Getting Started with OpenSmartMonitor

Devtank Ltd.

Marcus Holder

13.12.2022

This document will describe the first steps required to get up and running with a fresh version of the OpenSmartMonitor Firmware repository.

Contents

1. Compile

After you have cloned osm_firmware.git, the first step is to initialise the git submodule and update it.

```
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ make
make -C ./libs/libopencm3 TARGETS=stm32/l4
make[1]: Entering directory '/media/sf_my_p/osm_firmware/libs/libopencm3'
make[1]: *** No targets specified and no makefile found. Stop.
make[1]: Leaving directory '/media/sf_my_p/osm_firmware/libs/libopencm3'
make: *** [ports/stm.mk:39: libs/libopencm3/lib/libopencm3_stm32l4.a] Error 2
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ git submodule --hel
p
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ git submodule init
Submodule 'libopencm3' (https://github.com/libopencm3/libopencm3.git) registere
d for path 'libs/libopencm3'
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ git submodule updat
e
Cloning into '/media/sf_my_p/osm_firmware/libs/libopencm3'...
Rubbish Bin ath 'libs/libopencm3': checked out '44e142d4f97863e669737707a1a22bf4
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ make
make -C ./libs/libopencm3 TARGETS=stm32/l4
make[1]: Entering directory '/media/sf_my_p/osm_firmware/libs/libopencm3'
  GENHDR  stm32/l4
  BUILD   lib/stm32/l4
```

Figure 1: Git Submodule missing

You will need to compile the software which can be done by executing the Makefile with the command 'make' at the top level of the repository.

If make fails to compile, this may be due to missing packages and modules. For example, on this fresh Ubuntu virtual machine I received the error 'NO PICOLIB FOUND'.

```
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ make

make: pkg-config: No such file or directory
make: pkg-config: No such file or directory
find: ‘./build’: No such file or directory
mkdir -p ./build
FOUND=0; for P in /usr/lib/picolibc/arm-none-eabi/picolibc.specs /usr/local/lib
/picolibc/arm-none-eabi/picolibc.spec /usr/lib/gcc/arm-none-eabi/`arm-none-eabi
-gcc  -dumpversion`/picolibc.specs /usr/local/lib/gcc/arm-none-eabi/`arm-none-e
abi-gcc  -dumpversion`/picolibc.specs; do \
        if [ -e $P ]; then FOUND=1; break; fi \
done; \
if [ "$FOUND" = "0" ]; then echo "NO PICOLIB FOUND"; exit 1; fi
/bin/sh: 1: arm-none-eabi-gcc: not found
/bin/sh: 1: arm-none-eabi-gcc: not found
NO PICOLIB FOUND
make: *** [ports/stm.mk:43: build/.stm_build_env] Error 1
```

Figure 2: Running make on a fresh machine.

The Makefile will you tell you about missing files and packages that you will need to install to compile successfully.

In this case, I needed the picolibc-arm-none-eabi package to continue as well as pkg-config.

```
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ make
mkdir -p "build/tool/src"
cc -c -I./ports/stm/include -I./tools/img_json_interpretter/include -I./core/in
clude -I./sensors/include -I./comms/include -I./libs/libopencm3/include/ -I/usr
/include/json-c -Wno-address-of-packed-member -MMD -MP -g -DSTM32L4 -D__CONFIGT
OOL__ -Dfw_name=tool -DFW_NAME=TOOL -pedantic -Wall -Wextra -Werror -Wno-unused
-parameter -Wno-address-of-packed-member -I./model/env01/ -I./model/penguin/ -I
./model/sens01/ tools/img_json_interpretter/src/main.c -o build/tool/src/main.o
make: cc: No such file or directory
make: *** [tools/img_json_interpretter/tool.mk:37: build/tool/src/main.o] Error
 127
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ which gcc
```

Figure 3: Makefile trying to compile with cc rather than gcc.

Another issue you may find is that the Makefile will fail if it attempts to compile using cc, the build-essential package was required by my virtual machine in order for it to use the correct tools.

