Week 6
OpenCR 및 ROS Serial

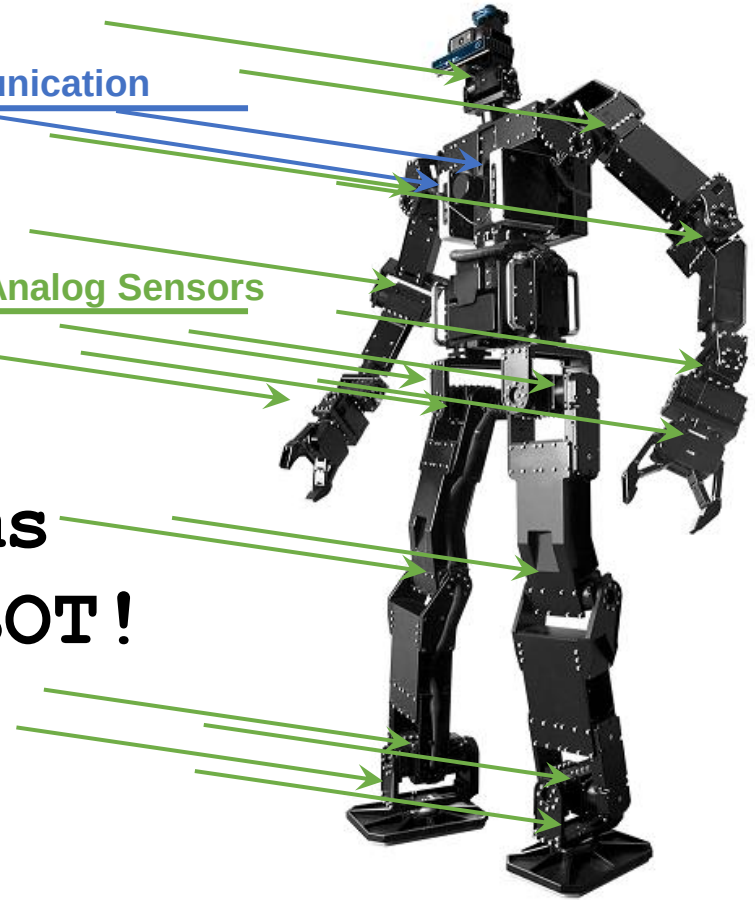
Youngjoon Han

young@ssu.ac.kr

[본 강의자료는 ROBOTIS Open Source Team의 표윤석 박사의 공개자료를 활용함]

Intel NUC
for Sensor and Communication

ARM Cortex-M
for Motor control and Analog Sensors



Small embedded systems are everywhere in ROBOT!

THORMANG3

Scope of computing resources : computation vs control



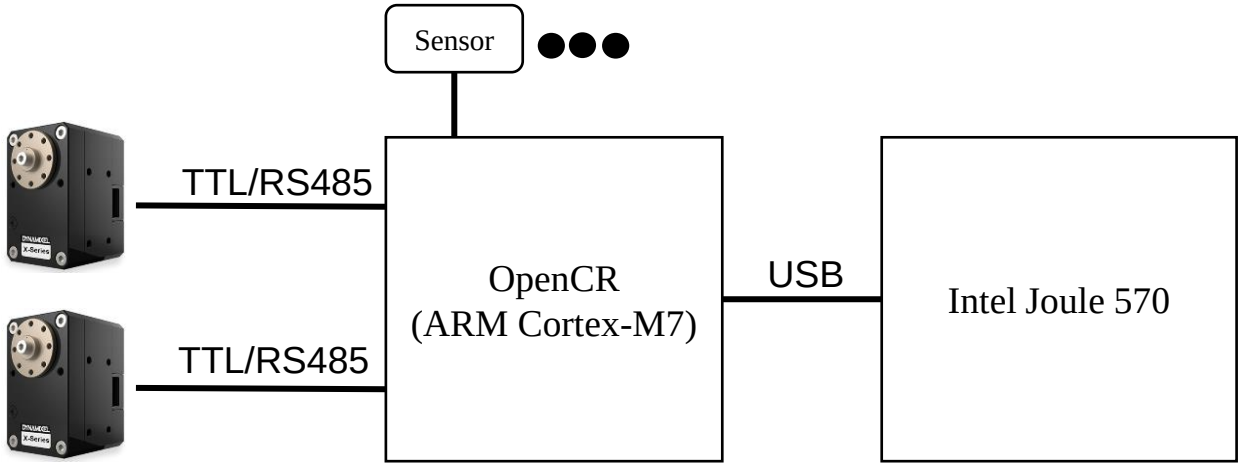


	8/16-bit MCU	32-bit MCU		ARM A-class (smartphone without screen)	x86 (laptop without screen)
		"small" 32-bit MCU	"big" 32-bit MCU		
Example Chip	Atmel AVR	ARM Cortex-M0	ARM Cortex-M7	Samsung Exynos	Intel Core i5
Example System	Arduino Leonardo	Arduino M0 Pro	SAM V71	ODROID	Intel NUC
MIPS	10's	100's	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

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

ROS에서 임베디드 시스템

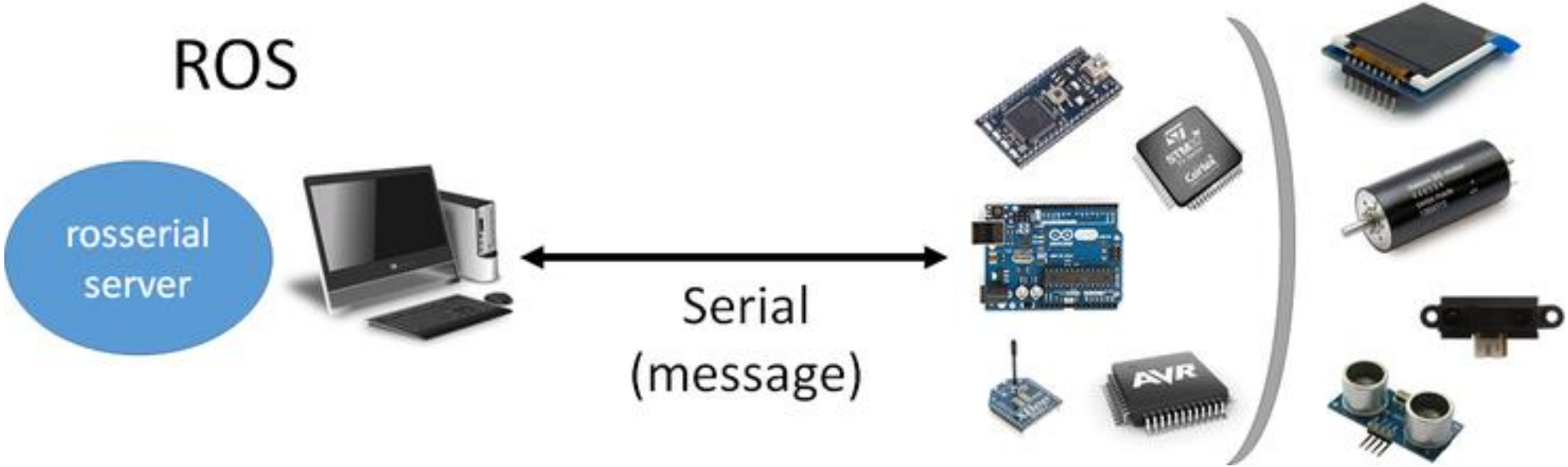
- PC와는 달리 임베디드 시스템에서는 ROS 설치가 불가능
- 실시간성 확보 및 하드웨어 제어를 위해서는
  ROS가 설치된 PC와 임베디드 시스템 간의 연결이 요구됨
- ROS에서는 이를 위해 rosserial 이라는 패키지를 제공!



