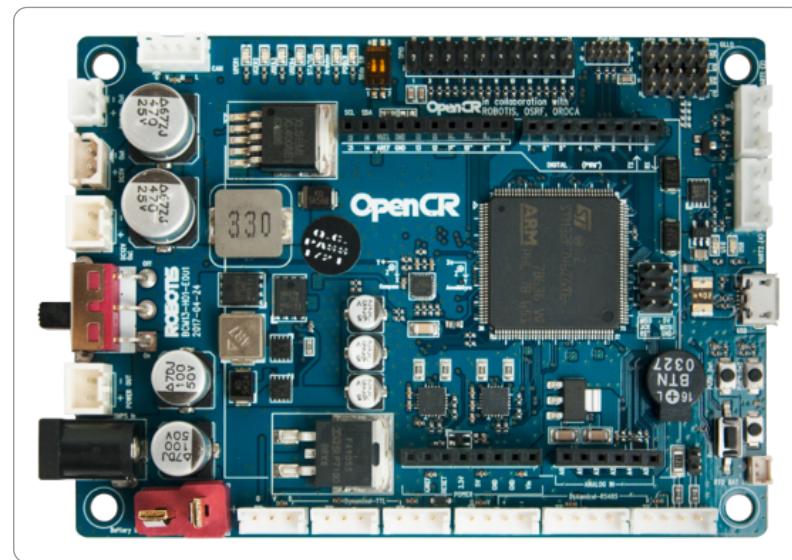


OpenCR1.0

- [1. Introduction](#)
- [2. Specifications](#)
- [3. Hardware \[+\]](#)
- [4. Bootloader \[+\]](#)
- [5. Downloader \[+\]](#)
- [6. Arduino IDE \[+\]](#)
- [7. Examples \[+\]](#)
- [8. Downloads](#)
- [9. References \[+\]](#)

[Back to Top ▲](#)

OpenCR 1.0

1. Introduction

OpenCR1.0 is developed for ROS embedded systems to provide completely open-source hardware and software.

Everything about the board; Schematics, PCB Gerber, BOM and the firmware source code for the TurtleBot3 and OP3 are free to distribute under open-source licenses for users and the ROS community.

The STM32F7 series chip inside the OpenCR1.0 board is based on a very powerful ARM Cortex-M7 with floating point unit.

The development environment for OpenCR1.0 is wide open from Arduino IDE and Scratch for young students to traditional firmware development for the expert.

2. Specifications

Items	Specifications
Microcontroller	STM32F746ZGT6 / 32-bit ARM Cortex®-M7 with FPU (216MHz, 462DMIPS)
Sensors	Gyroscope 3Axis, Accelerometer 3Axis, Magnetometer 3Axis (MPU9250)
Programmer	ARM Cortex 10pin JTAG/SWD connector USB Device Firmware Upgrade (DFU) Serial
Digital I/O	32 pins (L 14, R 18) *Arduino connectivity 5Pin OLLO x 4 GPIO x 18 pins PWM x 6 I2C x 1 SPI x 1
Analog INPUT	ADC Channels (8bit) x 6
Communication Ports	USB x 1 (Micro-B USB connector/USB 2.0/Host/Peripheral/OTG) TTL x 3 (B3B-EH-A / Dynamixel) RS485 x 3 (B4B-EH-A / Dynamixel) UART x 2 (20010WS-04) CAN x 1 (20010WS-04)

OpenCR1.0

Items	Specifications
LEDs and buttons	LD2 (red/green) : USB communication User LED x 4 : LD3 (red), LD4 (green), LD5 (blue) User button x 2
Powers	External input source 5 V (USB VBUS), 7-24 V (Battery or SMPS) Default battery : LI-PO 11.1V 1,800mAh 19.98Wh Default SMPS: 12V 5A External output source *12V max 5A(SMW250-02), *5V max 4A(5267-02A), 3.3V@800mA(20010WS-02) External battery Port for RTC (Real Time Clock) (Molex 53047-0210) Power LED: LD1 (red, 3.3 V power on) Reset button x 1 (for power reset of board) Power on/off switch x 1
Dimensions	105(W) X 75(D) mm
Mass	60g

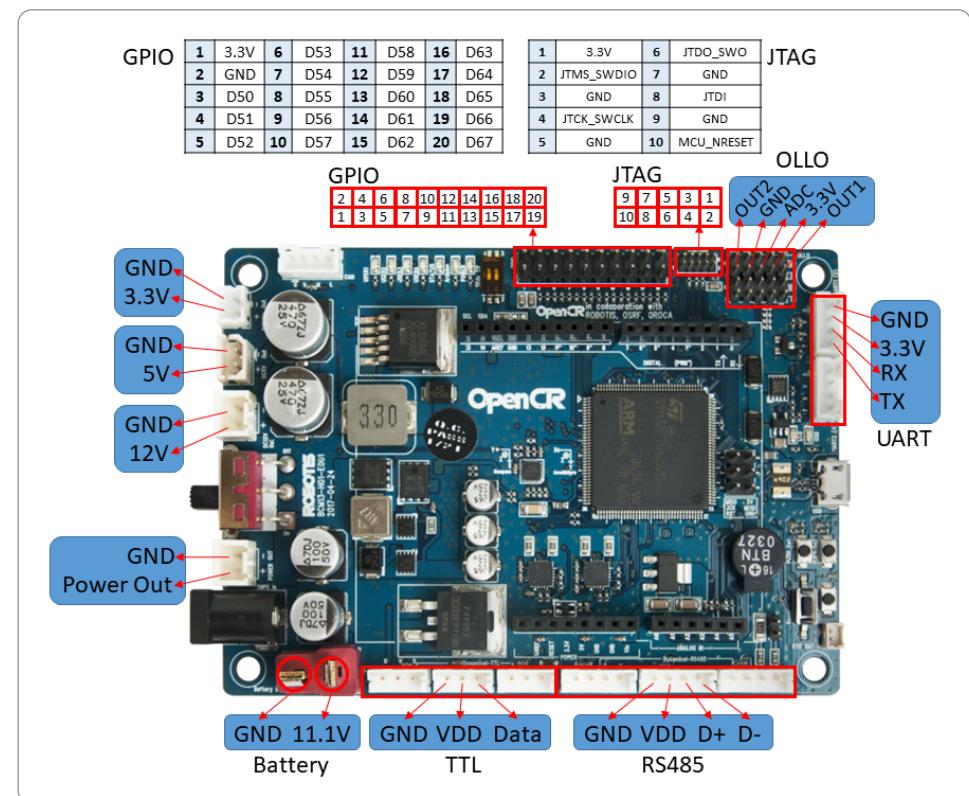
[Back to Top ▲](#)

* 5V power source is supplied from regulated 12V output.

Note Hot swap power switch between "shore power"(12V, 5A SMPS) and "mobile power"(battery) from OpenCR1.0 board enables UPS(Uninterrupted Power Supply) feature.

3. Hardware

3. 1. Layout/Pin Map

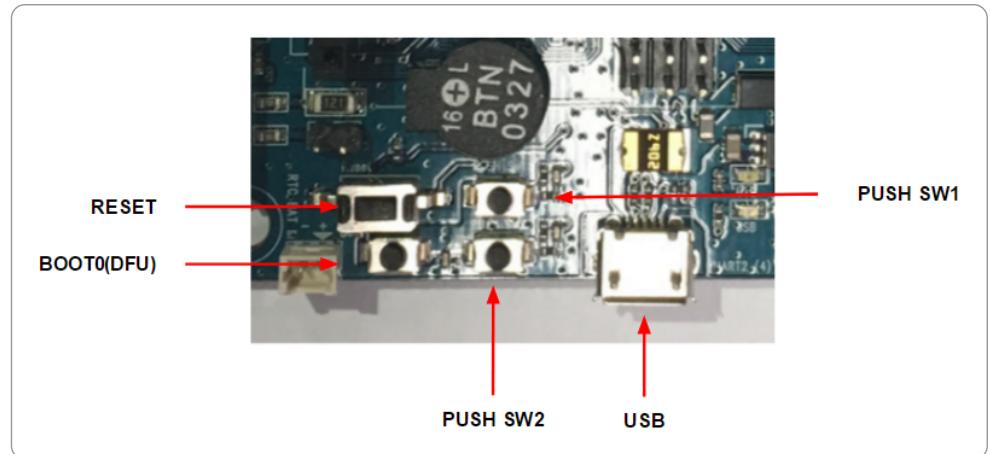


4. Bootloader

The bootloader is responsible for initializing the board and downloading and executing the firmware into flash memory.

Item	Description
Supported OS	Windows, Linux, Mac

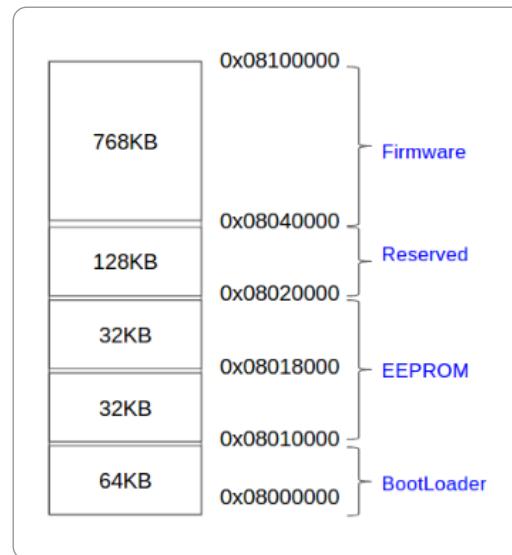
Item	Description
Compiler	gcc arm 5.4 2016q2

OpenCR1.0[Back to Top ▲](#)

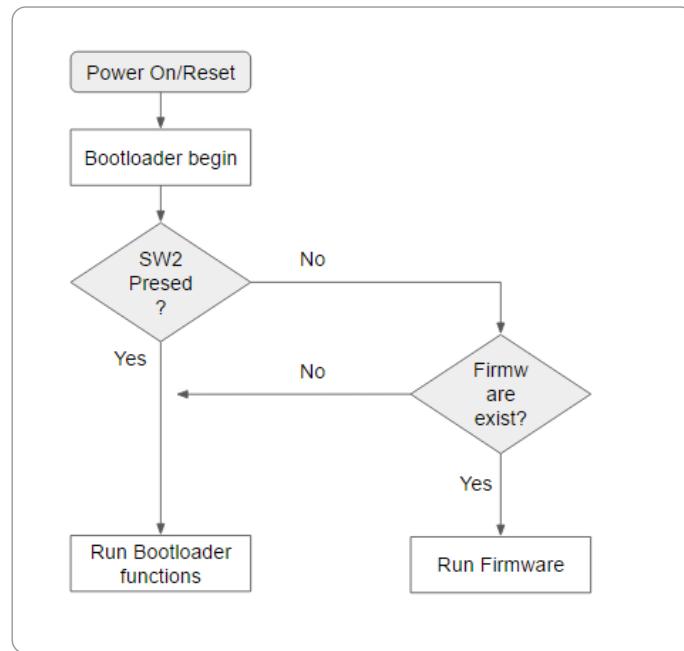
- **USB Port**
 - Connected to PC and recognized as serial port
 - A communication cable for downloading firmware through the bootloader.
- **PUSH SW2**
 - Press and hold the button when the power is on or reset to execute the bootloader
 - If the button is not pressed when the power is turned on, the bootloader is executed. If the firmware is in the flash memory, the bootloader executes the firmware.

4. 1. Memory Map

The STM32F746 used in OpenCR has an internal flash memory of 1024KB, and each area is defined as follows. The bootloader is located at the lowest address in the flash memory and the bootloader is first executed when the power is turned on and reset.

**4. 2. Boot Sequence**

OpenCR1.0

[Back to Top ▲](#)

If the board is powered on or reset, if the SW2 button is pressed, it waits for commands from the PC in the boot loader state. If the SW2 button is not pressed, jump to the firmware if the firmware exists in the firmware area of the flash memory and execute it.

4. 3. Communication Protocol

4. 3. 1. MAVLink

The communication protocol for downloading firmware from the boot loader uses MAVLink. MAVLink was created for communication in UAV, see the link below for details.

As a feature, when the command name and parameters are generated as an XML file, the communication code for each command is automatically generated by the code such as java / c / python. It creates a function that performs CRC checking or data parsing for communication automatically, so if you define only necessary commands as XML file, you can generate the source automatically.

It is also easy to port because it generates in various languages.

<http://qgroundcontrol.org/mavlink/start>

4. 3. 2. Composition of Commands

```

<?xml version="1.0"?>
<mavlink>
  <!--<include>common.xml</include>-->
  <!-- NOTE: If the included file already contains a version tag, remove the version tag
  <!--><version>3</version>-->
<enums>
</enums>
<messages>
  <message id="150" name="ACK">
    <description></description>
    <field type="uint8_t" name="msg_id"></field>
    <field type="uint16_t" name="err_code"></field>
    <field type="uint8_t" name="length"></field>
    <field type="uint8_t[16]" name="data"></field>
  </message>
</messages>
  ... omit ...
</mavlink>
  
```

The final defined XML file is added to GitHub.

https://github.com/ROBOTIS-GIT/OpenCR/blob/master/arduino/opencr_bootloader/common/msg/xml/opencr.xml

The defined xml file should be converted to the command code of the language you want to use through the MavLink utility. Download the MavLink utility source code from github below.

<https://github.com/mavlink/mavlink/>

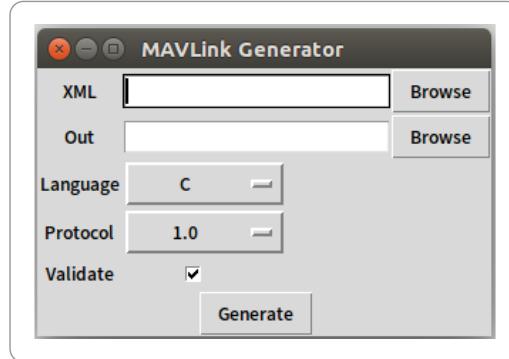
OpenCR1.0

It is written in Python and requires Python 2.7 or later. If it is not installed, add it.

```
$ sudo apt-get install python-tk
$ sudo apt-get install python-future
$ python mavgenerate.py
```

When you run MAVLink Generator, a GUI screen will appear, select the XML file already created in XML, set the language to C, and set the protocol version to 1.0. If you select the folder name for output in Out, and select Generate, a function header file that can use the command added based on the xml file is created and can be used in the firmware

[Back to Top ▲](#)



The final generated communication code can be seen in the link below.

