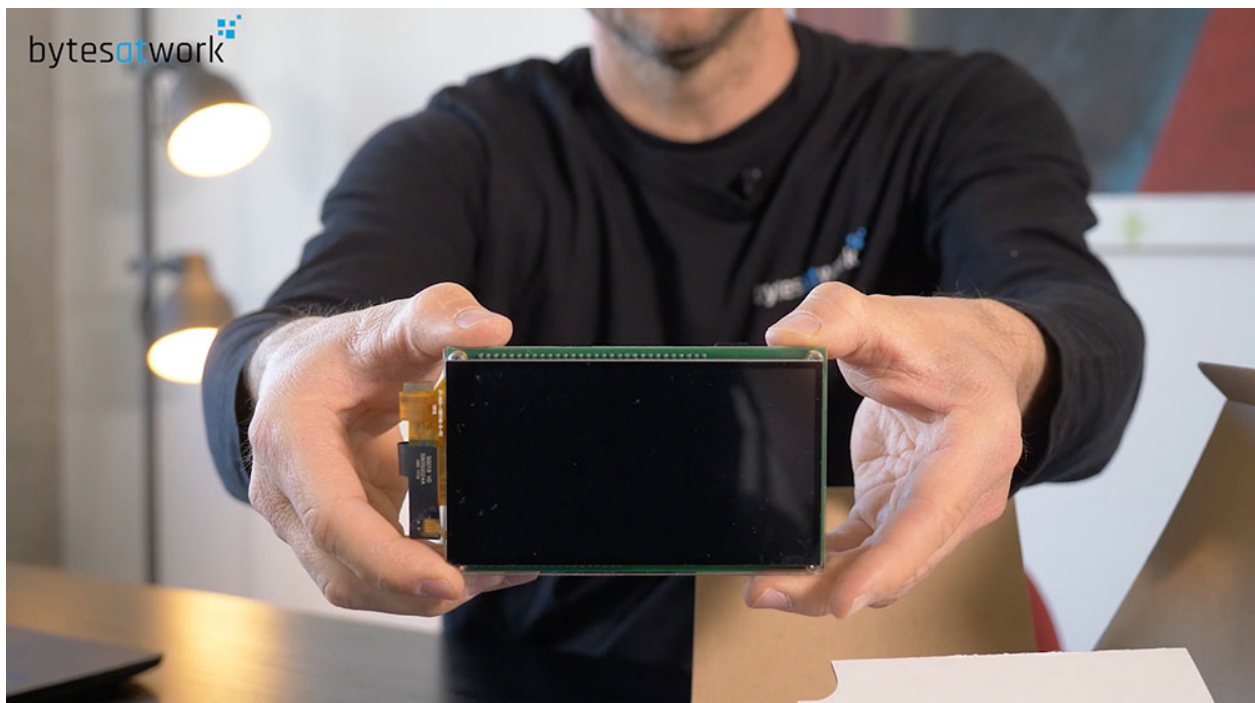
We from bytes at work are aware that the current persistent industrial development also has its darker side. This is our motivation to be exemplary in terms of use of resources. No wonder that unconditional reliability, long service life and low power consumption are main features of all our products.



Unboxing byteDEVKIT STM32MP1

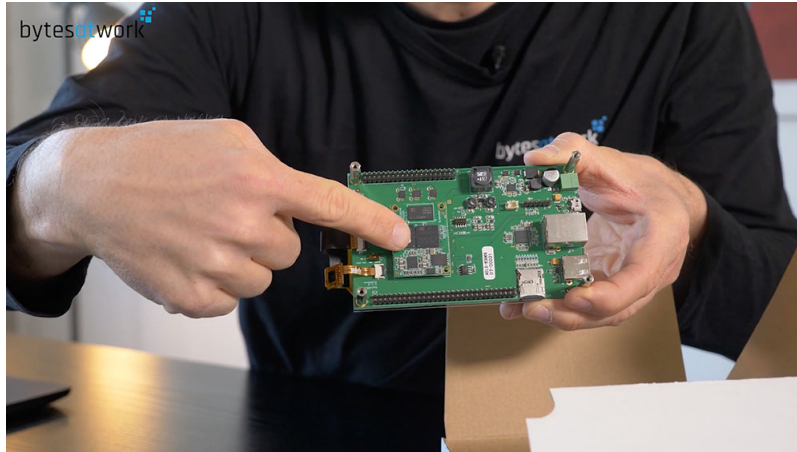
This guide delivers new users a brief overview of the package content and the functions of our byteDEVKIT STM32MP1. When unboxing you should find the following components:

- The byteDEVKIT STM32MP1 with a 5-inch touchscreen display

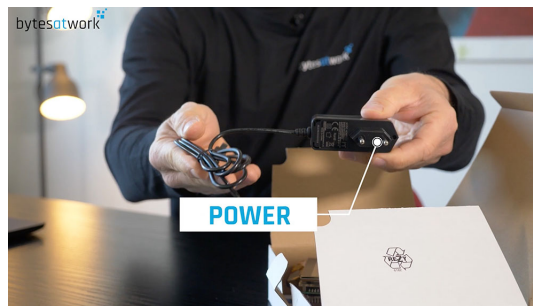


- The SOM STM32MP1x

Note: The SOM STM32MP1x is already connected with the byteDEVKIT STM32MP1.



- The power supply for the byteDEVKIT STM32MP1



- The USB serial cable for the byteDEVKIT STM32MP1

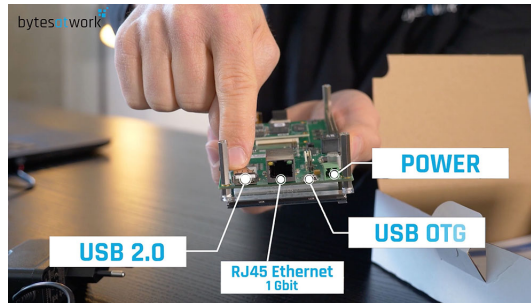


- micro-SD card with preinstalled Linux

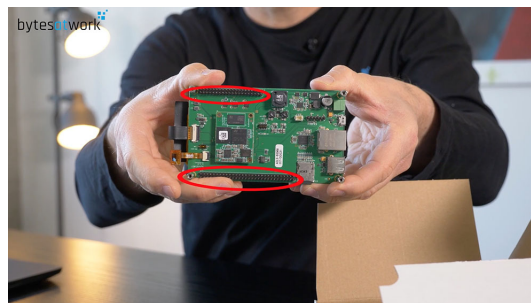


2.1 Technical overview byteDEVKIT STM32MP1

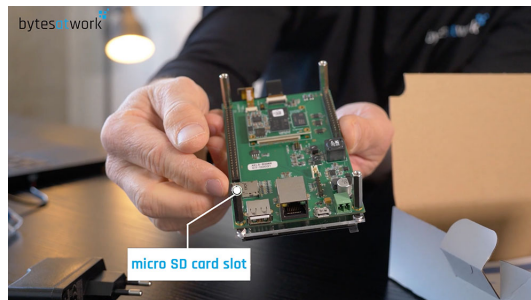
- The byteDEVKIT STM32MP1 offers the following connectors on the front side:
 - USB 2.0
 - RJ45 Ethernet 1 Gbit
 - USB OTG
 - Power connector



- You find the extension on the backside. The byteDEVKIT STM32MP1 offers:
 - 40 pin header compatible for the **rasperry pi**
 - 60 pin header with all the needed signals: **I2C, SPI, CAN, UART, I2S, LDC, GPIO and PWM**



- The micro-SD card slot contains a micro-SD card with preinstalled Linux OS:



Note: The micro-SD card is already slotted to the byteDEVKIT STM32MP1.

2.2 Unboxing Video Tutorial

First start byteDEVKIT STM32MP1

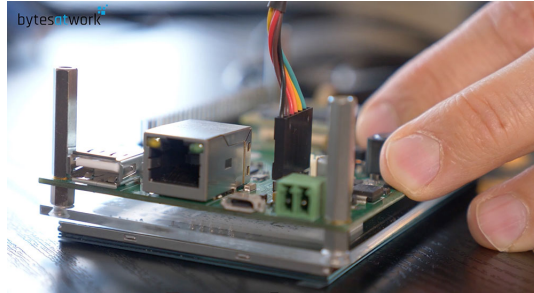
This guide helps with the first start of the byteDEVKIT STM32MP1:

3.1 Connecting the Hardware and first Booting

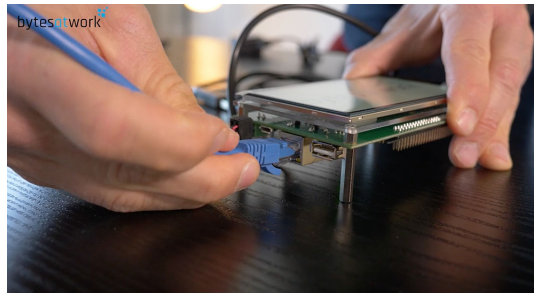
- Prepare the USB serial cable for connection
- Locate the black cable of the serial connector.