[터틀봇3에서의 PC와 임베디드 시스템의 구성]

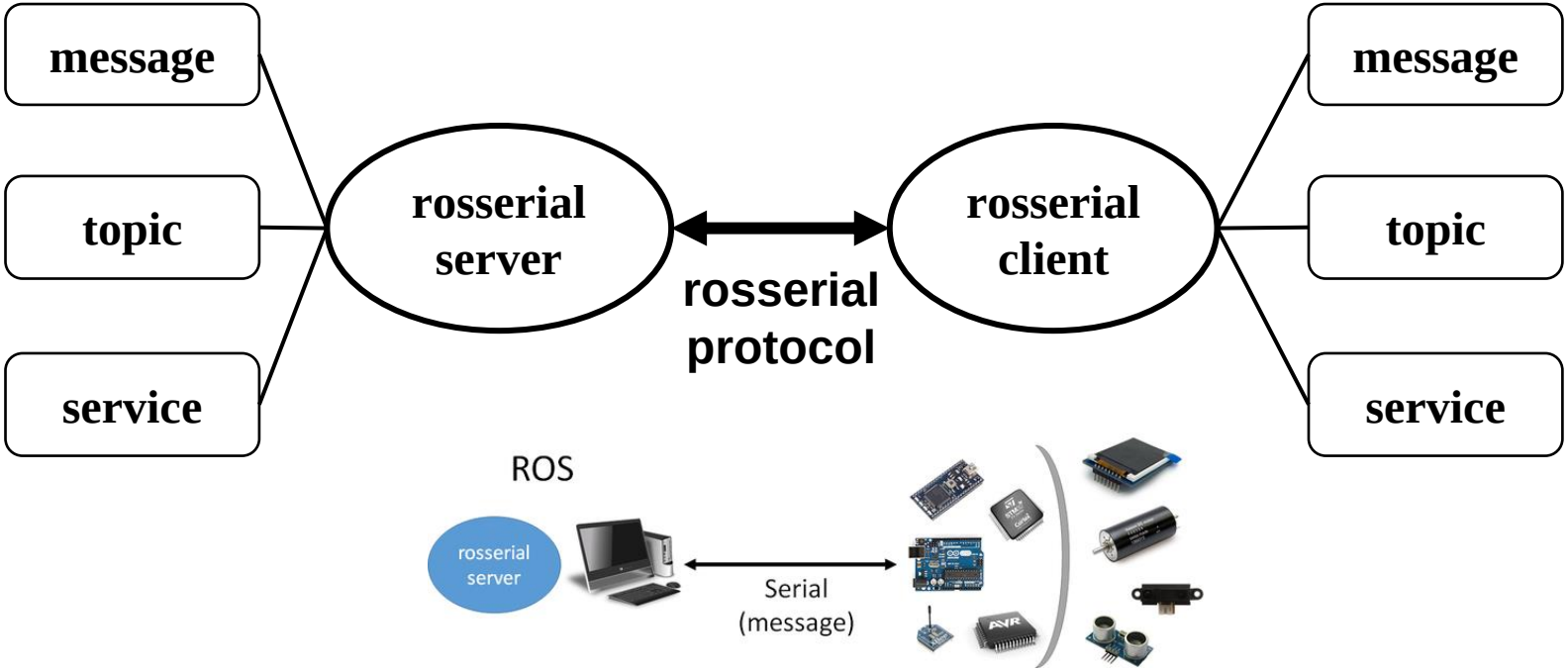
rosserial

- PC와 제어기 간의 메시지 통신을 위해 중계자 역할을 수행하는 ROS 패키지
  - 예) 제어기 → 시리얼(rosserial 프로토콜) → PC(ROS 메시지로 재전송)
  - 예) 제어기 ← 시리얼(rosserial 프로토콜) ← PC(ROS 메시지를 시리얼로 변경)



http://wiki.ros.org/rosserial

rosserial server와 client



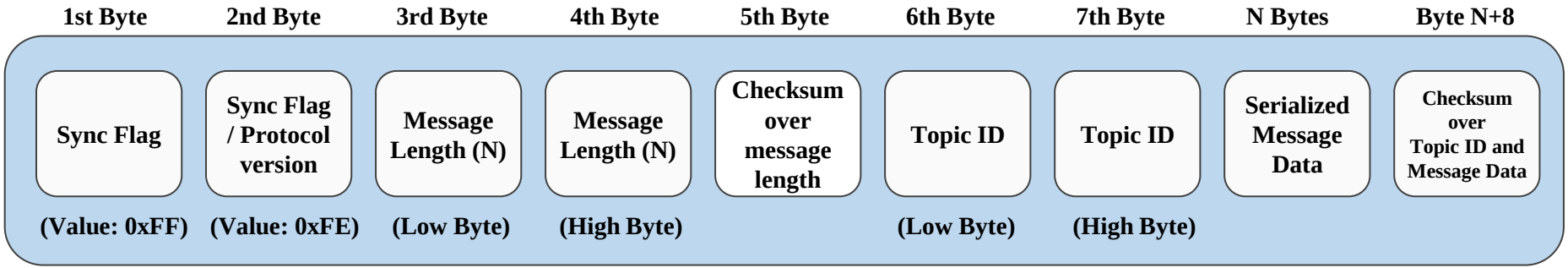
rosserial server와 client

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

- **rosserial_python**: Python 언어 기반의 rosserial server, 매우 많이 사용됨
- **rosserial_server**: C++ 언어 기반의 rosserial server, 일부 기능 제약 있음
- **rosserial_java**: Java 언어 기반의 rosserial server, 안드로이드 SDK와 함께 사용

- **rosserial client**
  - **rosserial_arduino**: Arduino와 Leonardo 지원, OpenCR은 수정하여 사용 중
  - **rosserial_embeddedlinux**: 임베디디용 리눅스용 라이브러리
  - **rosserial_windows**: 윈도우 운영체제 지원, 윈도우의 응용프로그램과 통신 지원
  - **rosserial_mbed**: ARM사의 mbed 지원
  - **rosserial_tivac**: TI사의 Launchpad 지원

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



Sync Flag	패킷의 시작 위치를 알기 위한 헤더로 항상 0xFF 이다.
Sync Flag / Protocol version	프로토콜 버전으로 ROS Groovy는 0xFF이고, ROS Hydro, Indigo, Jade 그리고 Kinetic은 0xFE 이다.
Message Length (N)	메시지의 데이터 길이, 2 Byte = Low Byte + High Byte, Low 바이트가 먼저 전송되고 이어서 High 바이트가 전송된다.
Checksum over message length	메시지 길이 헤더의 유효성 검증을 위한 체크섬 Checksum = 255 - ((Message Length Low Byte + Message Length High Byte) %256)
Topic ID	메시지의 형태를 구분하기 위한 ID이다. 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	송/수신 메시지를 시리얼 형태로 전송하기 위한 데이터이다. 예) IMU, TF, GPIO 데이터
Checksum over Topic ID and Message Data	Topic ID와 메시지 데이터의 유효성 검증을 위한 체크섬 Checksum = 255 - ((Topic ID Low Byte + Topic ID High Byte + data byte values) % 256)

Rosserial 제약사항 (http://wiki.ros.org/rosserial/Overview/Limitations)

1. Maximum Size of a Message, Maximum Number of Publishers/Subscribers

ros::NodeHandle_<HardwareType, MAX_PUBLISHERS=25, MAX_SUBSCRIBERS=25, IN_BUFFER_SIZE=512, OUT_BUFFER_SIZE=512> nh;

: 실제 노드핸들의 선언에서 퍼블리셔의 갯수, 서브스크라이버의 갯수, 수신버퍼, 송신 버퍼를 설정하게 되는데 이는 사용하는 MCU에 의존하게 된다.