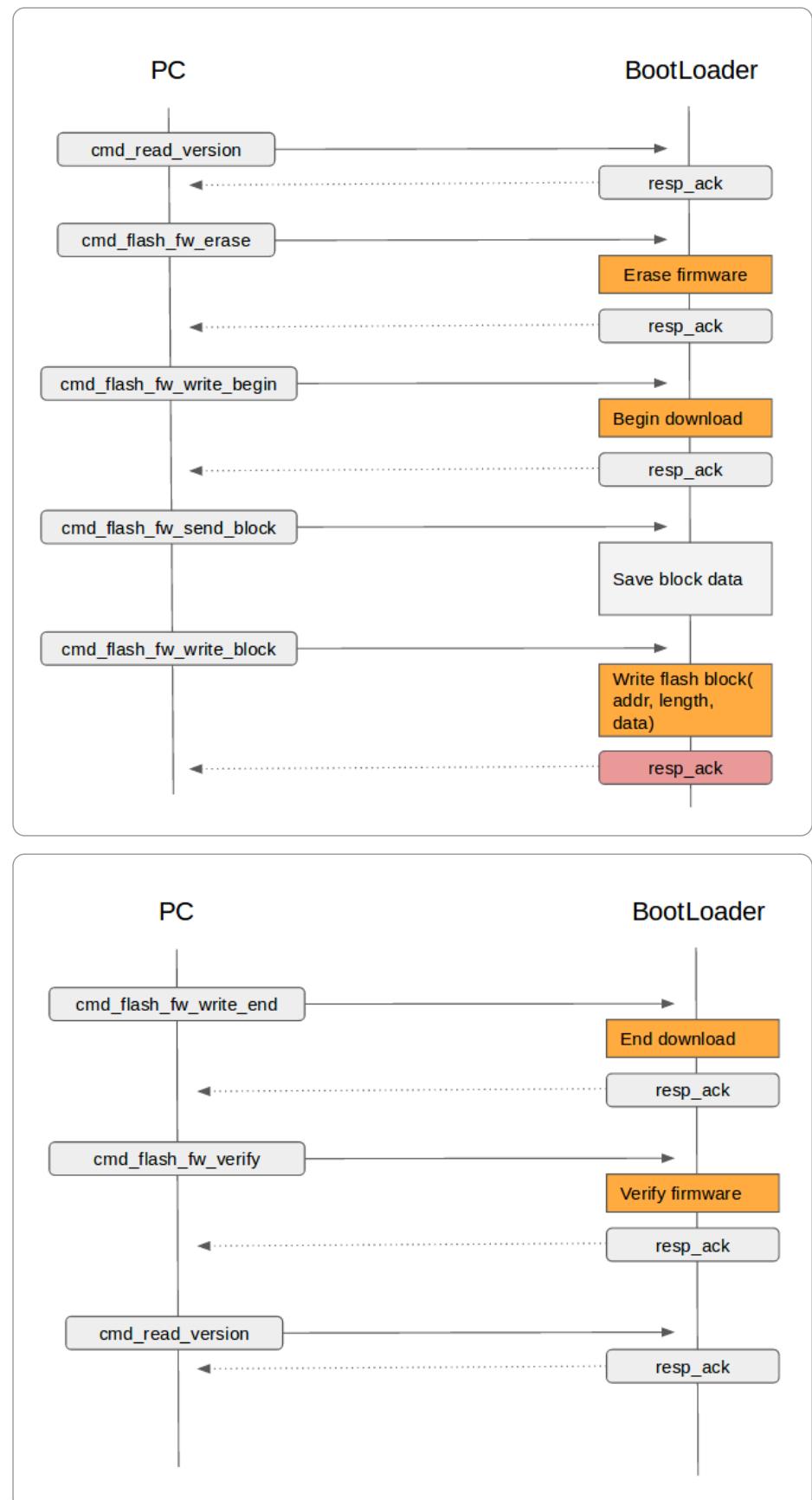
https://github.com/ROBOTIS-GIT/OpenCR/tree/master/arduino/opencr_develop/opencr_bootloader/common/msg/mavlink

The commands for the boot loader to download and execute the firmware through the MavLink protocol are as follows.

```
void cmd_read_version( msg_t *p_msg );
void cmd_read_board_name( msg_t *p_msg );
void cmd_jump_to_fw( msg_t *p_msg );
void cmd_flash_fw_write_packet( msg_t *p_msg );
void cmd_flash_fw_write_begin( msg_t *p_msg );
void cmd_flash_fw_write_end( msg_t *p_msg );
void cmd_flash_fw_write_block( msg_t *p_msg );
void cmd_flash_fw_erase( msg_t *p_msg );
void cmd_flash_fw_verify( msg_t *p_msg );
void cmd_flash_fw_read_block( msg_t *p_msg );
```

4. 3. 3. Download Sequence

The main function of the boot loader is to receive the firmware file from the PC, store it in flash, and execute the stored firmware. The order of downloading is as follows, and you can check how to proceed and how to proceed by looking at it. You can do the actual implementation based on this.

[OpenCR1.0](#)[Back to Top ▲](#)

4. 3. 4. Message Processing

In the code actually implemented, the main function calls the `msg_process_vcp()` function for message processing. In this case, if there is data coming from USB, `msg_recv` function is called to parse the message, and if any command is received, it returns TRUE to call the corresponding command function.

OpenCR1.0

```

int main(void)
{
    tTime = millis();
    while(1)
    {
        msg_process_vcp();
    }

    void msg_process_vcp(void)
    {
        BOOL ret;
        uint8_t ch;
        msg_t msg;

        while(vcp_is_available())
        {
            ch = vcp_getch();
            ret = msg_recv( 0, ch, &msg );

            if( ret == TRUE )
            {
                switch( msg.p_msg->msgid )
                {
                    case MAVLINK_MSG_ID_READ_VERSION:
                        cmd_read_version(&msg);
                        break;

                    case MAVLINK_MSG_ID_READ_BOARD_NAME:
                        cmd_read_board_name(&msg);
                        break;

                    ... omit ...

                    default:
                        cmd_send_error(&msg, ERR_INVALID_CMD);
                        break;
                }
            }
        }
    }
}

```

For example, the Mavlink_msg_read_version_decode() function parses the message data into the data structure of the command. If the field Responsive to parsed mav_data is active, it sends the boot loader version and revision via the ACK command.



```

void cmd_read_version( msg_t *p_msg )
{
    err_code_t err_code = OK;
    mavlink_ack_t mav_ack;
    mavlink_read_version_t mav_data;

    mavlink_msg_read_version_decode(p_msg->p_msg, &mav_data);

    if( mav_data.resp == 1 )
    {
        mav_ack.msg_id = p_msg->p_msg->msgid;
        mav_ack.err_code = err_code;
        mav_ack.data[0] = boot_version;
        mav_ack.data[1] = boot_version>>16;
        mav_ack.data[2] = boot_version>>24;
    }
}

```

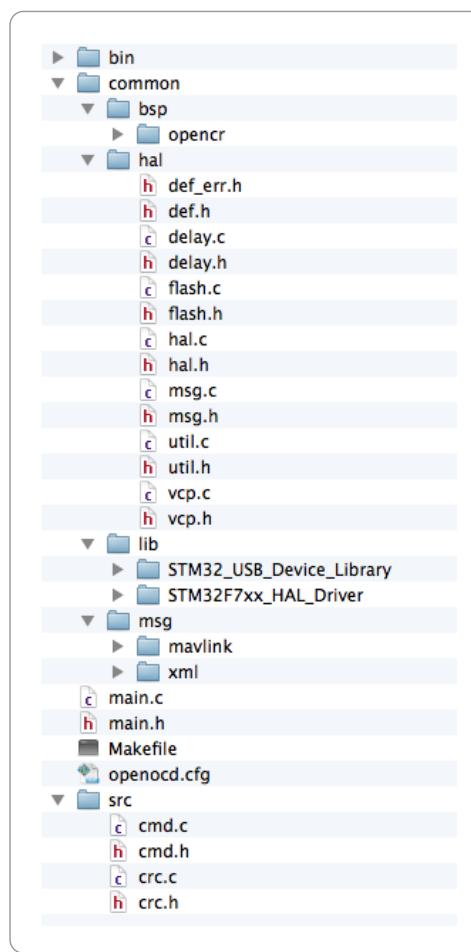
```

mav_ack.data[4] = boot_revision;
mav_ack.data[5] = boot_revision>>8;
mav_ack.data[6] = boot_revision>>16;
mav_ack.data[7] = boot_revision>>24;
mav_ack.length = 8;
resp_ack(p_msg->ch, &mav_ack);
}
}

```

OpenCR1.0**4. 3. 5. Folder Structure**

Item	Contents
bin	Save obj and bin files generated after build
common->bsp	Includes board-specific functions (LED / BUTTON / USB, etc.)
common->hal	Hardware independent function folders on the board
common->lib	Libraries used externally or publicly
msg->mavlink	Communication protocol source generated by Xml
msg->xml	The xml file folder where you defined the command
src	Function folder required for boot loader function

Back to Top ▲**4. 4. Update Bootloader(Linux)**

You can update the bootloader using the MCU's DFU mode on the OpenCR board. To update using DFU mode, you need to install dfu-util.

```
$ sudo apt-get install dfu-util
```

4. 4. 1. Enter DFU Mode

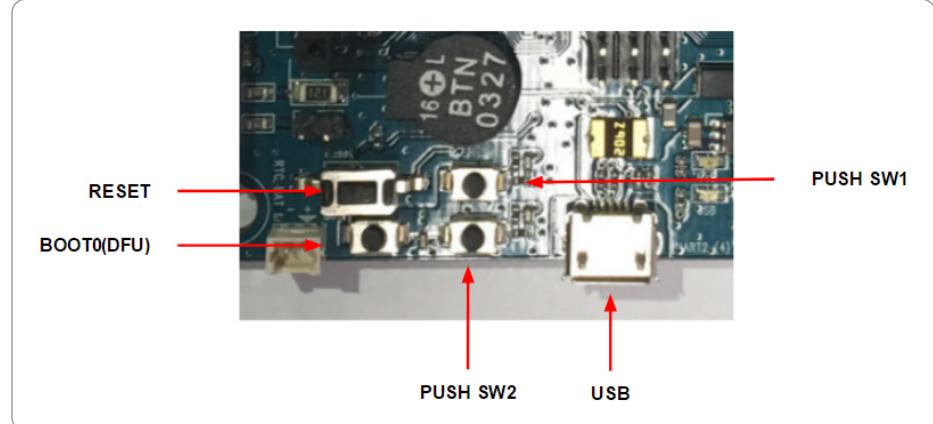
To run OpenCR in DFU mode, please follow the instruction below.

1. Hold down the **Boot** button.
2. Press the **Reset** button.
3. Release the **Reset** button.
4. Release the **Boot** button.

OpenCR will enter the DFU mode after reset by the built-in boot loader.

OpenCR1.0

[Back to Top ▲](#)



4. 4. 2. Check Boot Mode

If you run `lsusb`, you can check if it is in DFU mode. If the MCU is in DFU mode, the DFU device will be displayed after running `lsusb`.

```
$ lsusb
```

```
baram@baram-desktop:~$ lsusb
Bus 004 Device 002: ID 8087:8000 Intel Corp.
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 002: ID 8087:8008 Intel Corp.
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 058f:6364 Alcor Micro Corp. AU6477 Card Reader Controller
Bus 001 Device 003: ID a16d:c52h Logitech, Inc. Unifying Receiver
Bus 001 Device 007: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 001 Device 002: ID 05e3:0000 Genesys Logic, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
baram@baram-desktop:~$
```

4. 4. 3. Update Bootloader

After building the boot loader, move to the folder where the bin file is located and update it with `dfu-util`.

```
$ sudo dfu-util -d 0483:df11 -a 0 -s 0x08000000 -D ./opencr_boot.bin
```

OpenCR1.0[Back to Top ▲](#)

```
baram@baram-desktop: ~/develop/git/OpenCR/arduino/opencr_bootloader/bin
-util -d 0483:df11 -a 0 -s 0x08000000 -D ./opencr_boot.bin
dfu-util 0.5

(c) 2005-2008 by Weston Schmidt, Harald Welte and OpenMoko Inc.
(c) 2010-2011 Tormod Volden (DfuSe support)
This program is Free Software and has ABSOLUTELY NO WARRANTY

dfu-util does currently only support DFU version 1.0

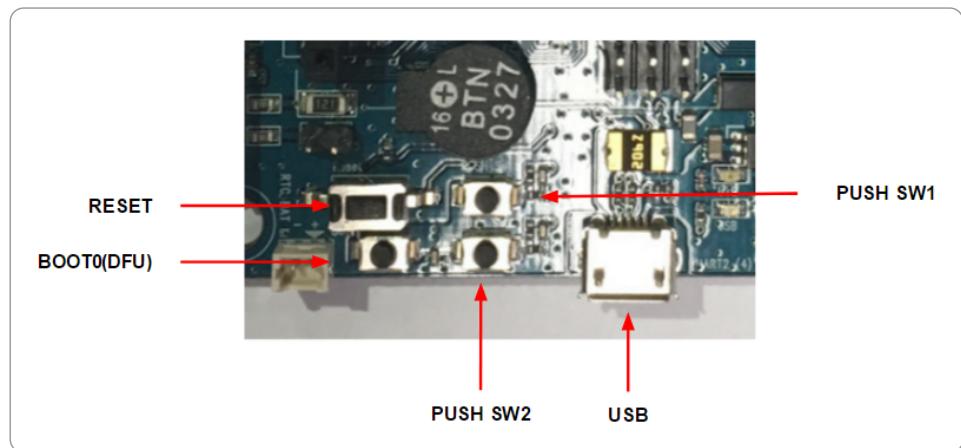
Filter on vendor = 0x0483 product = 0xdf11
Opening DFU USB device... ID 0483:df11
Run-time device DFU version 011a
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="@Internal Flash  /0
x08000000/04*032Kg,01*128Kg,03*256Kg"
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
No valid DFU suffix signature
Warning: File has no DFU suffix
Dfuse interface name: "Internal Flash "
baram@baram-desktop:~/develop/git/OpenCR/arduino/opencr_bootloader/bin$
```

4. 5. Firmware Recovery Mode

If corrupted or incompletely firmware is downloaded and the board freezes or does not work, you must enter the boot loader to be able to download the normal firmware. To execute the boot loader, please follow the instruction below.

1. Hold down the **PUSH SW2** button.
2. Press the **Reset** button.
3. Release the **Reset** button.
4. Release the **PUSH SW2** button.

OpenCR will enter the boot loader after reset. When the boot loader is running, the STATUS LED blinks every 100ms.



You can download the normal firmware while the boot loader is running.

5. Downloader

The PC program Downloader communicates with the boot loader and downloads the firmware from the PC to the firmware area of the OpenCR board flash.

Item	Description
Supported OS	Windows, Linux, Mac
Compiler	Linux : gcc Windows : Qt 5.7

5. 1. Usage

```
$ opencr_ld <Communication port> <Baudrate> <Firmware binary> <Firmware execution status [0|1]>
```



- Communication port: The serial port name is usually `/dev/ttyACM0` for Linux, and it should be the same as the serial port connected to OpenCR.
- Baudrate : The speed to communicate and input at 115,200bps.
- Firmware binary : The firmware binary image has an extension of bin.
- Firmware execution status : In case of 1, the firmware will be executed after downloading the firmware. If it is not input or if it is 0, only downloading the firmware is performed.

5. 2. Linux/Mac Example

```
$ sudo opencr_ld /dev/ttyACM0 115200 ./opencrfw.bin 1
```

5. 3. Windows Example

[Back to Top ▲](#)

```
opencr_ld.exe COM1 115200 ./opencrfw.bin 1
```

5. 4. Execution Result

```
opencr_ld ver 1.0.2
opencr_ld_main
>>
file name : /tmp/arduino_build_193791/Blink.ino.bin
tile size : 29 KB
Open port OK
Clear Buffer Start
Clear Buffer End
>>
Board Name : OpenCR R1.0
Board Ver : 0x16100600
Board Rev : 0x00000000
>>
flash_erase : 0 : 1.029000 sec
flash_write : 0 : 0.157000 sec
CRC OK 2BA000 2BA000 0.001000 sec
[OK] Download
jump_to_fw
```

5. 5. Download Executable File

- https://github.com/ROBOTIS-GIT/OpenCR/tree/master/arduino/opencr_arduino/tools/opencr_tools_1.0.0

6. Arduino IDE

6. 1. Install on Linux

6. 1. 1. USB Port Settings

Make the OpenCR USB port be able to upload the Arduino IDE program without root permission.

```
$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/99-opencr-cdc.rules
$ sudo cp ./99-opencr-cdc.rules /etc/udev/rules.d/
$ sudo udevadm control --reload-rules
$ sudo udevadm trigger
```

OpenCR1.0

```
willson@willson-XPS: ~ willson@WillSon-XPS: ~ 132x72
willson@willson-XPS: ~ wget https://raw.githubusercontent.com/ROBOTIS-OpenCR/master/99-opencr-cdc.rules
2017-05-26 15:24:19 -> "https://raw.githubusercontent.com/ROBOTIS-OpenCR/master/99-opencr-cdc.rules"
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.72.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.72.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 516 [text/plain]
Saving to: '99-opencr-cdc.rules.1'

99-opencr-cdc.rules.1                                         100%[=====]  59.2 MB/s

2017-05-26 15:24:20 (59.2 MB/s) - '99-opencr-cdc.rules.1' saved [516/516]

willson@willson-XPS: ~ sudo cp ./99-opencr-cdc.rules /etc/udev/rules.d/
[sudo] password for willson:
willson@willson-XPS: ~ sudo udevadm control --reload-rules
willson@willson-XPS: ~ sudo udevadm trigger
willson@willson-XPS: ~
```

6. 1. 2. Compiler Settings

Since the OpenCR libraries is built for 32 bit platform, 64 bit PC needs the 32 bit compiler relevants for the ArduinoIDE.

[Back to Top ▲](#)

```
$ sudo apt-get install libncurses5-dev:i386
```

6. 1. 3. Install Arduino IDE(Linux)

Download the latest version of Arduino IDE from the official arduino homepage, and install it. Currently, the OpenCR will be on service in the version 1.6.4 or later.

