# 드론 통신과 PID 튜닝

An **Unmanned aerial vehicle** (UAV) is a Unmanned Aerial Vehicle. UAVs include both autonomous (means they can do it alone) drones and remotely piloted vehicles (RPVs). A UAV is capable of controlled, sustained level flight and is powered by a jet, reciprocating, or electric engine.

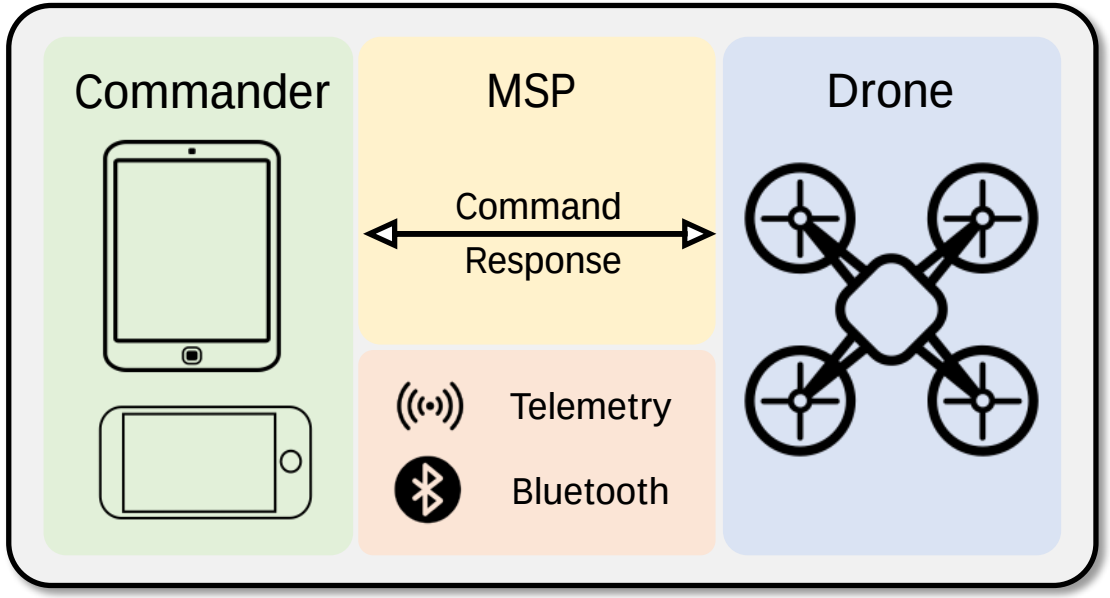

# Multi-Wii 프로토콜

# MSP 란?



- Multi-Wii
  - Wii 게임기용 Nunchuck의 센서를 이용하여 개발을 시작한 드론 오픈소스
- MSP(Multi-Wii Serial Protocol)
  - Multiwii 용 직렬통신 프로토콜
  - Tablet, Smartphone, PC와
    - 조정 명령 전송
    - 드론 상태 모니터링
    - 주요 설정
  - Bluetooth 또는 Telemetry로 이용

Nintendo Wii Nunchuck

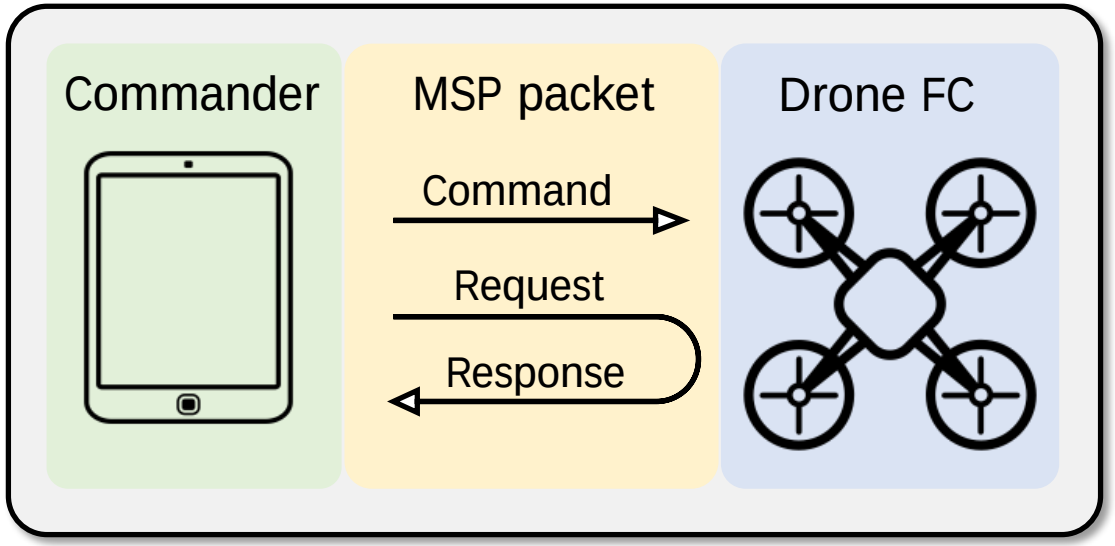


# MSP 개요



## MSP 패킷 종류

- 명령 전송(Command)
  - Commander → Drone FC
- 데이터 요청(Request)
  - Commander → Drone FC
- 응답(Response to Request)
  - Drone FC → Commander



# 명령 전송 패킷

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 명령 전송

- Commander → Drone FC
- 패킷

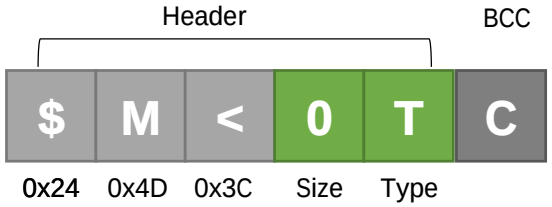

  - Header
    - $ M : preamble
    - < : 방향 Commander → 드론 표시
    - Size(N): Payload의 바이트 수
    - Type(T): 명령의 종류
  - BCC(Block Check Code): 데이터 무결성 확인, XOR사용

# 데이터 요청 패킷

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 데이터 요청 전송

- Commander → Drone FC
- 패킷




  - Header
    - $ M : preamble
    - < : 방향 Commander → 드론 표시
    - Size(N): Payload의 바이트 수
    - Type(T): 명령의 종류
  - BCC: 데이터 무결성 확인, XOR사용

# 응답 패킷

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 응답

- Drone FC → Commander
- 패킷


  - Header
    - $ M : preamble
    - > : 방향  드론 → Commander
    - Size(N): Payload의 바이트 수
    - Type(T): 명령의 종류
  - BCC(Block Check Code): 데이터 무결성 확인, XOR사용

# BCC 계산 방법

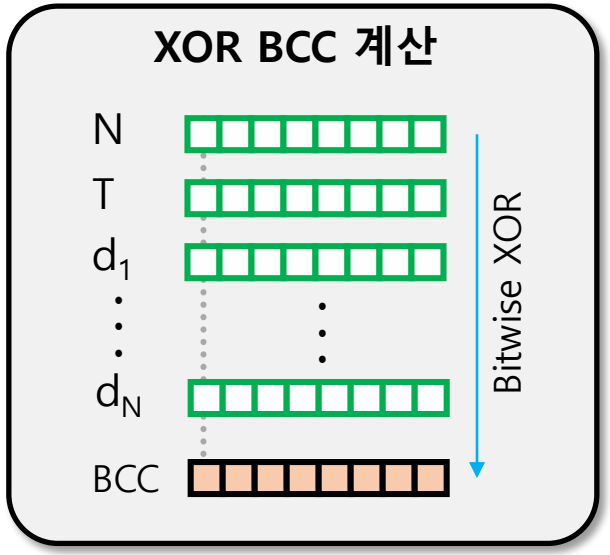
Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## XOR BCC

- 헤더와 BBC를 제외한 Size, Type, Payload 를 비트 단위 Excusive Or

C = bitwise XOR( Size N | Type T | Payload N -bytes )

- 오류 확인 방법
  - 계산 BCC와 수신된 BCC 비교
    - 일치하면 통신 에러 없다고 판단.