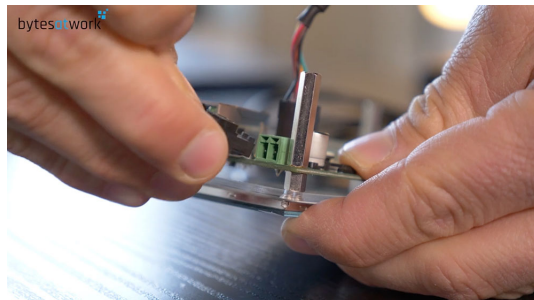
Caution: Connect the serial cable to the byteDEVKIT STM32MP1 as shown. The **black cable** must point towards the USB OTG connector.



- Connect the USB connector with USB port of your computer or laptop.
- Connect the ethernet RJ45 with the byteDEVKIT STM32MP1.

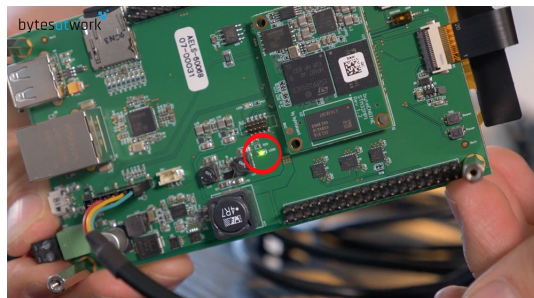


- Plug in the power socket.
- Connect the power supply cable to the power slot of the byteDEVKIT STM32MP1.



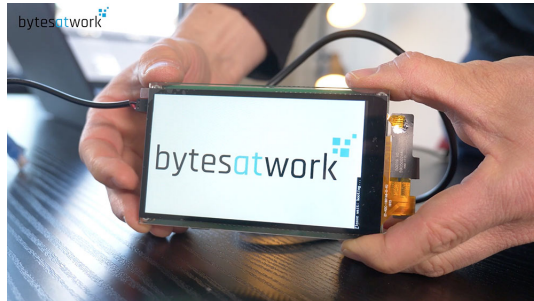
- A green LED on the backside of the byteDEVKIT STM32MP1 indicates the status of the power supply.

Attention: Your byteDEVKIT STM32MP1 is powered up, when the green LED lights up. If the LED doesn't light up, check the connection of the power socket.



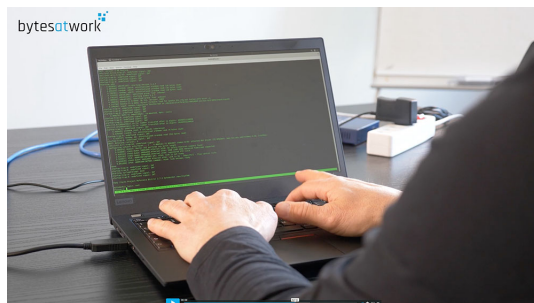
- The 5-inch touchscreen display shows the bytes at work-logo when booting.

Hint: The booting procedure will take a few seconds.



- Now you can access the byteDEVKIT STM32MP1 with your laptop.

Hint: For further information refer to: [“Bring-up_byteDEVKIT_STM32MP1”](#).



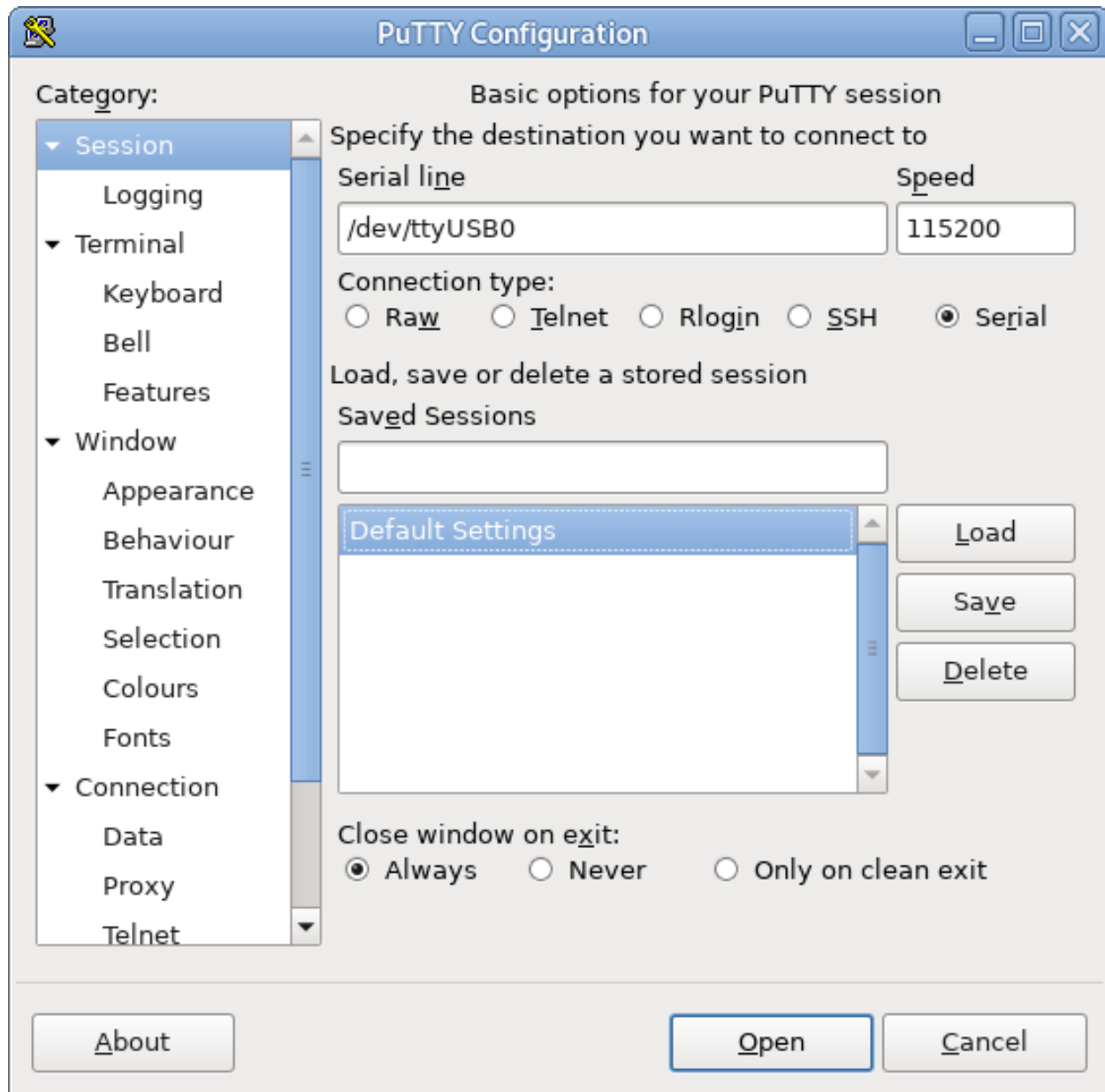
Bring-up byteDEVKIT STM32MP1

4.1 How do I connect to byteDEVKIT using the serial console?

- Use the serial port to connect the byteDEVKIT STM32MP1:
 - Connect the debug cable with the byteDEVKIT STM32MP1 and your computer/laptop
 - Start a serial communication program on your computer/laptop (<putty>, <minicom> or something else)
 - Set to 115200, 8N1, no flow control
 - login with: **user: “root”** and **password: “rootme”**

