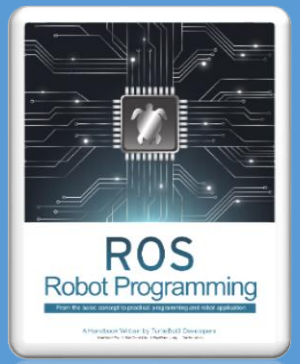
Embedded System

ROBOTIS KAIST



YouTube Subscribe

Textbook
P. 232~277

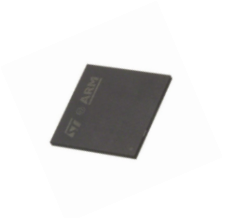
Contents

YouTube Subscribe

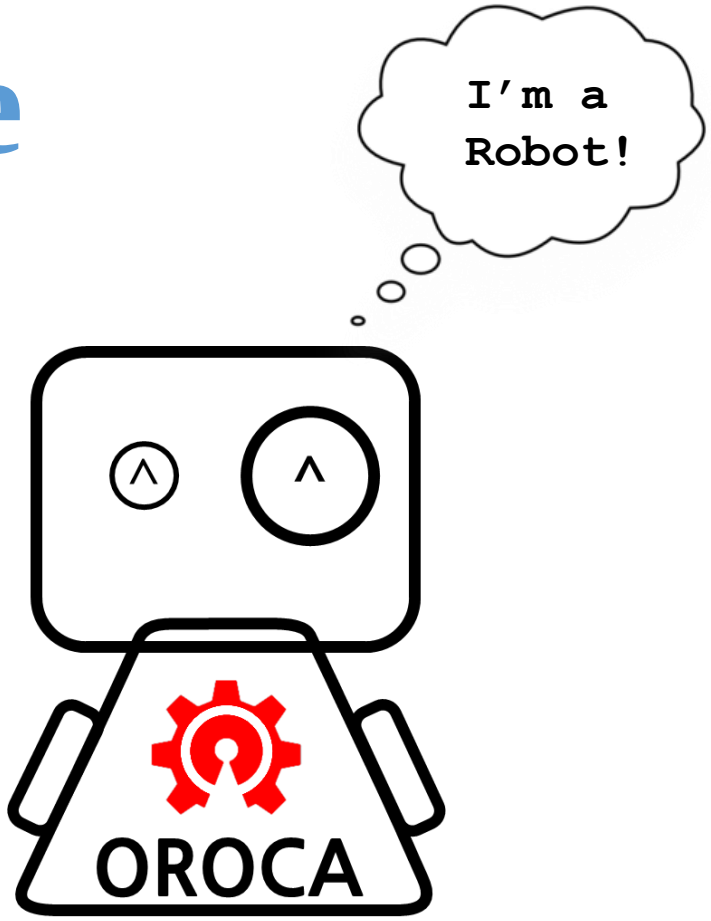
Textbook
P. 232~277

What is the Best Computing Resource for Your Robot?

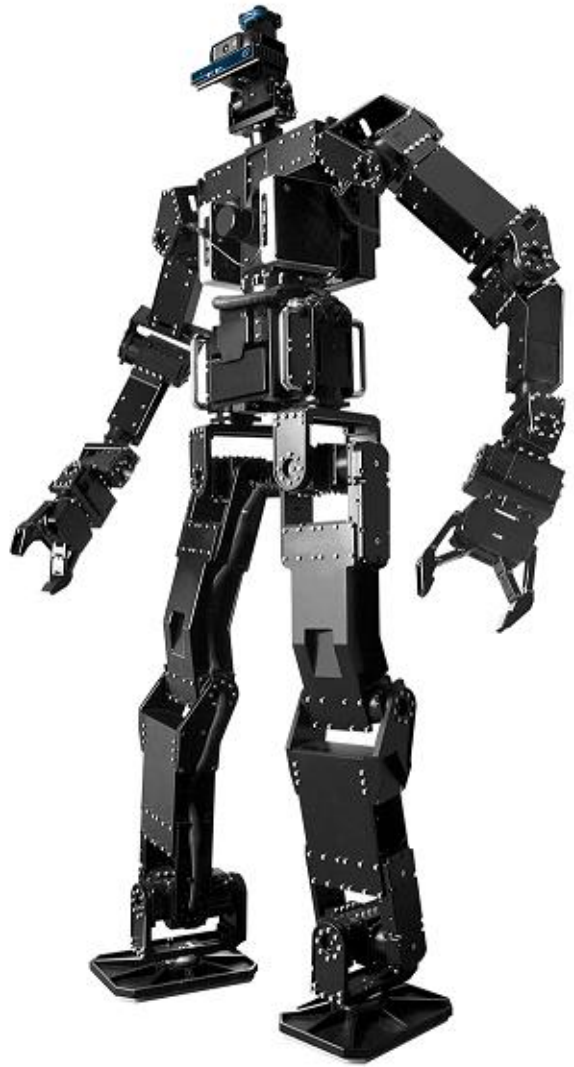
CPU?



MCU?



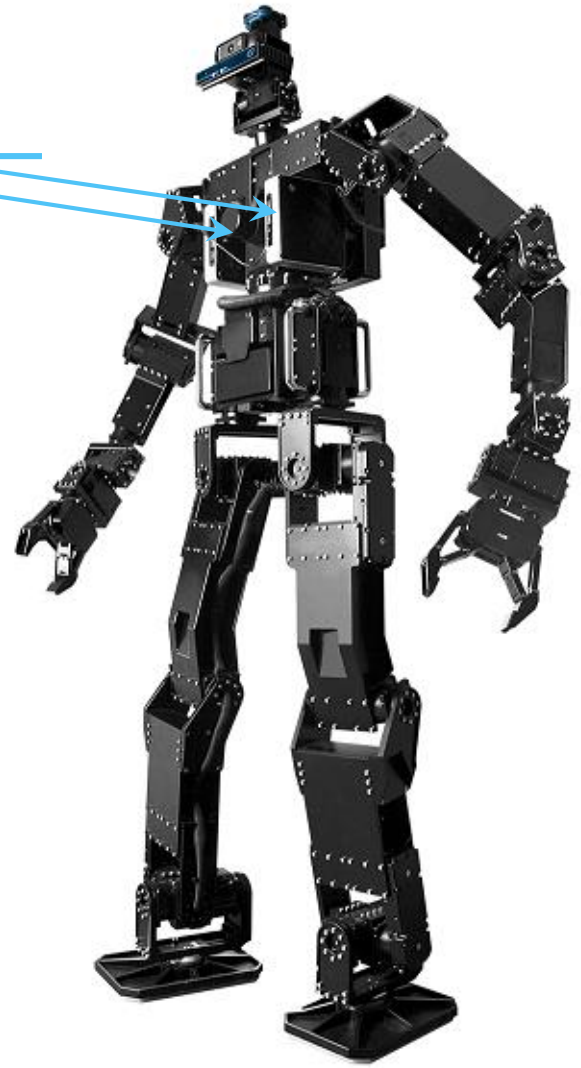
The proportion of embedded systems in robots



THORMANG3

The proportion of embedded systems in robots

Intel NUC
for Sensor and Communication

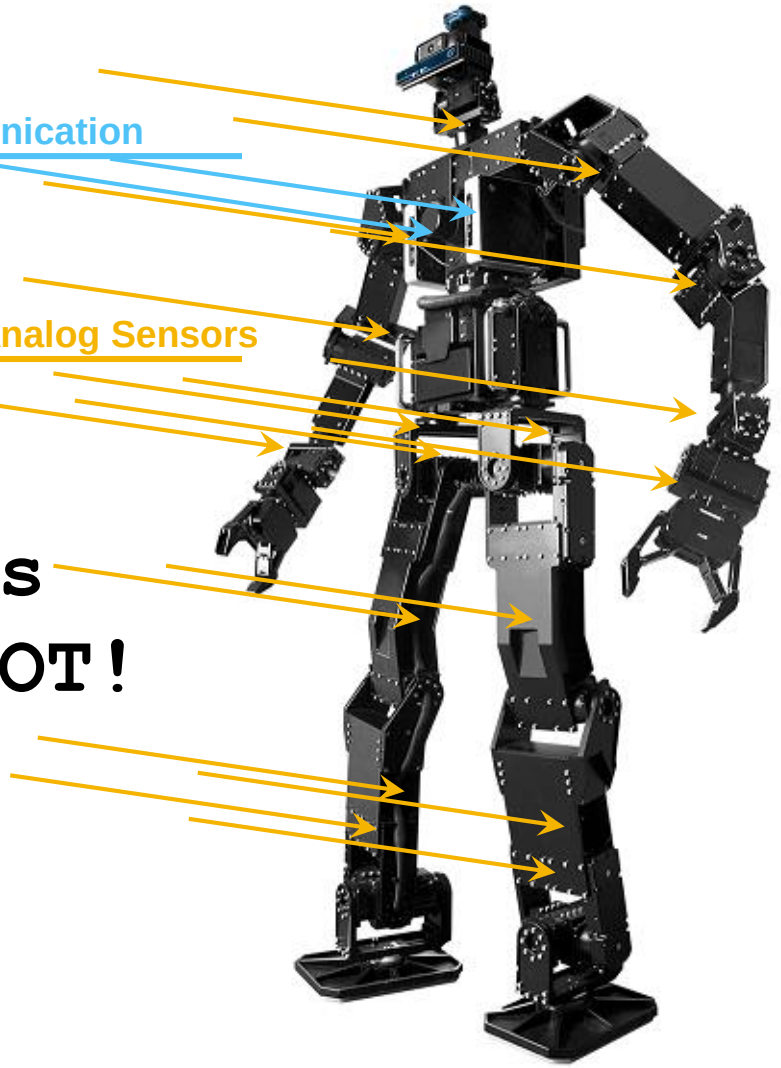


THORMANG3

The proportion of embedded systems in robots

Intel NUC
for Sensor and Communication

ARM Cortex-M
for Motor control and Analog Sensors



Small embedded systems are everywhere in ROBOT!