# 명령 타입

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 대표 명령 타입(T)

- 번호에 따른 구분
  - 100번대: 정보 요청
    - Response 있음.
    - ex: MSP_RAW_IMU
      MSP_RAW_GPS
  - 200번대: 명령 전송
    - Response 없음
    - ex: MSP_SET_RAW_RC
      MSP_SET_PID

| 요청 명령 | 번호 | 명령 전송 | 번호 |
|---|---|---|---|
| MSP_IDENT | 100 | MSP_SET_RAW_RC | 200 |
| MSP_STATUS | 101 | MSP_SET_RAW_GPS | 201 |
| MSP_RAW_IMU | 102 | MSP_SET_PID | 202 |
| MSP_SERVO | 103 | MSP_SET_BOX | 203 |
| MSP_MOTOR | 104 | MSP_SET_RC_TUNING | 204 |
| MSP_RC | 105 | MSP_ACC_CALIBRATION | 205 |
| MSP_RAW_GPS | 106 | MSP_MAG_CALIBRATION | 206 |
| MSP_COMP_GPS | 107 | MSP_SET_MISC | 207 |
| MSP_ATTITUDE | 108 | MSP_RESET_CONF | 208 |
| MSP_ALTITUDE | 109 | MSP_SELECT_SETTING | 210 |
| MSP_ANALOG | 110 | MSP_SET_HEAD | 211 |
| MSP_RC_TUNING | 111 | MSP_SET_SERVO_CONF | 212 |
| MSP_PID | 112 | MSP_SET_MOTOR | 214 |
| MSP_BOX | 113 | MSP_BIND | 241 |
| MSP_MISC | 114 | MSP_EEPROM_WRITE | 250 |
| MSP_MOTOR_PINS | 115 | MSP_DEBUGMSG | 253 |
| MSP_BOXNAMES | 116 | MSP_DEBUG | 254 |
| MSP_PIDNAMES | 117 | | |
| MSP_SERVO_CONF | 120 | | |

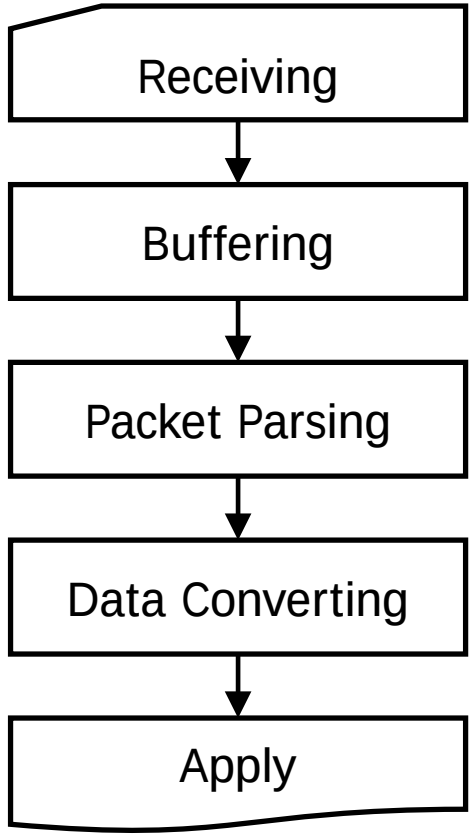

# MSP 수신 프로그래밍

# 수신 과정

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 주요 절차

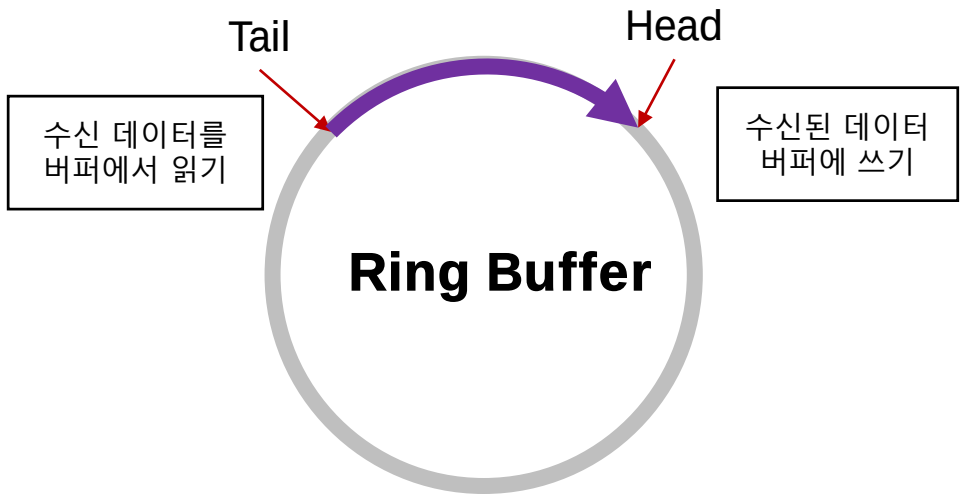
- **Buffering**:
  - ring 버퍼로 수신과 처리가 동시에 이뤄지지 않아도 일정 분량 데이터 보관.
  - serialBufferRX[] 에 저장
- **Packet Parsing**:
  - 패킷을 분석하여 데이터의 무결성을 검사하고 유용한 데이터 추출
  - inBuf[] 에 저장
- **Data Converting:**
  - Parsing 된 데이터를 원하는 변수에 맞게 형변환
  - rcSerial[] 에 저장



# Buffering

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- Buffer의 필요성
  - 수신과 처리가 동시에 일어날 때 유효
  - 처리가 늦어질 때 수신데이터가 싸여 Overrun 상태가 될 수 있음.
  - 유한한 바이트의 버퍼를 사용하여 원활한 처리가 가능하도록 Ring의 형태를 사용함.
  - 실제로는 1차원 배열로 구현

