EPOSMoteIII Quick-Start

This is a step-by-step guide of everything you need to get started working with EPOS on EPOSMoteIII. This guide only points to relevant documentation pieces and is meant to help new users get started. After completing the steps below, users are still encouraged to take a look at the complete documentation pages:

- Complete documentation for EPOS 2.2
- Embedded Systems Labs with EPOS and EPOSMote
- EPOSMoteIII usage documentation
- EPOSMoteIII hardware documentation

For further assistance, please check EPOS Project at LISHA's GitLab.

1.1. Requirements

To work with EPOSMoteIII, you will need:

1. A computer running Linux
2. At least one EPOSMoteIII device (with EPOS Bootloader, which is the default)
3. At least one microUSB data cable

The EPOSMoteIII and microUSB cable are only necessary for the last steps of this guide. If you do not have these right now but are planning to use EPOSMoteIII in the near future, you can still go through most of this guide to configure your Linux environment beforehand. If you are attending an introductory course by LISHA, we usually provide the EPOSMoteIII devices and microUSB cables, but it helps if you bring your own.

1.2. Register at EPOS' website

The first step is to register at EPOS' website to get access to all of EPOS. To register, click here.

1.3. Download and install the GCC toolchain for ARM

Follow the instructions in the sections Downloading the toolchain and Installing from EPOS 2.2 User Guide. Choose the correct version of GCC for ARMv7 for your host (32 or 64 bits).

1.4. Download the latest EPOS2 code for ARM

Clone the latest release of EPOS which supports EPOSMote III from LISHA's GitLab.

1.5. Configure your host computer's USB

If your EPOSMote III is loaded with an EPOS Bootloader (which is the default), you can program your EPOSMote III device via USB. A Linux host will recognize EPOSMote III as an ''Abstract Control Model' '(ACM) device using a default driver.

Install `python3` and the `pyserial`

To program the EPOSMote III via USB, you will need `python3` with the `pyserial` module. You can install them as follows:

- For Debian-based distros (including Ubuntu):

```
$ sudo apt-get install python3-pip
$ sudo pip3 install pyserial
```

- For RedHat-based distros (including Fedora):
  ```
  $ sudo dnf install python3-pip
  $ sudo pip3 install pyserial
  ```

  ***Note**: conflicts between pyserial from pip and native serial packages for python have been reported and it might be necessary to remove the native package before installing pyserial via pip.*

Check for conflicts with `modemmanager`

When using EPOSMote III with USB, the `modemmanager` Linux service might get in the way. It is recommended to stop this service. You can do this with the command:''

- For Debian-based distros (including Ubuntu):
  ```
  $ sudo stop modemmanager
  ```

  You can also disable this service permanently with:
  ```
  $ echo "echo manual > /etc/init/modemmanager.override" | sudo sh
  ```

- For RedHat-based distros (including Fedora):
  ```
  $ sudo systemctl stop modem-manager.service
  ```

  You can also disable this service permanently with:
  ```
  $ sudo systemctl disable modem-manager.service
  ```

Grant permissions

Some distros require manual authorization to the serial ports. If this is the case of your distro (it is for Ubuntu), add your user to the required groups and log in again for these configurations take effect.

- For Ubuntu:
  ```
  $ sudo usermod -a -G dialout,uucp <username>
  ```

Configure the Serial Port

The default serial port configuration for EPOSMote III is "115200 8N1":

- Baud rate: 115200
- Data bits: 8
- Parity bits: 0
- Stop bits: 1

You can configure the serial port from inside a communication program like `minicom`, or you can use `stty` to configure these parameters from the command-line (enabling the use of tools such as `cat`):

```
$ stty -F /dev/ttyACM0 ispeed 115200 ospeed 115200 cs8 -cstopb -parenb -crtscts
```

***Note**: if you want to be able to write to the serial port, thus reaching EPOSMote III, hardware flow control must be disabled. Check the documentation of your communication program to accomplish this.*

1.6. Compile the Hello World application for EPOSMoteIII

You can look at the application's source code at `app/hello/hello.cc`. This is a simple application that

prints "Hello World" on the console, which, by default is redirected to the serial port. Before compiling it, instruct EPOS to produce code for EPOSMoteIII by editing the application-specific configuration file `app/hello/hello_traits.h`. It should look like this:

```
template<> struct Traits<Build>: public Traits_Tokens
{
    // Basic configuration
    static const unsigned int MODE = LIBRARY;
    static const unsigned int ARCHITECTURE = ARMv7;
    static const unsigned int MACHINE = Cortex;
    static const unsigned int MODEL = eMote3;
    static const unsigned int CPUS = 1;
    static const unsigned int NODES = 1; // no networking
    static const unsigned int EXPECTED_SIMULATION_TIME = 0; // not simulated
...
};
```

Now you can use the following command to compile your application and produce a tailored instance of EPOS to support it at run-time:

```
$ make APPLICATION=hello all
```

1.7. Upload the Hello World application to EPOSMoteIII

After compiling, you have to upload, or *flash*, your application to the mote. The program will run very quickly, so you might need to reset the EPOSMote III by pressing the button to see the output. ***Note***: *Every time the mote is reset, the bootloader will run and wait for a handshake for one second. If none is received and there is an image already loaded in flash memory, it will start executing that image.*
You can upload with:

```
$ make APPLICATION=hello flash
```

After you start the EPOSMoteIII with the application, you can launch your serial port communication program of choice to see the output. For example, assuming EPOSMote III has been detected by the host as `/dev/ttyACM0`:

```
$ minicom -D /dev/ttyACM0
```

Note: *Type `<Ctrl+a> + z` to check minicom commands.*

1.8. Troubleshooting

1.8.1. Old Ubuntu Releases won't Compile EPOS 2.2

The GCC tool-chain distributed with older Ubuntu versions is not able to compile EPOS 2.2. Upgrade to version 18.04 or newer.