2. Float64

: 아두이노의 한계로 64-bit float은 32-bit 데이타형으로 변환된다.

Rosserial 제약사항

3. Strings
: 메모리 사용 문제로 String 대신에 "char *"를 사용한다.

4. Arrays
: 메모리 사용 문제로 길이를 지정하여 사용한다.

5. Speed and bandwidth
: serial의 하드웨어적인 제어로 인하여 전송 속도와 대역폭에 한계가 존재한다.

http://wiki.ros.org/rosserial/Overview/Limitations

OpenCR's Challenge

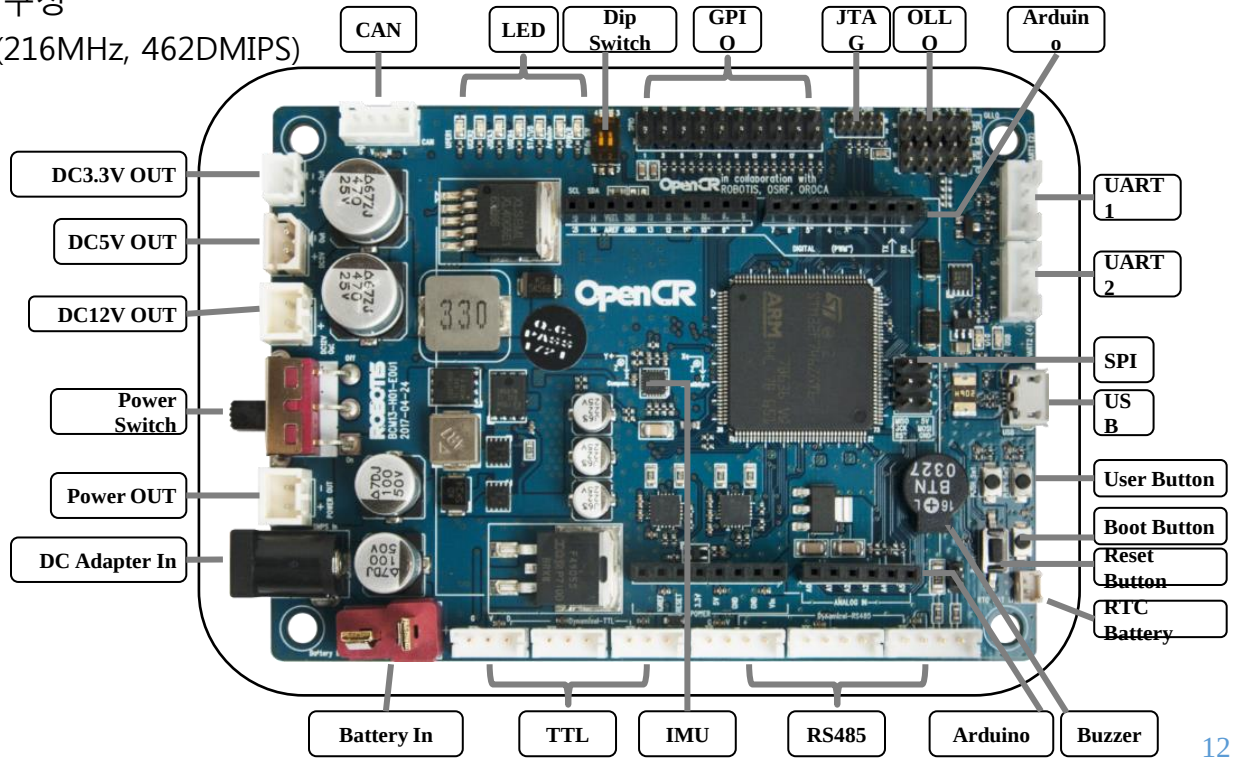
- rosserial 에서의 한계 =8Bit 계열의 MCU / Serial (UART)

- OpenCR은 32Bit 계열의 MCU이기에 이를 하드웨어적 성능으로 극복할 수 있으며, serial이 아닌 USB Stack 및 Ethernet Stack 을 이용하면 시리얼 통신에만 묶여 있던 제약을 해결할 수 있을 것으로 예상된다.

- 이 숙제를 풀어내는 것이 OpenCR이 “Open-source Control module for ROS”으로 거듭날 수 있는 길이라고 생각한다.

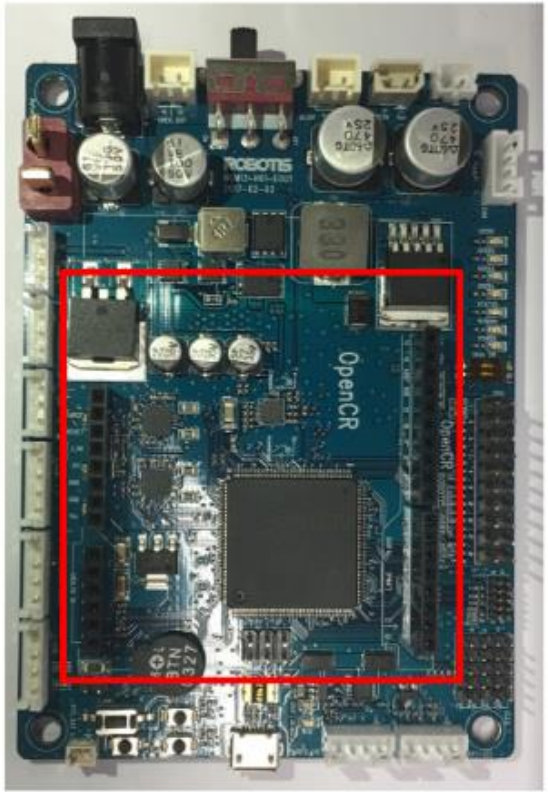
OpenCR (Open-source Control Module for ROS)

- **ROS를 지원하는 임베디드 보드**이며 **터틀봇3**에서 메인 제어기로 사용된다.
- **오픈소스 H/W, S/W** : 회로, BOM, 거버 데이터 등의 H/W 정보 및 OpenCR의 모든 S/W를 오픈소스로 공개
- **rosserial의 제약 사항을 극복**하기 위한 구성
  - 32-bit ARM Cortex-M7 with FPU (216MHz, 462DMIPS)
  - 1MB 플래시 메모리
  - 320KB SRAM
  - Float64 지원
  - UART가 아닌 USB 패킷 전송 사용
- SBC 계열의 컴퓨터와 다양한 센서와 함께 사용하기 위한 **전원 설계**
  - 12V@1A, 5V@4A, 3.3V@800mA
- **확장 포트**
  - 32 pins(L 14, R 18)
    *Arduino connectivity
  - OLLO Sensor module x 4 pins
  - Extension connector x 18 pins