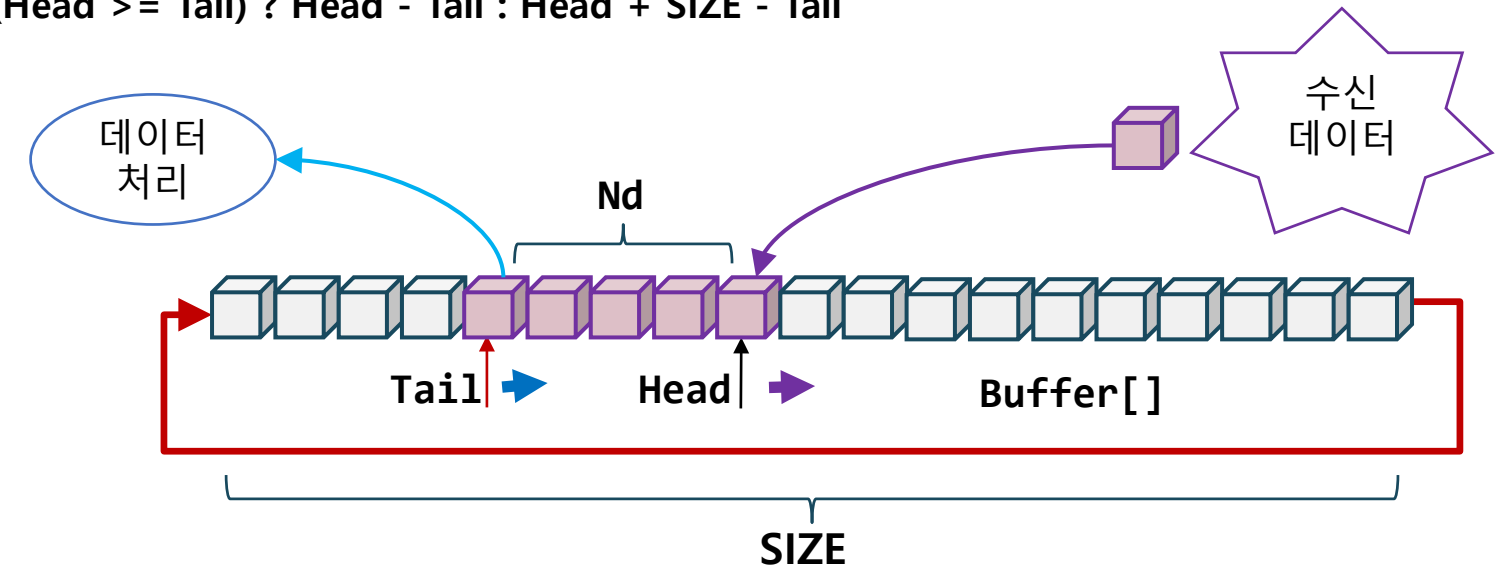



# Ring Buffer구현

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## Ring buffer의 구현

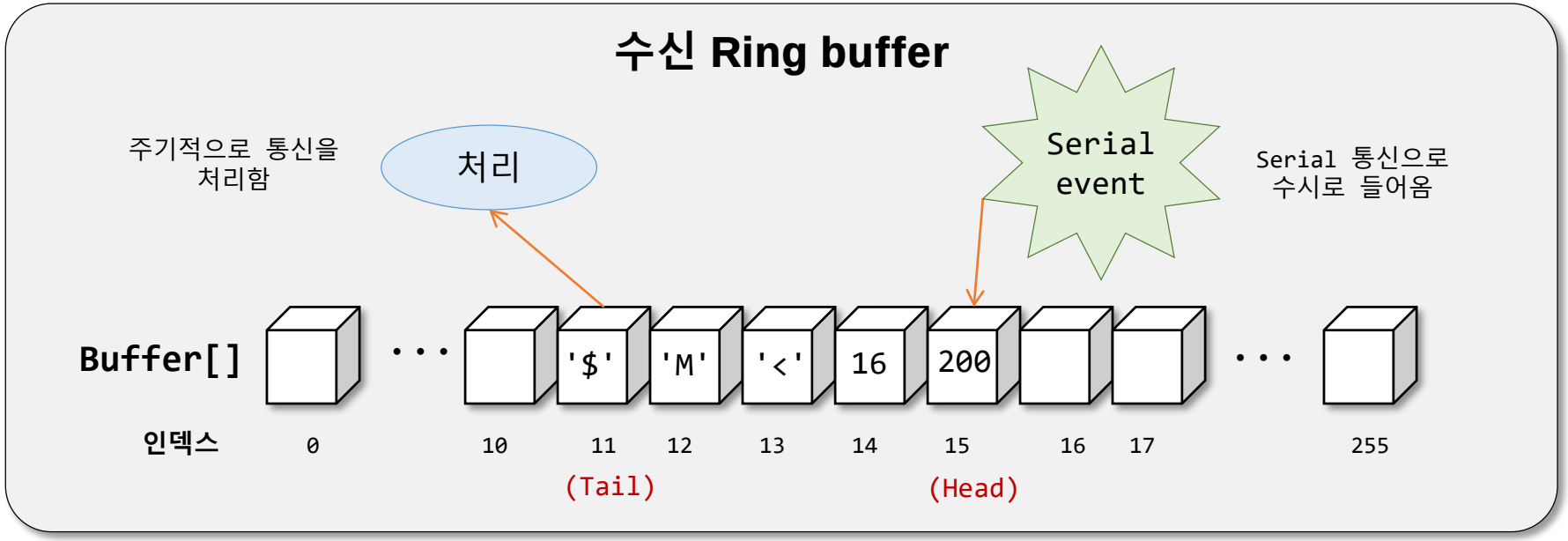
- 1차원 배열을 ring 형태로 사용하고자 Head와 Tail 인덱스를 적용
- 수신 데이터는 head에 놓고 head를 다음 자리로 옮김.
- 데이터 처리는 tail에서 읽고 tail을 다음 자리로 옮김.
- 인덱스는 배열 끝에 도달하면 다시 0으로 보냄.
- Head와 Tail의 차이, 즉 Nd 는 다음과 같이 구할 수 있음.
  - **Nd = (Head >= Tail) ? Head - Tail : Head + SIZE - Tail**



# Ring Buffer구현

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- Ring buffer에 쓰기
  - SerialEvent를 통해 수신 받은 데이터를 버퍼의 head 인덱스를 따라 차례대로 저장.
  - 인덱스는 끝에 도달하면 다시 0으로
- Ring buffer에 읽기
  - 버퍼로부터 데이터를 수신 받은 tail 인덱스로부터 순서대로 읽음.



# Ring Buffer에 쓰기 코드

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

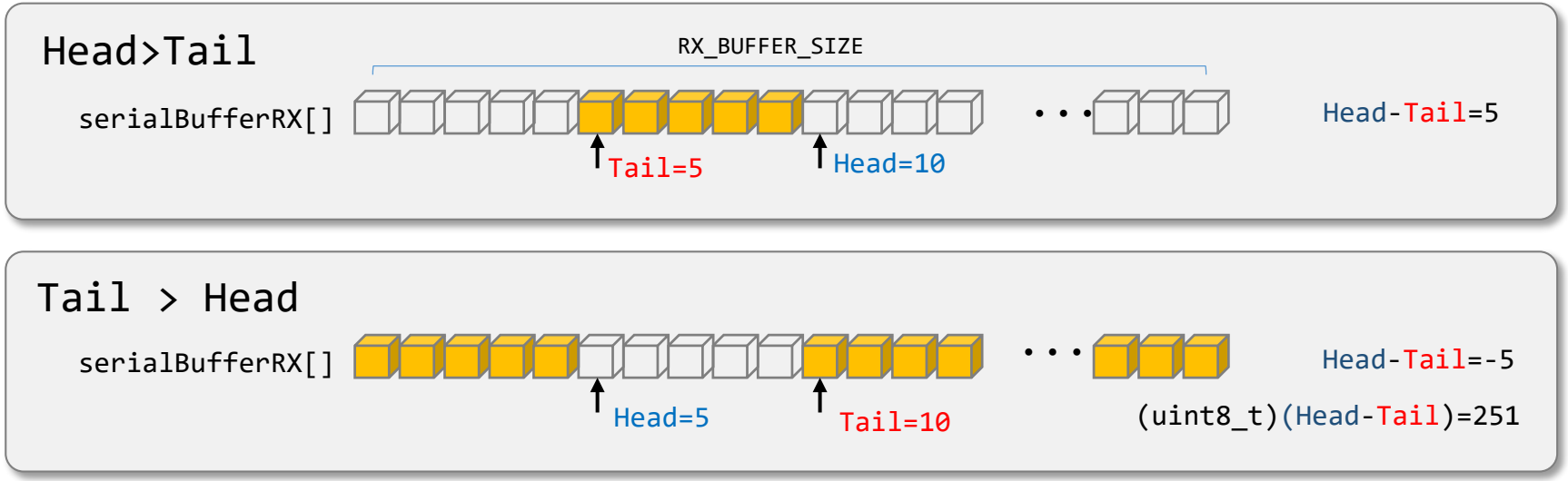
- SerialEvent 함수에 구현
  - 수신된 데이터를 링버퍼 serialBufferRX[]에 저장
  - 저장 후 serialHeadRX 를 1 증가
  - RX_BUFFER_SIZE 는 버퍼의 크기

```
void serialEvent(void) {
    // When data is arrived, keep it to the bufferRX[]
  while(Serial.available()) {          // available data >0
    uint8_t h = serialHeadRX;          // temp index h for Head index
    // store data to Buffer and increase index h
    serialBufferRX[h++] = (uint8_t)Serial.read();
    if(h >= RX_BUFFER_SIZE)  h = 0;    // to make ring buffer
    // keep h to seiralHeadRX for the next operations
    serialHeadRX = h;
  }             // repeat till available data ==0
}
```

# Ring Buffer 의 데이터 갯수

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- SerialAvailable[]
  - Head 와 Tail의 차이를 구하여 링 버퍼의 유효한 데이터 계산
  - 2가지 경우




```
uint8_t SerialAvailable(void) {
  return ((uint8_t)(serialHeadRX - serialTailRX)) % RX_BUFFER_SIZE;
  //check (a>=b)? a-b : a+SIZE-b;
}
```

# Ring Buffer 에서 1바이트 읽어내기

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

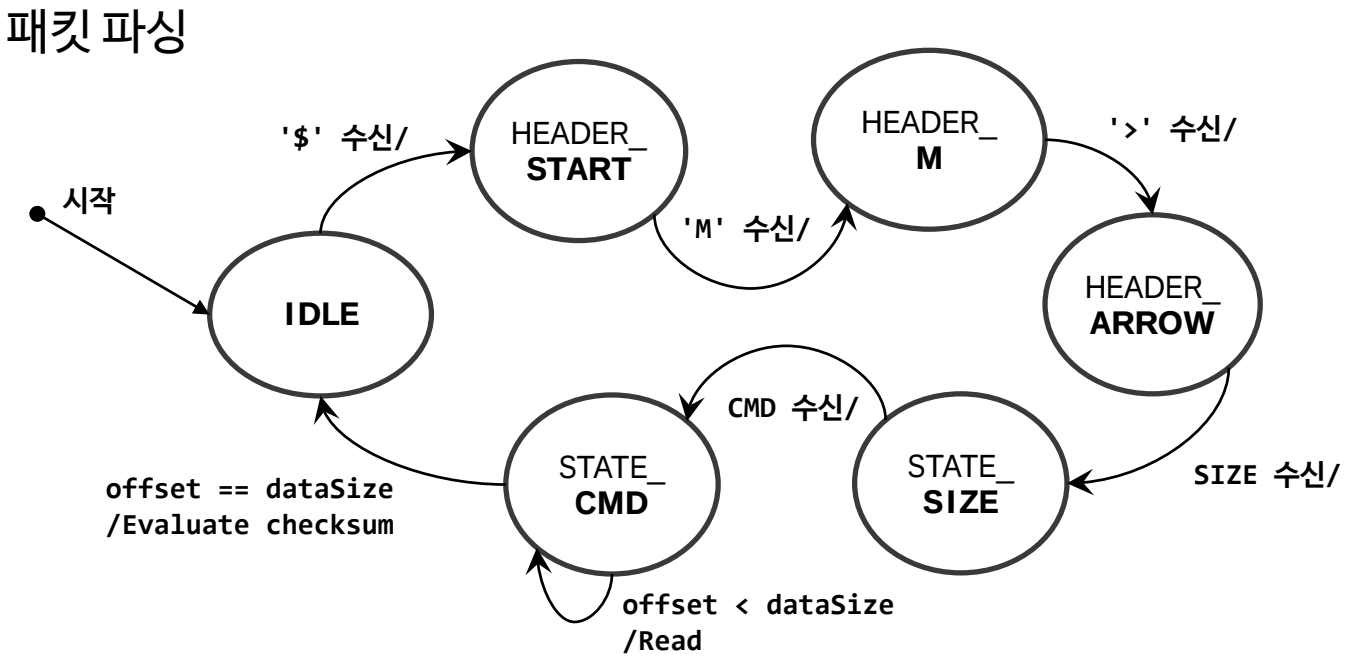
- SerialRead[]
  - 링 버퍼의 tail 에서 1바이트 읽어내기
  - head!=tail 이면
    - t를 1증가 시키고 t>=buffer_size 이면 t=0
      ← 배열에 끝에 도달하면 처음으로
  - head==tail 이면 처리 불가

```
uint8_t SerialRead(void)  {                 // read from seiralHeadRX[]
  uint8_t t = serialTailRX;                 // temp index h for Tail index
  uint8_t c = serialBufferRX[t];            // get a data
  if(serialHeadRX != t) { // to prevent Tail from exceeding Head
    if(++t >= RX_BUFFER_SIZE) t = 0 ;
    // increase 1 and if it exceeds buffer size, t=0 to make ring buffer
    serialTailRX = t ;                      // update seiralTailRX
  }
  return c;
}
```