<https://www.arduino.cc/en/Main/Software>

Then, extract the downloaded file to the desired folder and execute the installation file from the terminal. In this case, the example shown below makes the folder tools in the user's top folder (~/). This folder will act as the Arduino IDE folder.

```
$ cd ~/tools/arduino-1.16.0  
$ ./install.sh
```

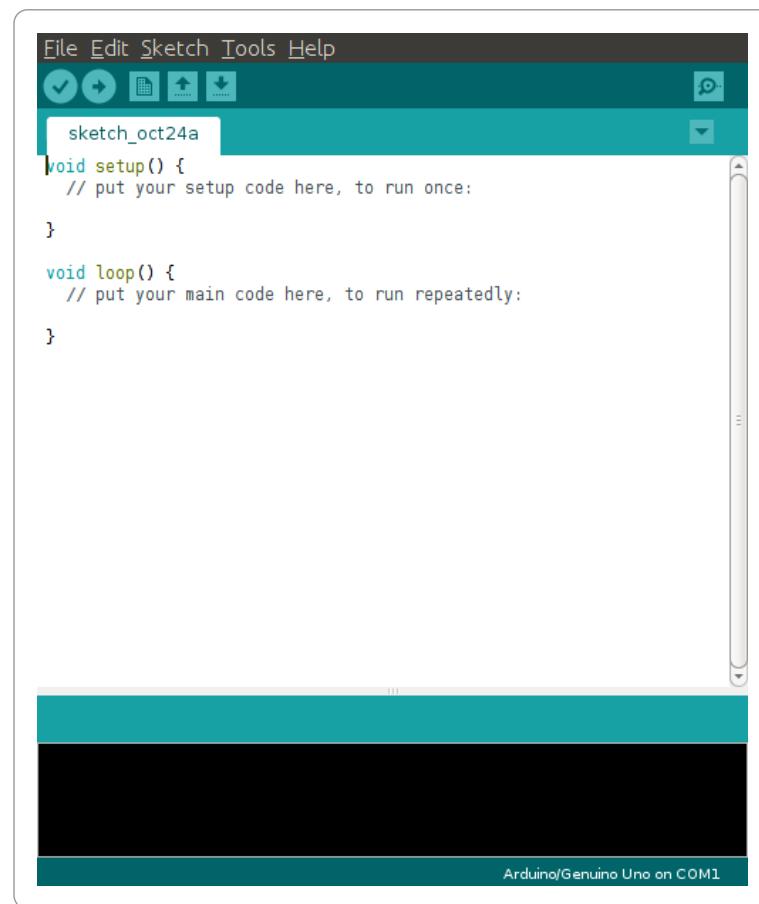
Set the file path of installed Arduino IDE as an absolute path named PATH in the bashrc file. Here recommends to use gedit editor. (Use another editor, if necessary.) Finally, source it to apply the changes.

```
$ gedit ~/.bashrc
$ export PATH=$PATH:$HOME/tools/arduino-1.16.0
$ source ~/.bashrc
```

6. 1. 4. Run Arduino IDE(Linux)

To run the Arduino IDE on Linux platform, type into the terminal as follows.

```
$ arduino
```

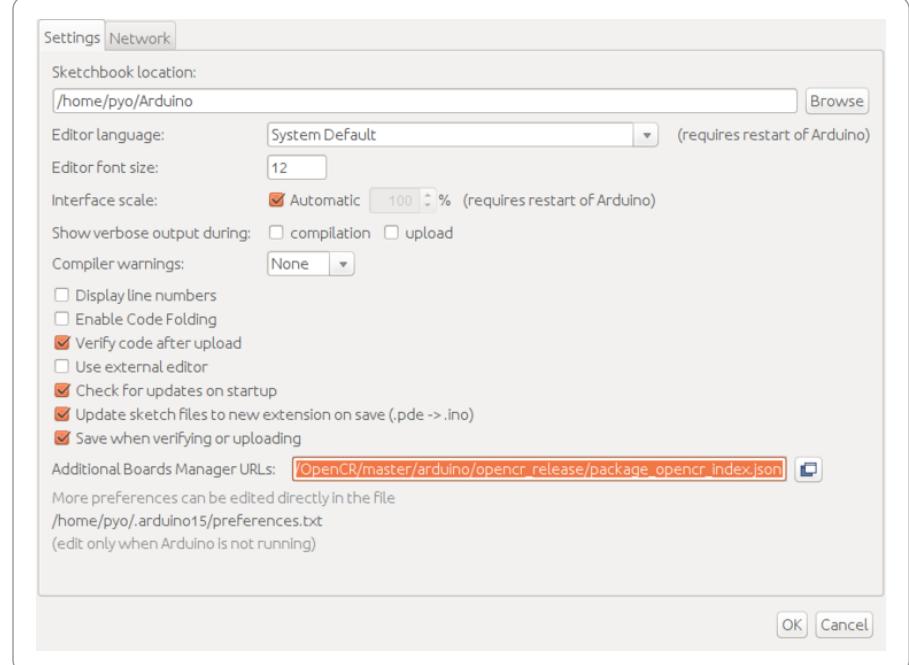


6. 1. 5. Porting to Arduino IDE(Linux)

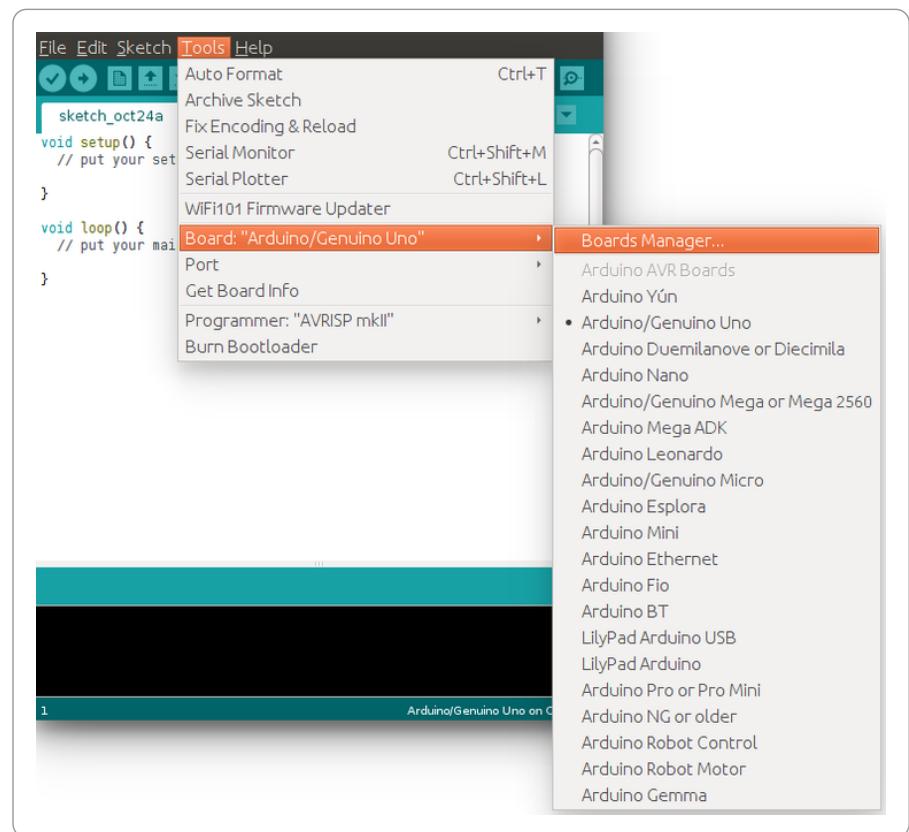
6. 1. 5. 1. Preferences

After Arduino IDE is run, click File → Preferences in the top menu of the IDE. When the Preferences window appears, copy and paste following link to the Additional Boards Manager URLs textbox. (This step may take about 20 min.)

https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/arduino/opencr_release/package_index.json

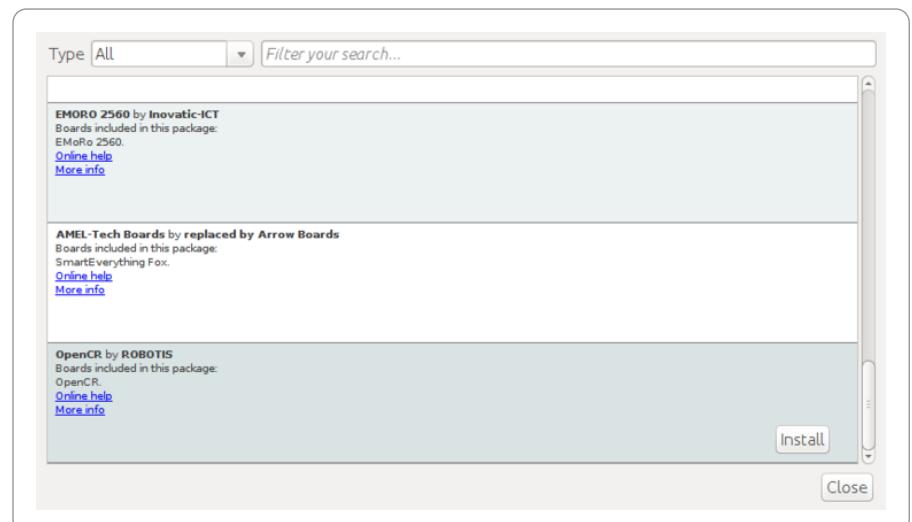
OpenCR1.0[Back to Top ▲](#)**6. 1. 5. 2. Install the OpenCR package via Boards Manager**

Click Tools → Board → Boards Manager.



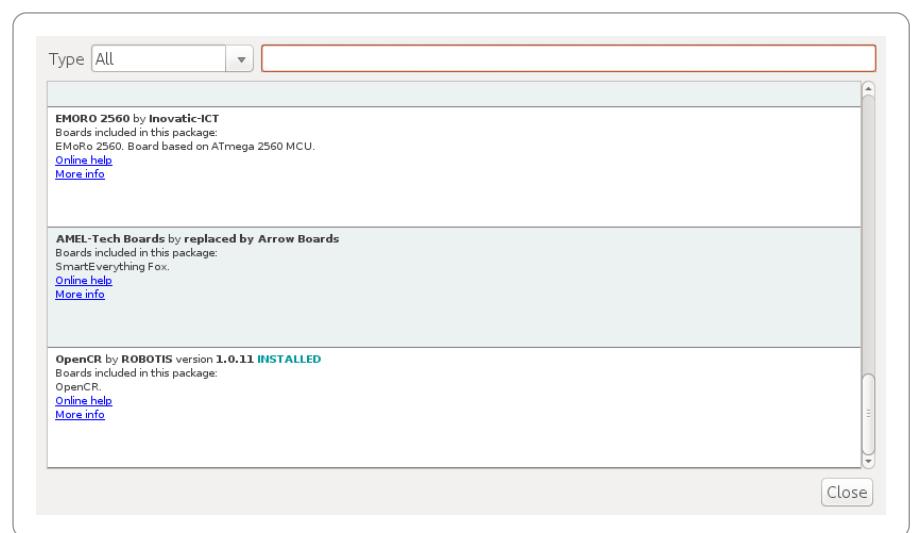
Type OpenCR into the textbox to find the OpenCR by ROBOTIS package. After it finds out, click Install.

OpenCR1.0

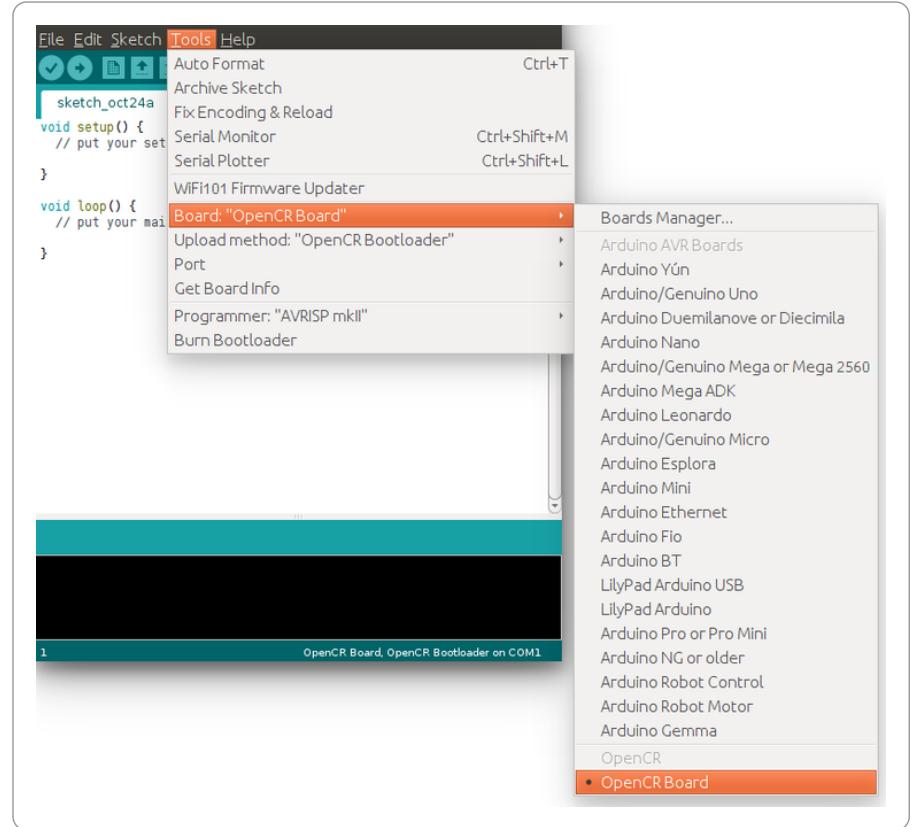


[Back to Top ▲](#)

After the installation, "INSTALLED" will be appeared.



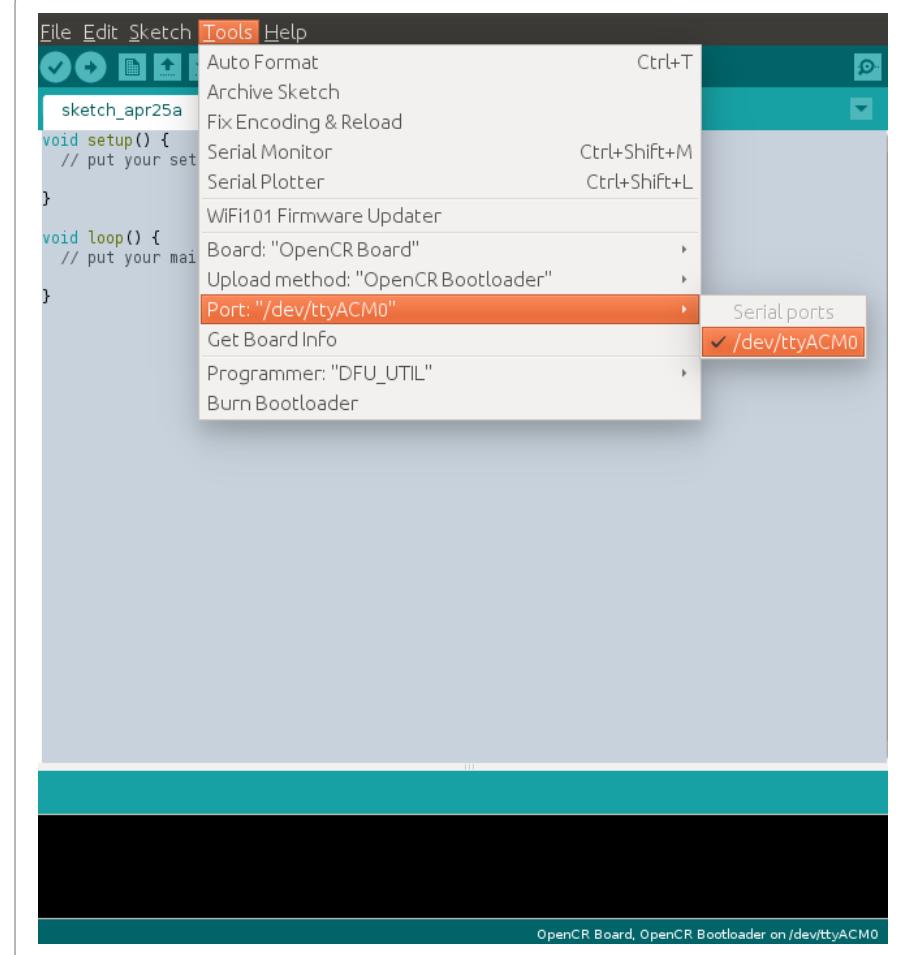
See if OpenCR Board is now on the list of Tools → Board. Click this to import the OpenCR Board source.

OpenCR1.0[Back to Top ▲](#)**6. 1. 5. 3. Port setting**

This step shows the port setting for the program uploads. The OpenCR should be connected to the PC and the OpenCR via the USB ports.

Select Tools → Port → /dev/ttyACM0.

Warning The last digit value `0` in the string `/dev/ttyACM0` might be different depend on the USB connection environment.