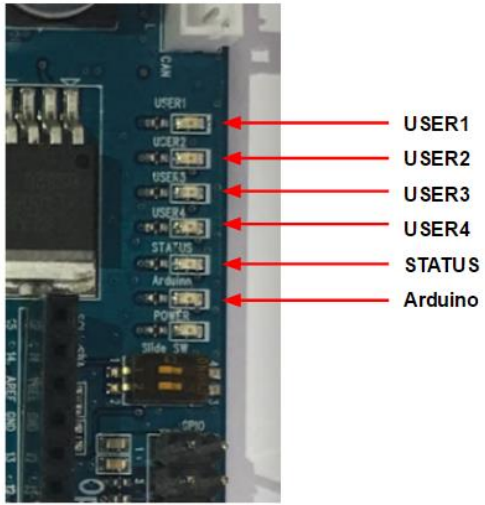


OpenCR (Open-source Control Module for ROS)

- Arduino Uno pinmap과 호환이 되는 커넥터를 갖고 있음

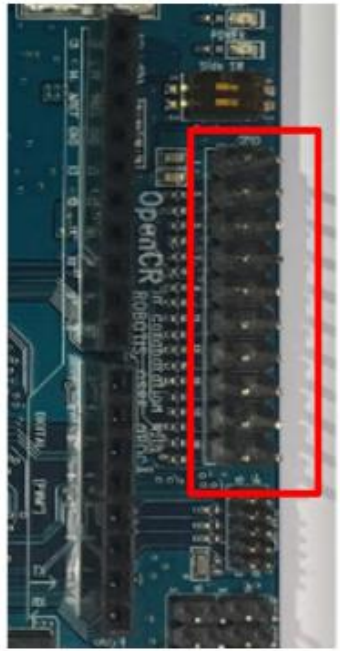


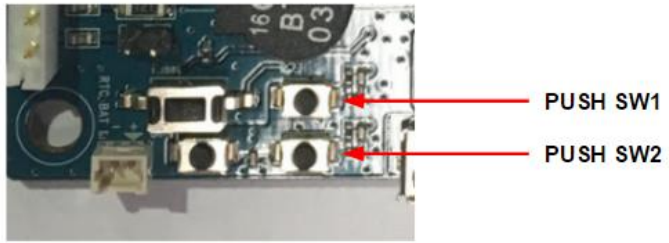
User LED



Dip Switch

GPIO





Push Switch

OpenCR (Open-source Control Module for ROS)

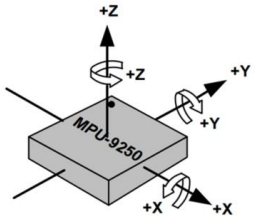
- **UART(Serial)**

Class Instance	Arduino Pin	Hardware
Serial	USB	USB
Serial1	0(RX), 1(TX)	USART6
Serial2 (SerialBT1)	UART1	USART2
Serial3	DXL Port	USART3
Serial4 (SerialBT2)	UART2	UART8

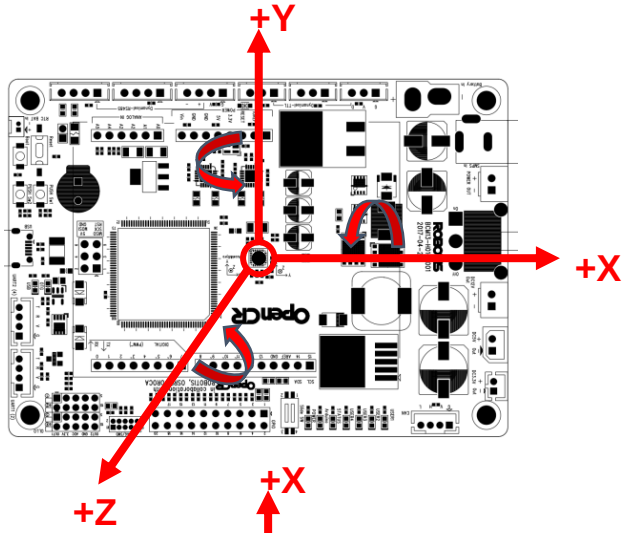
Since Serial3 is used for Dynamixel, its usage differs from other serial.

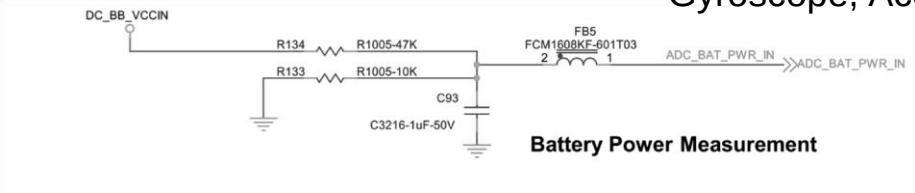
기본 장착 센서 및 통신 지원

- 기본 장착 센서
  - Gyroscope 3Axis
  - Accelerometer 3Axis
  - Magnetometer 3Axis
  - 전압 측정 회로