THORMANG3

Types of computer resources and ROS support




	8/16-bit MCU	32-bit MCU		ARM A-class	x86
		"small" 32-bit MCU	"big" 32-bit MCU		
Example Chip	Atmel AVR	ARM Cortex-M0	ARM Cortex-M7	Samsung Exynos	Intel Core i5
Example System	Arduino Leonardo	Arduino M0 Pro	SAM V71	ODROID	Intel NUC
MIPS	10's	100's	100's	1000's	10000's
RAM	1-32 KB	32 KB	384 KB	a few GB (off-chip)	2-16 GB (SODIMM)
Max power	10's of mW	100's of mW	100's of mW	1000's of mW	10000's of mW
Peripherals	UART, USB FS, ...	USB FS	Ethernet, USB HS	Gigabit Ethernet	USB SS, PCIe

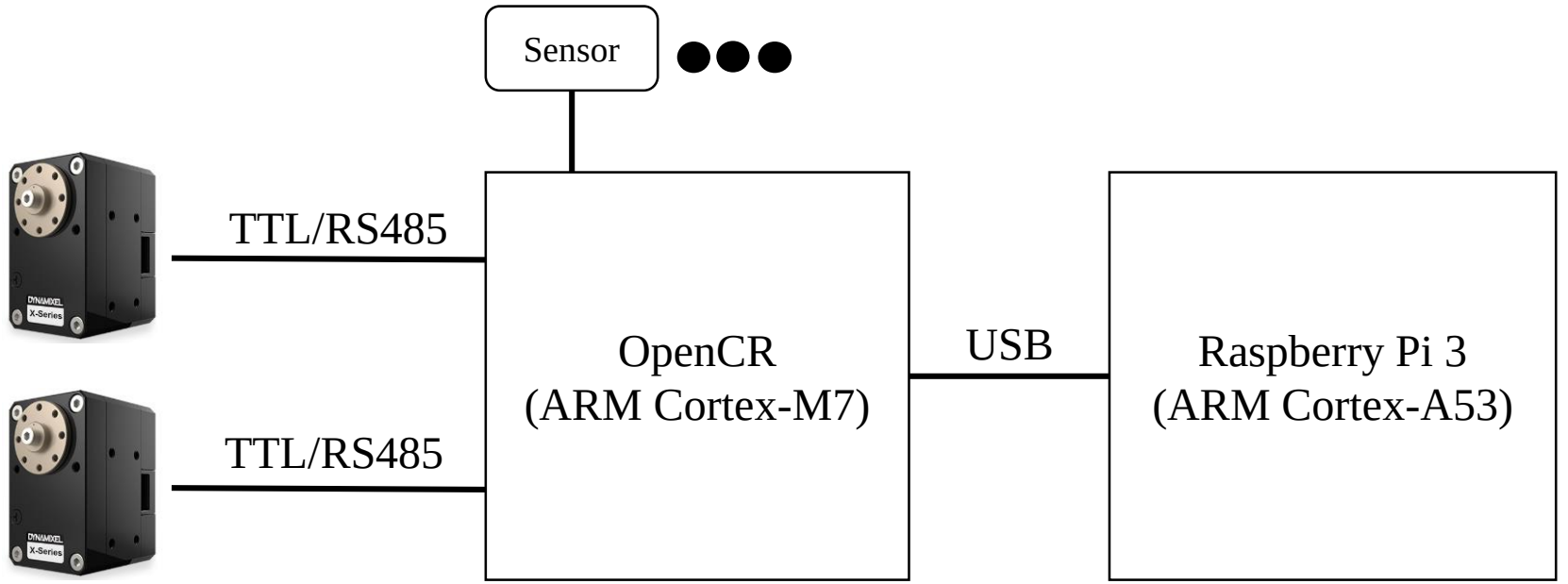
ROS not installable

ROS installable

Morgan Quigley (OSRF) "ROS 2 on “small” embedded systems", ROSCon2015

Embedded systems in ROS

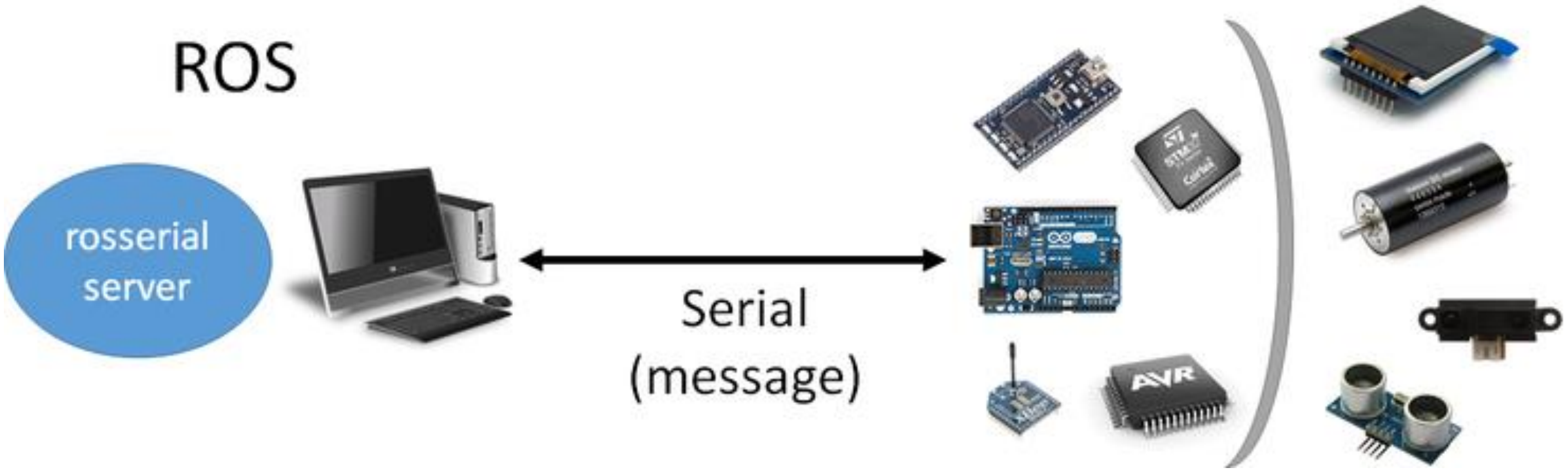
- Unlike PC, ROS can not be installed in embedded system
- For securing real-time factor and hardware control, connection between the embedded system and the ROS installed PC is required.
- ROS provides a package called ‘rosserial’ for this function!



[Configuration of PC and embedded system in TurtleBot3]

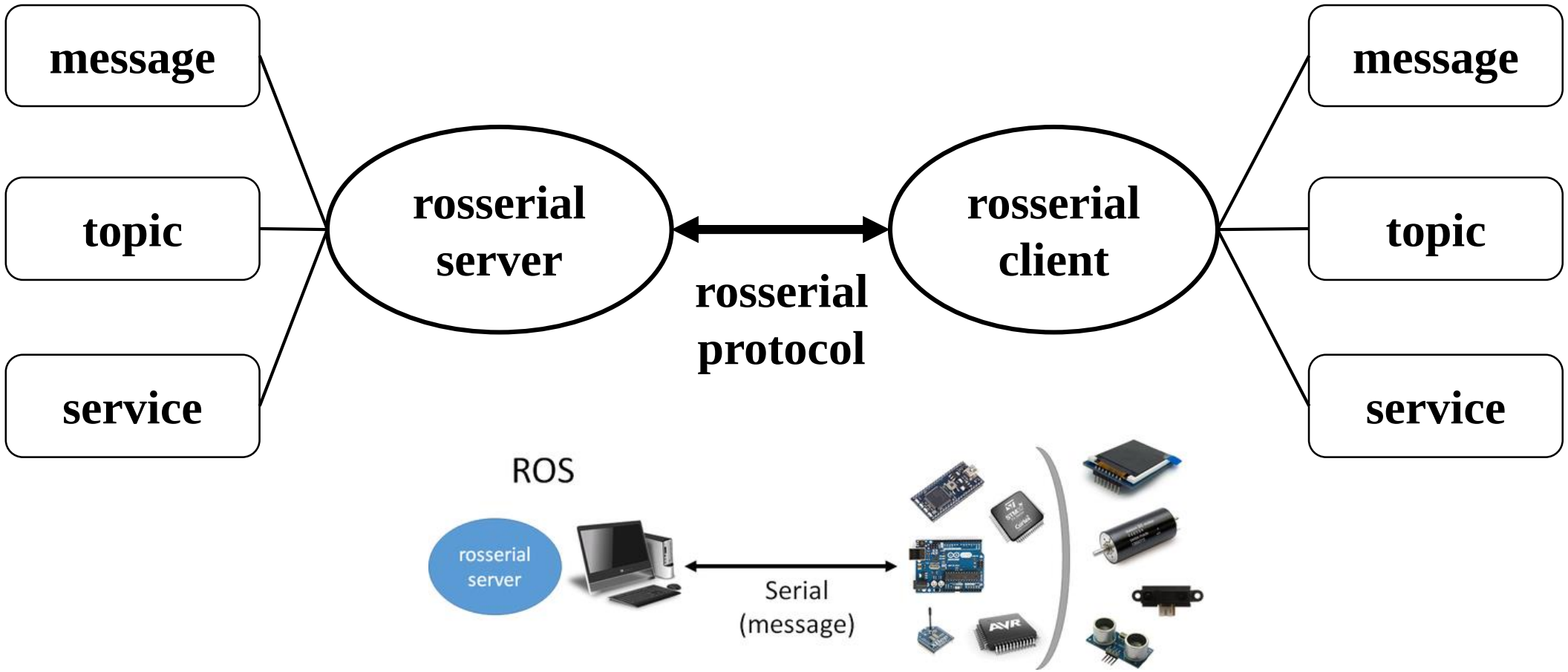
‘rosserial’