OpenCR1.0[Back to Top ▲](#)**6. 1. 6. Remove Modemmanager**

After programming with the Arduino IDE and uploading the program to the OpenCR, the OpenCR will be restarted and be reconnected. At the same moment, the modem-related packages of the Linux will send the AT command to manage the device. Thus indicates an connection error on the OpenCR, so this step should be done previously.

```
$ sudo apt-get purge modemmanager
```

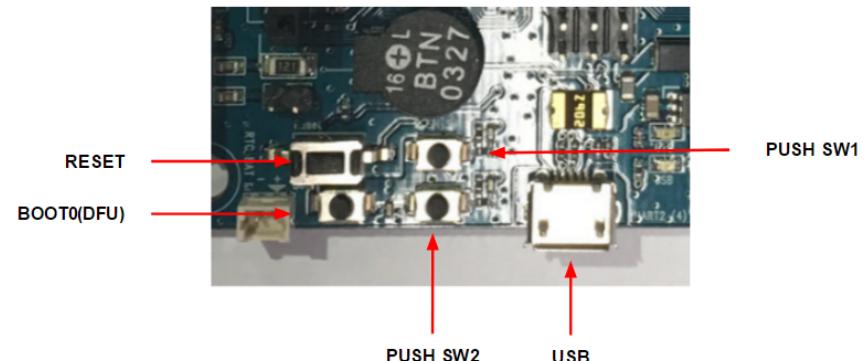
6. 1. 7. Writing Bootloader(Linux)

Caution Update only if the boot loader version has been changed.

The STM32F7xx, which is used for the main MCU on the OpenCR board, supports DFU(Device Firmware Upgrade). This enables the built-in bootloader of the MCU by itself to boot the DFU protocol by using USB, primarily for the bootloader initialization, the recovery mode, and the bootloader update. The biggest advantage to let the users be able to use bootloader with USB but no other JTAG equipment. Write the firmware by using the DFU mode which is embedded in MCU without writing / debugging equipment, such as STLink.

6. 1. 7. 1. Programmer Setting

Select Tools → DFU-UTIL

OpenCR1.0**6. 1. 7. 2. Run DFU mode.****Back to Top ▲**

Press the **Reset Button** while the **Boot Button** is being pushed. This activates the DFU mode.

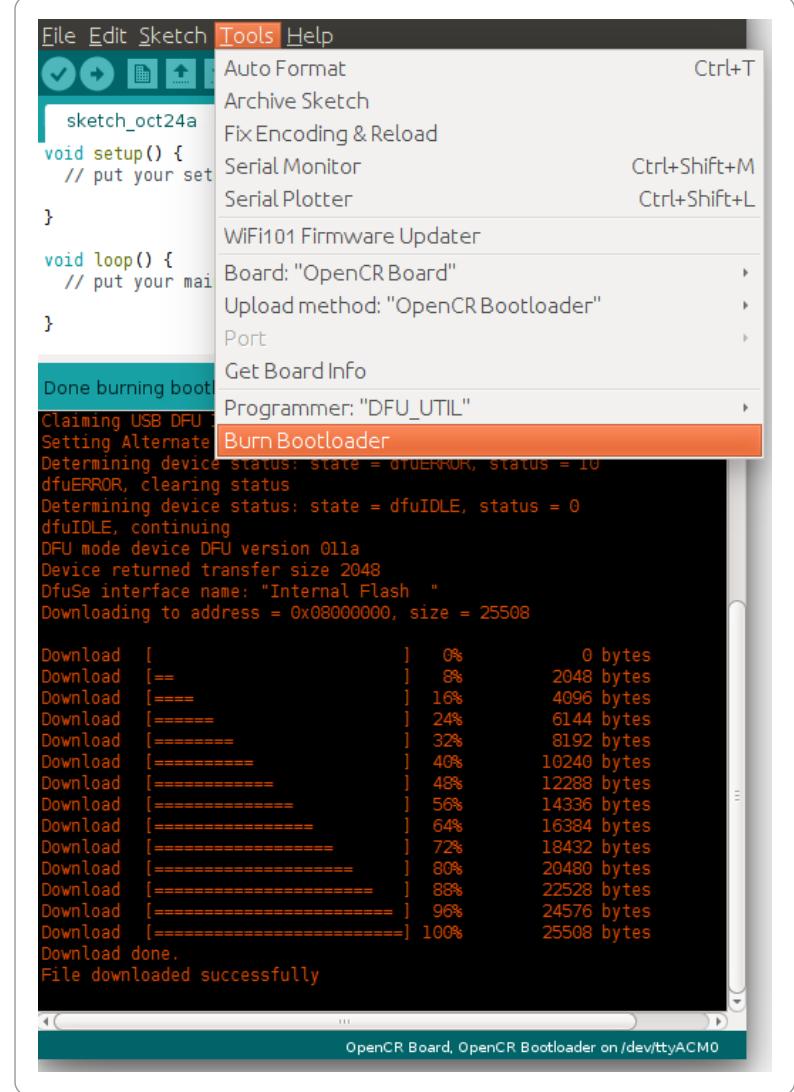
If you successfully entered to DFU mode, you will be able to find

STMicroelectronics STM Device in DFU Mode text string when **lsusb** is entered in the terminal.

```
leon@leon: ~
@ :~$ lsusb
Bus 002 Device 002: ID 8087:8000 Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 8087:8008 Intel Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 003: ID 8564:1000 Transcend Information, Inc. JetFlash
Bus 004 Device 002: ID 2109:0812 VIA Labs, Inc. VL812 Hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 008: ID 8087:07dc Intel Corp.
Bus 003 Device 007: ID 046d:c531 Logitech, Inc. C-U0007 [Unifying Receiver]
Bus 003 Device 005: ID 046d:c52b Logitech, Inc. Unifying Receiver
Bus 003 Device 003: ID 0bda:0129 Realtek Semiconductor Corp. RTS5129 Card Reader Controller
Bus 003 Device 006: ID 05e3:0608 Genesys Logic, Inc. Hub
Bus 003 Device 004: ID 046d:c07c Logitech, Inc. M-R0017 [G700s Rechargeable Gaming Mouse]
Bus 003 Device 002: ID 2109:2812 VIA Labs, Inc. VL812 Hub
Bus 003 Device 009: ID 2232:1045 Silicon Motion
Bus 003 Device 014: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
@ :~$
```

6. 1. 7. 3. Download the bootloader.

Click Tools → Burn Bootloader to download the bootloader.

OpenCR1.0[Back to Top ▲](#)**6. 2. Install on Mac****6. 2. 1. Install Arduino IDE(Mac)**

Download the latest version of Arduino IDE from the official arduino homepage, and install it. Currently, the OpenCR will be on service in the version 1.6.4 or later.

<https://www.arduino.cc/en/Main/Software>

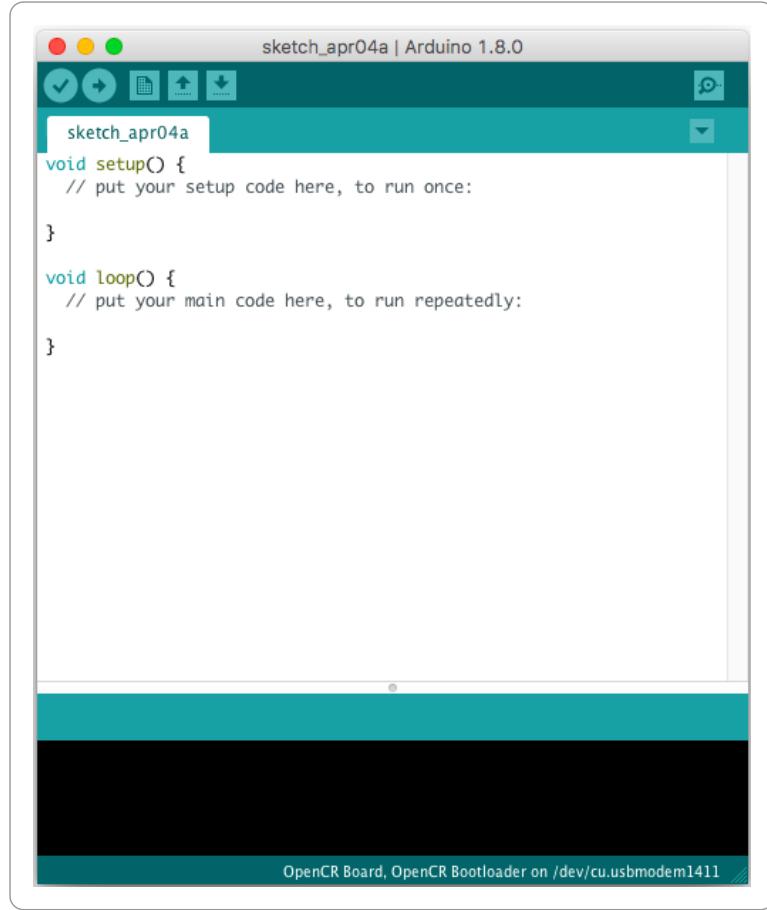
6. 2. 2. Run Arduino IDE(Mac)

To run the Arduino IDE on Mac platform, click the Arduino IDE icon as follows.



OpenCR1.0

[Back to Top ▲](#)

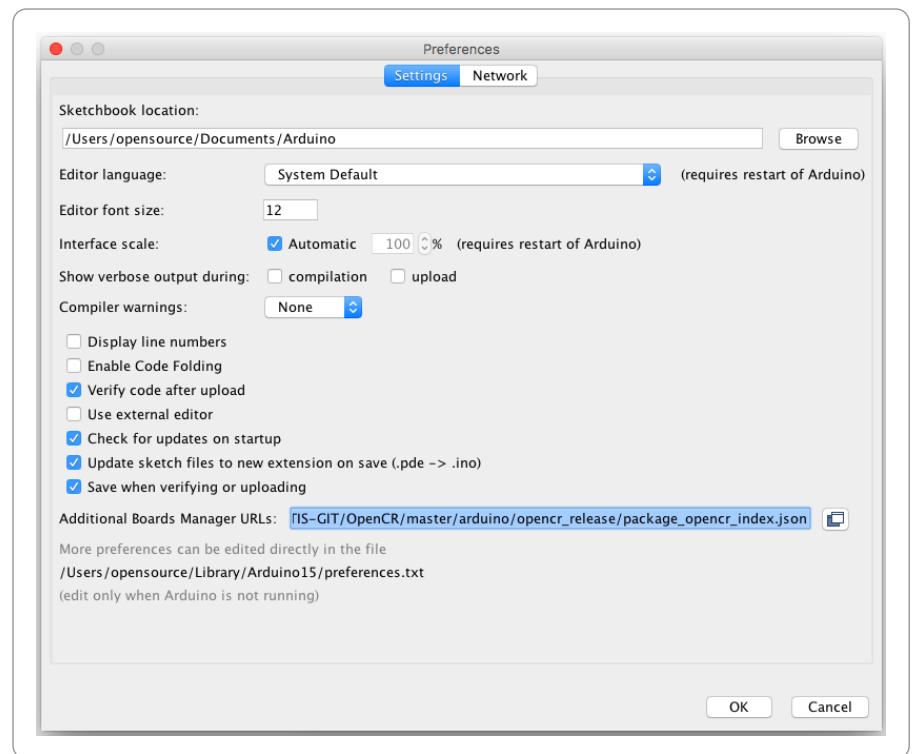


6.2.3. Porting to Arduino IDE(Mac)

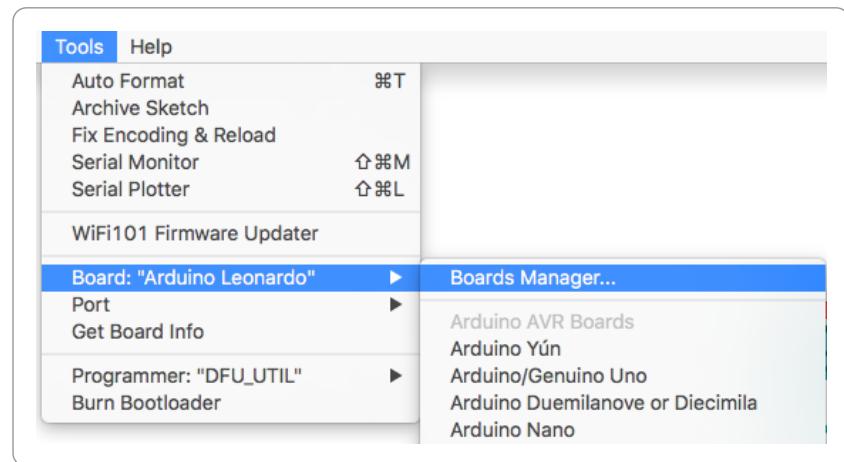
6.2.3.1. Preferences

After Arduino IDE is run, click File → Preferences in the top menu of the IDE. When the Preferences window appears, copy and paste following link to the Additional Boards Manager URLs textbox. (This step may take about 20 min.)

https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/arduino/opencr_release/package_opencr_index.json

OpenCR1.0[Back to Top ▲](#)**6.2.3.2. Install the OpenCR package via Boards Manager**

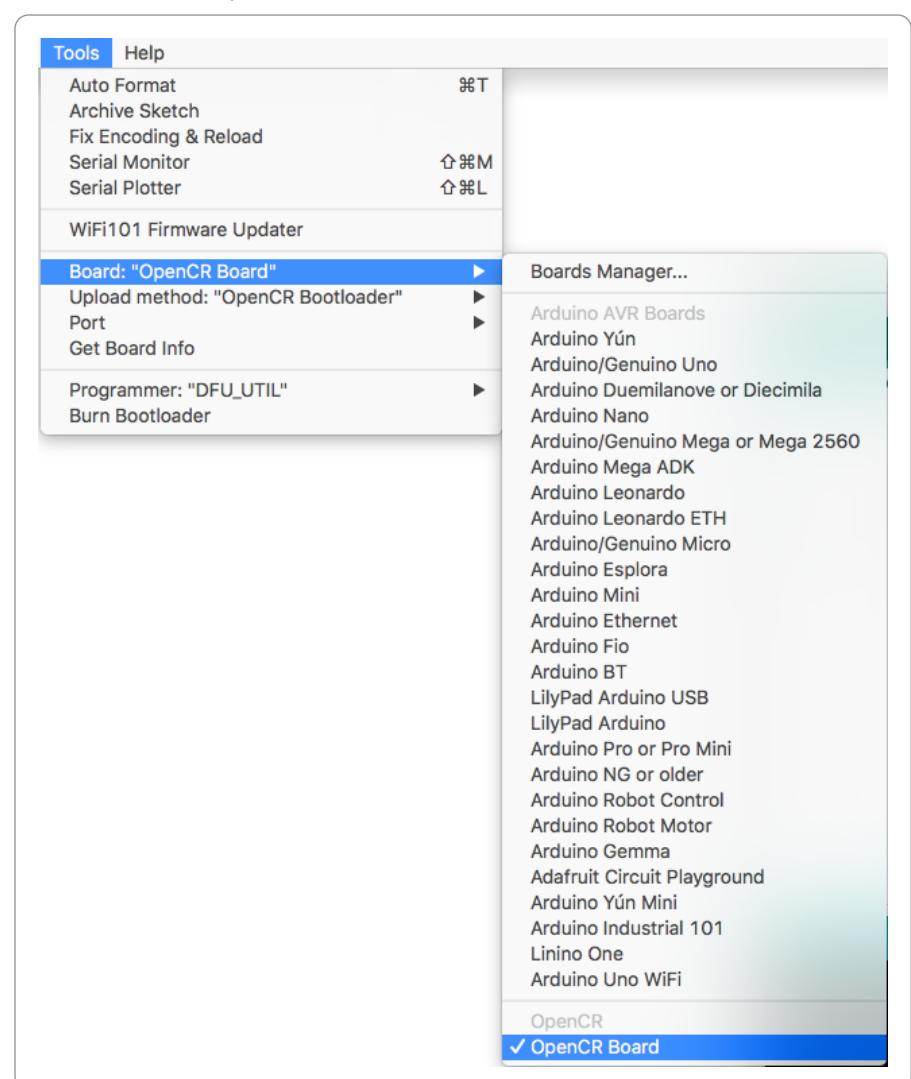
Click Tools → Board → Boards Manager.



Type OpenCR into the textbox to find the OpenCR by ROBOTIS package. Install of the OpenCR package.