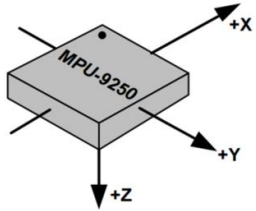
Gyroscope, Accelerometer



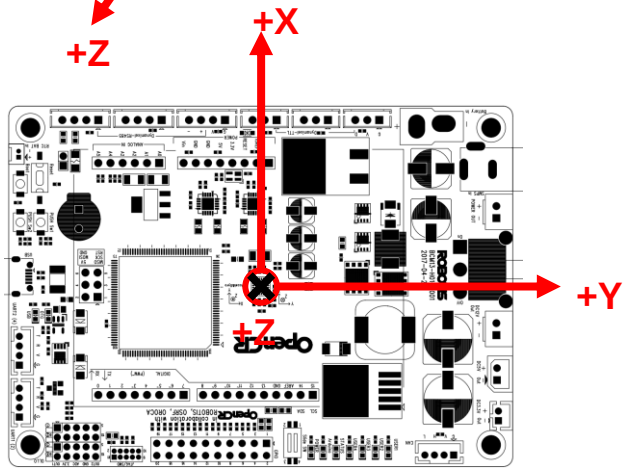


Battery Power Measurement

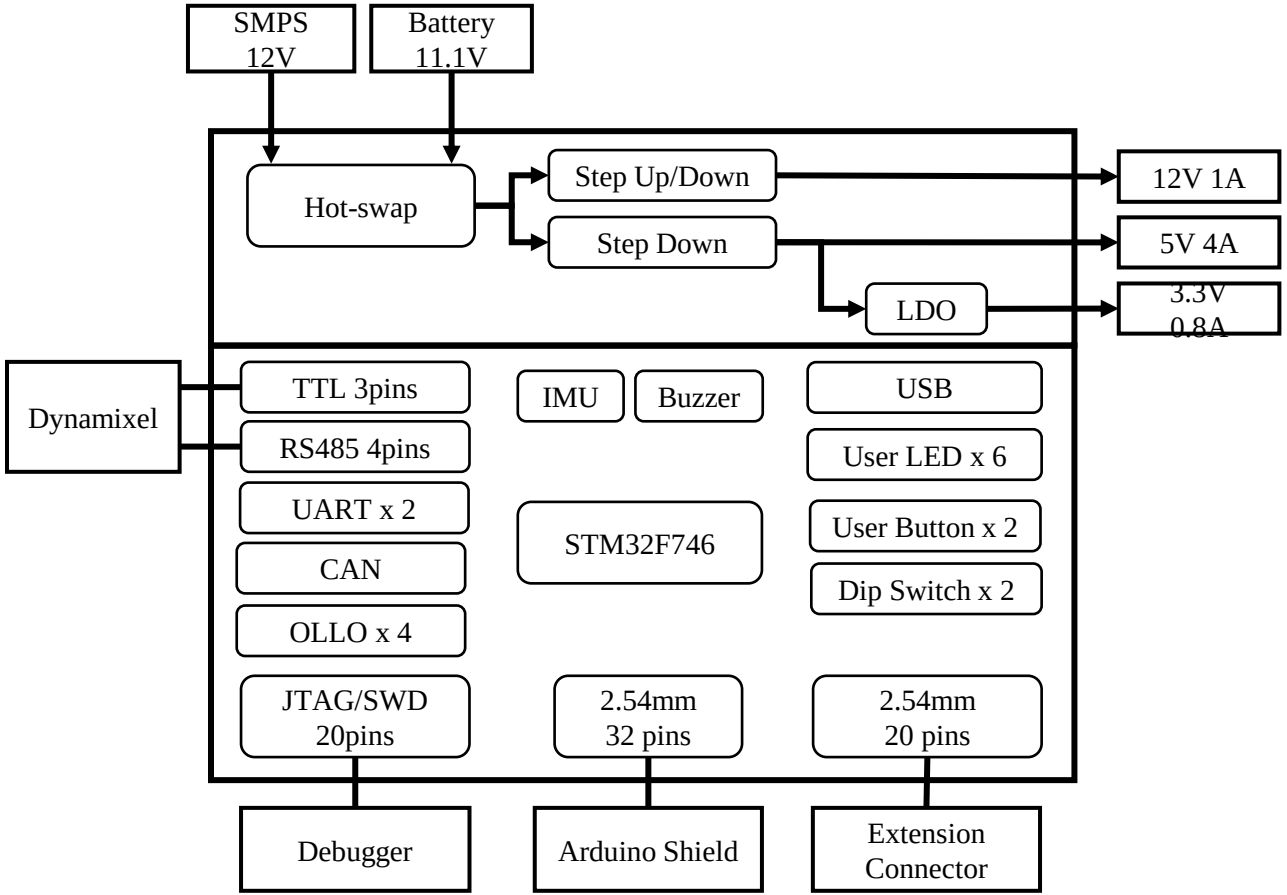
- 통신 지원
  - USB, SPI, I2C
  - TTL, RS485, CAN

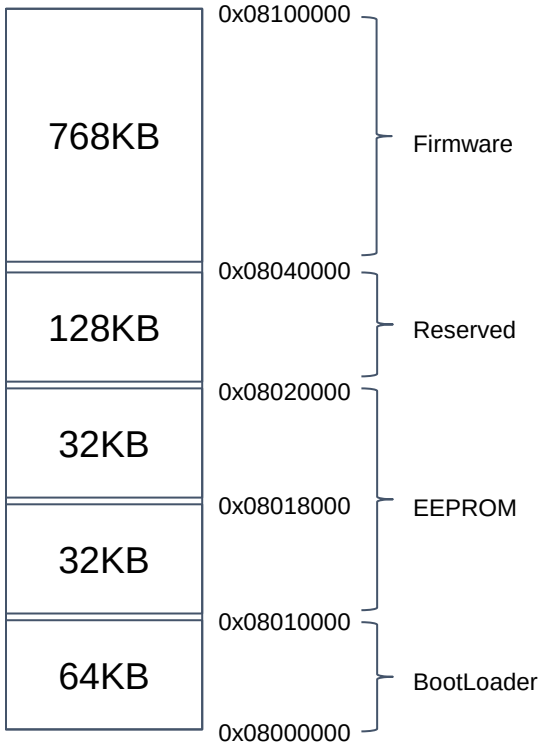


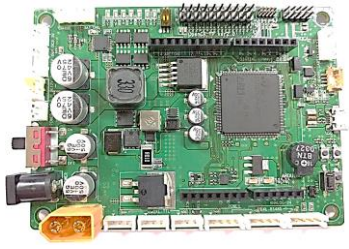
Magnetometer



블록 다이어그램 및 플래시 메모리 맵



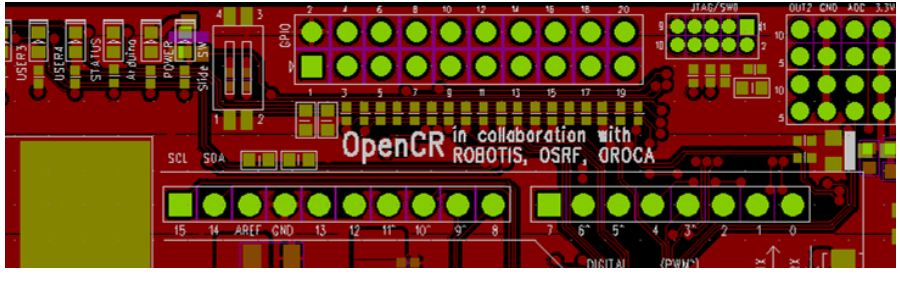




rosserial

OpenCR + ROSSERIAL

이하 설명에서는 컴퓨터에 Ubuntu 16.04 LTS OS와 ROS Kinetic 버전이 설치되어 있다는 전제 하에 OpenCR + rosserial 를 사용하는 법을 설명한다. F/W는 Arduino IDE 1.6.12 에서 개발함



rosserial install

다음 명령어로 rosserial과 아두이노 계열 지원 패키지를 설치하도록 하자.

그 이외에도 ros-kinetic-rosserial-windows ros-kinetic-rosserial-xbee, ros-kinetic-rosserial-embeddedlinux 등이 있는데 이는 필요할 경우에만 설치하도록 하자.

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

Build the ros libs for rosserial

- 아두이노 IDE 에서 사용가능한 라이브러리를 생성
- 다음과 같이 아두이노 IDE의 개인폴더의 라이브러리 폴더로 이동 →기존 ros_lib 를 모두 삭제

- 그 뒤, rosserial_arduino 패키지의 make_libraries.py 파일을 실행하여 ros_lib를 생성

```
$ cd ~/Arduino/libraries/
$ rm -rf ros_lib
$ rosrun rosserial_arduino make_libraries.py .
```

Build the ros libs for rosserial

- Arduino IDE 에서 ROS 프로그래밍이 가능하도록 ros.h와 같은 기본 헤더파일들과 설치된 ROS 패키지 및 catkin_ws 에 설치된 개인이 작성한 패키지들의 메시지 파일등의 헤더파일 등이 생성

OpenCR의 시리얼 통신은 기본 아두이노 보드와 조금 틀려서 설정을 하나 변경할 필요가 있다.
~/Arduino/libraries/ros_lib/ArduinoHardware.h 의 59번째 라인의 아래 내용을 변경하자.