4.1.1 LINUX

- Start PuTTY

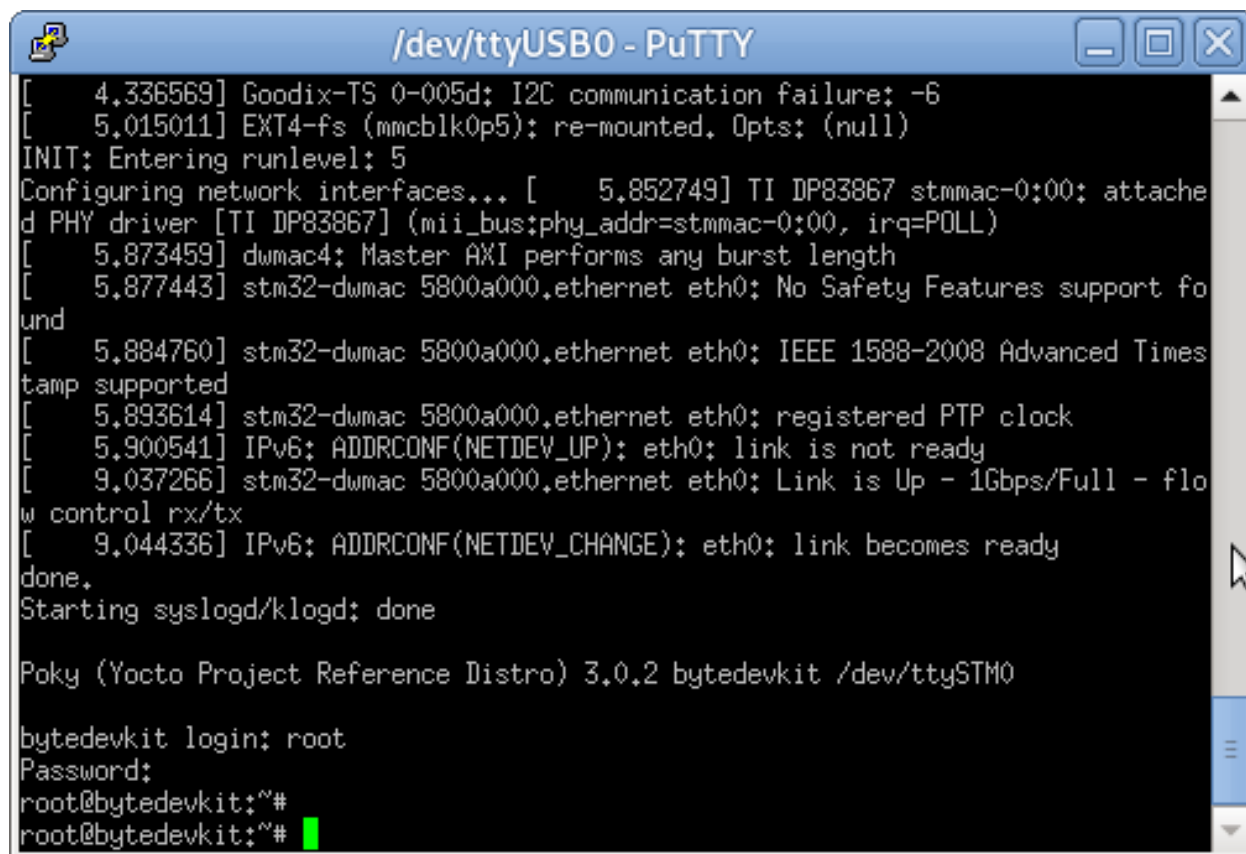


- Click “Serial”
- Change “Serial line” to “/dev/ttyUSB0”
- Change “Speed” to 115200
- Navigate to “Serial” in the menu “Connection”

Hint: make sure you have Data bits set to 8, Stop bits set to 1, Parity to None, Flow control to None

- Click “Open”

- Power up the byteDEVKIT STM32MP1



```
[ 4.336569] Goodix-TS 0-005d: I2C communication failure: -6
[ 5.015011] EXT4-fs (mmcblk0p5): re-mounted. Opts: (null)
INIT: Entering runlevel: 5
Configuring network interfaces... [ 5.852749] TI DP83867 stmmac-0:00: attached
PHY driver [TI DP83867] (mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[ 5.873459] dwmac4: Master AXI performs any burst length
[ 5.877443] stm32-dwmac 5800a000.ethernet eth0: No Safety Features support fo
und
[ 5.884760] stm32-dwmac 5800a000.ethernet eth0: IEEE 1588-2008 Advanced Times
tamp supported
[ 5.893614] stm32-dwmac 5800a000.ethernet eth0: registered PTP clock
[ 5.900541] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 9.037266] stm32-dwmac 5800a000.ethernet eth0: Link is Up - 1Gbps/Full - flo
w control rx/tx
[ 9.044336] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
done.
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 3.0.2 bytedevkit /dev/ttySTMO

bytedevkit login: root
Password:
root@bytedevkit:~#
root@bytedevkit:~#
```

- Once the login prompt appears, login with user “root” and password “rootme”

```

/dev/ttyUSB0 - PuTTY
[ 4.336569] Goodix-TS 0-005d: I2C communication failure: -6
[ 5.015011] EXT4-fs (mmcblk0p5): re-mounted. Opts: (null)
INIT: Entering runlevel: 5
Configuring network interfaces... [ 5.852749] TI DP83867 stmmac-0:00: attache
d PHY driver [TI DP83867] (mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[ 5.873459] dwmac4: Master AXI performs any burst length
[ 5.877443] stm32-dwmac 5800a000.ethernet eth0: No Safety Features support fo
und
[ 5.884760] stm32-dwmac 5800a000.ethernet eth0: IEEE 1588-2008 Advanced Times
tamp supported
[ 5.893614] stm32-dwmac 5800a000.ethernet eth0: registered PTP clock
[ 5.900541] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 9.037266] stm32-dwmac 5800a000.ethernet eth0: Link is Up - 1Gbps/Full - flo
w control rx/tx
[ 9.044336] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
done.
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 3.0.2 bytedevkit /dev/ttySTM0

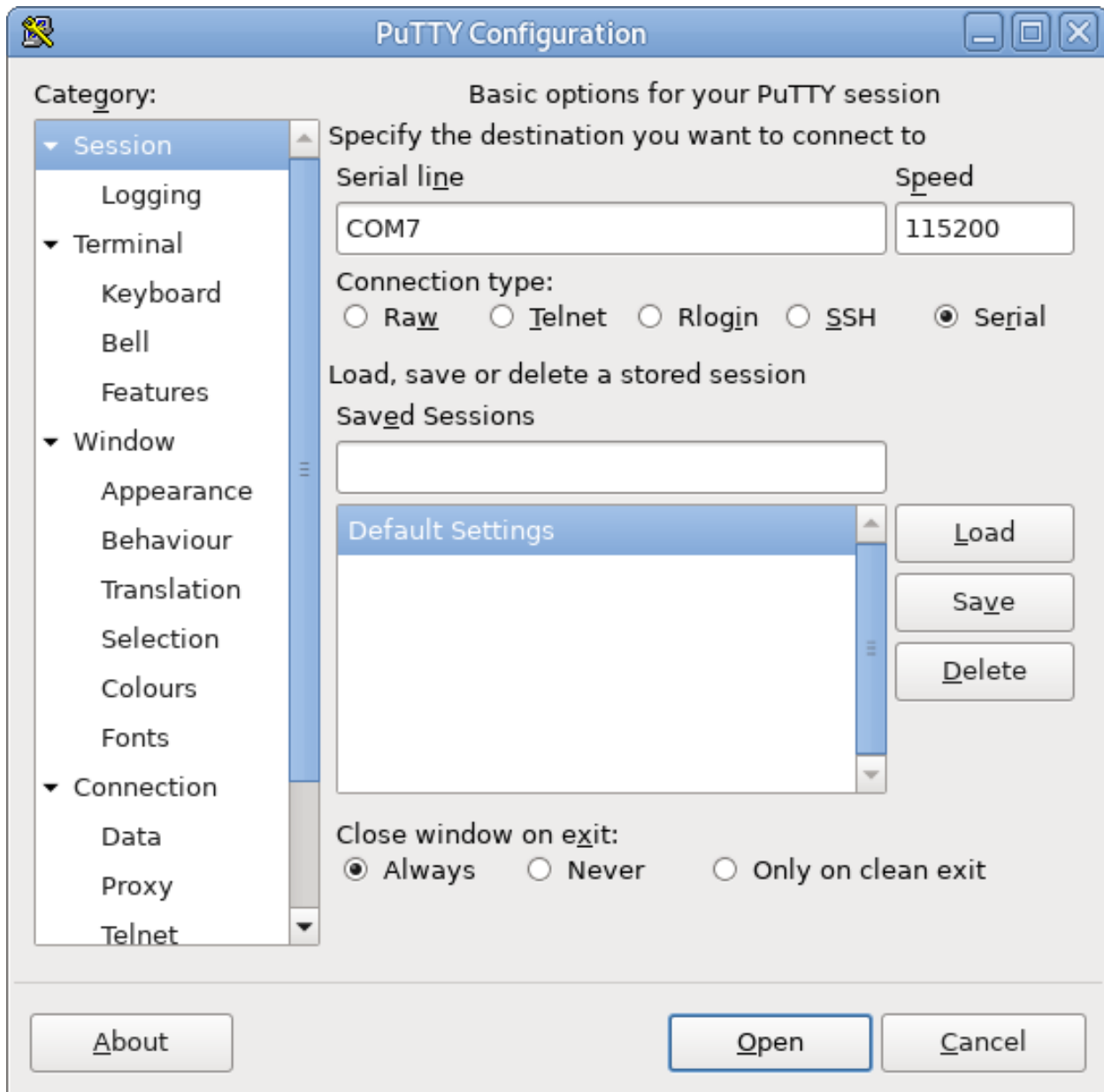
bytedevkit login: root
Password:
root@bytedevkit:~#
root@bytedevkit:~#