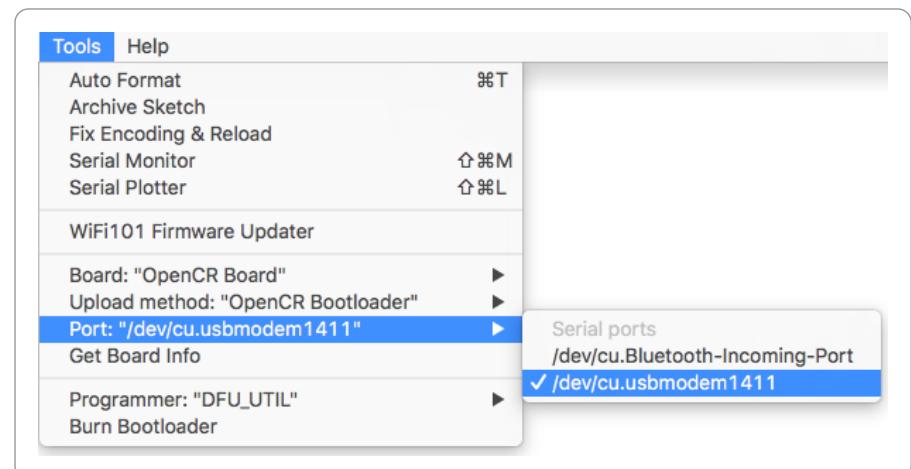
After the installation, "INSTALLED" will be appeared.

See if OpenCR Board is now on the list of Tools → Board. Click this to import the OpenCR Board source.

OpenCR1.0[Back to Top ▲](#)**6. 2. 3. 3. Port setting**

This step shows the port setting for the program uploads. The OpenCR should be connected to the PC and the OpenCR via the USB ports.
Select Tools → Port → /dev/cu.usbmodem1411

Caution The value of `/dev/cu.usbmodem1411` may be different depending on the environment connected to the PC.

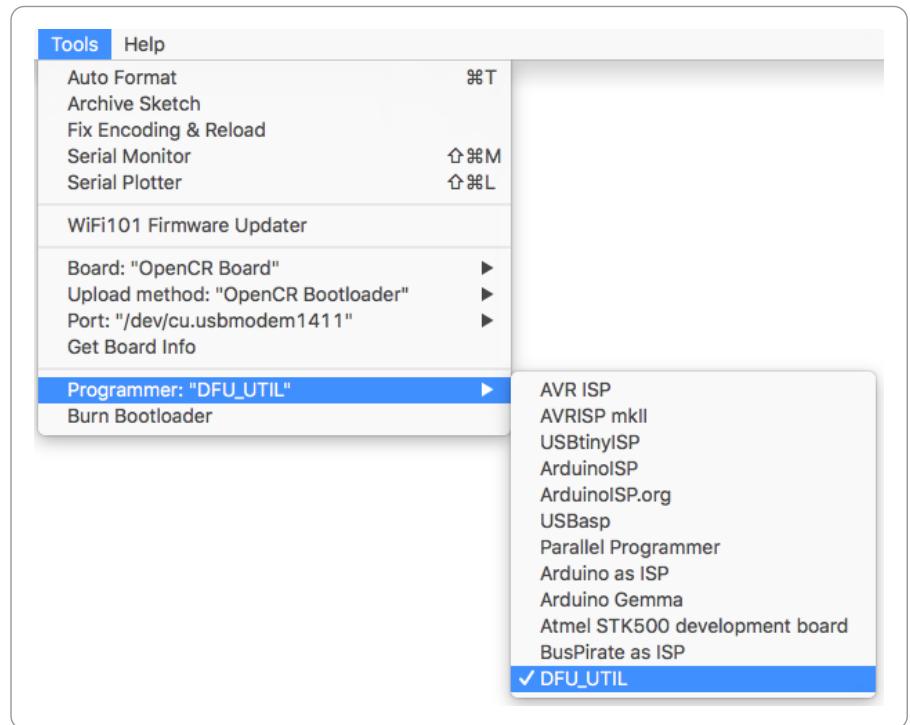
**6. 2. 4. Writing Bootloader(Mac)**

OpenCR1.0

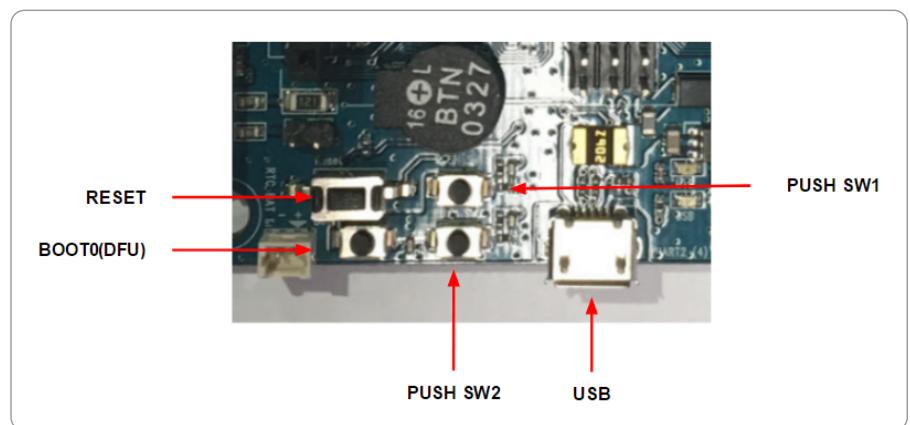
The STM32F7xx, which is used for the main MCU on the OpenCR board, supports DFU(Device Firmware Upgrade). This enables the built-in bootloader of the MCU by itself to boot the DFU protocol by using USB, primarily for the bootloader initialization, the recovery mode, and the bootloader update. The biggest advantage to let the users be able to use bootloader with USB but no other JTAG equipment. Write the firmware by using the DFU mode which is embedded in MCU without writing / debugging equipment, such as STLink.

6. 2. 4. 1. Programmer Setting

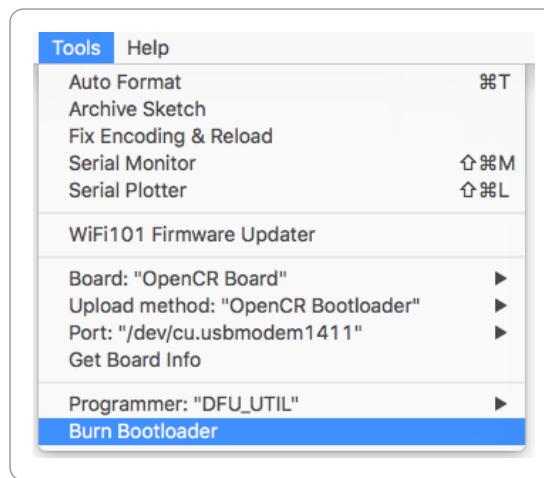
Select Tools → DFU-UTIL

**6. 2. 4. 2. Run DFU Mode**

Press the [Reset Button] while the [Boot Button] is being pushed. This activates the DFU mode.

**6. 2. 4. 3. Download bootloader**

Click Tools → Burn Bootloader to download the bootloader.

OpenCR1.0[Back to Top ▲](#)**6. 3. Install on Windows****6. 3. 1. Install Driver**

To use OpenCR's USB port as a serial port in Windows, you need a USB CDC driver. You can install the USB driver provided by ST.

<http://www.st.com/en/development-tools/stsw-stm32102.html>

6. 3. 2. Install Arduino IDE(Windows)

Download the latest version of Arduino IDE from the official arduino homepage, and install it. Currently, the OpenCR will be on service in the version 1.16.0 or later.

<https://www.arduino.cc/en/Main/Software>

The Arduino IDE for Windows is available as an installation version and a compressed version, so you can install it using your preferred method.

6. 3. 3. Porting to Arduino IDE(Windows)**6. 3. 3. 1. Preferences**

After Arduino IDE is run, click File → Preferences in the top menu of the IDE. When the Preferences window appears, copy and paste following link to the Additional Boards Manager URLs textbox. (This step may take about 20 min.)

https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/arduino/opencr_release/package_opencr_index.json

6. 3. 3. 2. Install the OpenCR package via Boards Manager

1. Click Tools → Board → Boards Manager.
2. Type OpenCR into the textbox to find the OpenCR by ROBOTIS package. After it finds out, click Install.
3. After the installation, "INSTALLED" will be appeared.
4. See if OpenCR Board is now on the list of Tools → Board. Click this to import the OpenCR Board source.

6. 3. 3. 3. Port setting

This step shows the port setting for the program uploads. The OpenCR should be connected to the PC and the OpenCR via the USB ports.

Select Tools → Port → COM1.

Caution The value of **COM1** may be different depending on the environment connected to the PC.

6. 4. Arduino Pinmap

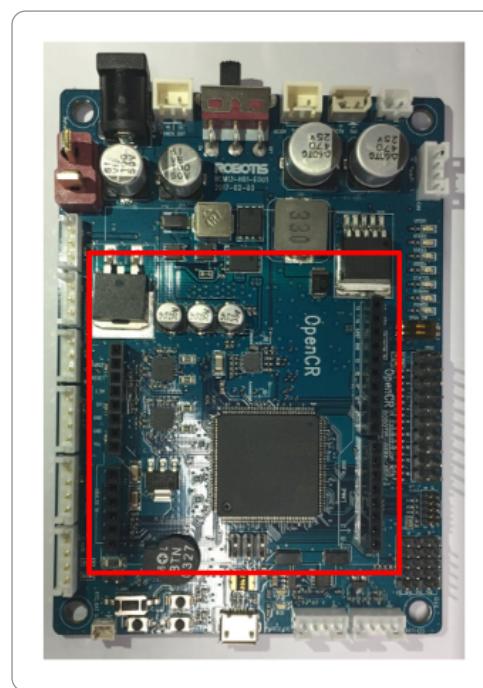
OpenCR includes a connector that is compatible with Arduino Uno pinmap.

6. 4. 1. Arduino Connector

The pins 0 to 21 are the same pin as the Arduino Uno, and thereafter they are mapped to the pins added to OpenCR.

[OpenCR1.0](#)

[Back to Top ▲](#)

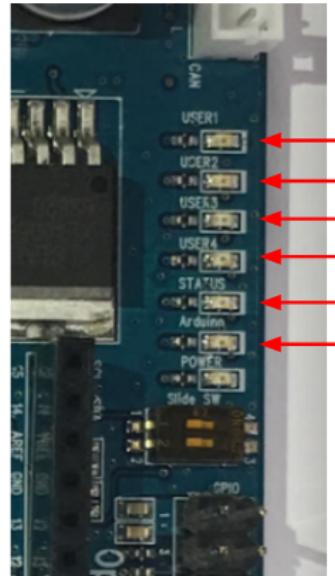


Pin No.	Function	1	2	3
0	UART RXD	UART6_RX		
1	UART TXD	UART6_TX		
2				EXTI_0
3	PWM	TIM3_CH1		EXTI_1
4				EXTI_2
5	PWM	TIM1_CH1		
6	PWM	TIM2_CH3		
7				EXTI_3
8				EXTI_4
9	PWM	TIM9_CH2		
10	PWM/NSS	TIM11_CH1	SPI2_NSS	
11	PWM/MOSI	TIM12_CH2	SPI2_MOSI	
12	MISO		SPI2_MISO	
13	SCK		SPI2_SCK	
14	SDA		I2C1_SDA	
15	SCL		I2C1_SCL	
16	ADC	A0		
17	ADC	A1		
18	ADC	A2		
19	ADC	A3		
20	ADC	A4		
21	ADC	A5		

6. 4. 2. User LED

The OpenCR additional LEDs consist of four LEDs and are mapped to Arupinopin 22-25.

[OpenCR1.0](#)



[Back to Top ▲](#)

Name	Arduino Pin	Pin Name
USER1	22	BDPIN_LED_USER_1
USER2	23	BDPIN_LED_USER_2
USER3	24	BDPIN_LED_USER_3
USER4	25	BDPIN_LED_USER_4
STS	36	BDPIN_LED_STATUS
Arduino	13	LED_BUILTIN

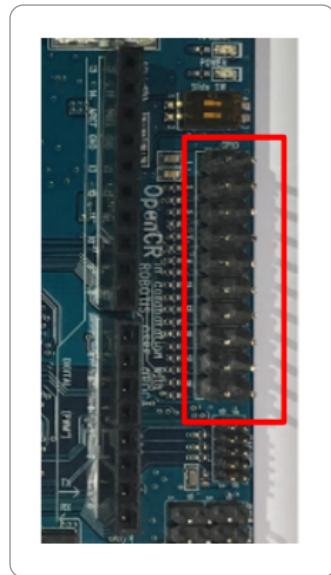
6. 4. 3. Dip Switch



Arduino Pin	Pin Name
26	BDPIN_DIP_SW_1
27	BDPIN_DIP_SW_2

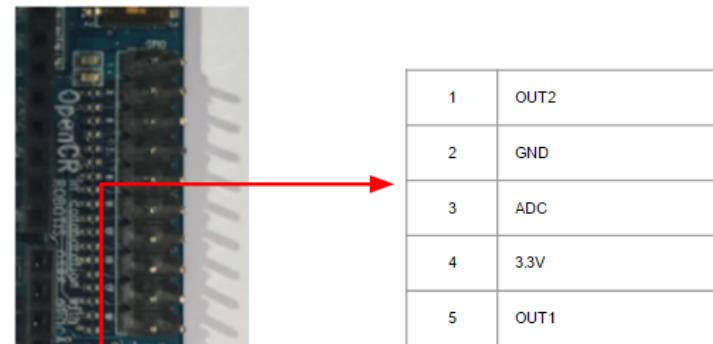
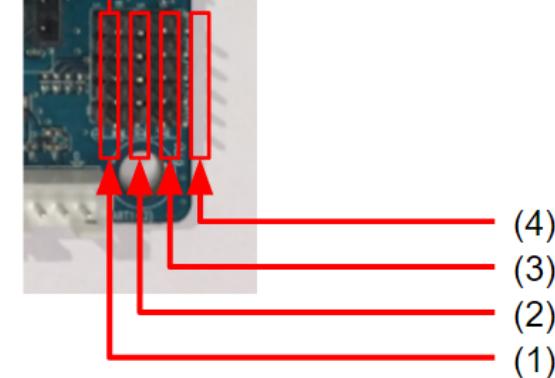
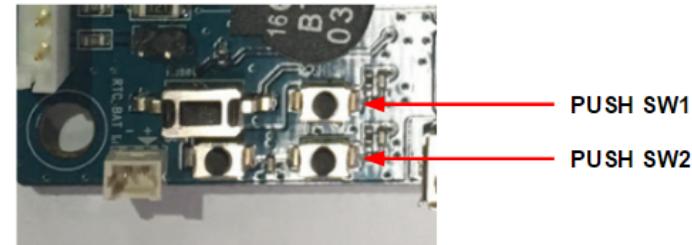
6. 4. 4. GPIO

It has an 18-pin common GPIO expansion connector and is mapped to the GPIO pin of the Arduino. The pin number below is the arduino pin number.