- ROS package acting as an intermediary for message communication between PC & Controller
  - Example) Controller → Serial(rosserial protocol) → PC(Retransmission with ROS messages)
  - Example) Controller ← Serial(rosserial protocol) ← PC(Change ROS messages to serial)



http://wiki.ros.org/rosserial

‘rosserial’ server & client



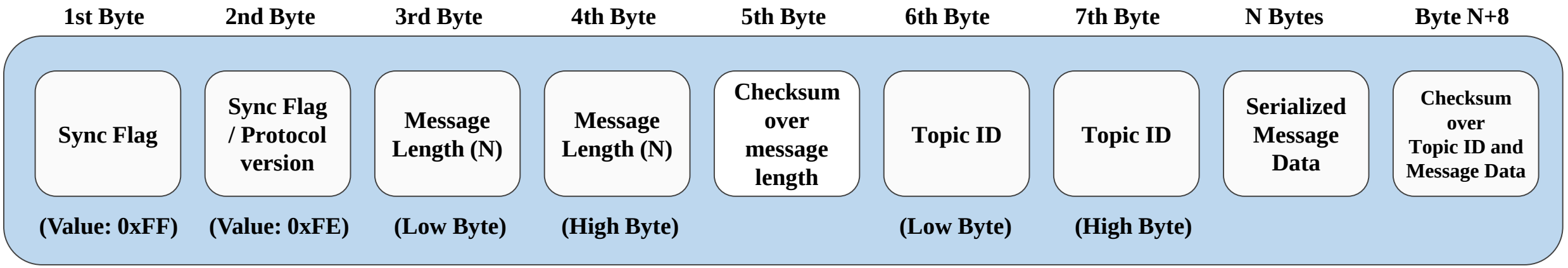
‘rosserial’ server & client

```
$ sudo apt-get install ros-kinetic-rosserial ros-kinetic-rosserial-server ros-kinetic-rosserial-arduino
```

- **rosserial server**
  - **rosserial_python**: Python language based rosserial server, very popular
  - **rosserial_server**: C++ language based rosserial server, Some functions are limited
  - **rosserial_java**: Java language based rosserial server, Used with android SDK

- **rosserial client**
  - **rosserial_arduino**: Support Arduino & Leonardo, OpenCR uses it with few modification
  - **rosserial_embeddedlinux**: Linux Library for Embedded System
  - **rosserial_windows**: Support Windows operating system, Windows application and communication support
  - **rosserial_mbed**: Support ARM’s mbed
  - **rosserial_tivac**: Support TI’s Launchpad

‘rosserial’ Protocol (http://wiki.ros.org/rosserial/Overview/Protocol)

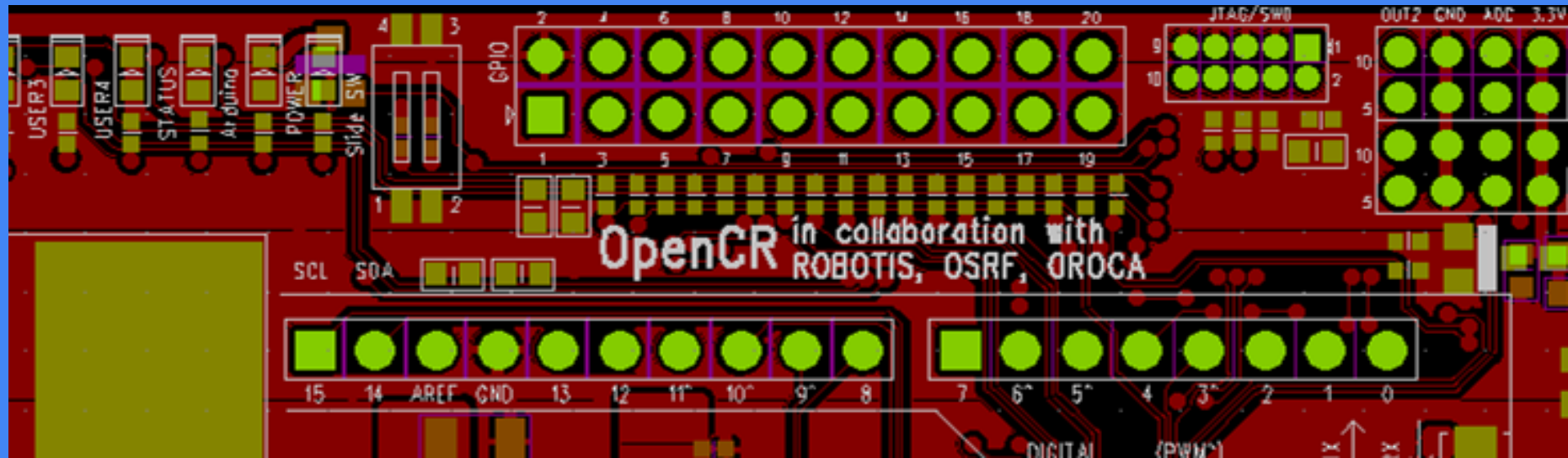


Sync Flag	Header to know the start position of the packet. It is always ‘0xFF’
Sync Flag / Protocol version	A protocol version, ROS Groovy is ‘0xFF’, & ‘ROS Hydro’, ‘Indigo’, ‘Jade’ and ‘Kinetic’ are ‘0xFE’
Message Length (N)	Data length of message, 2 Byte = Low Byte + High Byte, Low Byte are transmitted first, followed by ‘High Byte’
Checksum over message length	Checksum for validating message length headers Checksum = 255 - ((Message Length Low Byte + Message Length High Byte) %256)
Topic ID	It is an ID for identifying the type of message. 2 Byte = Low Byte + High Byte ID_PUBLISHER=0, ID_SUBSCRIBER=1, ID_SERVICE_SERVER=2, ID_SERVICE_CLIENT=4, ID_PARAMETER_REQUEST=6, ID_LOG=7, ID_TIME=10, ID_TX_STOP=11
Serialized Message Data	It is the data to transmit the ROS message in serial form EX) IMU, TF, GPIO
Checksum over Topic ID and Message Data	Topic ID and checksum to validate message data Checksum = 255 - ((Topic ID Low Byte + Topic ID High Byte + data byte values) % 256)

Limitations of ‘rosserial’

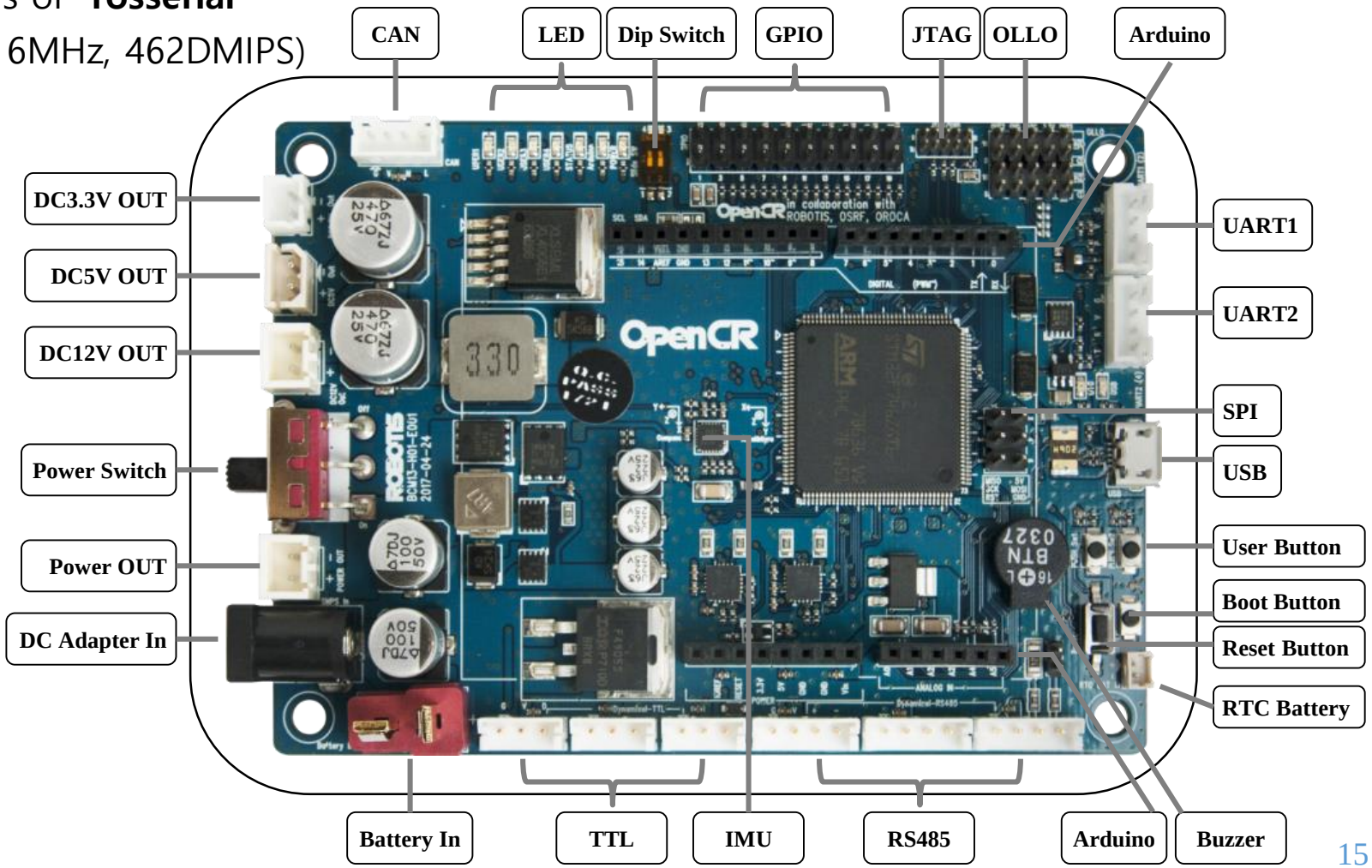
- **Memory**
  - The number of publishers, subscribers, and transmit/receive buffer size must be defined in advance
- **Float64**
  - Microcontroller does not support 64-bit real numbers, so it is converted to 32-bit
- **Strings**
  - Instead of storing string data in a string message, only pointer values of externally defined string data are stored in the message
- **Arrays**
  - Used with specified array size because of memory constraint
- **Communication Speed**
  - In case of a UART with the speed of 115200 bps, response time becomes slower as the number of messages increases