# 패킷 파싱

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

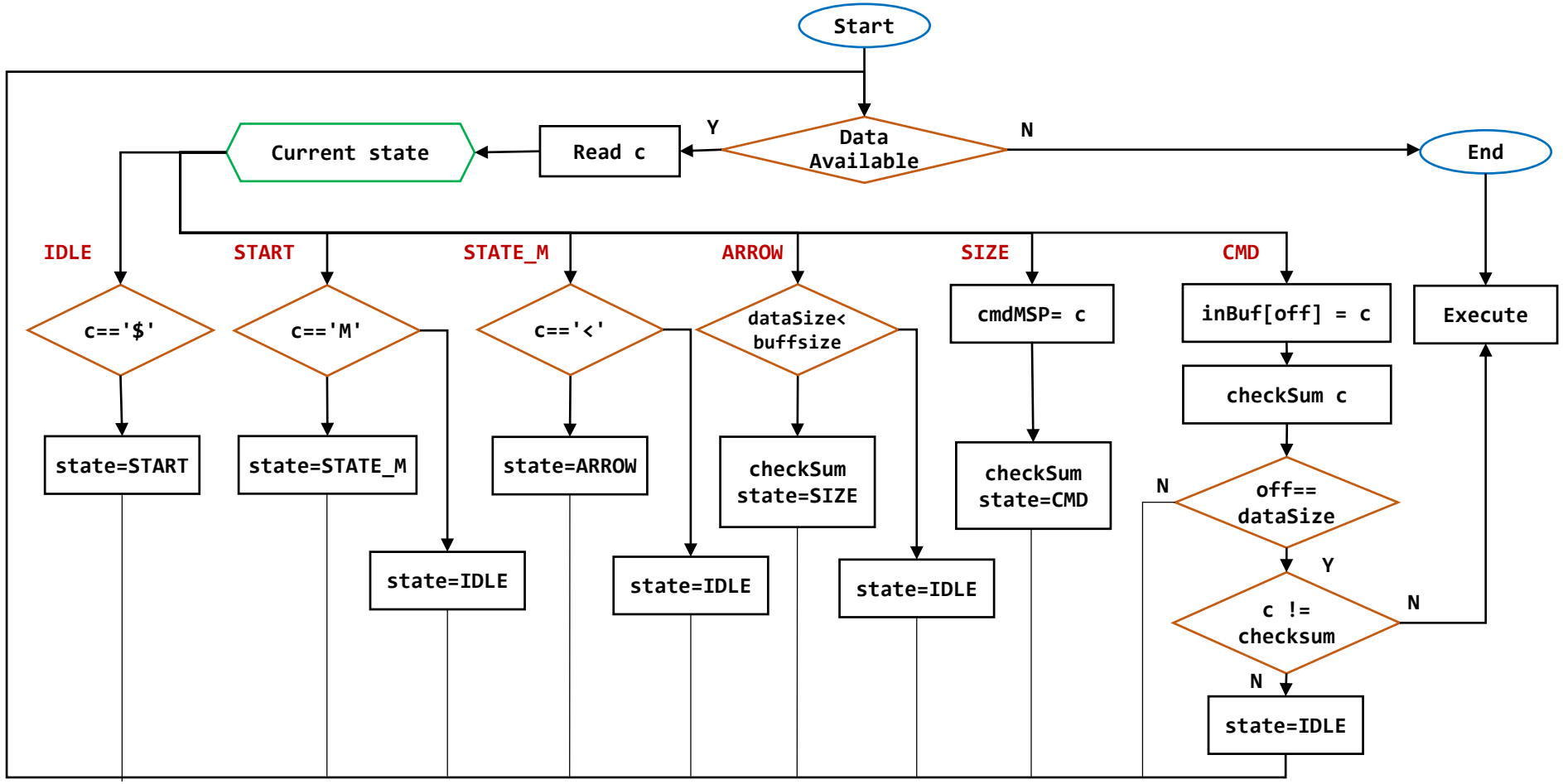
- 파싱에 State diagram 이용
  - 다음과 같은 상태를 정의
    - IDLE, HEADER_START, HEADER_M, HEADER_ARROW, HEADER_SIZE, HEADER_CMD



# 패킷 파싱

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 순서도



# 패킷 파싱 코드

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- serialCom 함수에 구현 [1/3]

```
void serialCom(void) {                          // parsing the sequence using protocol
  uint8_t c, availableData;                     // c =current data
  static uint8_t offset;                        // offset in inBuf
  static uint8_t dataLen;                       // Number of data to be recieved
  static uint8_t c_state;                       // current state sved
  uint8_t state;                                // tmep state of protocol
  availableData = SerialAvailable();            // read number of available data
  while(availableData--)  {                     // repeat till availableData=0
      c = SerialRead();                         // read a char
      state = c_state;                          // load current state
      if(state == IDLE) {
        if(c == '$')
          state = HEADER_START;                  // to the next state
      }
      else if(state == HEADER_START) {
        state = (c == 'M') ? HEADER_M : IDLE;
      }
      else if(state == HEADER_M) {
        state = (c == '<') ? HEADER_ARROW : IDLE;  }
```

# 패킷 파싱 코드

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- [2/3]

```
      else if(state == HEADER_ARROW) {    // dataLen received
        if(c > INBUF_SIZE) {             //  dataLen >INBUF_SIZE (64)
          state = IDLE;                  // give up
          continue;                      // exit  if statements
        }
        dataLen = c;                     // update dataLen with new date recieved
        checksum = c;                     // check sum start
        offset = 0;                      // make writing index of inBuf[] 0
        indRX = 0;                       // make reading index of inBuf[] 0
        state = HEADER_SIZE;
      }
      else if(state == HEADER_SIZE) {    // command received
        cmdMSP = c;                      // save command to cmdMSP
        checksum ^= c;                   // compute checksum (XOR)
        state = HEADER_CMD;              // next
      }
```