[OpenCR1.0](#)[Back to Top ▲](#)

Pin Number	Arduino Pin	Pin Name	Pin Number	Arduino Pin	Pin Name
1	-	3.3V	2	-	GND
3	50	BDPIN_GPIO_1	4	51	BDPIN_GPIO_1
5	52	BDPIN_GPIO_3	6	53	BDPIN_GPIO_1
7	54	BDPIN_GPIO_5	8	55	BDPIN_GPIO_1
9	56	BDPIN_GPIO_7	10	57	BDPIN_GPIO_1
11	58	BDPIN_GPIO_9	12	59	BDPIN_GPIO_1
13	60	BDPIN_GPIO_11	14	61	BDPIN_GPIO_1
15	62	BDPIN_GPIO_13	16	63	BDPIN_GPIO_1
17	64	BDPIN_GPIO_15	18	65	BDPIN_GPIO_1
19	66	BDPIN_GPIO_17	20	67	BDPIN_GPIO_1

6. 4. 5. OLLO Connector

OpenCR1.0[Back to Top ▲](#)**6.4.6. Push Switch**

Arduino Pin	Pin Name
34	BDPIN_PUSH_SW_1
35	BDPIN_PUSH_SW_2

6.4.7. Pin Definition

```
extern const Pin2PortMapArray g_Pin2PortMapArray[] =
{
    {GPIOC, GPIO_PIN_7,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , NO_EXTI }, // 0
    {GPIOC, GPIO_PIN_6,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , NO_EXTI }, // 1
    {GPIOG, GPIO_PIN_6,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , 0      }, // 2
    {GPIOB, GPIO_PIN_4,    NULL,    NO_ADC,      , &hTIM3,    TIM_CHANNEL_1, 1      }, // 3
    {GPIOG, GPIO_PIN_7,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , 2      }, // 4
    {GPIOA, GPIO_PIN_8,    NULL,    NO_ADC,      , &hTIM1,    TIM_CHANNEL_1, NO_EXTI }, // 5
    {GPIOA, GPIO_PIN_2,    NULL,    NO_ADC,      , &hTIM2,    TIM_CHANNEL_3, NO_EXTI }, // 6
    {GPIOC, GPIO_PIN_1,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , 3      }, // 7
    {GPIOC, GPIO_PIN_2,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , 4      }, // 8
    {GPIOA, GPIO_PIN_3,    NULL,    NO_ADC,      , &hTIM9,    TIM_CHANNEL_2, NO_EXTI }, // 9
    {GPIOB, GPIO_PIN_9,    NULL,    NO_ADC,      , &hTIM11,   TIM_CHANNEL_1, NO_EXTI }, // 10
    {GPIOB, GPIO_PIN_15,   NULL,    NO_ADC,      , &hTIM12,   TIM_CHANNEL_2, NO_EXTI }, // 11
    {GPIOB, GPIO_PIN_14,   NULL,    NO_ADC,      , NULL,      , NO_PWM,      , NO_EXTI }, // 12
    {GPIOA, GPIO_PIN_9,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , NO_EXTI }, // 13
    {GPIOB, GPIO_PIN_7,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , NO_EXTI }, // 14
    {GPIOB, GPIO_PIN_8,    NULL,    NO_ADC,      , NULL,      , NO_PWM,      , NO_EXTI }, // 15
}
```

OpenCR1.0

```

{GPIOA, GPIO_PIN_0, &hADC3, ADC_CHANNEL_0 , NULL , NO_PWM , NO_EXTI }, // 16
{GPIOF, GPIO_PIN_10, &hADC3, ADC_CHANNEL_8 , NULL , NO_PWM , NO_EXTI }, // 17
{GPIOF, GPIO_PIN_9, &hADC3, ADC_CHANNEL_7 , NULL , NO_PWM , NO_EXTI }, // 18
{GPIOF, GPIO_PIN_8, &hADC3, ADC_CHANNEL_6 , NULL , NO_PWM , NO_EXTI }, // 19
{GPIOF, GPIO_PIN_7, &hADC3, ADC_CHANNEL_5 , NULL , NO_PWM , NO_EXTI }, // 20
{GPIOF, GPIO_PIN_6, &hADC3, ADC_CHANNEL_4 , NULL , NO_PWM , NO_EXTI }, // 21

{GPIOG, GPIO_PIN_12, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 22
{GPIOE, GPIO_PIN_5, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 23
{GPIOE, GPIO_PIN_4, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 24
{GPIOG, GPIO_PIN_10, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 25
{GPIOG, GPIO_PIN_11, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 26
{GPIOE, GPIO_PIN_6, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 27
{GPIOA, GPIO_PIN_4, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 28
{GPIOC, GPIO_PIN_0, &hADC3, ADC_CHANNEL_10 , NULL , NO_PWM , NO_EXTI }, // 29
{GPIOC, GPIO_PIN_3, &hADC3, ADC_CHANNEL_13 , NULL , NO_PWM , NO_EXTI }, // 30
{GPIOF, GPIO_PIN_14, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 31
{GPIOF, GPIO_PIN_15, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 32
{GPIOG, GPIO_PIN_14, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 33
{GPIOG, GPIO_PIN_3, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 34
{GPIOC, GPIO_PIN_12, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 35
{GPIOG, GPIO_PIN_9, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 36
{GPIOA, GPIO_PIN_5, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 37
{GPIOA, GPIO_PIN_6, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 38
{GPIOB, GPIO_PIN_5, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 39

{GPIOB, GPIO_PIN_0, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 40
{GPIOC, GPIO_PIN_8, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 41
{GPIOA, GPIO_PIN_7, &hADC1, ADC_CHANNEL_7 , NULL , NO_PWM , 5 }, // 42
{GPIOC, GPIO_PIN_5, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 43
{GPIOB, GPIO_PIN_1, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 44
{GPIOC, GPIO_PIN_4, &hADC1, ADC_CHANNEL_14 , NULL , NO_PWM , 6 }, // 45
{GPIOD, GPIO_PIN_10, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 46
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 47
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 48
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 49

{GPIOB, GPIO_PIN_10, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 50
{GPIOB, GPIO_PIN_11, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 51
{GPIOC, GPIO_PIN_13, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 52
{GPIOD, GPIO_PIN_2, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 53
{GPIOE, GPIO_PIN_3, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 54
{GPIOG, GPIO_PIN_2, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 55
{GPIOE, GPIO_PIN_10, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 56
{GPIOE, GPIO_PIN_11, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 57
{GPIOE, GPIO_PIN_12, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 58
{GPIOE, GPIO_PIN_13, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 59
{GPIOE, GPIO_PIN_14, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 60
{GPIOE, GPIO_PIN_15, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 61
{GPIOF, GPIO_PIN_0, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 62
{GPIOF, GPIO_PIN_1, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 63
{GPIOF, GPIO_PIN_2, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 64
{GPIOD, GPIO_PIN_8, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 65
{GPIOF, GPIO_PIN_4, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 66
{GPIOD, GPIO_PIN_9, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 67
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 68
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 69

{GPIOF, GPIO_PIN_12, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 70
{GPIOF, GPIO_PIN_11, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 71
{GPIOF, GPIO_PIN_5, &hADC3, ADC_CHANNEL_15 , NULL , NO_PWM , 7 }, // 72
{GPIOE, GPIO_PIN_9, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 73
{GPIOE, GPIO_PIN_8, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 74
{GPIOF, GPIO_PIN_3, &hADC3, ADC_CHANNEL_9 , NULL , NO_PWM , 8 }, // 75
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 76
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 77
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 78
{GPIOF, GPIO_PIN_7, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 79

{GPIOD, GPIO_PIN_6, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 80
{GPIOD, GPIO_PIN_5, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 81
{GPIOE, GPIO_PIN_0, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 82
{GPIOE, GPIO_PIN_1, NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }, // 83

{NULL , 0 , NULL , NO_ADC , NULL , NO_PWM , NO_EXTI }
};


```

7. Examples

7. 1. LED

It is a built-in LED test on the OpenCR board.

[OpenCR1.0](#)

7. 1. 1. Code

There are 5 LEDs available in OpenCR, USER1 ~ 4, and the LED connected to base 13 of Arduino.

USER1 ~ 4 arduino pin numbers are defined as follows. When the corresponding pin is output as High / Low, the LED turns on / off.

```
#define BDPIN_LED_USER_1      22
#define BDPIN_LED_USER_2      23
#define BDPIN_LED_USER_3      24
#define BDPIN_LED_USER_4      25
```

[Back to Top ▲](#)

It is a code that sequentially turns on and off all the LEDs.

```
int led_pin = 13;
int led_pin_user[4] = { BDPIN_LED_USER_1, BDPIN_LED_USER_2, BDPIN_LED_USER_3, BDPIN_LED_USER_4 };

void setup() {
    // Set up the built-in LED pin as an output:
    pinMode(led_pin, OUTPUT);
    pinMode(led_pin_user[0], OUTPUT);
    pinMode(led_pin_user[1], OUTPUT);
    pinMode(led_pin_user[2], OUTPUT);
    pinMode(led_pin_user[3], OUTPUT);
}

void loop() {
    int i;

    digitalWrite(led_pin, HIGH); // set to as HIGH LED is turn-off
    delay(100); // Wait for 0.1 second
    digitalWrite(led_pin, LOW); // set to as LOW LED is turn-on
    delay(100); // Wait for 0.1 second

    for( i=0; i<4; i++ )
    {
        digitalWrite(led_pin_user[i], HIGH);
        delay(100);
    }
    for( i=0; i<4; i++ )
    {
        digitalWrite(led_pin_user[i], LOW);
        delay(100);
    }
}
```

7. 1. 2. Result

OpenCR Arduino Test - LED



7. 2. Button

It is a built-in BUTTON test on the OpenCR board.

7. 2. 1. Code

[OpenCR1.0](#)

There are Push switches SW1 ~ 2 and Dip switches 1 ~ 2 in OpenCR. The pin number is defined as below, so you can see the status of the current button when you input the data of that pin.

```
#define BDPIN_DIP_SW_1      26
#define BDPIN_DIP_SW_2      27
#define BDPIN_PUSH_SW_1      34
#define BDPIN_PUSH_SW_2      35
```

It is a code that outputs the button input status in serial.

```
void setup(){
    Serial.begin(115200);

    pinMode(BDPIN_DIP_SW_1, INPUT);
    pinMode(BDPIN_DIP_SW_2, INPUT);
    pinMode(BDPIN_PUSH_SW_1, INPUT);
    pinMode(BDPIN_PUSH_SW_2, INPUT);

}

void loop(){
    int dip_state;
    int push_state;

    dip_state = digitalRead(BDPIN_DIP_SW_1)<<0;
    dip_state |= digitalRead(BDPIN_DIP_SW_2)<<1;

    push_state = digitalRead(BDPIN_PUSH_SW_1)<<0;
    push_state |= digitalRead(BDPIN_PUSH_SW_2)<<1;

    Serial.print("dip_state = ");
    Serial.print(dip_state, BIN);

    Serial.print("\tpush_state = ");
    Serial.println(push_state, BIN);

    delay(100);
}
```

7. 2. 2. Result

OpenCR Arduino Test - BUTTON



7. 3. Buzzer

It is a BUZZER related test built in the OpenCR board and uses the Tone function of Arduino.

7. 3. 1. Code

OpenCR has a built-in BUZZER, and the sound varies depending on the frequency. The built-in BUZZER is also mapped to the arduino pin number, and the arduino pin number is as follows. Arduino's Tone function is ported, so you can use BUZZER by using this function.

```
#define BDPIN_BUZZER 31
```

OpenCR1.0

It outputs the melody according to the scale defined in the pitches.h header. The following code is a change from OpenCR's BUZZER to only the PIN number in the example provided in the Arduino IDE.

```
#include "pitches.h"

// notes in the melody:
int melody[] = {
    NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
    4, 8, 8, 4, 4, 4, 4
};

void setup() {
    // iterate over the notes of the melody:
    for (int thisNote = 0; thisNote < 8; thisNote++) {

        // to calculate the note duration, take one second
        // divided by the note type.
        // e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
        int noteDuration = 1000 / noteDurations[thisNote];
        tone(BDPIN_BUZZER, melody[thisNote], noteDuration);

        // to distinguish the notes, set a minimum time between them.
        // the note's duration + 30% seems to work well:
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        // stop the tone playing:
        noTone(BDPIN_BUZZER);
    }
}
```

[Back to Top ▲](#)

7. 3. 2. Result

OpenCR Arduino Test - BUZZER



7. 4. PWM

This is the PWM output test from the Arundin pin of the OpenCR board.

7. 4. 1. Code

OpenCR has the same pin configuration as Arduino Uno. The PWM output is also mapped to the same port. Therefore, analogueWrite is used to output the PWM duty ratio to the corresponding ports. The resolution is 8 bits, from 0 to 255, and the frequency is 50 KHz.

OpenCR1.0[Back to Top ▲](#)

This is an example of PWM output on all six pins.

```

int pwm_pins[6] = { 3, 5, 6, 9, 10, 11 };

void setup() {
    // put your setup code here, to run once:
}

void loop() {
    // put your main code here, to run repeatedly:
    int i;
    static uint8_t pwm_out = 0;

    for( i=0; i<6; i++ )
    {
        analogWrite(pwm_pins[i], pwm_out++);
    }
    delay(100);
}

```

7. 4. 2. Result

OpenCR Test - PWM



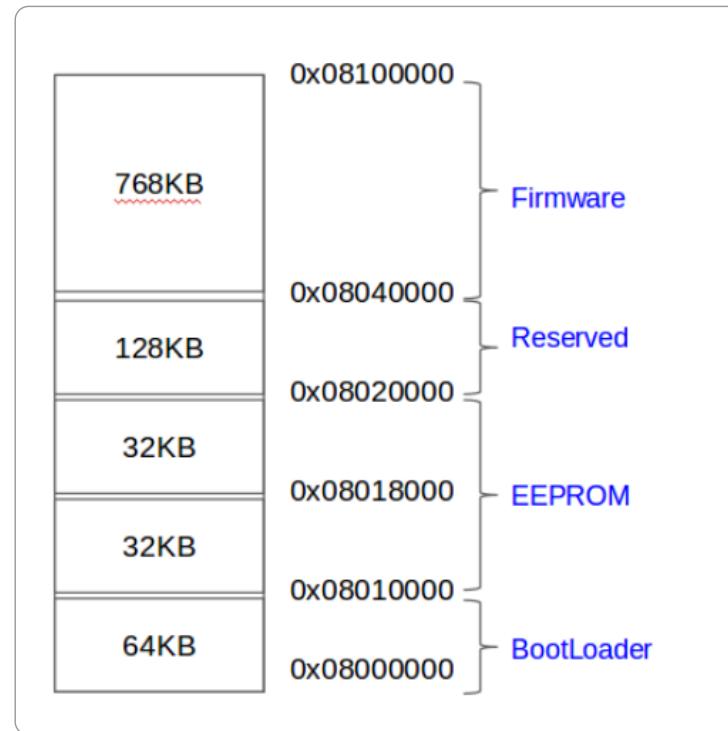
[OpenCR1.0](#)

[Back to Top ▲](#)

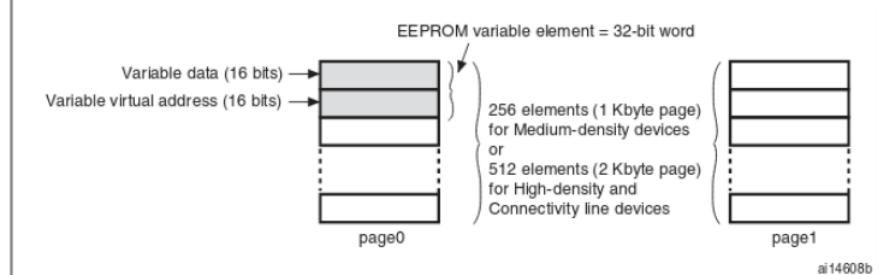
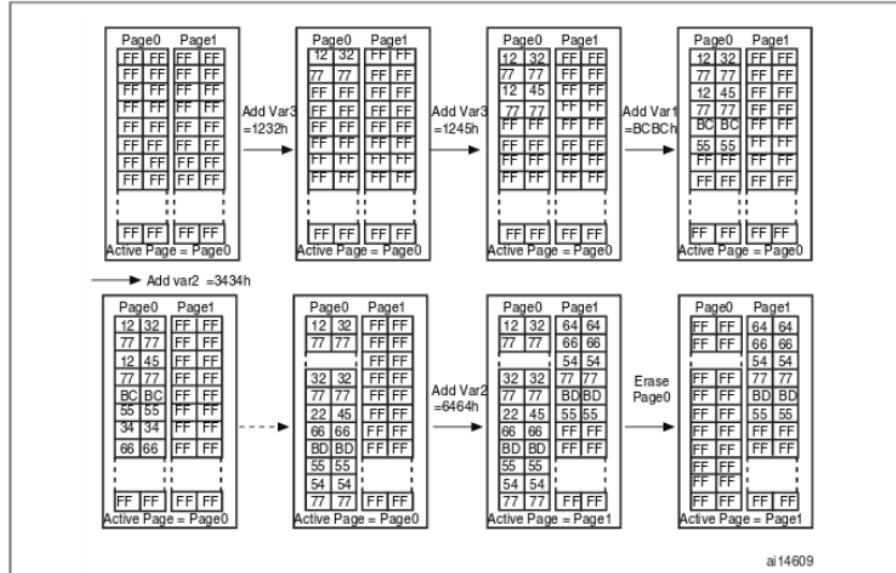
7. 5. EEPROM

It is the EEPROM library test of OpenCR board.

OpenCR does not have EEPROM memory, so it emulates a part of flash memory built in STM32F746 into EEPROM. The method of emulation was provided by ST as an example. The area used as EEPROM is 0x08010000 ~ 0x08020000 as shown below. Two sectors are used.



32 bits are used to store one data, the lower 16 bits are the data to be stored, and the upper 16 bits indicate the address of the corresponding data. When storing data, it is always stored in the new location. When you use one page while saving the data, only the latest values from the saved page are copied to the new page and the existing page is deleted. As a result, the number of flash memory erasures is reduced, thereby increasing the write-through life.

Figure 2. EEPROM variable format**Figure 3. Data update flow**

To use the EEPROM library, a header must be added, and the maximum size of the current EEPROM is 1 KBytes. Since the EEPROM library has ported what is supported in Arduino, the basic usage method is the same as that used in other existing Arduino boards. For more information on how to use it, please refer to the Adunion site.