OpenCR; **Open**-source Control module for **ROS**



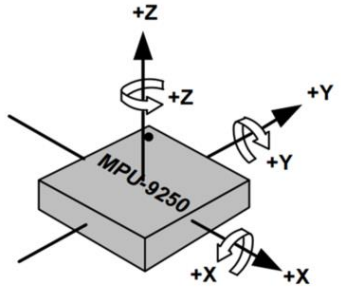
OpenCR (Open-source Control Module for ROS)

- It is an embedded board that **supports ROS** and is used as the main controller in **TurtleBot3**
- **Open source H/W, S/W** : H/W information such as circuit, BOM, Gerber data, and all S/W of OpenCR as open source
- Configuration to overcome the limitations of **rosserial**
  - 32-bit ARM Cortex-M7 with FPU (216MHz, 462DMIPS)
  - 1MB flash memory
  - 320KB SRAM
  - Float64 support
  - Using USB packet transmission instead of UART
- **Power design** for use with SBC series computers and various sensors
  - 12V@1A, 5V@4A, 3.3V@800mA
- **Expansion port**
  - 32 pins(L 14, R 18)
    *Arduino connectivity
  - OLLO Sensor module x 4 pins
  - Extension connector x 18 pins

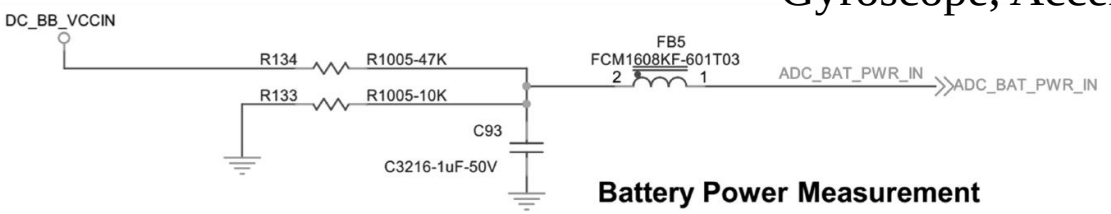


Built-in Sensor & Communication Support

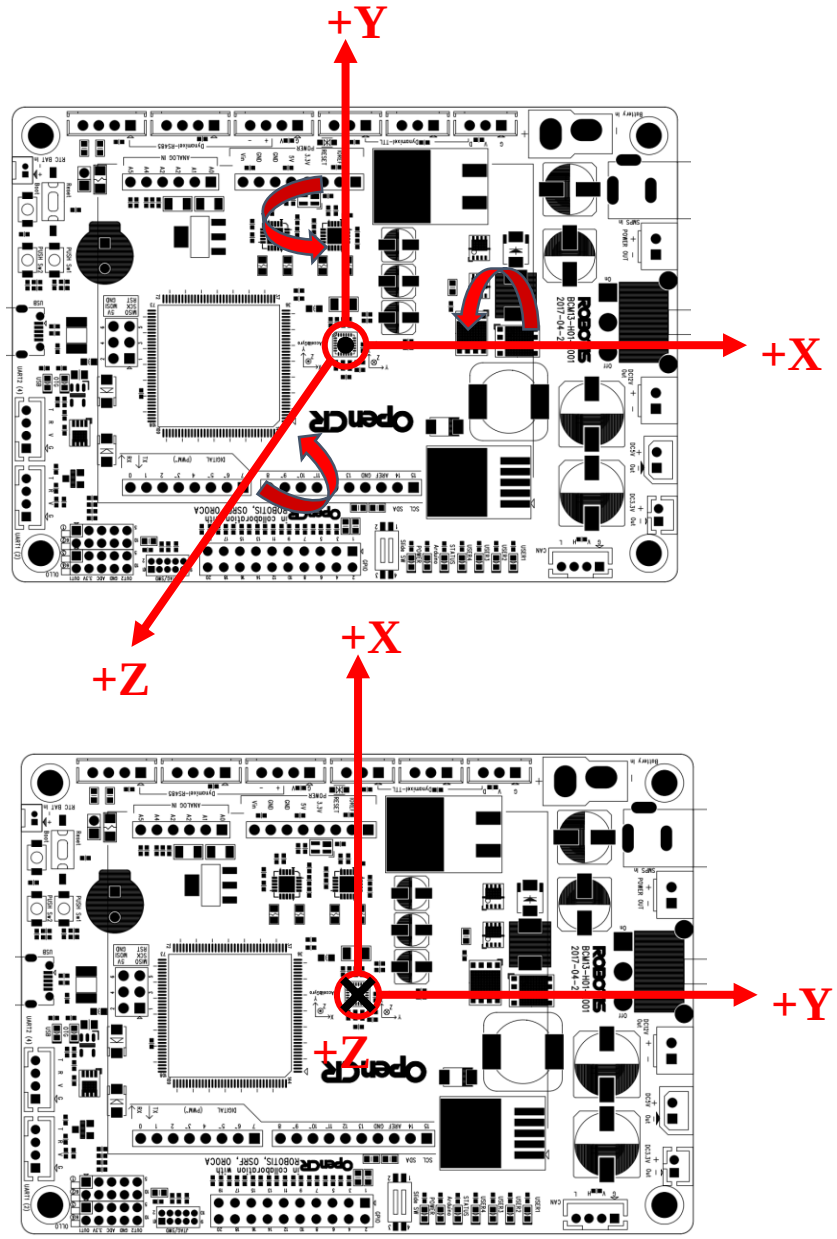
- Built-in Sensor
  - Gyroscope 3Axis
  - Accelerometer 3Axis
  - Magnetometer 3Axis
  - Voltage measuring circuit