(수정전) #define SERIAL_CLASS HardwareSerial

(수정후) #define SERIAL_CLASS USBSerial

Communicating ROS messages between the computer and the OpenCR board

- 마지막으로 컴퓨터와 OpenCR 보드간의 ROS 메시지 통신을 시도함
- OpenCR 보드를 컴퓨터의 USB단자에 연결하고 다음과 같이 아두이노 IDE를 실행

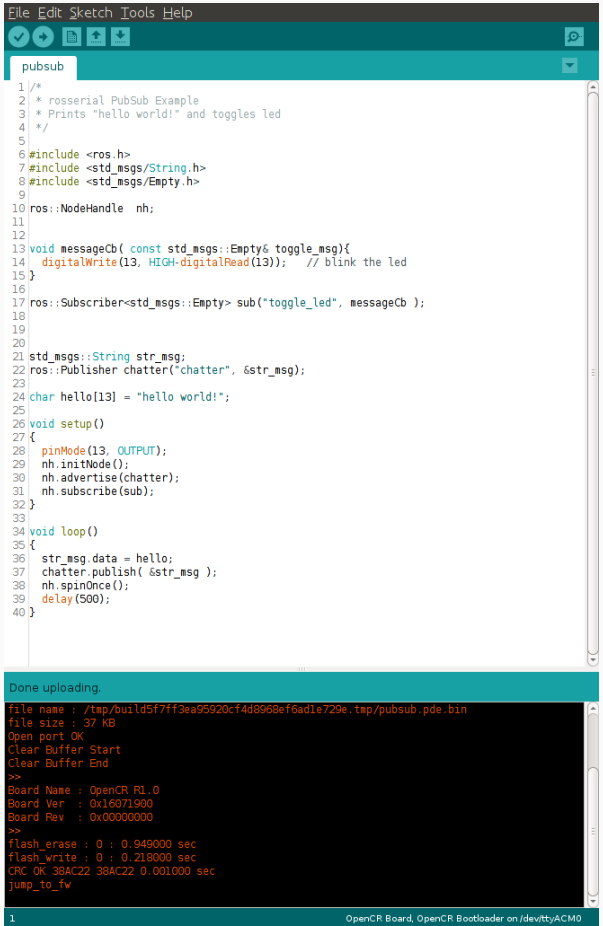
```
$ arduino
```

- 메뉴에서 [File] > [Sketchbook] > [libraries] > [ros_lib] > [pubsub] 를 선택하고 업로드 버튼을 클릭하여 OpenCR 보드에 업로드함
- 0.5초 단위로 "hello world!"라는 string 타입의 메시지를 퍼블리시(publish)하고 toggle_led 라는 토픽 이름의 LED 제어 신호를 서브스크라이브(subscribe)하는 프로그램

Example of rosserial: pubsub

pubsub

```
/*
 * rosserial PubSub Example
 * Prints "hello world!" and toggles led
 */

#include <ros.h>
#include <std_msgs/String.h>
#include <std_msgs/Empty.h>

ros::NodeHandle  nh;


void messageCb( const std_msgs::Empty& toggle_msg){
  digitalWrite(13, HIGH-digitalRead(13));   // blink the led
}

ros::Subscriber<std_msgs::Empty> sub("toggle_led", messageCb );



std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);

char hello[13] = "hello world!";

void setup()
{
  pinMode(13, OUTPUT);
  nh.initNode();
  nh.advertise(chatter);
  nh.subscribe(sub);
}

void loop()
{
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(500);
}
```



Example of rosserial: pubsub

- 다음 예제와 같이 roscore를 실행

```
$ roscore
```

- 다른 터미널창에서 rosserial_python패키지의 serial_node.py를 실행하자.

```
$ rosrun rosserial_python serial_node.py
_port:=/dev/ttyACM0
```

Example of rosserial: pubsub

- OpenCR 보드에서 퍼블리시하고 있는 메시지를 확인
- 다음과 같이 rostopic echo 명령어로 "chatter" 라는 토픽의 메시지를 확인
- 0.5초 단위로 메시지를 받고 있음을 확인할 수 있음

```
$ rostopic echo /chatter
data: hello world!
---
data: hello world!
---
data: hello world!
```

Example of rosserial: pubsub

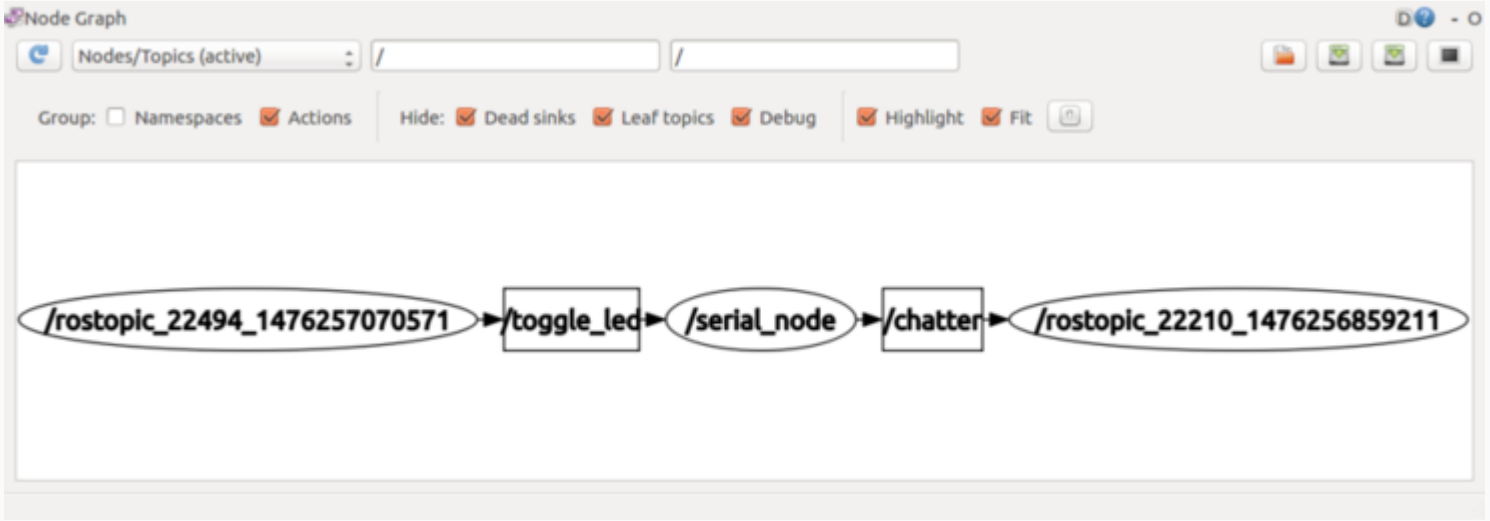
- 컴퓨터에서 std_msgs/Empty 타입의 "toggle_led"라는 토픽으로 OpenCR 보드에게 led 토글 신호를 퍼블리시함
- OpenCR 보드에서는 이를 서브스크라이브하여 led 를 on/off 되는지 확인
- 명령어를 보내면 led 는 켜졌다가 다시 한번 명령어를 보내면 꺼짐을 확인할 수 있음

```
$ rostopic pub -1 /toggle_led std_msgs/Empty "{}"
publishing and latching message for 3.0 seconds

$ rostopic pub -1 /toggle_led std_msgs/Empty "{}"
publishing and latching message for 3.0 seconds
```