There may be a few Python modules that will require installing, go through each of these until the Makefile compiles successfully.

```
ports/stm/src/sleep.o ./build/sens01/ports/stm/src/timers.o ./build/sens01/port
s/stm/src/uarts.o ./build/sens01/ports/stm/src/version.o ./build/sens01/ports/s
tm/src/w1.o ./build/sens01/sens01/sens01.o -L./libs/libopencm3/lib --static -no
startfiles -L./libs/libopencm3/lib/stm32/l4 -lopencm3_stm32l4 -Wl,--start-group
 -lc -lgcc -lnosys -Wl,--end-group -Wl,--gc-sections -mthumb -mcpu=cortex-m4 -p
edantic -mfloat-abi=hard -mfpu=fpv4-sp-d16 --specs=picolibc.specs -T./ports/stm
/stm32l4.ld -o build/sens01/firmware.elf
arm-none-eabi-objcopy -O binary build/sens01/firmware.elf build/sens01/firmware
.bin
./build/tool/json_x_img build/sens01/config.bin < model/sens01_default_mem.json
JSON version = 3
IMG  version = 3
Loaded modbus.
dd of=build/sens01/complete.bin if=./build/sens01/bootloader.bin bs=2k
1+1 records in
1+1 records out
2948 bytes (2.9 kB, 2.9 KiB) copied, 0.000628124 s, 4.7 MB/s
dd of=build/sens01/complete.bin if=./build/sens01/config.bin seek=2 conv=notrun
c bs=2k
2+0 records in
2+0 records out
4096 bytes (4.1 kB, 4.0 KiB) copied, 0.000620402 s, 6.6 MB/s
dd of=build/sens01/complete.bin if=./build/sens01/firmware.bin seek=4 conv=notr
unc bs=2k
38+1 records in
38+1 records out
78540 bytes (79 kB, 77 KiB) copied, 0.0117161 s, 6.7 MB/s
touch build/sens01/.complete
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$
```

Figure 4: Makefile finishes compiling.

2. Test

```
marcus@marcus-ThinkPad-X230:~/Work/SmartFactory/osm_linux$ make penguin
penguin           penguin_coverage  penguin_cppcheck  penguin_soak      penguin_test      penguin_valgrind
marcus@marcus-ThinkPad-X230:~/Work/SmartFactory/osm_linux$ make penguin_test
mkdir -p /tmp/osm/
cd ./python/; ./osm_test.py
2022-12-13T11:47:41.215Z osm_test.py:152 (21669) [INFO]: Starting Virtual OSM Test...
2022-12-13T11:47:41.215Z osm_test.py:272 (21669) [INFO]: Spawning virtual OSM.
2022-12-13T11:47:42.430Z osm_test.py:283 (21669) [INFO]: Connecting to the virtual OSM.
2022-12-13T11:47:47.883Z osm_test.py:100 (21669) [INFO]: Firmware = PASSED ("6a7daa0" = "6a7daa0" +/- 0)
2022-12-13T11:47:48.186Z osm_test.py:100 (21669) [INFO]: PM 10 = PASSED (30.0 = 30.0 +/- 0.0)
2022-12-13T11:47:48.490Z osm_test.py:100 (21669) [INFO]: PM 2.5 = PASSED (20.0 = 20.0 +/- 0.0)
2022-12-13T11:47:48.793Z osm_test.py:100 (21669) [INFO]: Current Clamp Phase 1 = PASSED (4982.0 = 5000.0 +/- 100.0)
2022-12-13T11:47:49.095Z osm_test.py:100 (21669) [INFO]: Current Clamp Phase 2 = PASSED (7509.0 = 7500.0 +/- 100.0)
2022-12-13T11:47:49.398Z osm_test.py:100 (21669) [INFO]: Current Clamp Phase 3 = PASSED (10018.0 = 10000.0 +/- 100.0)
2022-12-13T11:47:49.699Z osm_test.py:100 (21669) [INFO]: One Wire Probe = PASSED (25.062 = 25.0625 +/- 0.01)
2022-12-13T11:47:50.001Z osm_test.py:100 (21669) [INFO]: Temperature = PASSED (21.59 = 20.0 +/- 2.5)
2022-12-13T11:47:50.304Z osm_test.py:100 (21669) [INFO]: Humidity = PASSED (50.18 = 50.0 +/- 2.5)
2022-12-13T11:47:50.606Z osm_test.py:100 (21669) [INFO]: Battery = PASSED (100.0 = 100.0 +/- 0.0)
2022-12-13T11:47:50.908Z osm_test.py:100 (21669) [INFO]: Light = PASSED (997.0 = 1000.0 +/- 100.0)
2022-12-13T11:47:51.211Z osm_test.py:100 (21669) [INFO]: Power Factor = PASSED (1000.0 = 1000.0 +/- 0.0)
2022-12-13T11:47:52.111Z osm_test.py:100 (21669) [INFO]: Voltage Phase 1 = PASSED (24001.0 = 24001.0 +/- 0.0)
2022-12-13T11:47:53.116Z osm_test.py:100 (21669) [INFO]: Amps P1 = PASSED (30.1 = 30.1 +/- 0.0)
2022-12-13T11:47:54.113Z osm_test.py:100 (21669) [INFO]: Amps P2 = PASSED (30.2 = 30.2 +/- 0.0)
2022-12-13T11:47:54.415Z osm_test.py:114 (21669) [INFO]: IO off = PASSED (False = False)
2022-12-13T11:47:55.018Z osm_test.py:114 (21669) [INFO]: IO still off = PASSED (False = False)
2022-12-13T11:47:55.922Z osm_test.py:114 (21669) [INFO]: IO on = PASSED (True = True)
2022-12-13T11:47:56.828Z osm_test.py:202 (21669) [INFO]: Running measurement loop...
```