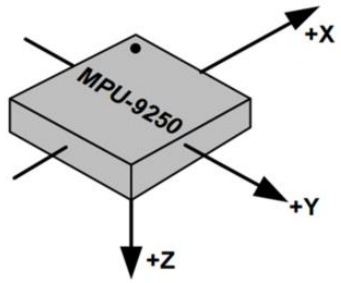
Gyroscope, Accelerometer



Battery Power Measurement

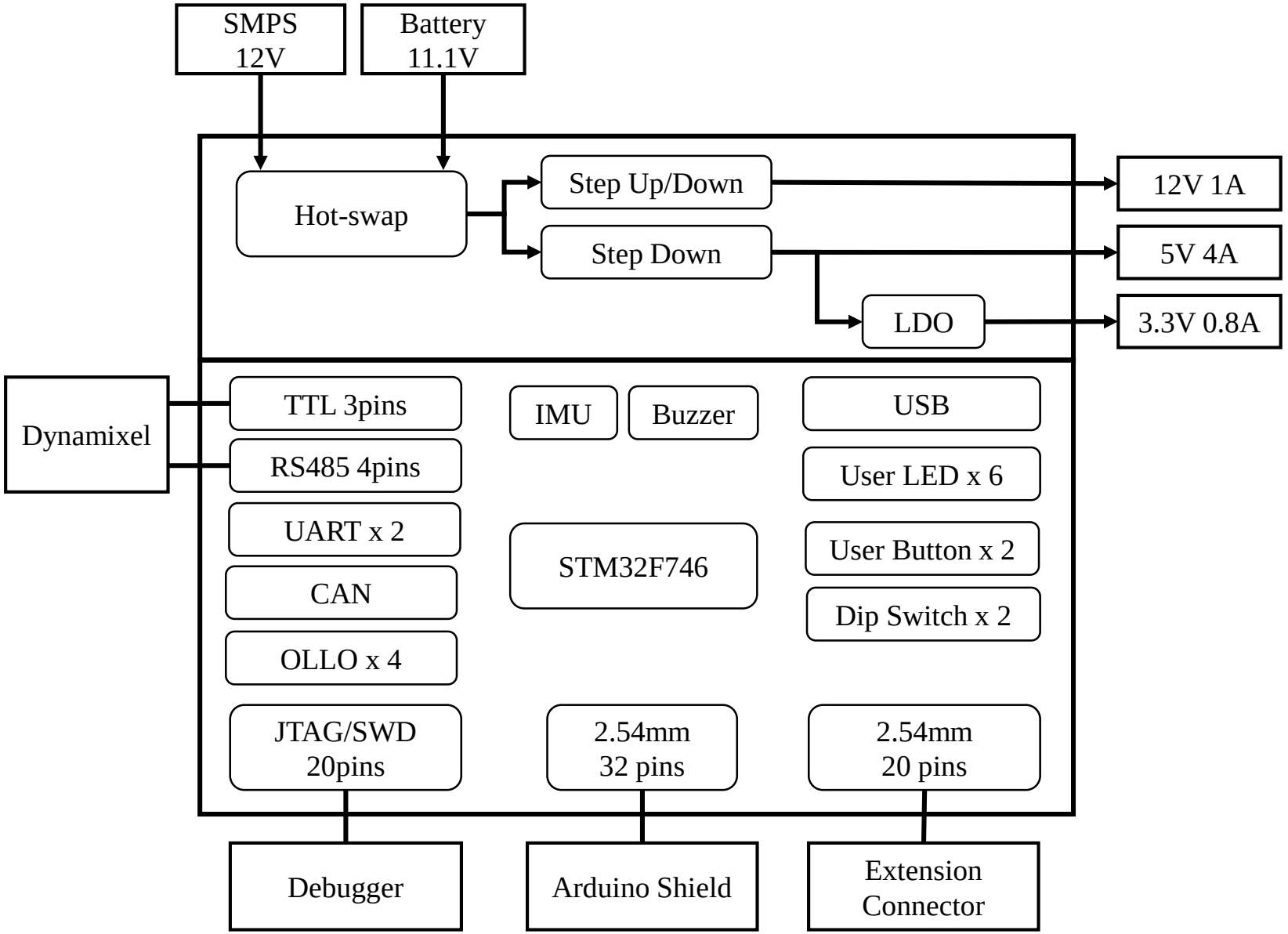


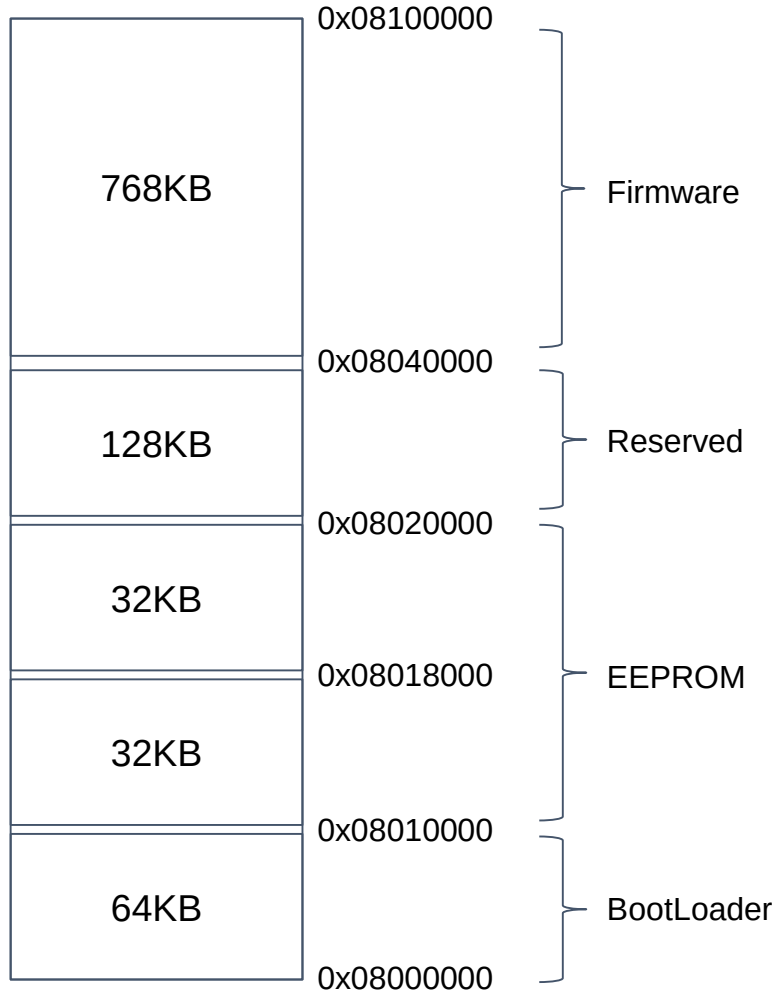
- Communication Support
  - USB, SPI, I2C
  - TTL, RS485, CAN



Magnetometer

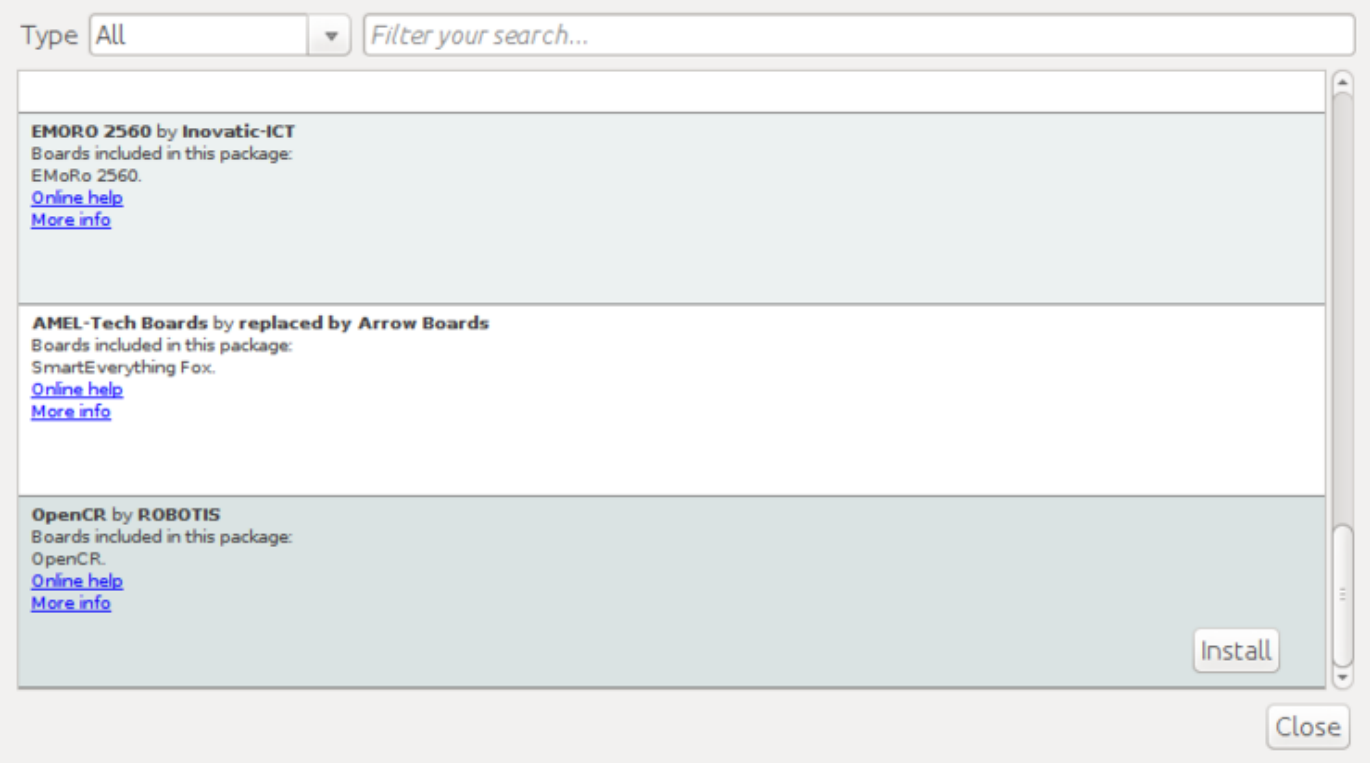
Block Diagram & Flash Memory Map

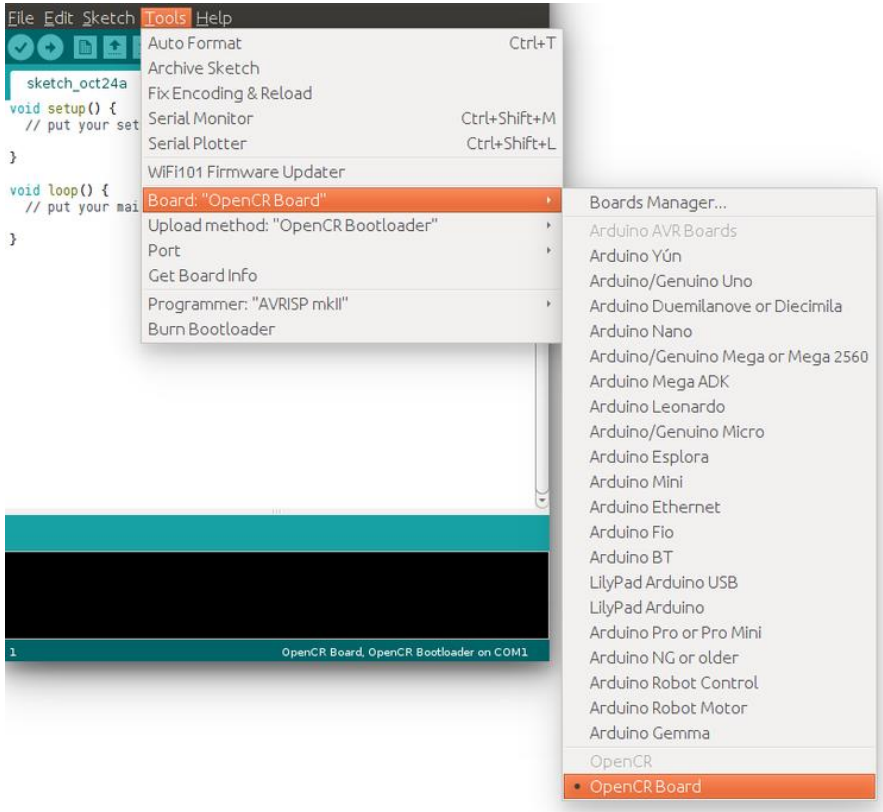




Establish Development Environment

- OpenCR supports Arduino IDE
- How to build OpenCR development environment
  - http://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_opencr1_0/
  - http://emanual.robotis.com/docs/en/parts/controller/opencr10/