# 패킷 파싱 코드

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

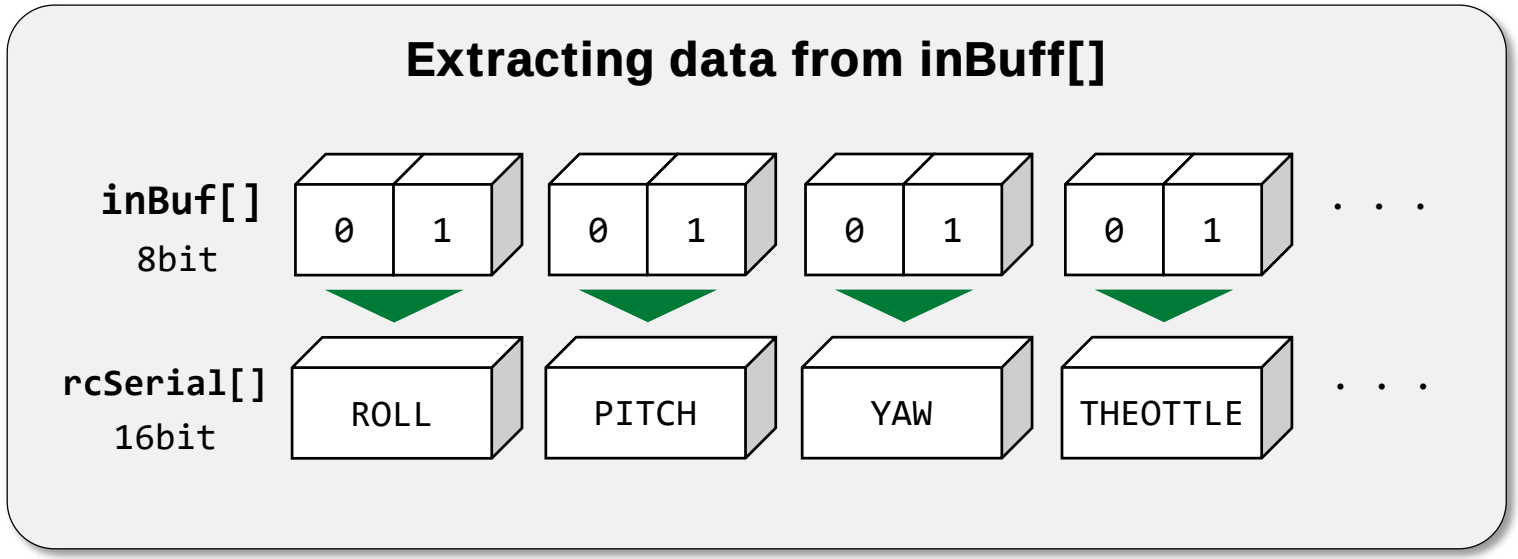
- [3/3]

```
    else if(state == HEADER_CMD) {
        if(offset < dataLen)   {          // updating data if not finished update
          checksum ^= c ;                 // compute checksum
          inBuf[offset++] = c;            // update inBuf
        }
        else {                            // finished
          if(checksum == c)               // checksum match !
            evaluateCommand(cmdMSP);      // execute cmdMSPcommand
          state = IDLE;                   // to the next
          availableData = 0;              // availableData=0 to exit while
        }
      }
      c_state = state;                    // keep cur state to c_state
  }
}
```

# 데이터 변환

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 8bit 배열 → 16bit 배열
  - uint8_t와 uint16_t 배열의 포인터를 이용



# 데이터 변환

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- evaluateCommand 함수에 구현

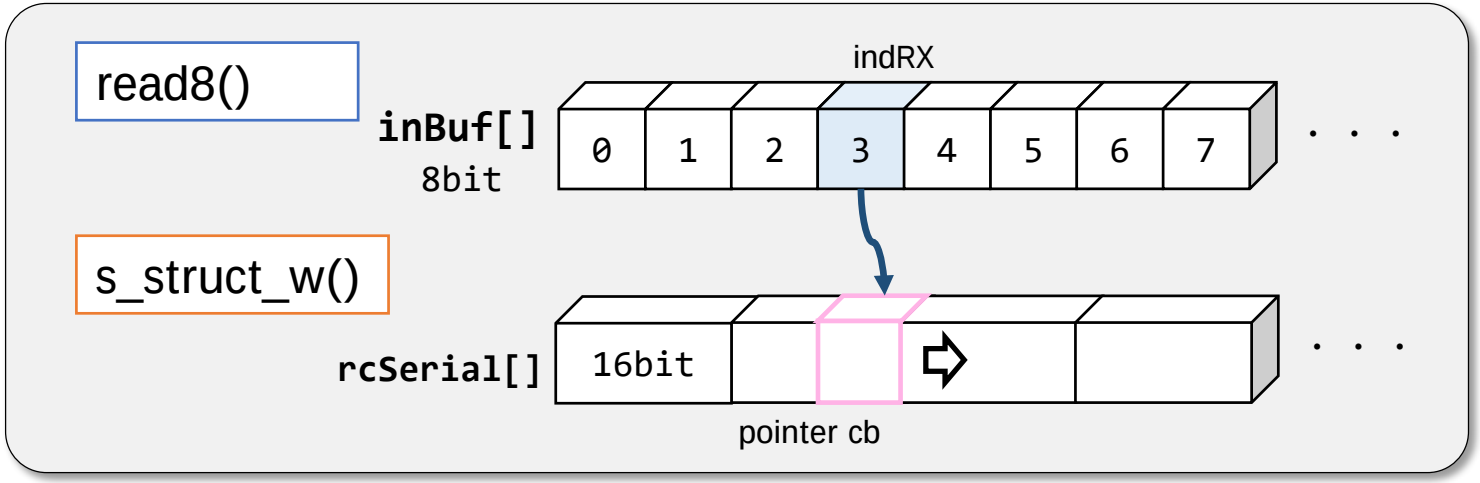
```
void evaluateCommand(uint8_t c)  {      // execute command
  switch(c){
    case MSP_SET_RAW_RC:                      // cmdMSP is Drone RC control command
      // update rcSerial[](16 bit)  by reading inBuf[] (8bit).
      s_struct_w((uint8_t*)&rcSerial, 16);
      // Causion:(uint8_t*)&rcSerial  -->
      // the start address of rcSerial in the form of uint8_t*
      Serial.print(rcSerial[ROLL]); Serial.print(",");
      Serial.print(rcSerial[PITCH]);Serial.print(",");
      Serial.print(rcSerial[YAW]);Serial.print(",");
      Serial.println(rcSerial[THROTTLE]);
      break;
    default:                                  // any other commands
      break;
  }
}
```

# 데이터 변환

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 형 변환

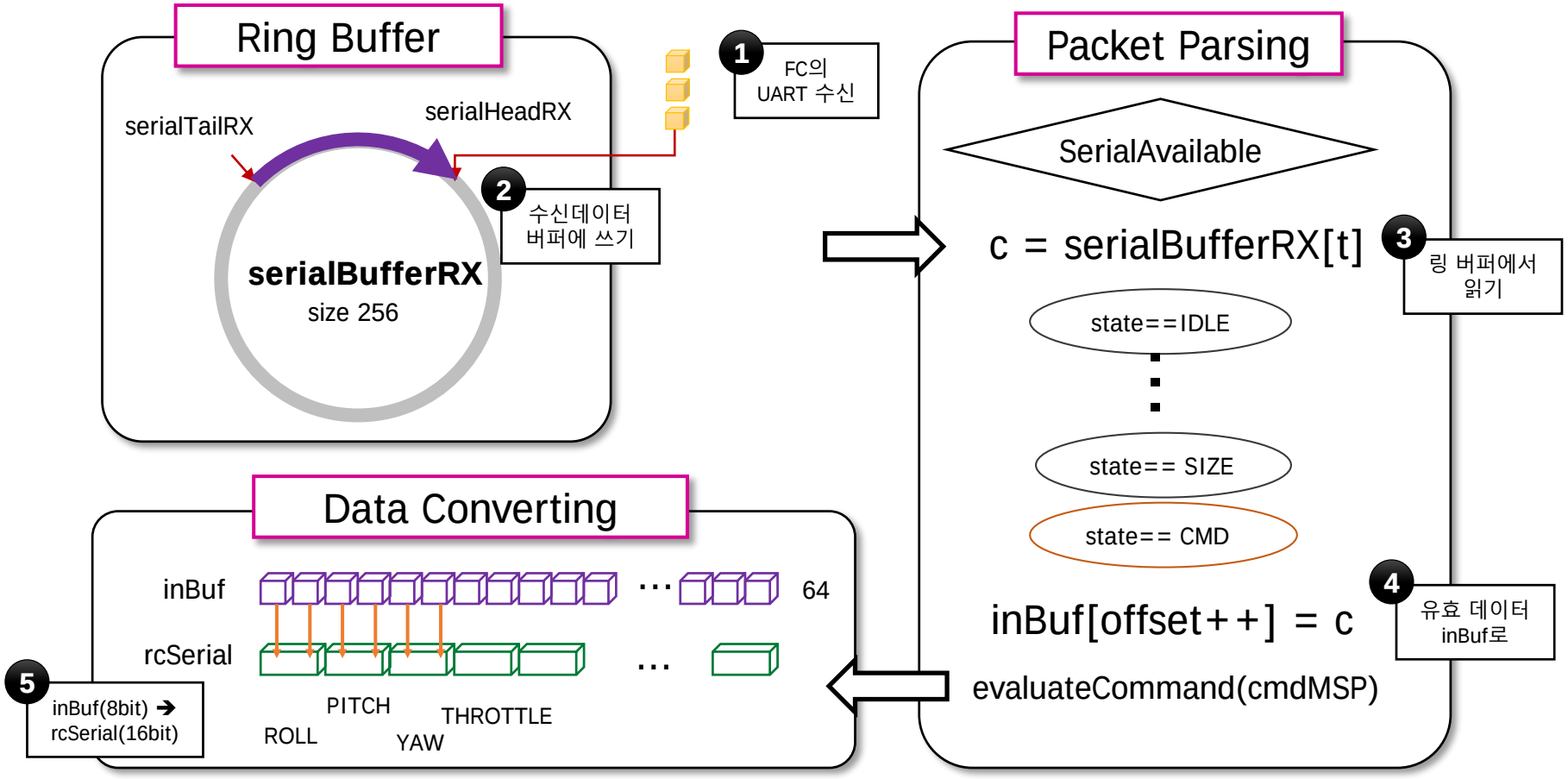
```
static void s_struct_w(uint8_t *cb,uint8_t siz){
  while(siz--)
    *cb++ = read8();
}
static uint8_t read8(void) {
  return inBuf[indRX++]&0xff;
}
```