Example of rosserial: pubsub

이 퍼블리시와 서브스크라이브 메시지 통신을 rqt_graph로 확인하면 다음과 같다.

```
$ rqt_graph
```



Example of rosserial: pubsub

- 퍼블리시와 서브스크라이브 관련 프로그램은 일반적인 ROS 프로그래밍과 같은 방법으로 작성됨
- 다만 다른 것은 그 대상이 OpenCR 라는 마이크로컨트롤러 보드임

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

ros::NodeHandle  nh;

void messageCb( const std_msgs::Empty& toggle_msg) {
  digitalWrite(13, HIGH-digitalRead(13));   // blink the led
}

ros::Subscriber<std_msgs::Empty> sub("toggle_led", messageCb );
std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);

char hello[13] = "hello world!";
```

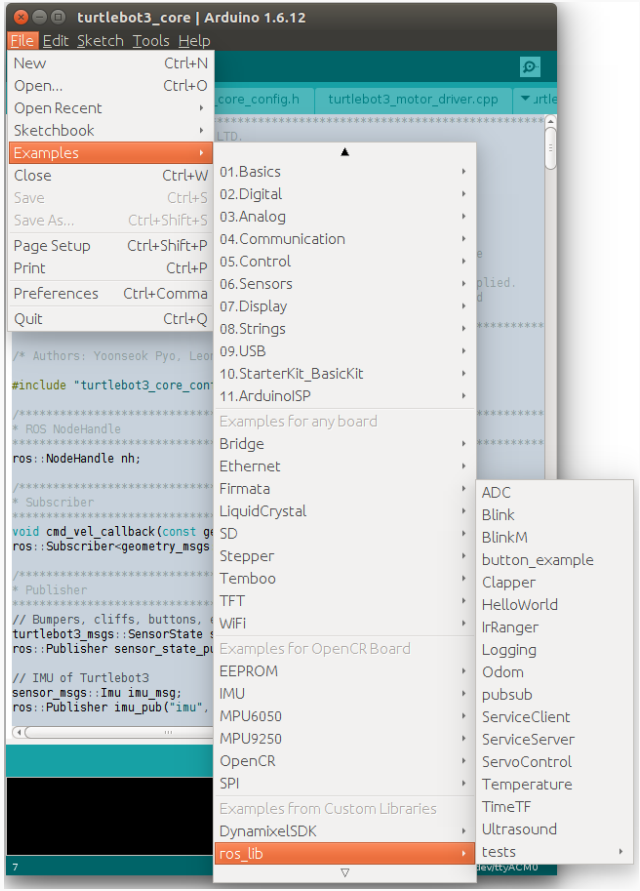
```
void setup() {
  pinMode(13, OUTPUT);
  nh.initNode();
  nh.advertise(chatter);
  nh.subscribe(sub);
}