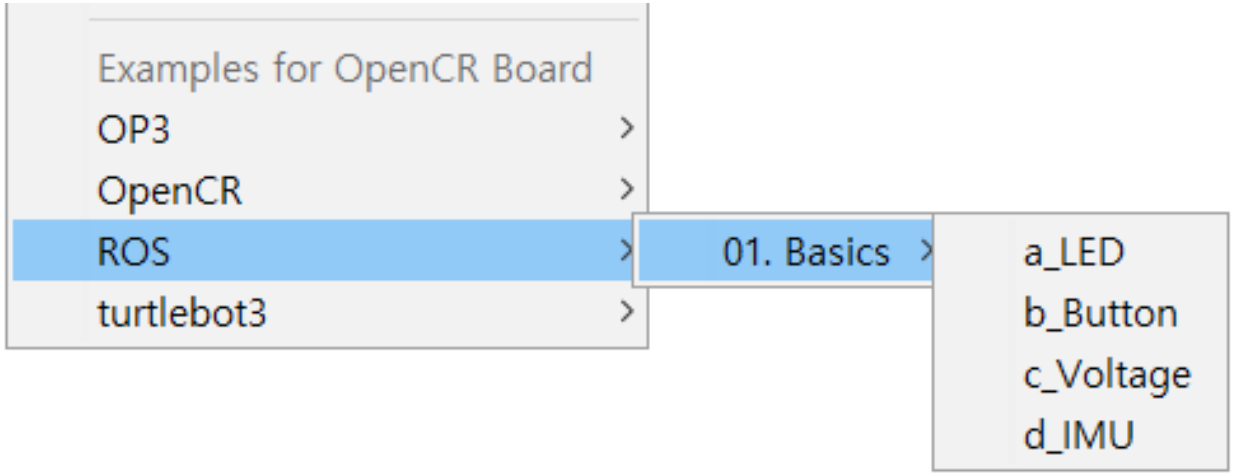


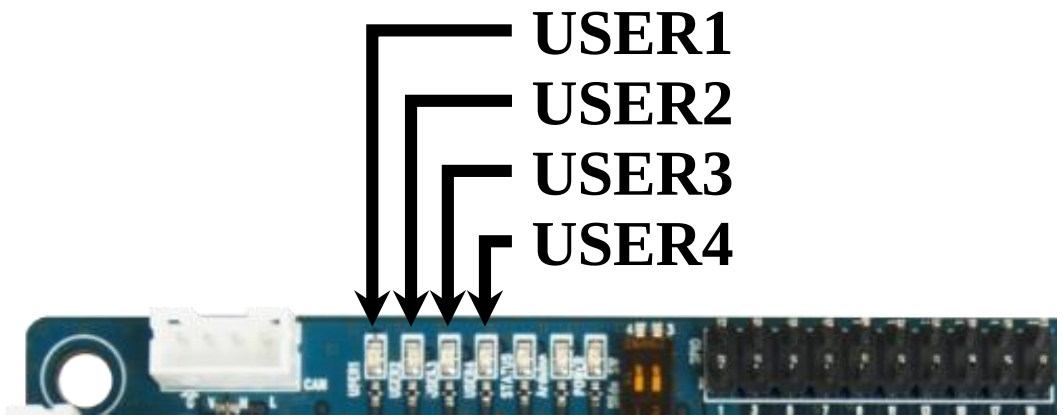


‘rosserial’ Example (LED Control)

$ arduino

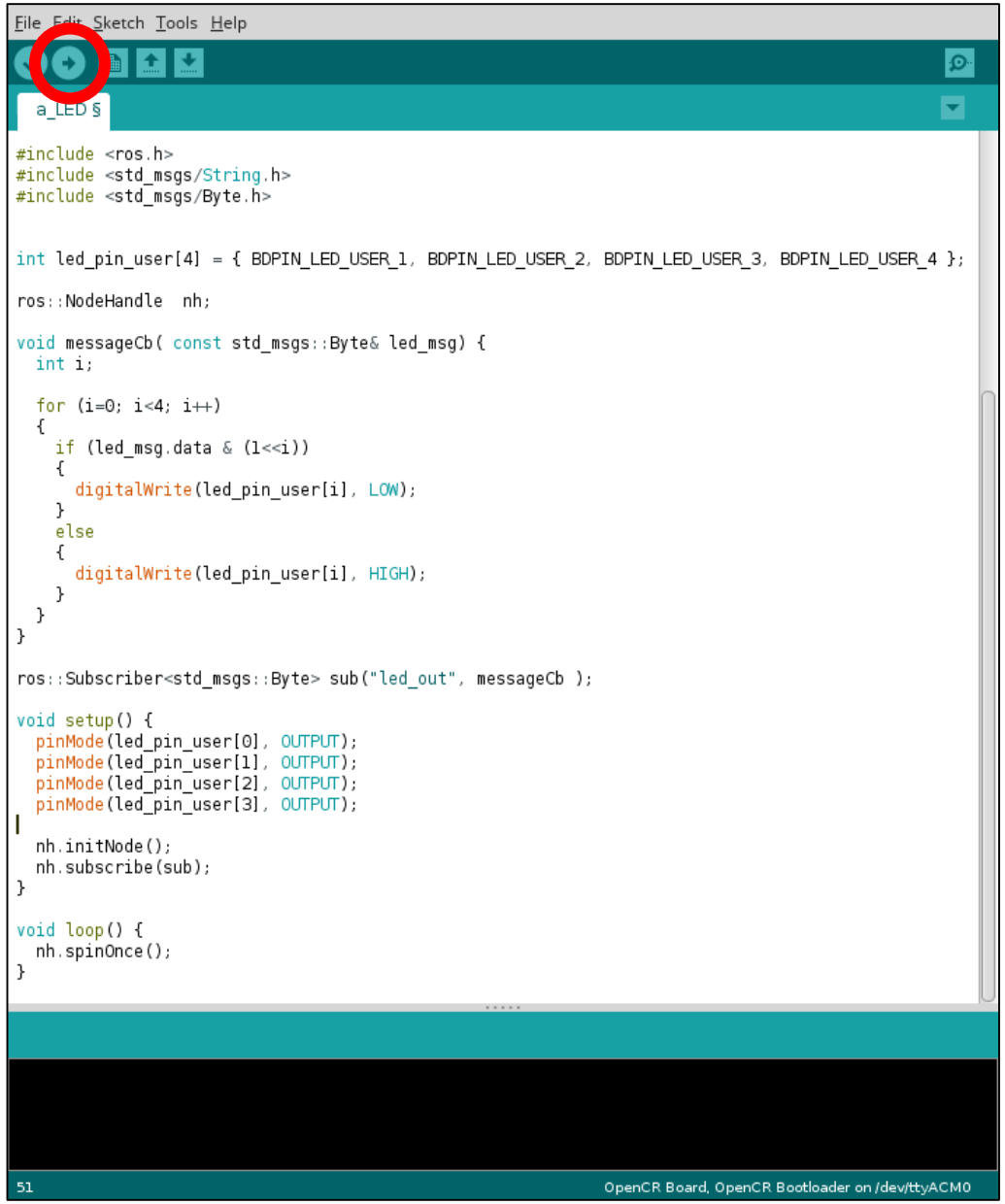
- After running Arduino, Load basic example [File] > [Examples] > [ROS] > [01. Basics] > [a_LED], build and upload
- This example defines the ‘led_out’ subscriber with 4 LEDs using ‘std_msg/Byte’ which is a ROS standard data type
- When the subscriber callback function is called, if the bit is 1, the LED is turned on; if it is 0, the LED is turned off





‘rosserial’ Example (LED Control)

```
#include <ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Byte.h>

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

ros::NodeHandle  nh;

void messageCb( const std_msgs::Byte& led_msg) {
  int i;

  for (i=0; i<4; i++)
  {
    if (led_msg.data & (1<<i))
    {
      digitalWrite(led_pin_user[i], LOW);
    }
    else
    {
      digitalWrite(led_pin_user[i], HIGH);
    }
  }
}

ros::Subscriber<std_msgs::Byte> sub("led_out", messageCb );

void setup() {
  pinMode(led_pin_user[0], OUTPUT);
  pinMode(led_pin_user[1], OUTPUT);
  pinMode(led_pin_user[2], OUTPUT);
  pinMode(led_pin_user[3], OUTPUT);

  nh.initNode();
  nh.subscribe(sub);
}

void loop() {
  nh.spinOnce();
}
```