# 수신 과정 정리

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 데이터 흐름

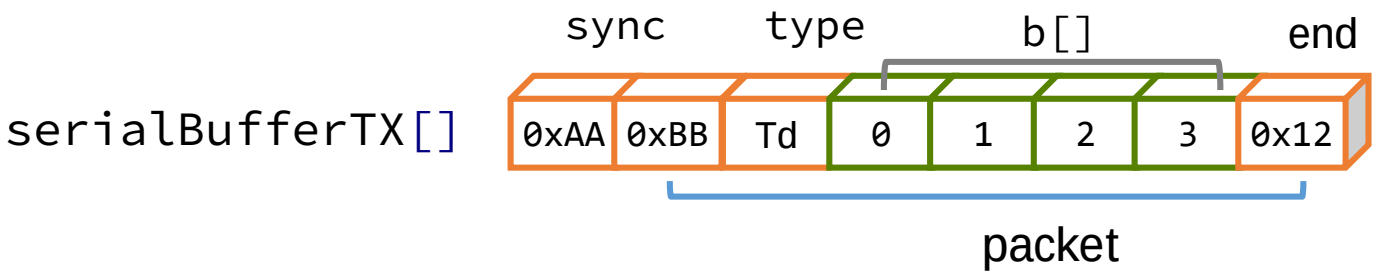



# 송신 프로그래밍

# 독자 송신 프로토콜

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

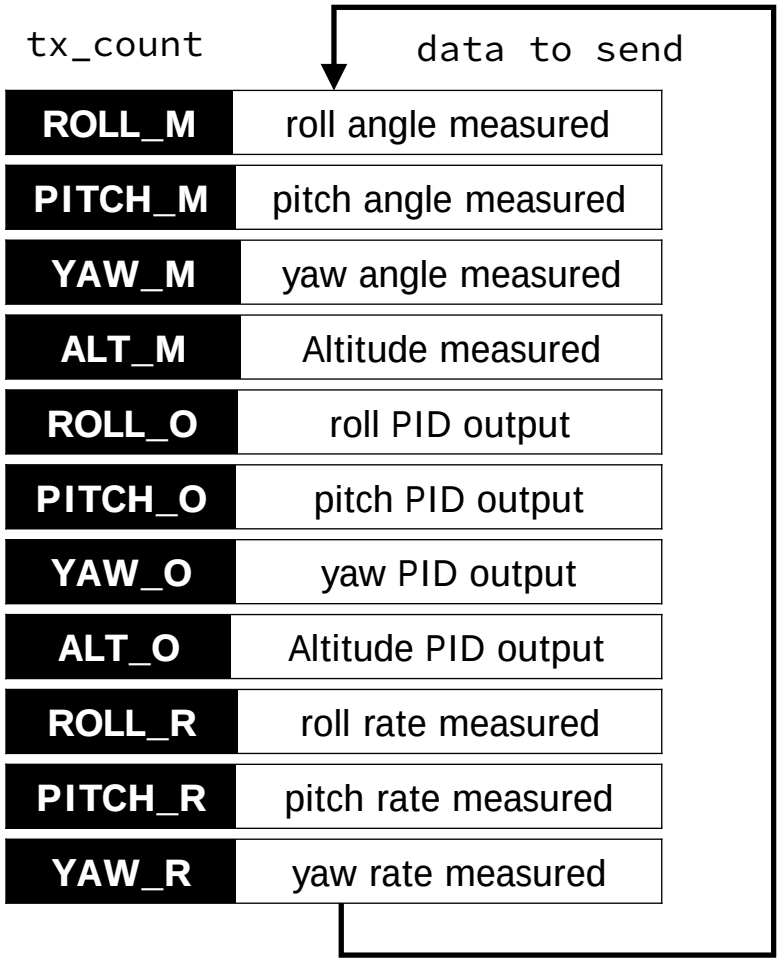
- 송신 프로토콜 필요
  - 컴퓨터에서 PID 튜닝을하려면 내부 계측 값들 실시간으로 읽어야 함
  - Multiwii의 프로토콜의 경우  요청하면 보내는 방식이므로 느림
  - 요구되는 각도, 각속도 등 정보량이 많기 때문에 순차적 전송
- 프로토콜 설정
  - 1개의 float 값을 4개의 unint8_t 배열 b[] 로 전송



# 데이터 전송 방법

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 방법
  - 매 샘플 마다 한꺼번에 모든 데이터를 전송하면 부하가 많이 걸림.
  - 일련 번호를 주고 샘플 마다 하나씩 전송
    - enum 을 이용 11개의 상태를 만듬.
      - ROLL_M, PITCH_M, YAW_M, ALT_M, ROLL_O, PITCH_O, YAW_O, ALT_O, ROLL_R, PITCH_R, YAW_R
    - 불필요한 내용은 이동하여 시간 단축 가능



# 데이터 전송

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- sendTXdata()함수에 sin,cos 파형 전송구현

```
enum  Packet_n{ROLL_M, PITCH_M, YAW_M, ALT_M, ROLL_O, PITCH_O, YAW_O, ALT_O,
                ROLL_R, PITCH_R, YAW_R };
void Multiwii::sendTXdata(){
  float tx_data;
  static uint16_t k=0;
  switch(tx_count){
    case ROLL_M:
      tx_data=  20.* sin(0.01*k);k++; break; // test function
    case PITCH_M:
      tx_data=20.* sin(0.03*k);k++; break;   // test function
    case YAW_M:
      tx_data=45.* cos(0.03*k);k++; break;   // test function
...
   }
   sendPacket(tx_count, tx_data);
   tx_count++;
   if (tx_count>YAW_M) tx_count=ROLL_M;
}
```

# 데이터 전송

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

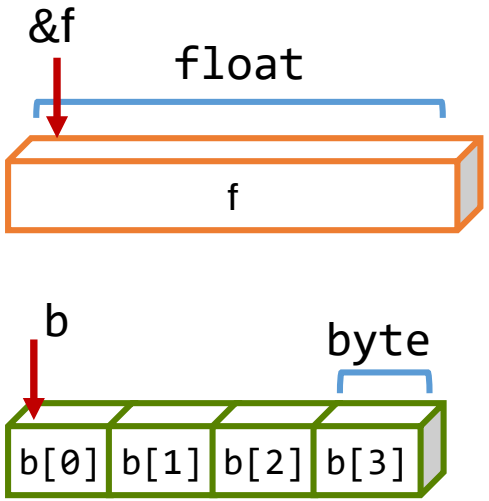
- sendPacket()함수에 구현

```
void Multiwii::sendPacket(byte n, float f) {
  byte* b = (byte *) &f;
  serialBufferTX[2]=n;
  for(int i=0;i<4;i++)
    serialBufferTX[i+3]=b[i];
   NeoSerial.write(serialBufferTX,8);
}
```