```

Note: You are now successfully connected to the byteDEVKIT STM32MP1

4.1.2 WINDOWS

- Connect the USB serial adapter to the computer
- Windows installs the driver automatically (if the windows doesn't install the driver reconnect the serial adapter cable)
- Open device manager and navigate to "Ports (COM & LPT)"
- The serial adapter shows up in the device tree: "Prolific USB-to-Serial Comm Port (COM7)"
- "COM7" is your serial port
- Install a serial terminal application, e.g. PuTTY (version 0.59 and newer) <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- Start PuTTY

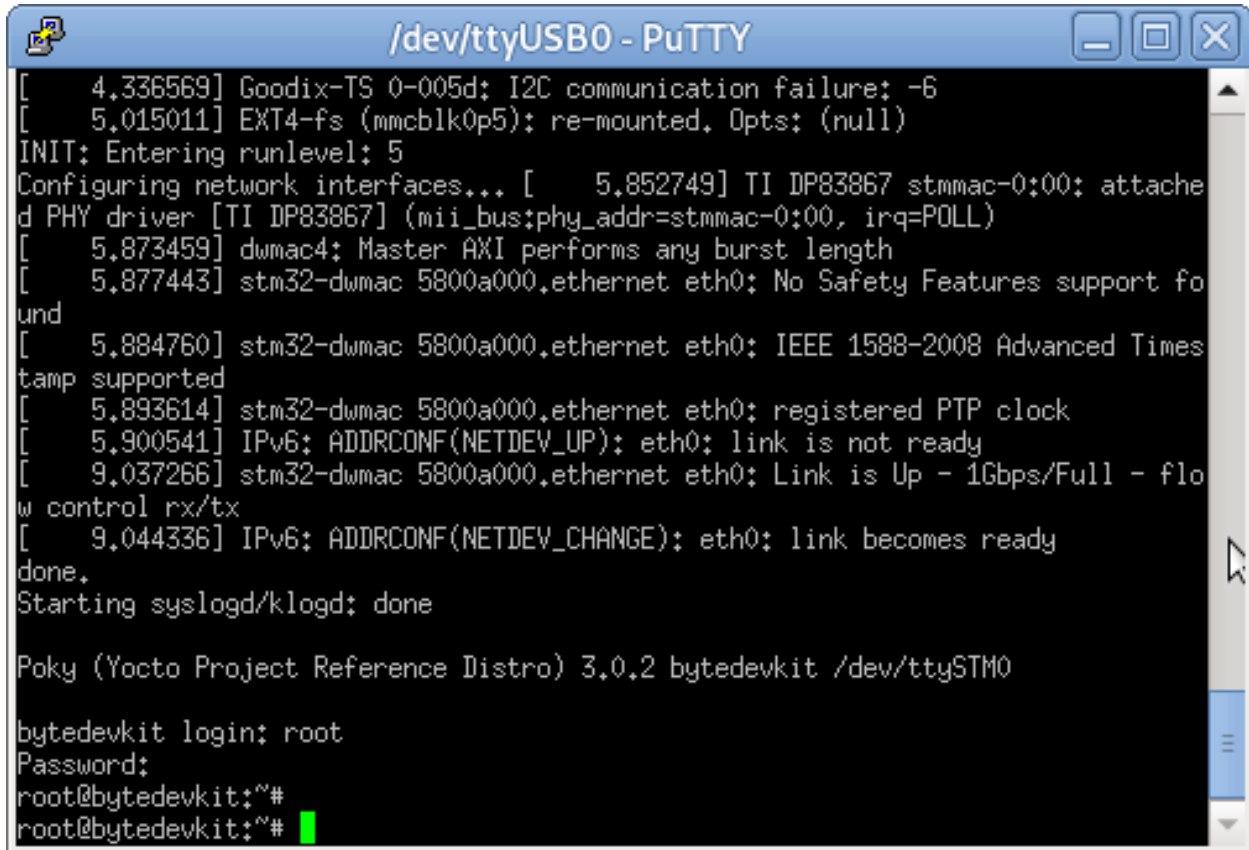


- Click “Serial”
- Change “Serial line” to serial port you found in device manager
- Change “Speed” to 115200
- Navigate to “Serial” in the menu “Connection”

Hint: make sure you have Data bits set to 8, Stop bits set to 1, Parity to None, Flow control to None

- Click “Open”
-

Power up the byteDEVKIT STM32MP1



```
/dev/ttyUSB0 - PuTTY
[ 4.336569] Goodix-TS 0-005d: I2C communication failure: -6
[ 5.015011] EXT4-fs (mmcblk0p5): re-mounted. Opts: (null)
INIT: Entering runlevel: 5
Configuring network interfaces... [ 5.852749] TI DP83867 stmmac-0:00: attache
d PHY driver [TI DP83867] (mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[ 5.873459] dwmac4: Master AXI performs any burst length
[ 5.877443] stm32-dwmac 5800a000.ethernet eth0: No Safety Features support fo
und
[ 5.884760] stm32-dwmac 5800a000.ethernet eth0: IEEE 1588-2008 Advanced Times
tamp supported
[ 5.893614] stm32-dwmac 5800a000.ethernet eth0: registered PTP clock
[ 5.900541] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 9.037266] stm32-dwmac 5800a000.ethernet eth0: Link is Up - 1Gbps/Full - flo
w control rx/tx
[ 9.044336] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
done.
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 3.0.2 bytedevkit /dev/ttySTMO

bytedevkit login: root
Password:
root@bytedevkit:~#
root@bytedevkit:~#
```