Running ‘rosserial server’ and Publishing Topics for LED Control

- After running ‘roscore’, run ‘rosserial sever’

```
$ roscore
```

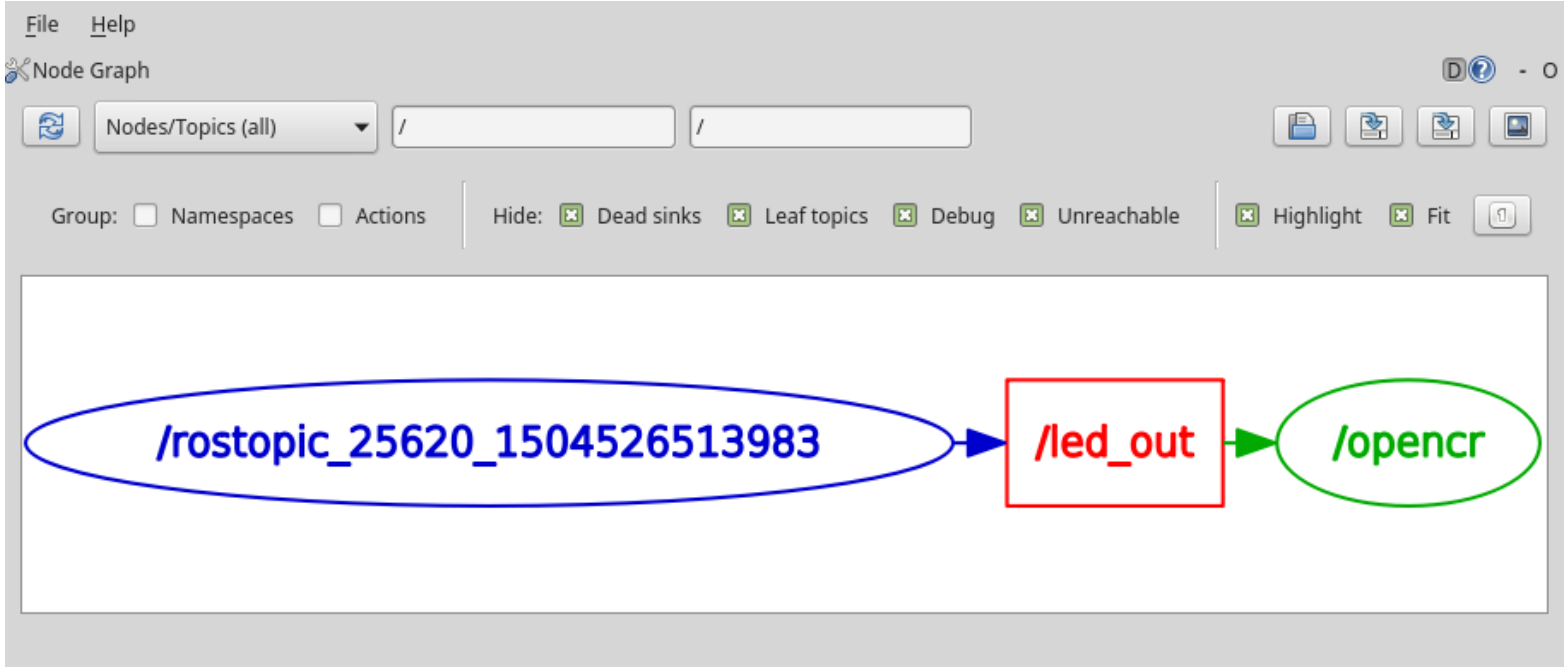
```
$ rosrun rosserial_python serial_node.py __name:=opencr _port:=/dev/ttyACM0 _baud:=115200
[INFO] [1495609829.326019]: ROS Serial Python Node
[INFO] [1495609829.336151]: Connecting to /dev/ttyACM0 at 115200 baud
[INFO] [1495609831.454144]: Note: subscribe buffer size is 1024 bytes
[INFO] [1495609831.454994]: Setup subscriber on led_out [std_msgs/Byte]
```

- Let’s use ‘rostopic pub’ to control the LED by entering a value in ‘led_out’

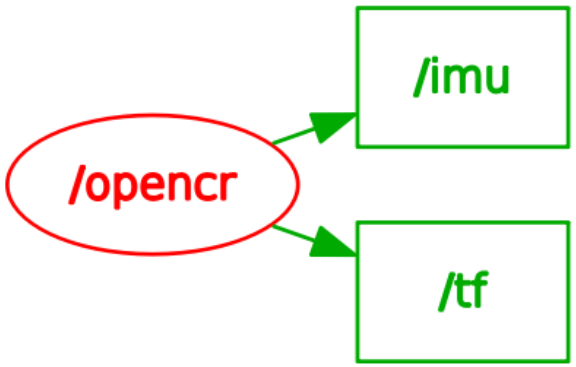
```
$ rostopic pub -1 led_out std_msgs/Byte 1      → USER1 LED On
$ rostopic pub -1 led_out std_msgs/Byte 2      → USER2 LED On
$ rostopic pub -1 led_out std_msgs/Byte 4      → USER3 LED On
$ rostopic pub -1 led_out std_msgs/Byte 8      → USER4 LED On
$ rostopic pub -1 led_out std_msgs/Byte 0      → LED Off
```

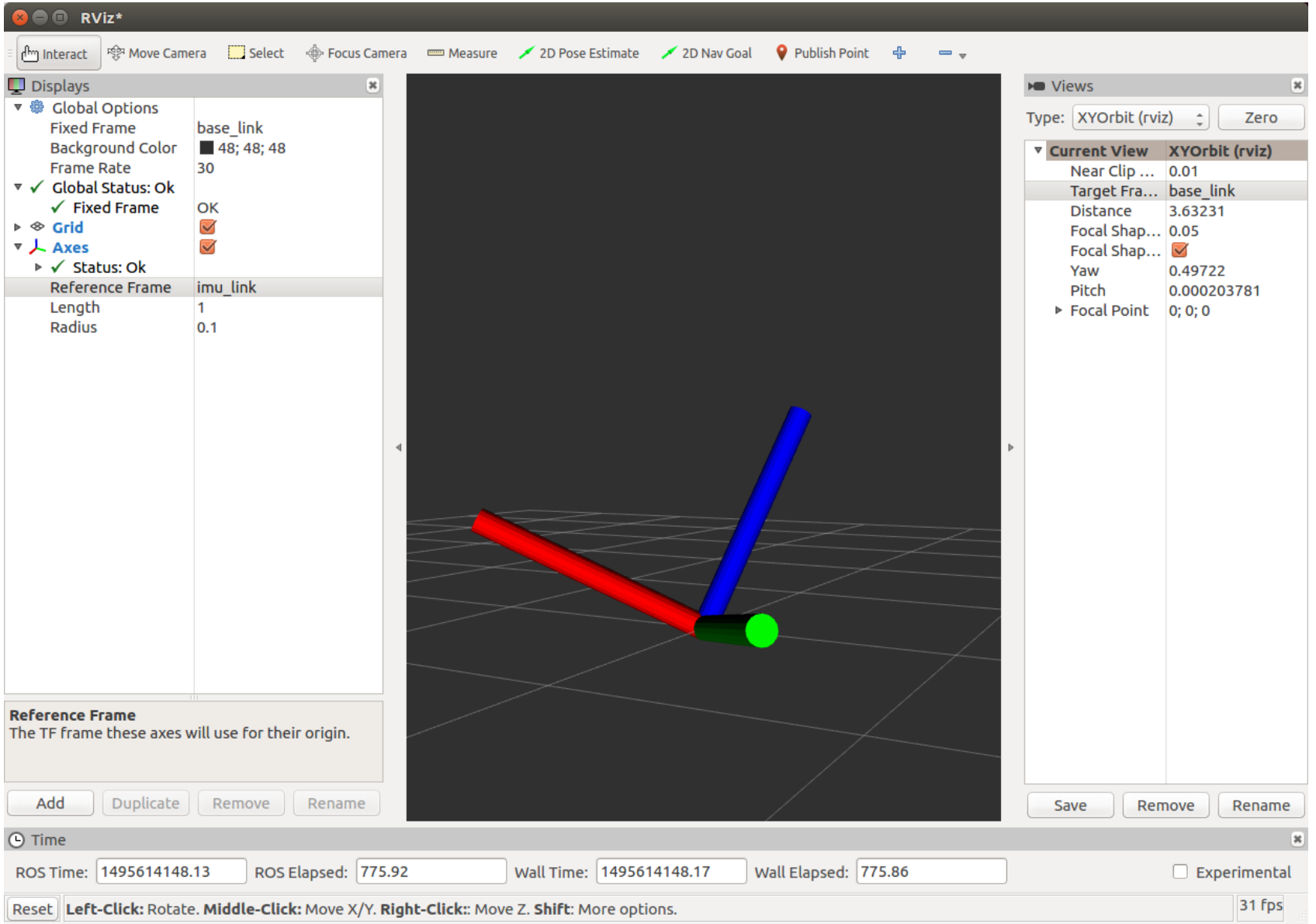
Publisher Node & Subscriber Node for LED Control

- Let's run rqt_graph
- It shows that rostopic command acts as a publisher node and opencr(rosserial server) is acting as a subscriber
- It is possible to confirm that information is being transmitted and received between the two nodes with the topic name '/led_out'