Figure 5: Running Test on Virtual OSM.

Now that osm_firmware has compiled, we can begin running tests and communicating with the fake osm. To run a test for the virtual OSM, enter make penguin_test in the top level directory.

3. Run

To communicate with the virtual OSM, you will need to run firmware.elf which is located in osm_firmware/build/penguin/firmware.elf. Once this is running, you can use minicom to open up communications with the fake sensor.

```
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ ./build/penguin/fir
mware.elf
-------------
Process ID: 13407
2022-12-13T11:54:56.380Z hpm_virtual.py:12 (13410) [INFO]: INITIALISED VIRTUAL
HPM
2022-12-13T11:54:56.439Z w1_server.py:79 (13411) [INFO]: W1 SERVER INITIALISED
2022-12-13T11:54:56.842Z i2c_server.py:50 (13412) [INFO]: I2C SERVER INITIALISE
D WITH 2 DEVICES
----start----
DEBUG:0000000740:SYS:Frequency : 0
DEBUG:0000000740:SYS:Version : [1783]-6a7daa0-Improve-STM-Linux-dependency-chec
k
```

Figure 6: Running Virtual OSM

The device should spawn in /tmp/osm/ and UART_DEBUG_slave is the special device that you want to connect to. Supply the baudrate of 115200 to minicom with the flag -b

```
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ ls /tmp/osm/
debug.log         osm.img           UART_EXT_slave    UART_LW_slave
i2c_socket        UART_DEBUG_slave  UART_HPM_slave    w1_socket
marcusosm@marcusosm-VirtualBox:/media/sf_my_p/osm_firmware$ minicom -b 115200 -
D /tmp/osm/UART_DEBUG_slave
```

Figure 7: Connecting to OSM via minicom

and supply the device with the flag -D.

```
DEBUG:0000508652:SYS:Command "q"
============{
Unknown command "q"
=============
     count : Counts of controls.
   version : Print version.
     debug : Set hex debug mask
     timer : Test usecs timer
       bat : Get battery level.
        cc : CC value
    cc_cal : Calibrate the cc
     cc_mp : Set the CC midpoint
   cc_gain : Set the max int and ext
  can_impl : Send example CAN message
 cal_sound : Set the cal coeffs.
      save : Save config
     reset : Reset device.
      wipe : Factory Reset
measurements : Print measurements
meas_enable : Enable measuremnts.
  get_meas : Get a measurement
  no_comms : Dont need comms for measurements
  interval : Set the interval
samplecount : Set the samplecount
interval_mins : Get/Set interval minutes
     repop : Repopulate measurements.
       ios : Print all IOs.
```