Once the login prompt appears, login with user “root” and password “rootme”

```

/dev/ttyUSB0 - PuTTY
[ 4.296669] Goodix-TS 0-005d: i2c test failed attempt 2: -6
[ 4.336504] Goodix-TS 0-005d: I2C communication failure: -6
[ 5.027199] EXT4-fs (mmcblk0p5): re-mounted. Opts: (null)
INIT: Entering runlevel: 5
Configuring network interfaces... [ 5.861469] TI DP83867 stmmac-0:00: attached
PHY driver [TI DP83867] (mii_bus:phy_addr=stmmac-0:00, irq=POLL)
[ 5.881887] dwmac4: Master AXI performs any burst length
[ 5.885779] stm32-dwmac 5800a000,ethernet eth0: No Safety Features support fo
und
[ 5.893308] stm32-dwmac 5800a000,ethernet eth0: IEEE 1588-2008 Advanced Times
tamp supported
[ 5.902098] stm32-dwmac 5800a000,ethernet eth0: registered PTP clock
[ 5.908959] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 9.037209] stm32-dwmac 5800a000,ethernet eth0: Link is Up - 1Gbps/Full - flo
w control rx/tx
[ 9.044282] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
done.
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 3.0.2 bytedevkit /dev/ttySTM0

bytedevkit login: root
Password:
root@bytedevkit:~#

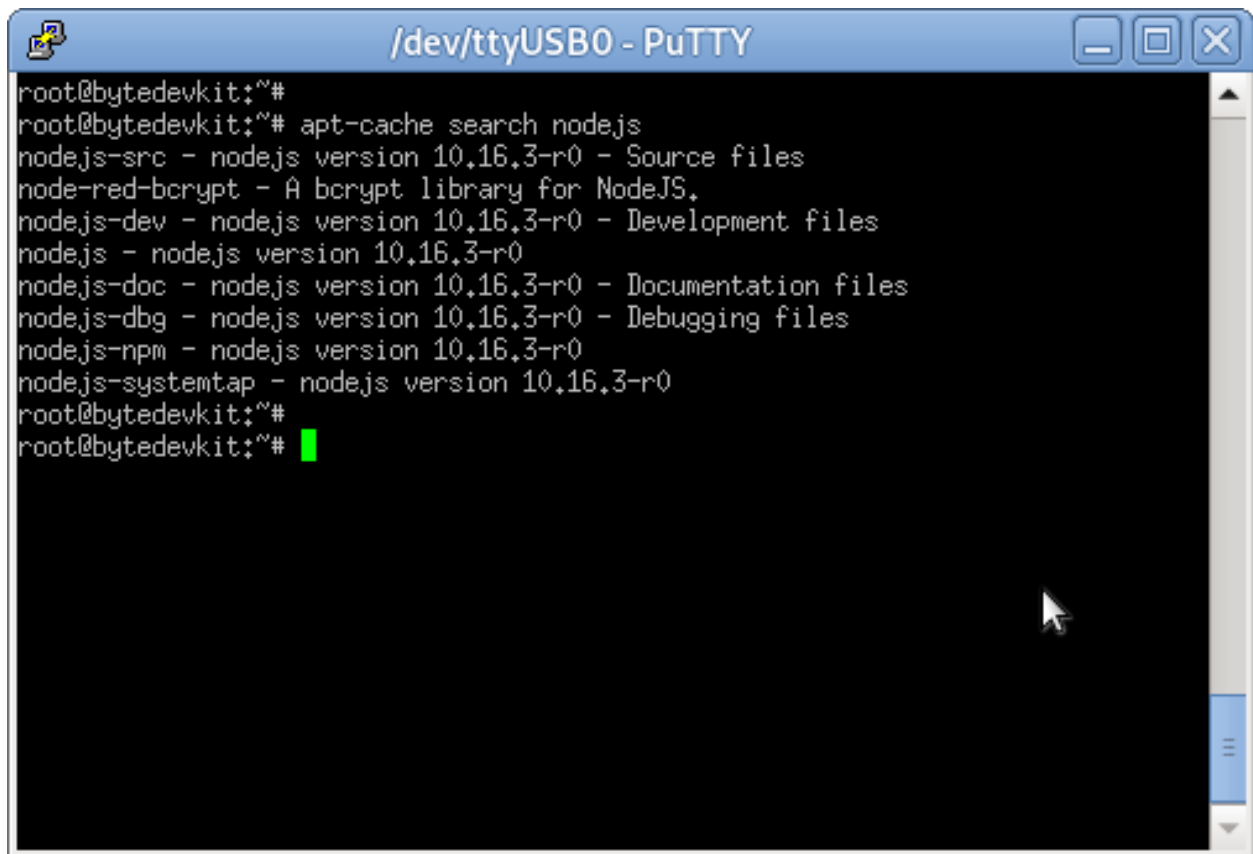
```

Note: You are now successfully connected to the byteDEVKIT STM32MP1

4.2 How to install additional software using apt

Hint: Follow the link for additional information about “apt”: <https://help.ubuntu.com/community/AptGet/Howto>

1. Connect the embedded device’s ethernet to your LAN
2. Run: *apt-get update*
3. Run: *apt-cache search <software component>* to search for available packages e.g.: *apt-cache search nodejs*



The image shows a PuTTY terminal window titled "/dev/ttyUSB0 - PuTTY". The terminal output is as follows:

```
root@bytedevkit:~#  
root@bytedevkit:~# apt-cache search nodejs  
nodejs-src - nodejs version 10.16.3-r0 - Source files  
node-red-bcrypt - A bcrypt library for NodeJS.  
nodejs-dev - nodejs version 10.16.3-r0 - Development files  
nodejs - nodejs version 10.16.3-r0  
nodejs-doc - nodejs version 10.16.3-r0 - Documentation files  
nodejs-dbg - nodejs version 10.16.3-r0 - Debugging files  
nodejs-npm - nodejs version 10.16.3-r0  
nodejs-systemtap - nodejs version 10.16.3-r0  
root@bytedevkit:~#  
root@bytedevkit:~#
```

The terminal window has a blue title bar with standard window controls (minimize, maximize, close) on the right. The background is black with white text. A green cursor is visible at the end of the last command line.

5. Run: *apt-get install <software component>* to install additional software e.g.: *apt-get install nodejs*

```

/dev/ttyUSB0 - PuTTY
root@bytedevkit:~#
root@bytedevkit:~# apt-get install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcudata64 libcui18n64 libcuc64
The following NEW packages will be installed:
  libcudata64 libcui18n64 libcuc64 nodejs
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 13.1 MB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
WARNING: The following packages cannot be authenticated!
  libcudata64 libcuc64 libcui18n64 nodejs
Install these packages without verification? [y/N] y
Get:1 http://packages.bytesatwork.io/gocto/3.0.2/bytedevkit/cortexa7t2hf-neon-vfpv4 ./ libcudata64 64.2-r0 [6704 kB]
Get:2 http://packages.bytesatwork.io/gocto/3.0.2/bytedevkit/cortexa7t2hf-neon-vfpv4 ./ libcuc64 64.2-r0 [524 kB]
Get:3 http://packages.bytesatwork.io/gocto/3.0.2/bytedevkit/cortexa7t2hf-neon-vfpv4 ./ libcui18n64 64.2-r0 [778 kB]
Get:4 http://packages.bytesatwork.io/gocto/3.0.2/bytedevkit/cortexa7t2hf-neon-vfpv4 ./ nodejs 10.16.3-r0 [5074 kB]
Fetched 13.1 MB in 4s (2901 kB/s)
Selecting previously unselected package libcudata64.
(Reading database ... 2959 files and directories currently installed.)
Preparing to unpack .../libcudata64_64.2-r0_armhf.deb ...
Unpacking libcudata64 (64.2-r0) ...
Selecting previously unselected package libcuc64.
Preparing to unpack .../libcuc64_64.2-r0_armhf.deb ...
Unpacking libcuc64 (64.2-r0) ...
Selecting previously unselected package libcui18n64.
Preparing to unpack .../libcui18n64_64.2-r0_armhf.deb ...
Unpacking libcui18n64 (64.2-r0) ...
Selecting previously unselected package nodejs.
Preparing to unpack .../nodejs_10.16.3-r0_armhf.deb ...
Unpacking nodejs (10.16.3-r0) ...
Setting up libcudata64 (64.2-r0) ...
Setting up libcuc64 (64.2-r0) ...
Setting up libcui18n64 (64.2-r0) ...
Setting up nodejs (10.16.3-r0) ...
root@bytedevkit:~#
root@bytedevkit:~#

```



The entire development lifecycle is done in-house with transparent project management and customer involvement. We have proven experience in a wide range of industries, including industrial automation and custom solutions for consumer electronics. This section helps you step by step initiating the software development process:

5.1 1. Where do you get the toolchain?

5.1.1 1.1 byteDEVKIT

- **Yocto** **3.0** Download LINK: https://download.bytesatwork.io/transfer/bytesatwork/m5/3.0/poky-bytesatwork-glibc-x86_64-bytesatwork-minimal-image-cortexa7t2hf-neon-vfpv4-bytedevkit-toolchain-3.0.2.sh
- **Yocto** **2.7** Download LINK: https://download.bytesatwork.io/transfer/bytesatwork/poky-bytesatwork-glibc-x86_64-devbase-image-bytesatwork-cortexa7t2hf-neon-vfpv4-bytedevkit-toolchain-2.7.1.sh

5.1.2 1.2 bytePANEL

- **Yocto** **3.0** Download LINK: https://download.bytesatwork.io/transfer/bytesatwork/m2/3.0/poky-bytesatwork-glibc-x86_64-bytesatwork-minimal-image-armv7at2hf-neon-bytepanel-emmc-toolchain-3.0.2.sh
 - **Yocto** **2.7** Download LINK: https://download.bytesatwork.io/transfer/bytesatwork/poky-bytesatwork-glibc-x86_64-devbase-image-bytesatwork-armv7at2hf-neon-bytepanel-toolchain-2.7.3.sh
-

5.2 2. How do you install the toolchain?