void loop() {
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(500);
}
```

ros_libs

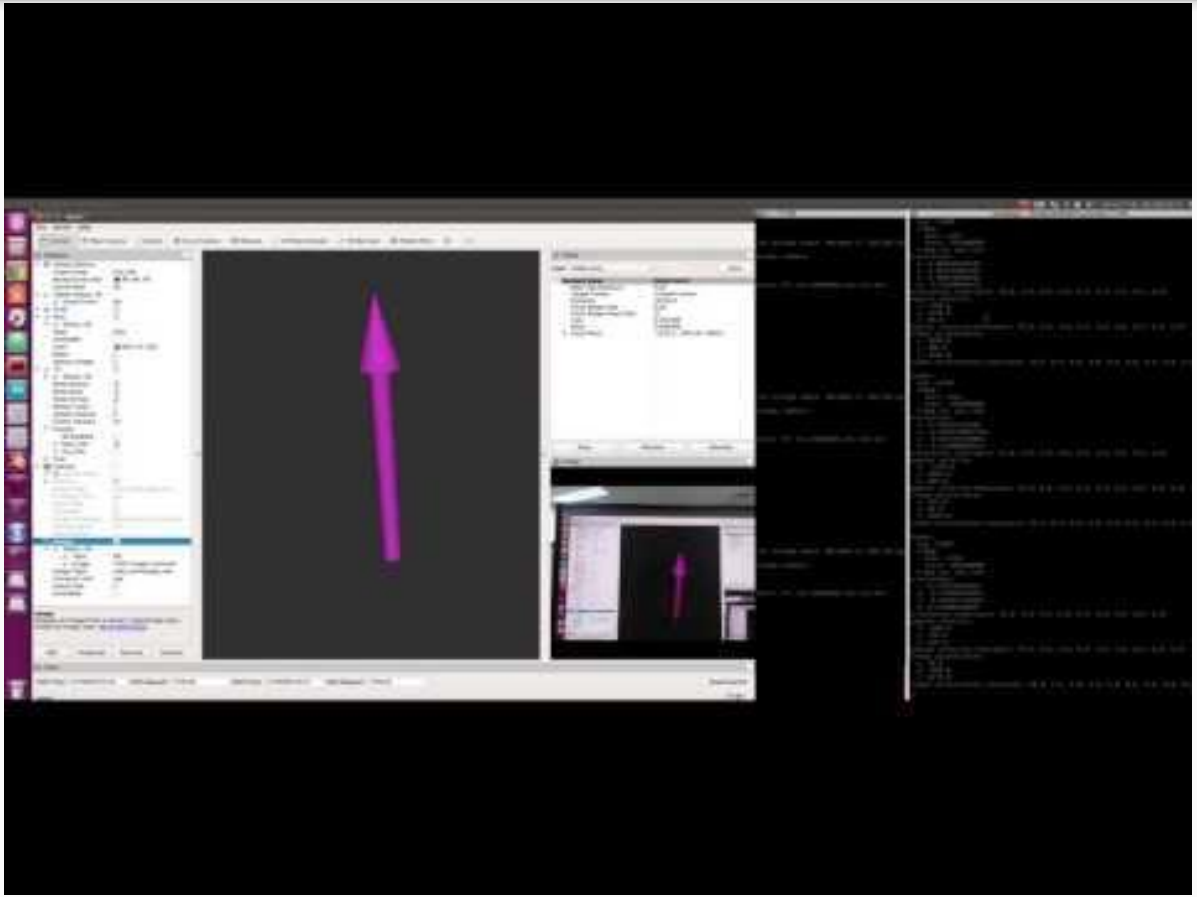
예제 프로그램(pubsub)이외에도

ADC, Blink, BlinkM, button_example,
Clapper, HelloWorld, IrRanger, Logging,
Odom, pubsub, ServiceClient, ServiceServer,
ServoControl, Temperature, TimeTF,
Ultrasound, tests

등이 준비 되어 있다.



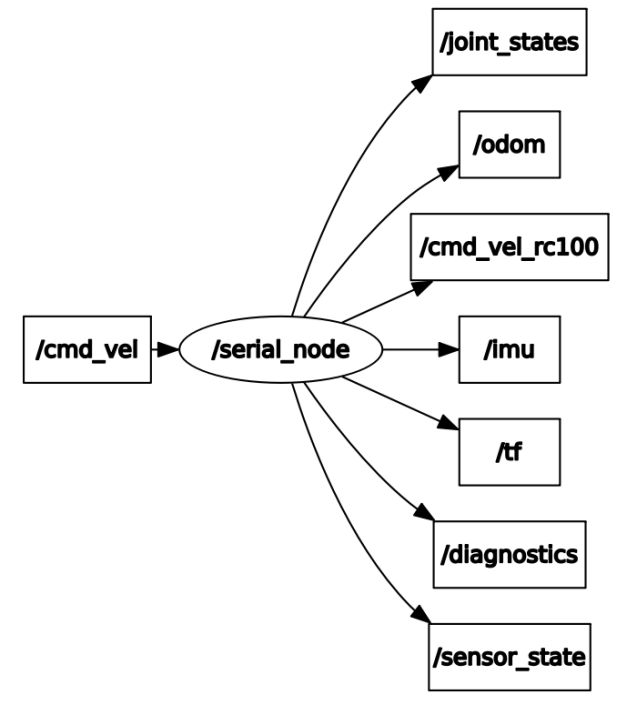
Practical examples: IMU on RViz

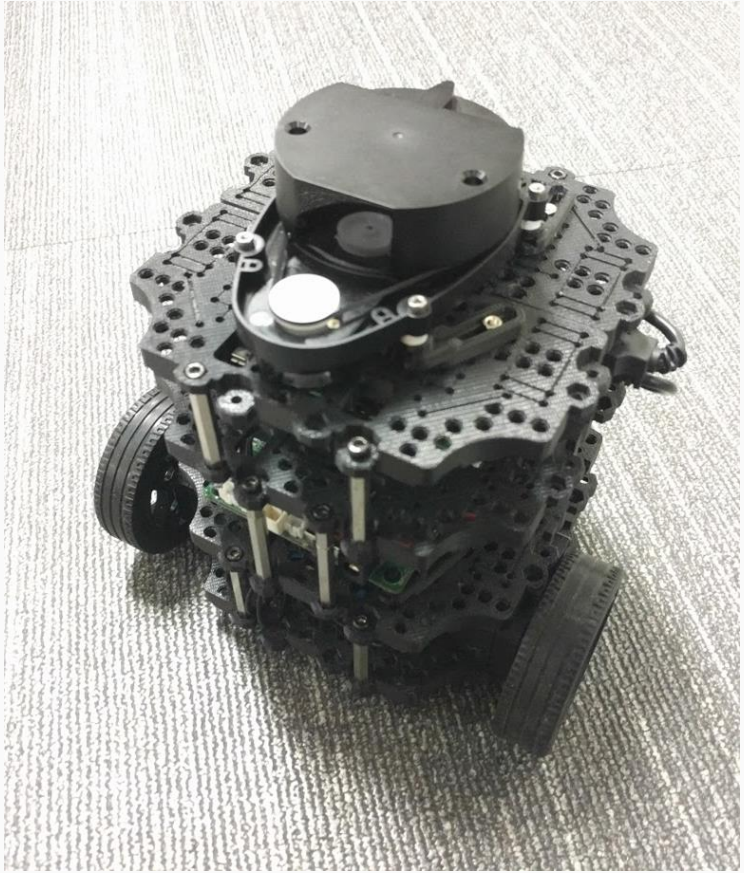


https://youtu.be/wXN_7oRHst0

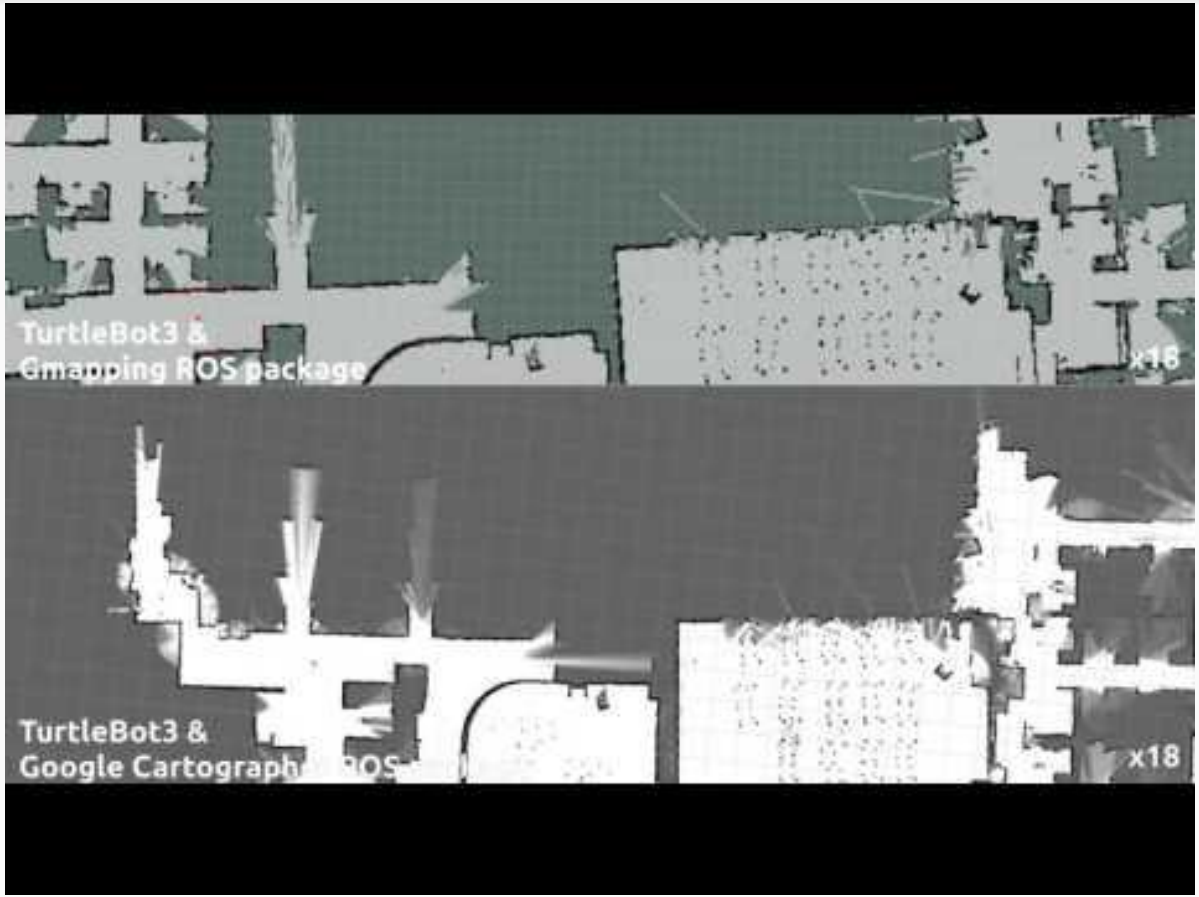
Practical examples: TurtleBot3 with OpenCR + rosserial

모바일 로봇에 사용할 때의 예제





Practical examples: TurtleBot3 with OpenCR + rosserial



https://youtu.be/lkW4-dG2BCY