Figure 8: Sending Command to Virtual OSM

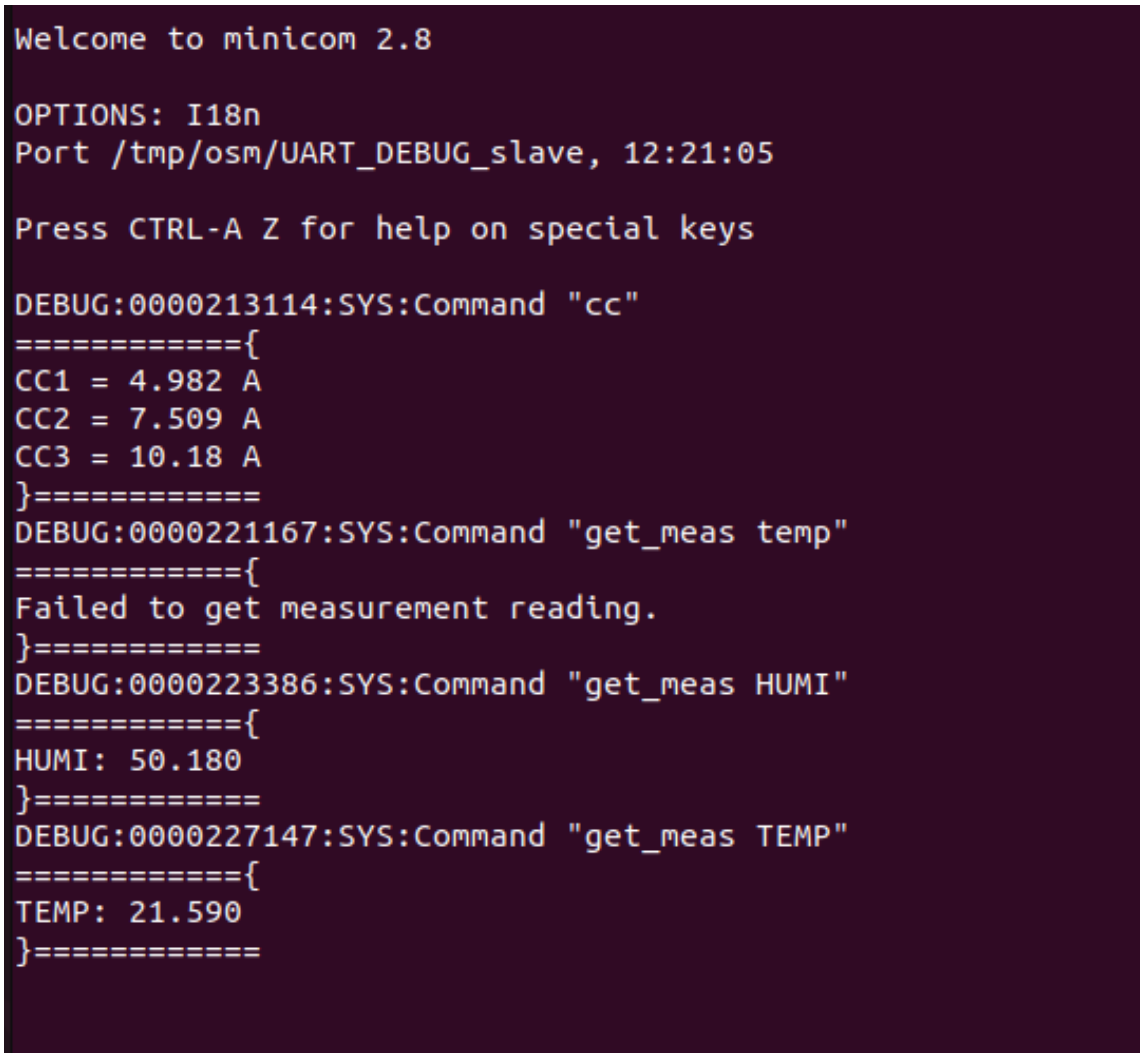

Figure 9: Sending Commands to Virtual OSM