# 데이터 전송

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

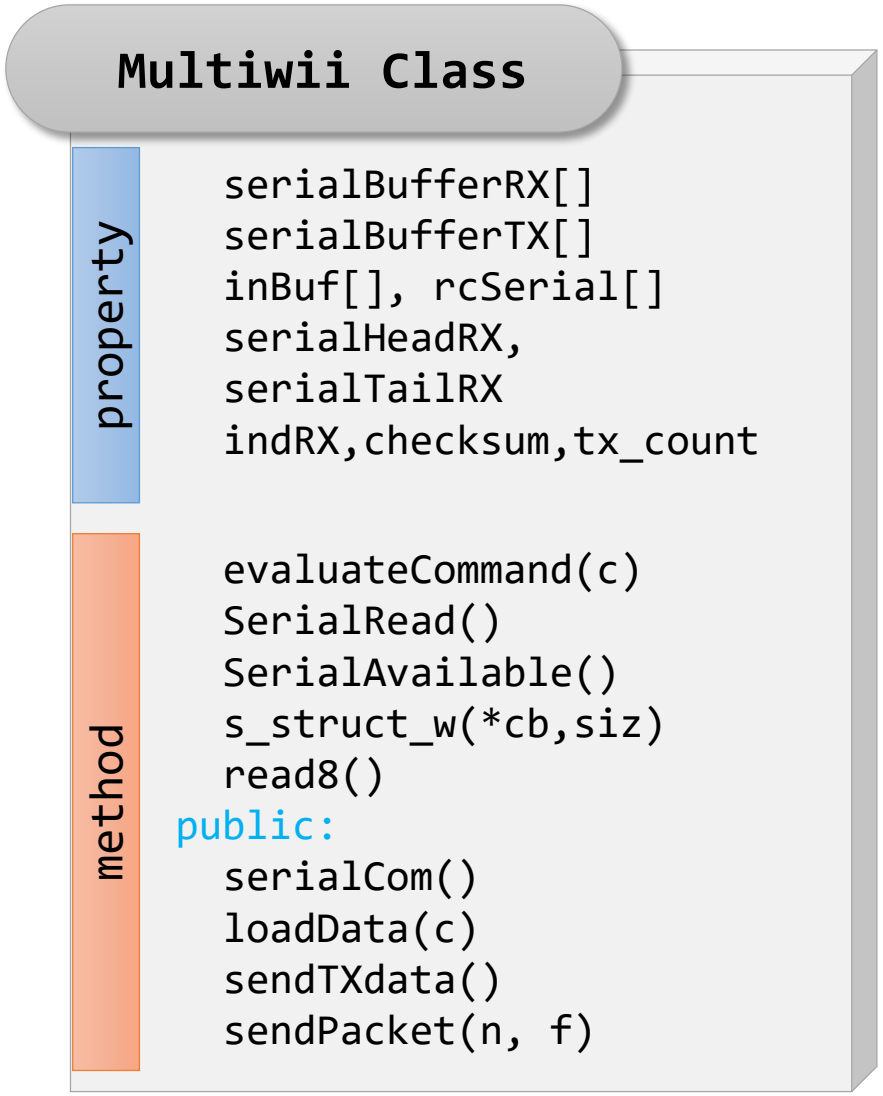
- `byte* b = (byte *) &f;`
  - f의 주소를 byte의 주소로 바꾸어 byte 포인터 b에 대입
  - 그러므로 b가 가리키는 곳은 f의 주소가 됨.
  - 포인터 b는 byte 배열처럼 쓸 수 있게 됨.
  - union 을 사용해도 같은 결과



# Multiwii: Class로 만들기

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- Multiwii class 정의
  - Property
    - serialBufferRX[]: 수신 링버퍼
    - inBuf[]: 패킷 파싱의 결과
    - rcSerial[]: inbuf에서 변환된 16비트 데이터
  - Method
    - evaluateCommand(c): 커멘드 종류 분석
    - SerialRead(): 링버퍼에서 데이터 읽기
    - SerialAvailable(): 링버퍼의 유효 데이터 갯수 읽기
    - serialCom(): 패킷 파싱
    - loadData(c): 링 버퍼에 수신 데이터 입력



# Multiwii Class 정의

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

SerialLib.h [1/2]

```
#include <Arduino.h>
#include <NeoHWSerial.h>
#define RX_BUFFER_SIZE 256                      // size of serialBuffRX
#define TX_BUFFER_SIZE 8
#define INBUF_SIZE 64                           // size of inBuf
#define MSP_SET_RAW_RC 200                      // control command
enum  Packet_n{ROLL_M, PITCH_M, YAW_M, ALT_M,
               ROLL_O, PITCH_O, YAW_O, ALT_O,
               ROLL_R, PITCH_R, YAW_R   };
class Multiwii{
    enum  MSP_protocol_bytes{IDLE, HEADER_START, HEADER_M,
       HEADER_ARROW, HEADER_SIZE, HEADER_CMD}; //state of protocol
    enum rcData{ROLL, PITCH, YAW, THROTTLE, AUX1, AUX2, AUX3,
       AUX4, AUX5, AUX6, AUX7, AUX8};   //contents of pyload
    static uint8_t inBuf[INBUF_SIZE];                   // internal buffer
    static uint8_t indRX;                               // index for reading inBuf
    static uint8_t cmdMSP, checksum;;
    static int16_t rcSerial[8];                         // vales (범위: 1000 ~ 2000)
```

# Multiwii Class 정의

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

SerialLib.h [2/2]

```
    static volatile uint8_t serialHeadRX, serialTailRX; // Head,Tail of Ring buffer
    static uint8_t serialBufferRX[RX_BUFFER_SIZE];      // Ring buffer
    static uint8_t serialBufferTX[TX_BUFFER_SIZE];
    static int tx_count;
    void evaluateCommand(uint8_t c);                    // 명령어 추출 함수
    uint8_t SerialRead(void);
    uint8_t SerialAvailable(void);
    static void s_struct_w(uint8_t *cb, uint8_t siz); // 8비트 데이터를 16비트 배열에 저장
    static uint8_t read8(void);                       // inBuf배열의 데이터를 8비트로 읽기
  public:
    void serialCom(void);                             // 패킷 파싱 함수
    void loadData(uint8_t c);
    void sendTXdata();
    void sendPacket(byte n, float f);
};
```

# Multiwii 아두이노

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- Multiwii_test.ino

Multiwii_test.ino

```
#include "serialLib.h"
#define ARDUINO_PLLOTER
#define SERIAL_COM_SPEED 115200                    // 115,200 bps
uint32_t prevTime,intVal=5000;
Multiwii mw;
void setup(void){
  NeoSerial.begin(SERIAL_COM_SPEED);
  NeoSerial.attachInterrupt(uartInt);
  prevTime=micros();
}
void loop(void){
  uint32_t curTime=micros();
  while (curTime-prevTime<intVal) curTime=micros();
  prevTime=curTime;
  mw.serialCom();
  mw.sendTXdata();
}
void uartInt(uint8_t c) {
  mw.loadData(c);
}
```