5.2.1 2.1 byteENGINE STM32MP1x

Download the Toolchain and install it

```
./poky-bytesatwork-glibc-x86_64-bytesatwork-minimal-image-cortexa7t2hf-neon-  
↳vfpv4-bytedevkit-toolchain-3.0.2.sh
```

5.2.2 2.2 byteENGINE AM335x

Download the Toolchain and install it

```
./poky-bytesatwork-glibc-x86_64-bytesatwork-minimal-image-armv7at2hf-neon-  
↳bytepanel-emmc-toolchain-3.0.2.sh
```

Hint: If you encounter problems when trying to install the toolchain, make sure the downloaded toolchain is executable. Run `chmod +x /<path>/<toolchain-file>.sh` to make it executable.

5.3 3. How do you use the toolchain?

5.3.1 3.1 byteENGINE STM32MP1x

Source the installed Toolchain:

```
source /opt/poky-bytesatwork/3.0.2/environment-setup-cortexa7t2hf-neon-vfpv4-poky-  
↳linux-gnueabi
```

Check if Cross-compiler is available in environment:

```
echo $CC
```

You should see the following output:

```
arm-poky-linux-gnueabi-gcc -mthumb -mcpu=cortex-a7 -mfpu=neon-vfpv4 -mfloat-abi=hard -mcpu=cortex-a7 -  
↳fstack-protector-strong -D_FORTIFY_SOURCE=2 -Wformat -Wformat-security -  
↳Werror=format-security --sysroot=/opt/poky-bytesatwork/3.0.2/sysroots/cortexa7t2hf-  
↳neon-vfpv4-poky-linux-gnueabi
```

Crosscompile the source code, e.g. by:

```
$CC helloworld.c -o helloworld
```

Check generated binary:

```
file helloworld
```

The output that is shown in prompt afterwards:

```
helloworld: ELF 32-bit LSB pie executable, ARM, EABI5 version 1
```

5.3.2 3.2 byteENGINE AM335x

Source the Toolchain

```
source /opt/poky-bytesatwork/3.0.2/environment-setup-armv7at2hf-neon-poky-linux-
↪gnueabi
```

Check if Cross-compiler is available in environment:

```
echo $CC
```

You should see the following output:

```
arm-poky-linux-gnueabi-gcc -march=armv7-a -mthumb -mfpu=neon -mfloat-abi=hard --
↪sysroot=/opt/poky-bytesatwork/3.0.2/sysroots/armv7at2hf-neon-poky-linux-gnueabi
```

Cross-compile the source code, e.g. by:

```
$CC helloworld.c -o helloworld
```

Check generated binary:

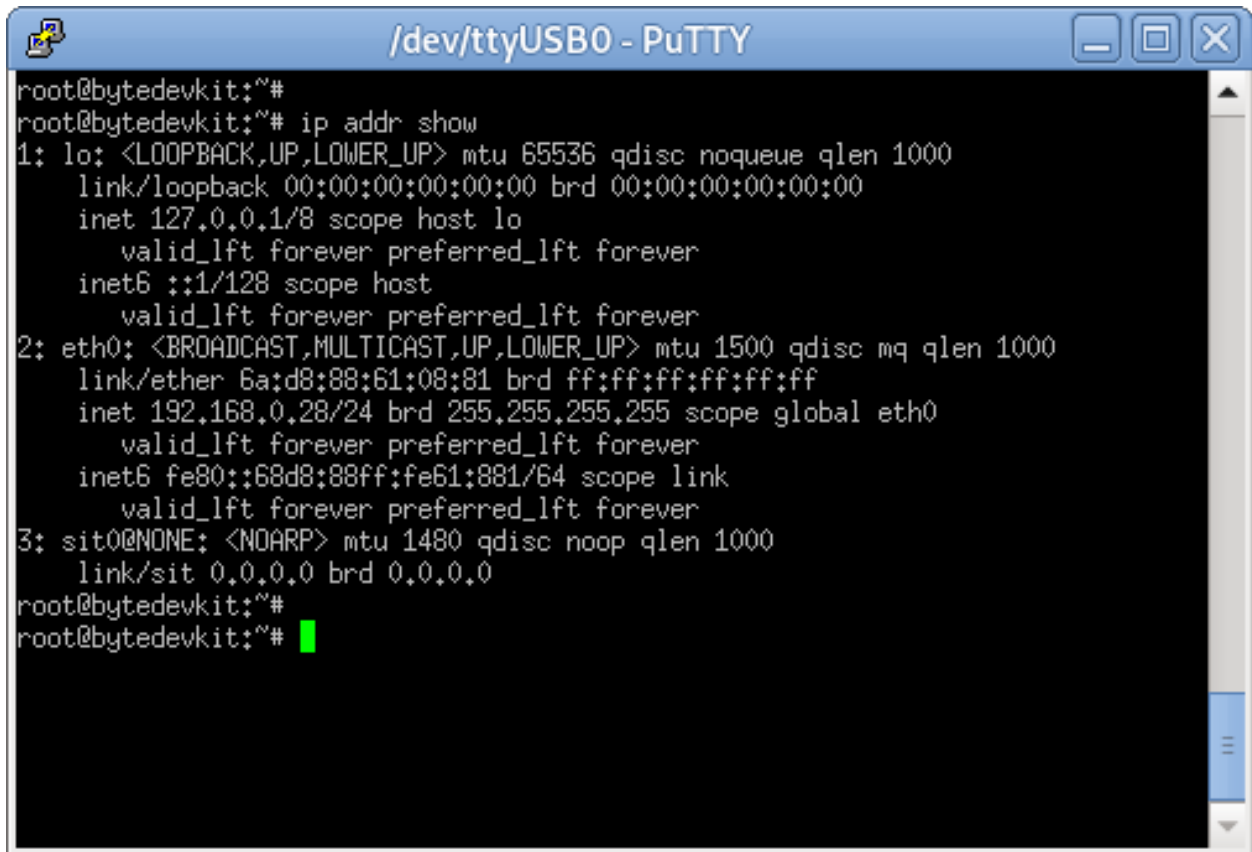
```
file helloworld
```

The output that is shown in prompt afterwards:

```
helloworld: ELF 32-bit LSB pie executable, ARM, EABI5 version 1
```

5.4 4. How to bring your binary to the target?

1. Connect the embedded device's ethernet to your LAN
2. determine the embedded target ip address by *ip addr show*

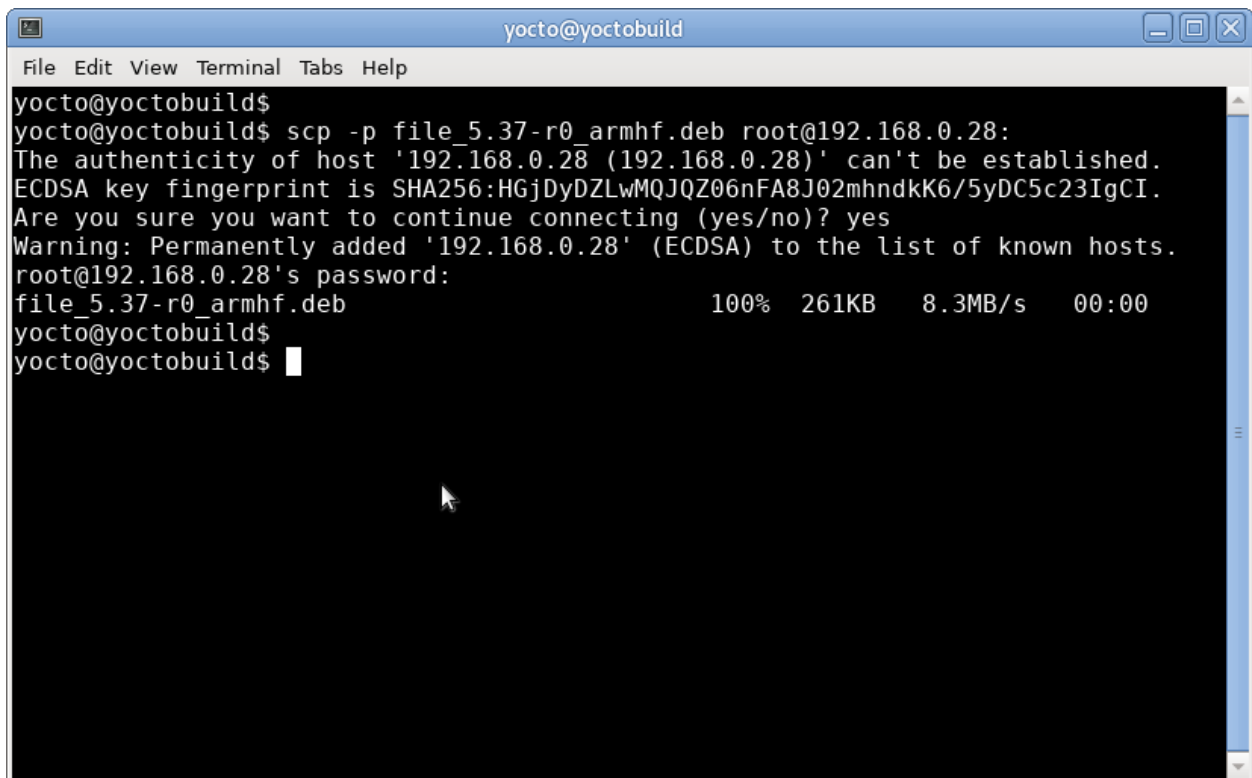


```

/dev/ttyUSB0 - PuTTY
root@bytedevkit:~#
root@bytedevkit:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq qlen 1000
    link/ether 6a:d8:88:61:08:81 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.28/24 brd 255.255.255.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::68d8:88ff:fe61:881/64 scope link
        valid_lft forever preferred_lft forever
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
root@bytedevkit:~#
root@bytedevkit:~# █