<https://www.arduino.cc/en/Reference/EEPROM>

7.5.1. Code

```
#include <EEPROM.h>

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
}

void loop() {
    // put your main code here, to run repeatedly:
    uint32_t tTime;
    static int i = 0;

    if( (millis()-tTime) > 100 )
    {
        Serial.print(EEPROM.read(0));
        Serial.print("\r");
        Serial.print(EEPROM.read(1));
        Serial.print("\r");
        Serial.print(EEPROM.read(2));
        Serial.println("\r");

        tTime = millis();
    }
}
```

OpenCR1.0

```

if (Serial.available())
{
    uint8_t inByte = Serial.read();

    if( inByte == '1' )
    {
        EEPROM.write(0, i+1);
        EEPROM.write(1, i+2);
        EEPROM.write(2, i+3);

        i++;
    }
}
}

```

7. 5. 2. Result[Back to Top ▲](#)**OpenCR Arduino Test - EEPROM****7. 6. OP3**

OpenCR is used for power and sensor control in OP3, a humanoid robot. If the OpenCR firmware for OP3 has been changed, follow the procedure below to update it.

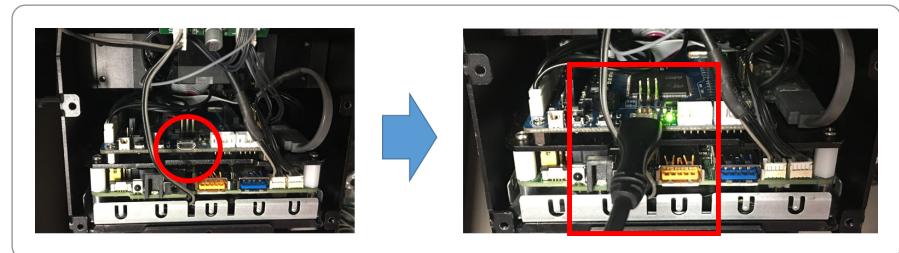
7. 6. 1. Preparations

OpenCR develops and downloads firmware through the Arduino IDE. Therefore, you must install the Arduino IDE in advance and install the OpenCR board package. Install through the following link document.

- [Install Arduino IDE and OpenCR](#)

7. 6. 2. Download OP3 firmware

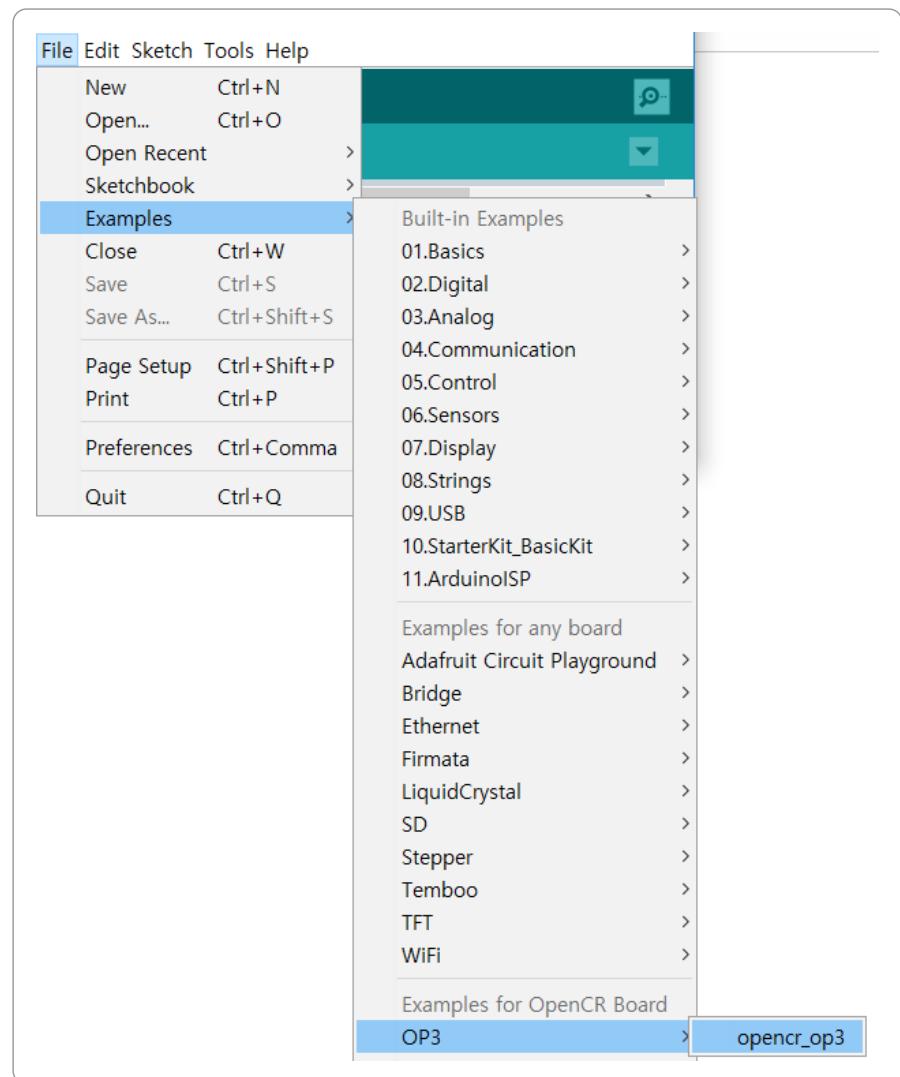
1. To update OpenCR's firmware, open the front cover of OP3 and connect USB to PC as shown below.



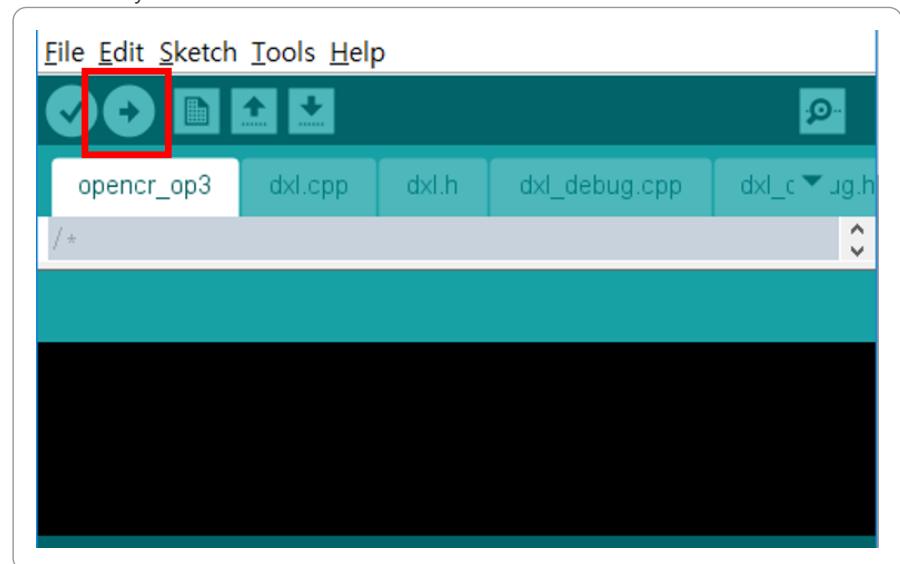
2. After connecting USB, select Tools-> Board-> OpenCR Board in Arduino IDE.

3. Change Tools-> Port to the port to which the board is connected.

4. In the Arduino IDE Examples, select the firmware for OP3.

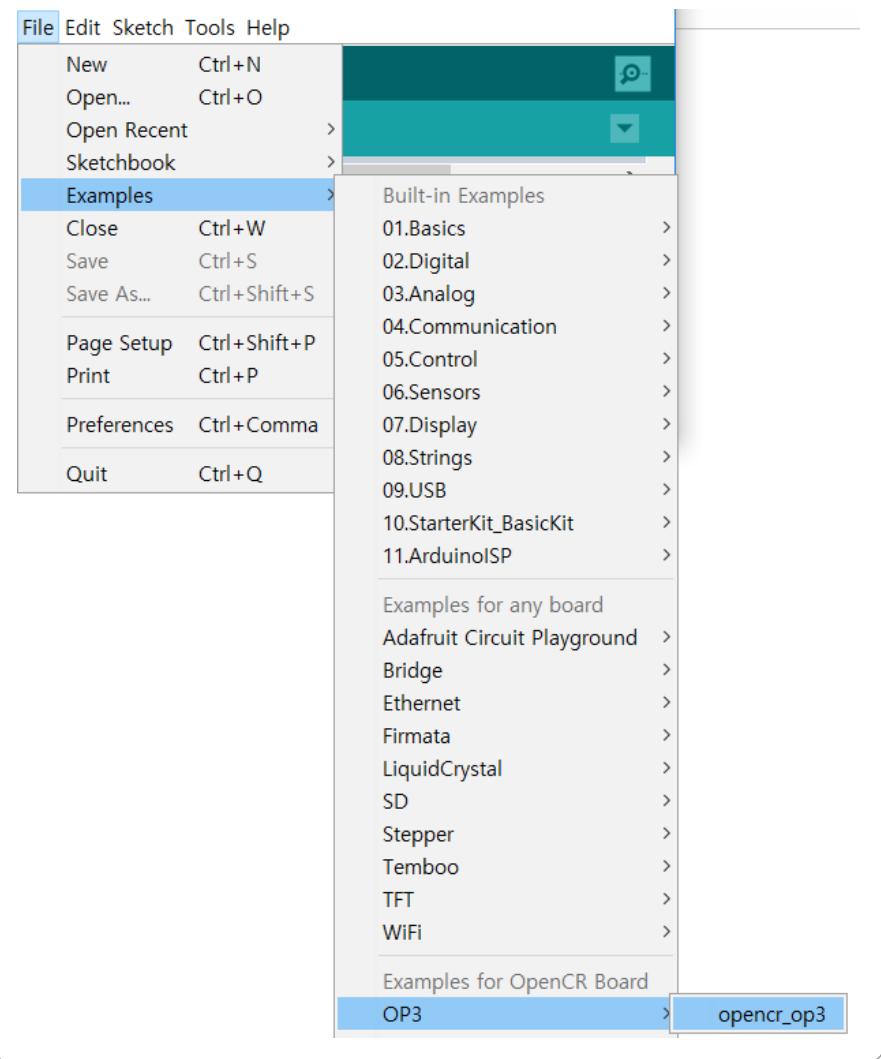
OpenCR1.0[Back to Top ▲](#)

5. Click on the icon in the Arduino IDE that displays the red circle in the following figure to build and download the firmware. When the download is completed, the firmware is automatically executed.

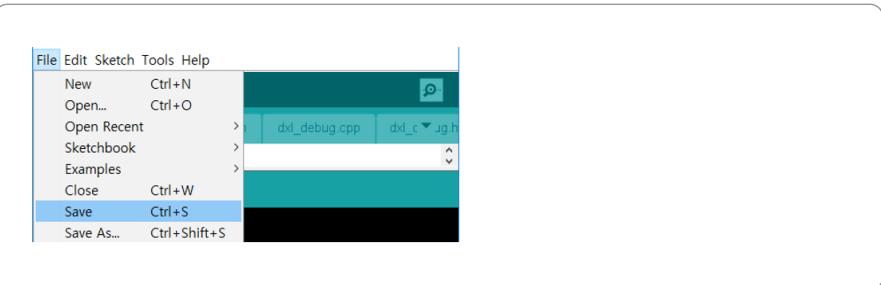
**7. 6. 3. Editing OP3 Firmware**

The firmware that is provided as a basic example of OpenCR is read-only. If you want to edit it, you have to save it to a new folder and work on it.

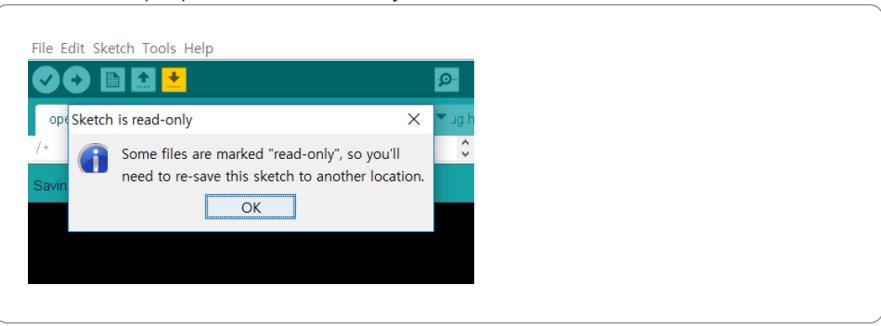
1. Open the OP3 example.

OpenCR1.0[Back to Top ▲](#)

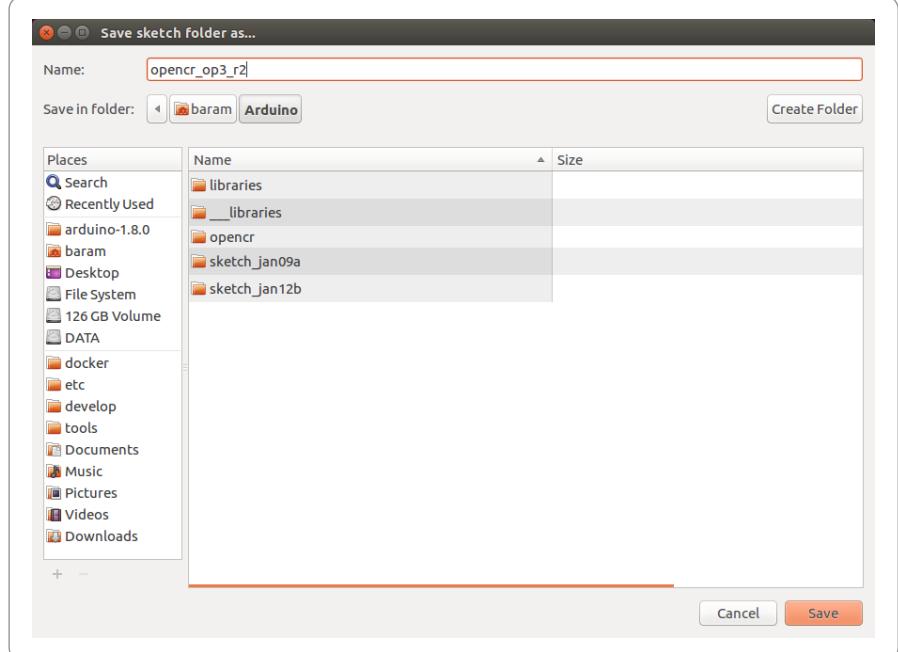
2. Select File-> Save.



3. Since the example provided is Read-Only, select OK to save it as a new file.



4. Save it to a new folder and edit it. Once editing is complete, repeat the process of building and downloading the firmware.

OpenCR1.0[Back to Top ▲](#)

7. 7. Sensors

7. 7. 1. Ambient Light Sensor

It is ambient light sensor test on the OpenCR board.

- Pinouts
 - Green : Signal
 - Red : Vcc
 - Black : Gnd
- Specification
 - [ambient light sensor specification](#)
 - Supply Voltage : 3.3V to 5V
 - Illumination range : 1 Lux to 6000 Lux
 - Interface : Analog

7. 7. 1. 1. Code

LED turns off/on sequentially depending on the light received by the sensor.

LED turns off in bright place. If it is dark place, the LED turns on.

This sensor is an analog sensor, connect it to port A0.

```
#define BDPIN_LED_USER_1    23
#define BDPIN_LED_USER_2    24
#define BDPIN_LED_USER_3    25
```

It is a code that turns on/off the LED depending on the brightness of light changes.

```
void setup()
{
  Serial.begin(9600);
  pinMode(BDPIN_LED_USER_1, OUTPUT);
  pinMode(BDPIN_LED_USER_2, OUTPUT);
  pinMode(BDPIN_LED_USER_3, OUTPUT);
}

void loop()
{
  if(analogRead(0)<200)
  {
    digitalWrite(BDPIN_LED_USER_1, LOW);
    digitalWrite(BDPIN_LED_USER_2, LOW);
    digitalWrite(BDPIN_LED_USER_3, LOW);
  }
  else if(analogRead(0)>200 && analogRead(0)<300)
```

```

    {
        digitalWrite(BDPIN_LED_USER_1, HIGH);
        digitalWrite(BDPIN_LED_USER_2, LOW);
        digitalWrite(BDPIN_LED_USER_3, LOW);
    }
    else if(analogRead(0)>300 && analogRead(0)<400)
    {
        digitalWrite(BDPIN_LED_USER_1, HIGH);
        digitalWrite(BDPIN_LED_USER_2, HIGH);
        digitalWrite(BDPIN_LED_USER_3, LOW);
    }
    else if(analogRead(0)>400 && analogRead(0)<500)
    {
        digitalWrite(BDPIN_LED_USER_1, HIGH);
        digitalWrite(BDPIN_LED_USER_2, HIGH);
        digitalWrite(BDPIN_LED_USER_3, HIGH);
    }
    Serial.println(analogRead(0), DEC);
    delay(100);
}

```

OpenCR1.0

[Back to Top ▲](#)

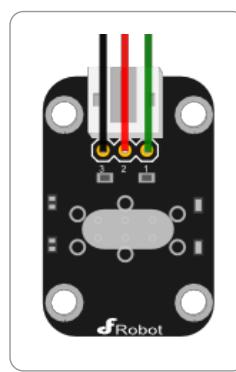
7.7.1.2. Result

OpenCR 01 Ambient Light Sensor Example



7.7.2. Tilt Sensor

It is tilt sensor test on the OpenCR.