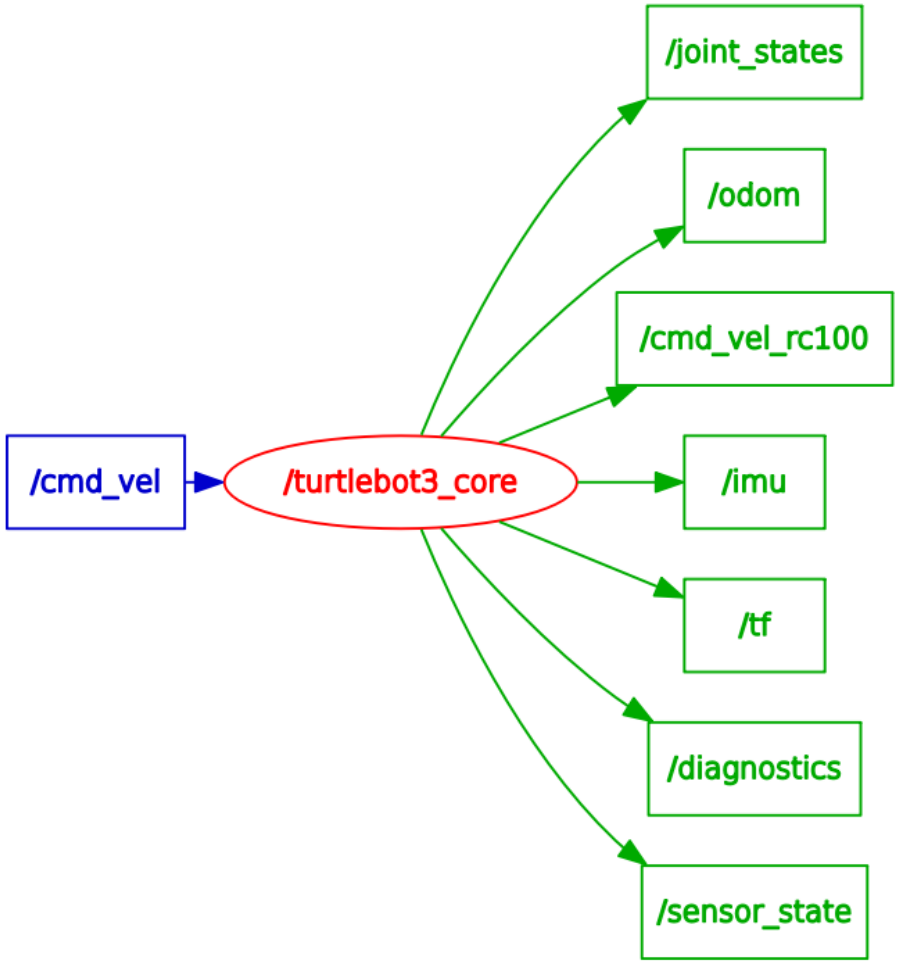


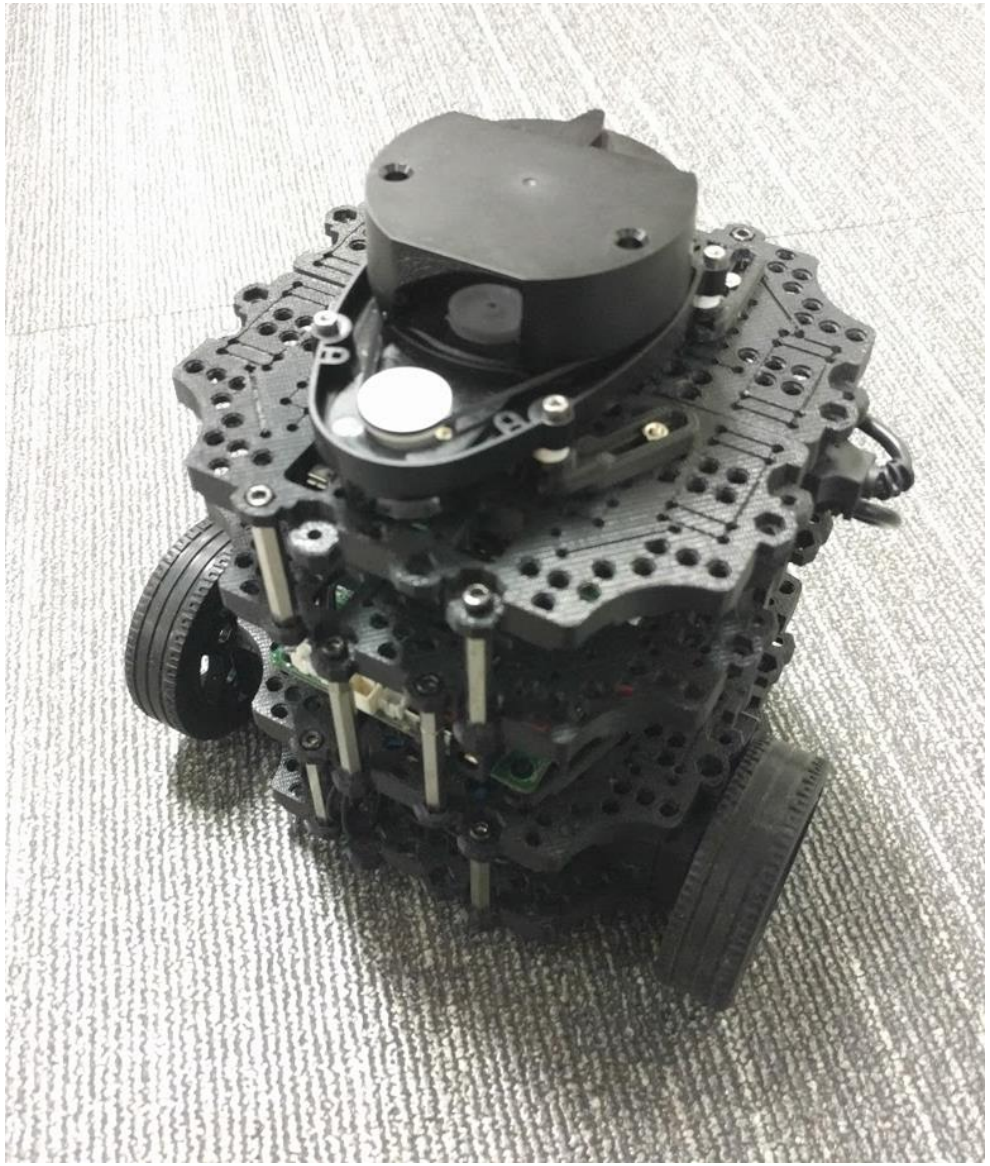
‘rosserial’ Example (IMU Control)





‘rosserial’ Example (TurtleBot3 Burger)





Question Time!

Advertisement #1

✓ Download link

✓ Language:

English, Chinese, Japanese, Korean

“ROS Robot Programming”

A Handbook is written by TurtleBot3 Developers

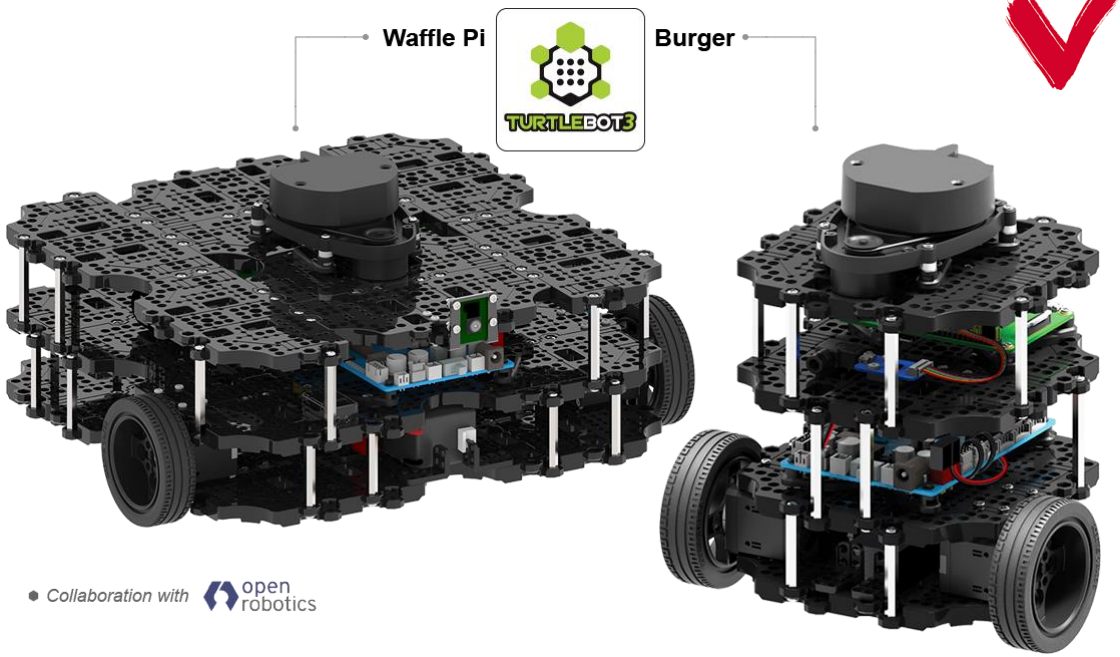
Advertisement #2

TURTLEBOT3

AI Research Starts Here
ROS Official Platform

TurtleBot3 is a new generation mobile robot that’s modular, compact and customizable. Let's explore ROS and create exciting applications for education, research and product development.

Direct Link



Advertisement #3

www.robotsource.org

The ‘RobotSource’ community is the space for people making robots.

We hope to be a community where we can share knowledge about robots, share robot development information and experiences, help each other and collaborate together. Through this community, we want to realize open robotics without disti nguishing between students, universities, research institutes and companies.

Join us in the Robot community ~

END.