```

3. scp your binary, e.g. hello world to the target by `scp helloworld root@<ip address of target>:/tmp`



```

yocto@yoctobuild
yocto@yoctobuild$ scp -p file_5.37-r0_armhf.deb root@192.168.0.28:
The authenticity of host '192.168.0.28 (192.168.0.28)' can't be established.
ECDSA key fingerprint is SHA256:HGjDyDLwMQJQZ06nFA8J02mhndkK6/5yDC5c23IgCI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.28' (ECDSA) to the list of known hosts.
root@192.168.0.28's password:
file_5.37-r0_armhf.deb                                100% 261KB   8.3MB/s   00:00
yocto@yoctobuild$
yocto@yoctobuild$ █

```

4. run `chmod +x` on the target to make your binary executable: `chmod +x /<path>/<binary name>`
5. run your binary on the target: `/<path>/<binary name>`

5.5 5. Where do you get the Image for your SD-Card?

5.5.1 5.1 byteDEVKIT

- **Yocto 3.0** Download LINK: <https://download.bytesatwork.io/transfer/bytesatwork/m5/3.0/bytesatwork-minimal-image-bytedevkit.wic.gz>
- **Yocto 2.7** Download LINK: https://download.bytesatwork.io/transfer/bytesatwork/m5/2.7/flashlayout_bytesatwork-minimal-image_FlashLayout_sdcard_stm32mp157c-bytedevkit.raw.gz

5.5.2 5.2 bytePANEL

- **Yocto 3.0** Download LINK: <https://download.bytesatwork.io/transfer/bytesatwork/m2/3.0/bytesatwork-minimal-image-bytepanel-emmc.wic.gz>
- **Yocto 2.7** Downlad LINK: <https://download.bytesatwork.io/transfer/bytesatwork/m2/2.7/devbase-image-bytesatwork-bytepanel-emmc-20190729194430.sdimg.gz>

5.6 6. How do you flash the Image?

Attention:

- You need is a micro-SD card with **minimal 8GB** capacity.
- **All existing data** on the micro-SD card will be lost.
- **Do not format** the micro-SD card before flashing.

5.6.1 6.1 byteDEVKIT

- **Yocto 3.0**

WINDOWS

```
Unzip the <file.wic.gz> (e.g. with 7-zip)
Write the resulting <file.wic> to the uSD-card with a tool like Roadkils Disk_
↳ Image: https://www.roadkil.net/program.php?ProgramID=12
```

LINUX

```
gunzip -c <file.wic.gz> | dd of=/dev/mmcblk<X> bs=8M conv=fdatasync_
↳ status=progress
```

Hint: To improve write performance, you could use bmap-tools under Linux:

bmaptool copy <file.wic.gz> /dev/mmcblk<X>

- **Yocto 2.7**

WINDOWS

```
Unzip the <file.raw.gz> (e.g. with 7-zip)
Write the resulting <file.raw> to the uSD-card with a tool like Roadkils Disk_
↪Image: https://www.roadkil.net/program.php?ProgramID=12
```

LINUX

```
gunzip -c <file.raw.gz> | dd of=/dev/mmcblk<X> bs=8M conv=fdatasync_
↪status=progress
```

5.6.2 6.2 bytePANEL

- **Yocto 3.0**

WINDOWS

```
Unzip the <file.wic.gz> (e.g. with 7-zip)
Write the resulting <file.wic> to the uSD-card with a tool like Roadkils Disk_
↪Image: https://www.roadkil.net/program.php?ProgramID=12
```

LINUX

```
gunzip -c <file.wic.gz> | dd of=/dev/mmcblk<X> bs=8M conv=fdatasync_
↪status=progress
```

Hint: To improve write performance, you could use bmap-tools under Linux:

bmaptool copy <file.wic.gz> /dev/mmcblk<X>

- **Yocto 2.7**

WINDOWS

```
Unzip the <file.sdimg.gz> (e.g. with 7-zip)
Write the resulting <file.sdimg> to the uSD-card with a tool like Roadkils Disk_
↪Image[https://www.roadkil.net/program.php?ProgramID=12]
```

LINUX

```
gunzip -c <file.sdimg.gz> | dd of=/dev/mmcblk<X> bs=8M conv=fdatasync_
↪status=progress
```

5.7 7. How do you build an image?

5.7.1 7.1 byteDEVKIT

- **Yocto 3.0**

Use repo to download all necessary repositories:

```
repo init -u https://github.com/bytesatwork/bsp-platform-st.git -b zeus
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for byteDEVKIT:

```
MACHINE=bytedevkit DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

The final command builds the development image:

```
bitbake bytesatwork-minimal-image
```

The output is found in:

```
tmp/deploy/images/bytedevkit
```

Hint: For additional information about yocto images and howto build them, please visit: <https://www.yoctoproject.org/docs/3.0/mega-manual/mega-manual.html#brief-building-your-image>

- **Yocto 2.7**

Use repo to download all necessary repositories:

```
repo init -u https://github.com/bytesatwork/bsp-platform-st.git -b warrior
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for byteDEVKIT:

```
MACHINE=bytedevkit DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

The final command builds the development image:

```
bitbake devbase-image-bytesatwork
```

The output is found in:

```
tmp/deploy/images/bytedevkit
```

5.7.2 7.2 bytePANEL

- **Yocto 3.0**

Use repo to download all necessary repositories:

```
repo init -u https://github.com/bytesatwork/bsp-platform-ti.git -b zeus
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for bytePANEL:

```
MACHINE=bytepanel DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

the final command builds the development image:

```
bitbake bytesatwork-minimal-image
```

The output is found in:

```
tmp/deploy/images/bytepanel
```

Hint: For additional information about yocto images and howto build them, please visit: <https://www.yoctoproject.org/docs/3.0/mega-manual/mega-manual.html#brief-building-your-image>

- **Yocto 2.7**

Use repo to download all necessary repositories:

```
repo init -u https://github.com/bytesatwork/bsp-platform.git -b warrior
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for bytePANEL:

```
MACHINE=bytepanel DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

the final command builds the development image:

```
bitbake devbase-image-bytesatwork
```

The output is found in:

```
tmp/deploy/images/bytepanel
```

5.7.3 7.3 How to modify the image

- **bytesatwork delivers tips for customizing an image**

The image recipes can be found in *sources/meta-bytesatwork/recipes-core/images*

This is relative to where you started your repo command to check out all the sources.

Edit the minimal-image recipe *bytesatwork-minimal-image.bb*

Add the desired software-package to *IMAGE_INSTALL* variable, for example add *net-tools* to *bytesatwork-minimal-image.bb*

Rebuild the image by:

```
bitbake bytesatwork-minimal-image
```

5.7.4 7.4 How to rename the image

- If you want to rename or copy an image, simple rename or copy the image recipe by:

```
cp bytesatwork-minimal-image.bb customer-example-image.bb
```

5.7.5 7.5 Troubleshooting

- Image size is to small

If you encounter that your image size is to small to install additional software, please have a look at the *IMAGE_ROOTFS_SIZE* variable under *meta-bytesatwork/recipes-core/images/bytesatwork-minimal-image.bb*. Increase the size if necessary.

5.8 8. How do you build a toolchain?