- Pinouts
 - Green : Signal
 - Red : Vcc
 - Black : Gnd
- Specification
 - [Tilt Sensor Specification](#)
 - Supply Voltage : 3.3V to 5V
 - Interface : Digital

7.7.2.1. Code

tilt sensor and led are connected to OpenCR. so that red/blue led is on/off when tilted and red/blue led is off/on when not tilted.

Connect the Tilt Sensor, Led_blue, and Led_red signal pins to D0, D1, and D2.

OpenCR1.0

```
#define tilt      0
#define led_blue 1
#define led_red  2

void setup()
{
    pinMode(tilt, INPUT);
    pinMode(led_blue, OUTPUT);
    pinMode(led_red, OUTPUT);
}

void loop()
{
    if(digitalRead(tilt) == HIGH)
    {
        digitalWrite(led_blue, HIGH);
        digitalWrite(led_red, LOW);
    }
    else
    {
        digitalWrite(led_blue, LOW);
        digitalWrite(led_red, HIGH);
    }
}
```

[Back to Top ▲](#)

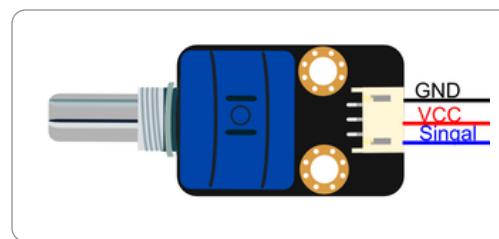
7.7.2.2. Result

OpenCR 02 Tilt Sensor Example



7.7.3. Rotation Sensor

It is rotation sensor test on the OpenCR board.



- Specification
 - [Rotation Sensor Specification](#)
 - Rotation Angle : 3600 degrees
 - Supply Voltage : 3.3V to 5V
 - Interface : Analog

7. 7. 3. 1. Code

Rotation sensor is an analog sensor, the output value depending on the degree of rotation.

The LED turned on/off depending on the degree of rotation.

The signal pin is connected to A0 of OpenCR.

[OpenCR1.0](#)

```
#define BDPIN_LED_USER_1      22
#define BDPIN_LED_USER_2      23
#define BDPIN_LED_USER_3      24
#define BDPIN_LED_USER_4      25

const int analogInPin = A0;
int sensorValue = 0;

void setup()
{
    Serial.begin(9600);
    pinMode(BDPIN_LED_USER_4, OUTPUT);
    pinMode(BDPIN_LED_USER_3, OUTPUT);
    pinMode(BDPIN_LED_USER_2, OUTPUT);
    pinMode(BDPIN_LED_USER_1, OUTPUT);
}

void loop()
{
    sensorValue = analogRead(analogInPin);
    Serial.print(" sensorValue : ");
    Serial.println(sensorValue);

    if(sensorValue>0 && sensorValue<50)
    {
        digitalWrite(BDPIN_LED_USER_4, LOW);
        digitalWrite(BDPIN_LED_USER_3, HIGH);
        digitalWrite(BDPIN_LED_USER_2, HIGH);
        digitalWrite(BDPIN_LED_USER_1, HIGH);
    }
    if(sensorValue>50 && sensorValue<100)
    {
        digitalWrite(BDPIN_LED_USER_4, LOW);
        digitalWrite(BDPIN_LED_USER_3, LOW);
        digitalWrite(BDPIN_LED_USER_2, HIGH);
        digitalWrite(BDPIN_LED_USER_1, HIGH);
    }
    if(sensorValue>100 && sensorValue<150)
    {
        digitalWrite(BDPIN_LED_USER_4, LOW);
        digitalWrite(BDPIN_LED_USER_3, LOW);
        digitalWrite(BDPIN_LED_USER_2, LOW);
        digitalWrite(BDPIN_LED_USER_1, HIGH);
    }
    if(sensorValue>200 && sensorValue<250)
    {
        digitalWrite(BDPIN_LED_USER_4, LOW);
        digitalWrite(BDPIN_LED_USER_3, LOW);
        digitalWrite(BDPIN_LED_USER_2, LOW);
        digitalWrite(BDPIN_LED_USER_1, LOW);
    }
    delay(100);
}
```

7. 7. 3. 2. Result

OpenCR 03 Rotation Sensor Example

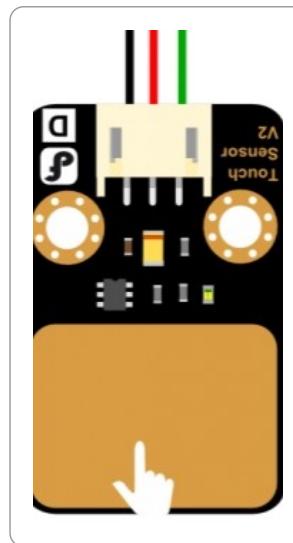


[OpenCR1.0](#)

[Back to Top ▲](#)

7. 7. 4. Capacitive Touch Sensor

It is capacitive touch sensor test on the OpenCR board.



- Pinouts
 - Green : Signal
 - Red : Vcc
 - Black : Gnd
- Specification
 - [Capacitive Touch Sensor Specification](#)
 - Supply Voltage : 3.3V to 5V
 - Interface : Digital

7. 7. 4. 1. Code

When you put your hand on the sensor, the led turn on/off sequentially and then the LED turns off when you take your hand.

Tilt sensor is a digital sensor, signal of sensor is connected to D0 of OpenCR.

```
#define SensorINPUT          0
#define BDPIN_LED_USER_1      22
#define BDPIN_LED_USER_2      23
#define BDPIN_LED_USER_3      24
#define BDPIN_LED_USER_4      25
```

```
int LED[] = {BDPIN_LED_USER_1, BDPIN_LED_USER_2, BDPIN_LED_USER_3, BDPIN_LED_USER_4};
int i = 0;

void setup()
```

OpenCR1.0

```

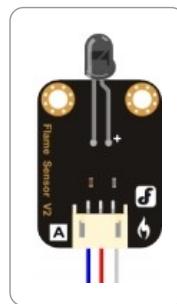
{
    pinMode(SensorINPUT, INPUT);
    pinMode(BDPIN_LED_USER_1, OUTPUT);
    pinMode(BDPIN_LED_USER_2, OUTPUT);
    pinMode(BDPIN_LED_USER_3, OUTPUT);
    pinMode(BDPIN_LED_USER_4, OUTPUT);
}

void loop()
{
    if (digitalRead(SensorINPUT) == HIGH )
    {
        for(i=0; i<4; i++)
        {
            digitalWrite(LED[i], LOW);
            delay(100);
            digitalWrite(LED[i], HIGH);
        }
    }
    if (digitalRead(SensorINPUT) == LOW )
    {
        for(i=0; i<4; i++)
        {
            digitalWrite(LED[i], HIGH);
        }
    }
}

```

[Back to Top ▲](#)**7. 7. 4. 2. Result****OpenCR 04 Capacitive Touch Sensor Example****7. 7. 5. Flame Sensor**

It is flame sensor test on the OpenCR board.



- Pinouts
 - Blue : Signal
 - Red : Vcc
 - Black : Gnd
- Specification
 - [Flame Sensor Specification](#)
 - Detection range : 20cm(4.8V) ~ 100cm(1V)

- o Supply Voltage : 3.3V to 5V
- o Interface : Analog

7. 7. 5. 1. Code

[OpenCR1.0](#)

If the flame is detected, turns on the led.
 Fire near the sensor, it outputs a high value close to 1024.
 If the output exceeds 800, led will turn on.
 Signal is connected to A0 of Arduino.

```
#define BDPIN_LED_USER_1 22
#define flame 0

void setup()
{
  Serial.begin(9600);
  pinMode(BDPIN_LED_USER_1, OUTPUT);
}

void loop()
{
  int val;
  val=analogRead(flame);

  if(val>800)
  {
    digitalWrite(BDPIN_LED_USER_1, LOW);
  }
  else
  {
    digitalWrite(BDPIN_LED_USER_1, HIGH);
  }

  Serial.println(val,DEC);
  delay(100);
}
```

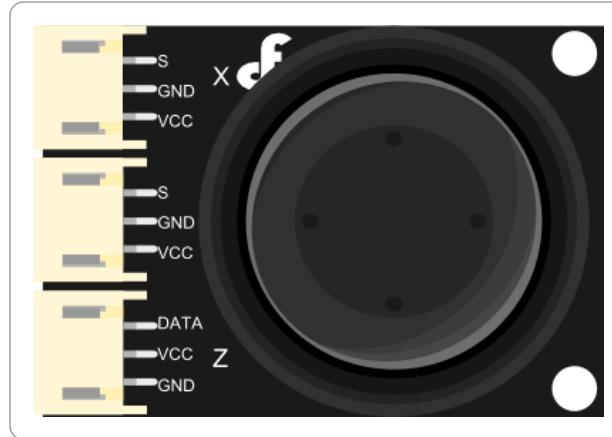
7. 7. 5. 2. Result

OpenCR 06 Flame Sensor Example



7. 7. 6. Joystick Sensor

It is joystick test on the OpenCR board.

OpenCR1.0

- Specification
 - [Joystick Sensor Specification](#)
 - Interface : Analog

[Back to Top ▲](#)**7.7.6.1. Code**

Joystick is to get the output value according to the input.

We will look at the X Y Z values that change depending on how we move.

Signal of x,y and z is connected to A0, A1, A2 of Arduino.

```
#define X A0
#define Y A1
#define Z A2

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int x,y,z;
    x=analogRead(X);
    y=analogRead(Y);
    z=analogRead(Z);

    Serial.print(" X = ");
    Serial.print(x,DEC);

    Serial.print(" Y = ");
    Serial.print(y,DEC);

    Serial.print(" Z = ");
    Serial.println(z,DEC);
    delay(100);
}
```

7.7.6.2. Result

OpenCR 05 Joystick Example



[OpenCR1.0](#)

[Back to Top ▲](#)

7. 8. Dynamixel Workbench

7. 8. 1. Find Dynamixel

When you get a Dynamixel first, you need to know what ID and Baud rate is. This example find out ID and Baud rate of connected Dynamixels.

begin function set PortHandler and PacketHandler. **scan** function ping all Dynamixels. After get Dynamixels, you can check ID and Baudrate of its.

```
/****************************************************************************
 * Copyright 2016 ROBOTIS CO., LTD.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *****/
/* Authors: Taehun Lim (Darby) */

#include <DynamixelWorkbench.h>

#define DXL_BUS_SERIAL1 "1"           //Dynamixel on Serial1(USART1) <-OpenCM9.04
#define DXL_BUS_SERIAL2 "2"           //Dynamixel on Serial2(USART2) <-LN101,BT210
#define DXL_BUS_SERIAL3 "3"           //Dynamixel on Serial3(USART3) <-OpenCM 485EXP
#define DXL_BUS_SERIAL4 "/dev/ttyUSB0" //Dynamixel on Serial3(USART3) <-OpenCR

#define BAUDRATE_NUM 3

DynamixelWorkbench dxl_wb;

void setup()
{
    Serial.begin(57600);
    while(!Serial); // Open a Serial Monitor

    uint8_t scanned_id[16] = {0, };
    uint8_t dxl_cnt = 0;
    uint32_t baud[BAUDRATE_NUM] = {9600, 57600, 1000000};
    uint8_t index = 0;
    uint8_t range = 100;

    while (index < BAUDRATE_NUM)
    {
        Serial.println(String(baud[index]) + " bps");

        dxl_wb.begin(DXL_BUS_SERIAL4, baud[index]);
        dxl_wb.scan(&scanned_id[0], &dxl_cnt, range);
```

OpenCR1.0

```

for (int i = 0; i < dxl_cnt; i++)
{
    Serial.println("  id : " + String(scanned_id[i]) + "  Model Name : " + String(dxl_wb.{ 
    }

    index++;
}
Serial.println("End");
}

void loop()
{
}

```


7. 8. 2. Position

This example shows position control using Dynamixel. You need to set parameters of BAUDRATE and ID.

[Back to Top ▲](#)

begin function set an portHandler and packetHandler. **ping** function get an item of connected Dynamixel.

jointMode function make joint(position) mode.

If Dynamixel is set correctly, **goalPosition** function make it move to position.

```

/****************************************************************************
 * Copyright 2016 ROBOTIS CO., LTD.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
****/

/* Authors: Taehun Lim (Darby) */

#include <DynamixelWorkbench.h>

#define DXL_BUS_SERIAL1 "1"           //Dynamixel on Serial1(USART1)  <-OpenCM9.04
#define DXL_BUS_SERIAL2 "2"           //Dynamixel on Serial2(USART2)  <-LN101,BT210
#define DXL_BUS_SERIAL3 "3"           //Dynamixel on Serial3(USART3)  <-OpenCM 485EXP
#define DXL_BUS_SERIAL4 "/dev/ttyUSB0" //Dynamixel on Serial3(USART3)  <-OpenCR

#define BAUDRATE  57600
#define DXL_ID    1

DynamixelWorkbench dxl_wb;

void setup()
{
    Serial.begin(57600);
    while(!Serial); // Open a Serial Monitor

    dxl_wb.begin(DXL_BUS_SERIAL4, BAUDRATE);
    dxl_wb.ping(DXL_ID);

    dxl_wb.jointMode(DXL_ID);
}

void loop()
{
    dxl_wb.goalPosition(DXL_ID, 0);

    delay(2000);

    dxl_wb.goalPosition(DXL_ID, 2000);

    delay(2000);
}

```

7. 8. 3. Speed

This example shows velocity control using Dynamixel. You need to set parameters of BAUDRATE and ID.

begin function set an portHandler and packetHandler. **ping** function get an item of connected Dynamixel.

wheelMode function make wheel(velocity) mode.

If Dynamixel is set correctly, **goalSpeed** function make it turn to position.

```
/****************************************************************************
 * Copyright 2016 ROBOTIS CO., LTD.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *****/
/* Authors: Taehun Lim (Darby) */

#include <DynamixelWorkbench.h>

#define DXL_BUS_SERIAL1 "1"           //Dynamixel on Serial1(USART1)  <-OpenCM9.04
#define DXL_BUS_SERIAL2 "2"           //Dynamixel on Serial2(USART2)  <-LN101,BT210
#define DXL_BUS_SERIAL3 "3"           //Dynamixel on Serial3(USART3)  <-OpenCM 485EXP
#define DXL_BUS_SERIAL4 "/dev/ttyUSB0" //Dynamixel on Serial3(USART3)  <-OpenCR

#define BAUDRATE  57600
#define DXL_ID    1

DynamixelWorkbench dxl_wb;

void setup()
{
    Serial.begin(57600);
    while(!Serial); // Open a Serial Monitor

    dxl_wb.begin(DXL_BUS_SERIAL4, BAUDRATE);
    dxl_wb.ping(DXL_ID);

    dxl_wb.wheelMode(DXL_ID);
}

void loop()
{
    dxl_wb.goalSpeed(DXL_ID, 300);

    delay(3000);

    dxl_wb.goalSpeed(DXL_ID, -300);

    delay(3000);
}
```

7. 9. OpenManipulator

7. 9. 1. OpenManipulator Chain

- [e-Manual](#) [OpenManipulator Chain](#)

7. 9. 2. OpenManipulator SCARA

- [e-Manual](#) [OpenManipulator SCARA](#)

7. 9. 3. OpenManipulator Link

- [e-Manual](#) [OpenManipulator Link](#)

8. Downloads

- [Download BOM](#)
- [Download Schematic](#)
- [Download PCB](#)

[OpenCR1.0](#)

9. References

9. 1. Recovery Mode

- [e-Manual OpenCR Firmware Recovery](#)

[Back to Top ▲](#)

© 2018 ROBOTIS. Powered by Jekyll & Minimal Mistakes.