5.8.1 8.1 byteDEVKIT

- Yocto 3.0

```
repo init -u https://github.com/bytesatwork/bsp-platform-st.git -b zeus
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for byteDEVKIT:

```
MACHINE=bytedevkit DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

The final command builds an installable toolchain:

```
bitbake bytesatwork-minimal-image -c populate_sdk
```

- Yocto 2.7

```
repo init -u https://github.com/bytesatwork/bsp-platform-st.git -b warrior
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for byteDEVKIT:

```
MACHINE=bytedevkit DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

The final command builds an installable toolchain:

```
bitbake devbase-image-bytesatwork -c populate_sdk
```

5.8.2 8.2 bytePANEL

- Yocto 3.0

```
repo init -u https://github.com/bytesatwork/bsp-platform-ti.git -b zeus
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for bytePANEL:

```
MACHINE=bytepanel DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

The final command builds an installable toolchain:

```
bitbake bytesatwork-minimal-image -c populate_sdk
```

- **Yocto 2.7**

```
repo init -u https://github.com/bytesatwork/bsp-platform.git -b warrior
repo sync
```

If those commands are completed successfully, the following command will setup a Yocto Project environment for bytePANEL:

```
MACHINE=bytepanel DISTRO=poky-bytesatwork EULA=1 . setup-environment build
```

The final command builds an installable toolchain:

```
bitbake devbase-image-bytesatwork -c populate_sdk
```

5.8.3 8.3 How to modify your toolchain

Currently the bytesatwork toolchain is generated out of the bytesatwork-minimal-image recipe. If you want to add additional libraries and development headers to customize the toolchain, you need to modify the bytesatwork-minimal-image recipe. It can be found under *sources/meta-bytesatwork/recipes-core/images*

For example if you want to develop your own ftp client and you need libftp and the corresponding header files, edit the recipe *bytesatwork-minimal-image.bb* and add *ftplib* to the *IMAGE_INSTALL* variable.

This will provide the ftplib libraries and development headers in the toolchain. After adding additional software components, the toolchain needs to be rebuilt by:

```
bitbake bytesatwork-minimal-image -c populate_sdk
```

The newly generated toolchain will be available under:

```
tmp/deploy/sdk
```

For additional information, please visit: <https://www.yoctoproject.org/docs/3.0.2/overview-manual/overview-manual.html#cross-development-toolchain-generation>

5.8.4 8.4 Troubleshooting

- **Errors when building the toolchain**

If you get the error below, please revert commit: *179c5cb7fd0f06970135187f1203507aa55d6bde* in the poky repository (sources/poky). See also Bug 13338 https://bugzilla.yoctoproject.org/show_bug.cgi?id=13338.

```
ERROR: bytesatwork-minimal-image-1.0-r0 do_populate_sdk: Unable to install packages.
↳ Command '/home/daniel/workspace/bytesatwork/yocto/ti-m2-zeus/build/tmp/work/
↳ bytepanel_emmc-poky-linux-gnueabi/bytesatwork-minimal-image/1.0-r0/recipe-sysroot-
↳ native/usr/bin/apt-get install --force-yes --allow-unauthenticated openssh-ssh
↳ openssh-sshd apt dpkg coreutils base-passwd dhcp-client target-sdk-provides-dummy
↳ shadow openssh-scp packagegroup-core-standalone-sdk-target packagegroup-core-boot
↳ vim openssh-sftp-server run-postinsts' returned 100:
Reading package lists...
Building dependency tree...
Reading state information...
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
  target-sdk-provides-dummy : Conflicts: coreutils
E: Unable to correct problems, you have held broken packages.
```



Hardware Development

We provide the development for a wide range of embedded systems, from small-scale embedded components to sophisticated embedded systems with increased security requirements. Our engineers are certified hardware experts and provide long experience in business.

6.1 byteENGINE AM335x

- **General Information:** The byteENGINE AM335x is a high performance industrial oriented computing module. It allows a short time-to-market, while reducing development costs and substantial design risks. The system on module (SOM) uses the Texas Instruments AM335x industrial applications processor family. The AM335x features a PowerVR™ SGX Graphics Accelerator Subsystem for 3D graphics acceleration. The Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) allows independent operation from the ARM processor. PRU-ICSS enables real-time protocols such as EtherCAT, PROFINET, EtherNet/IP, PROFIBUS, Ethernet Powerlink and Sercos.

The byteENGINE AM335x is a high performance industrial oriented computing module. It allows a short time-to-market, while reducing development costs and substantial design risks.

The system on module (SOM) uses the Texas Instruments AM335x industrial applications processor family. The AM335x features a PowerVR™ SGX Graphics Accelerator Subsystem for 3D graphics acceleration. The Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) allows independent operation from the ARM processor. PRU-ICSS enables real-time protocols such as EtherCAT, PROFINET, EtherNet/IP, PROFIBUS, Ethernet Powerlink and Sercos.

- **Datasheet AM335x:** https://www.bytesatwork.io/wp-content/uploads/2019/03/Datasheet_byteENGINE_AM335x-12.pdf
- **Prepared Pinmux file AM335x:** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/byteEngineM2-20160922.pinmux>
- **Detailed pinout AM335x:** https://download.bytesatwork.io/documentation/byteENGINE/ressources/PinmuxConfigSummary_byteEngineM2-20160922.xlsx
- **Datasheet Connectors Neltron 2001S-100G-270-020:** https://download.bytesatwork.io/documentation/byteENGINE/ressources/Neltron_2000P.pdf

- **Schematic of the connectors X1 and X2:** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/m2-connector.pdf>
 - **Texas Instruments Sitara™ AM335x Processors:** <http://www.ti.com/processors/sitara-arm/am335x-cortex-a8/overview.html>
 - **AM335x Technical Reference Manual:** <https://www.ti.com/lit/ug/spruh73q/spruh73q.pdf>
 - **TPS65910x Integrated Power-Management Unit:** <http://www.ti.com/lit/ds/symlink/tps65910.pdf>
-

6.2 byteENGINE STM32MP1x

- **General Information:** The byteENGINE STM32MP1x is a high performance industrial oriented computing module. It allows you a short time-to-market, reducing development costs and substantial design risks.

The system on module (SOM) uses the STM32MP15xxAC devices which are based on the high-performance dual-core ARM® Cortex®-A7 32-bit RISC core operating at up to 650MHz/800MHz. The STM32MP15xxAC devices also embed a Cortex®-M4 32-bit RISC core operating at up to 200 MHz frequency. The Cortex®-M4 core features a floating point unit (FPU) single precision which supports ARM® single-precision dataprocessing instructions and data types.

Furthermore, the STM32MP15xxAC devices embed a 3D graphic processing unit (Vivante® - OpenGL® ES 2.0) running at up to 533 MHz, with performances up to 26 Mtriangle/s, 133 Mpixel/s.

- **Factsheet STM32MP1x:** https://www.bytesatwork.io/wp-content/uploads/2019/04/Fact-Sheet-byteENGINE_STM32MP1x.pdf
- **Datasheet STM32MP1x:** https://www.bytesatwork.io/wp-content/uploads/2019/12/Datasheet_byteENGINE_STM32MP1x-6.pdf
- **Detailed pinout STM32MP1x:** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/byteENGINE-M5-pinout.xlsx>
- **Datasheet Connectors Neltron 2001S-100G-270-020:** https://download.bytesatwork.io/documentation/byteENGINE/ressources/Neltron_2000P.pdf
- **Schematic of the connectors X1 and X2:** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/m5-connector-pinout.pdf>
- **STMicroelectronics STM32MP1:** <https://www.st.com/en/microcontrollers-microprocessors/stm32mp1-series.html>
- **STPMIC1 power management IC:** <https://www.st.com/en/power-management/stpmic1.html>
- **Datasheet STM32MP157C:** <https://www.st.com/resource/en/datasheet/stm32mp157c.pdf>
- **STM32CubeMX Software Download:** <https://www.st.com/en/development-tools/stm32cubemx.html>
- **STM32MP1x prepared CubeMX Project:** https://download.bytesatwork.io/documentation/byteENGINE/ressources/byteENGINE_STM32MP1.ioc
- **Prepared project: step model STM32MP1x:** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/byteengine-m5.step>
- **Altium Library Neltron 2001S-100G-270-020:** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/2001s-100G-270-020.zip>
- **Altium Library byteENGINE STM32MP1x (X1/X2 position mask on layer 21):** <https://download.bytesatwork.io/documentation/byteENGINE/ressources/Footprint-byteENGINE-M5.zip>