# PID 튜닝 준비

# PID 튜닝을 위한 구성



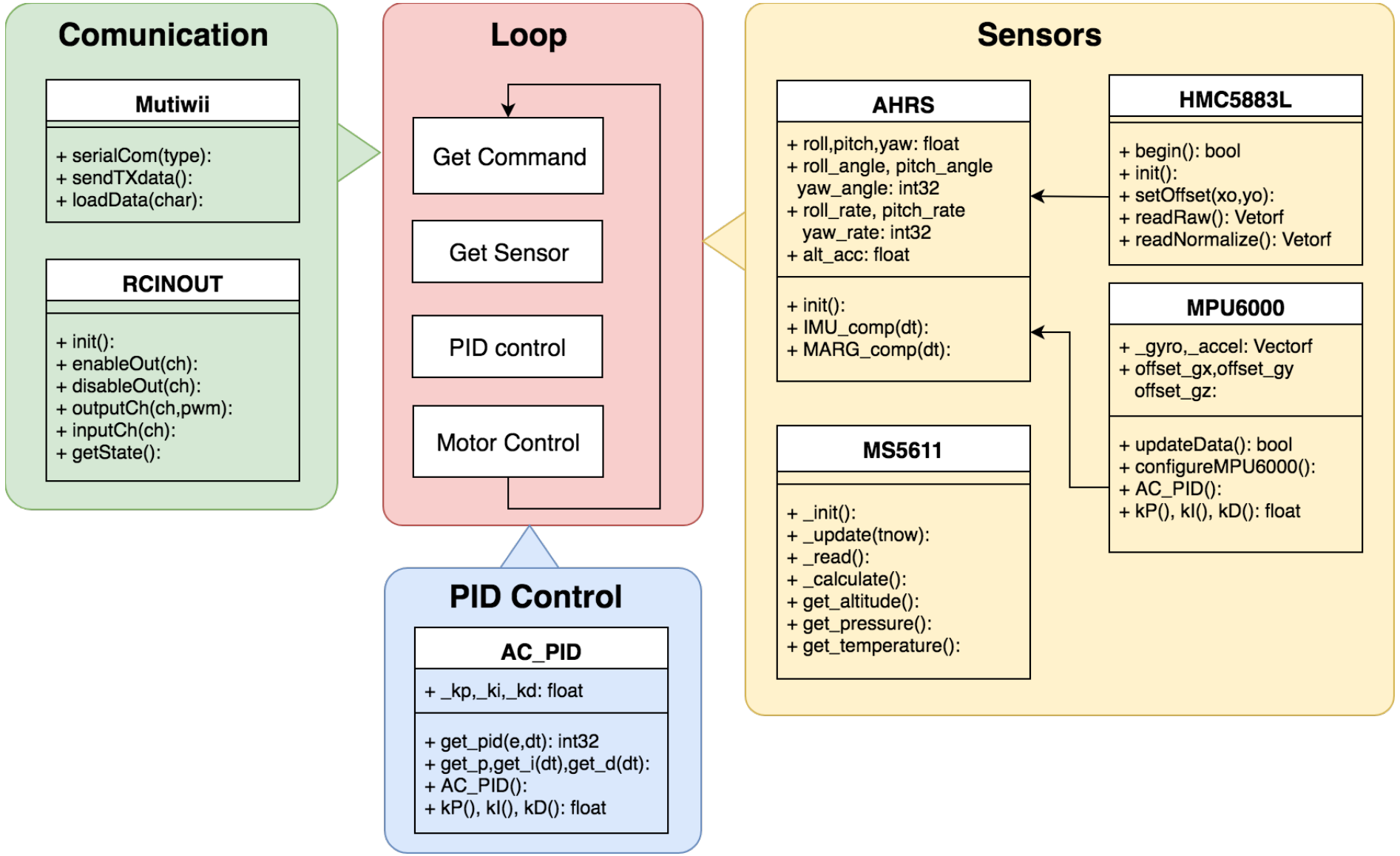
## PID 튜닝을 위한 객체

- Multiwii
  - 실시간 튜닝을 위한 Commander로부터 PID Gain 수신
- RCINOUT
  - RC 송신기로부터의 커맨드를 수신
  - 각 모터의 속도 지령 PWM 발생
- AC_PID
  - roll, pitch, yaw 각도 및 고도에 대한 PID 알고리즘 적용
- AHRS
  - MPU6000 과 HMC5883L로 부터 현재의 각도, 각속도, 가속도 정보를 얻음
- MS5611
  - Barometer로부터 고도 정보를 얻음

# PID 튜닝을 위한 구성

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 주요 Class
  - Multiwii
  - RCINOUT
  - AC_PID
  - AHRS
  - MS5611

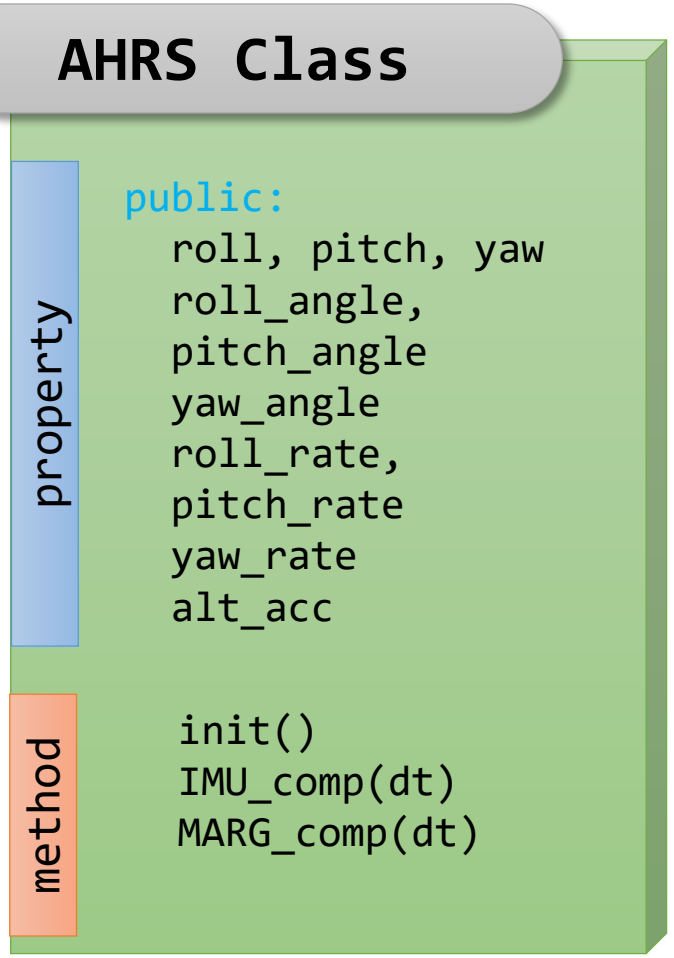


# AHRS Class 만들기

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## AHRS Class

- Property
  - roll, pitch, yaw : radian 단위의 각도
  - roll_angle, pitch_angle, yaw_angle:
    - degree x100 의 32bit 정수
  - roll_rate, pitch_rate, yaw_rate:
    - degree/sec x100 의 32bit 정수
  - alt_acc: m/s² 단위의 z 방향 가속도
- Method
  - init(): 초기화 및 설정
  - IMU_comp(dt):
    - IMU 만으로 roll, pitch를 구하는 필터 계산
  - MARG_comp(dt):
    - Magnetometer를 포함하여 roll,pitch,yaw 모두 구함



# AHRS Class 헤더

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- Class 정의

AHRS.h

```
#ifndef AHRS_H
#define AHRS_H
#include "MPU6k.h"
#include "HMC5883L.h"
#include "common.h"
#include <SPI.h>
#define ALPHA 0.92
class AHRS{
public:
  void IMU_comp(float dt);
  void MARG_comp(float dt);
  void init();
  float  roll, pitch,  yaw;
  float  alt_acc;
  int32_t   roll_angle, pitch_angle, yaw_angle;
  int32_t   roll_rate, pitch_rate, yaw_rate;
};
#endif
```

# AHRS Class C++ file

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## Class 구현

AHRS.cpp [1/3]

```
#include "AHRS.h"
extern MPU6000 mpu;
HMC5883L hmc;
void AHRS::init(){
  pinMode(BARO_CS_PIN, OUTPUT);  pinMode(12, OUTPUT);
  pinMode(MPU6K_CS_PIN, OUTPUT); digitalWrite(MPU6K_CS_PIN, HIGH);
  digitalWrite(BARO_CS_PIN, HIGH);   //Deselect Barometer
  SPI.begin();
  SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));delay(100);
  mpu.configureMPU6000();  // configure chip
  while (!hmc.begin()) delay(500);
  hmc.init();
}
void AHRS::IMU_comp(float dt){
  mpu.updateData( );
  Vectorf m = hmc.readNormalize();
  float accAngleX=-atan2(mpu._accel.y,-mpu._accel.z);
```

# AHRS Class C++ file

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- Class 구현

AHRS.cpp [2/3]

```
  float accAngleX=-atan2(mpu._accel.y,-mpu._accel.z);
  float srtAccXY=sqrt(mpu._accel.y*mpu._accel.y+mpu._accel.z*mpu._accel.z);
  float accAngleY=atan2(mpu._accel.x,srtAccXY);
    // complementary filter roll, pitch
  roll=ALPHA* (roll+mpu._gyro.x*dt)+(1-ALPHA)*accAngleX;
  pitch=ALPHA* (pitch+mpu._gyro.y*dt)+(1-ALPHA)*accAngleY;
  roll_angle=roll*DEG100;  pitch_angle=pitch*DEG100;
  roll_rate=mpu._gyro.x*DEG100;   pitch_rate=mpu._gyro.y*DEG100;
  yaw_rate=mpu._gyro.z*DEG100;  alt_acc=-mpu._accel.z-9.1;
}
 void AHRS::MARG_comp(float dt){
   mpu.updateData( );
  Vectorf m = hmc.readNormalize();
  float accAngleX=-atan2(mpu._accel.y,-mpu._accel.z);
  float srtAccXY=sqrt(mpu._accel.y*mpu._accel.y+mpu._accel.z*mpu._accel.z);
  float accAngleY=atan2(mpu._accel.x,srtAccXY);
```

# AHRS Class C++ file

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## Class 구현

AHRS.cpp [2/3]

```
  // complementary filter roll, pitch
  roll=ALPHA* (roll+mpu._gyro.x*dt)+(1-ALPHA)*accAngleX;
  pitch=ALPHA* (pitch+mpu._gyro.y*dt)+(1-ALPHA)*accAngleY;
  // magnetometer --> heading angle
  float xd=-m.y, yd=m.x, zd=-m.z;// internal HMC
  //compensate heading angle to remove roll, pitch effects
  float s_r=sin(roll), c_r=cos(roll),s_p=sin(pitch),c_p=cos(pitch);
  float xh=xd*c_p+yd*s_r*s_p+zd*c_r*s_p;
  float yh=yd*c_r+zd*s_r;
  float heading = atan2(yh, xh);
  // complementary filter yaw
  yaw=ALPHA*(yaw+mpu._gyro.z*dt)+(1-ALPHA)*heading;
  roll_angle=roll*DEG100;   pitch_angle=pitch*DEG100;
  yaw_angle=yaw*DEG100;   roll_rate=mpu._gyro.x*DEG100;
  pitch_rate=mpu._gyro.y*DEG100;   yaw_rate=mpu._gyro.z*DEG100;
  alt_acc=-mpu._accel.z-9.1;
}
```

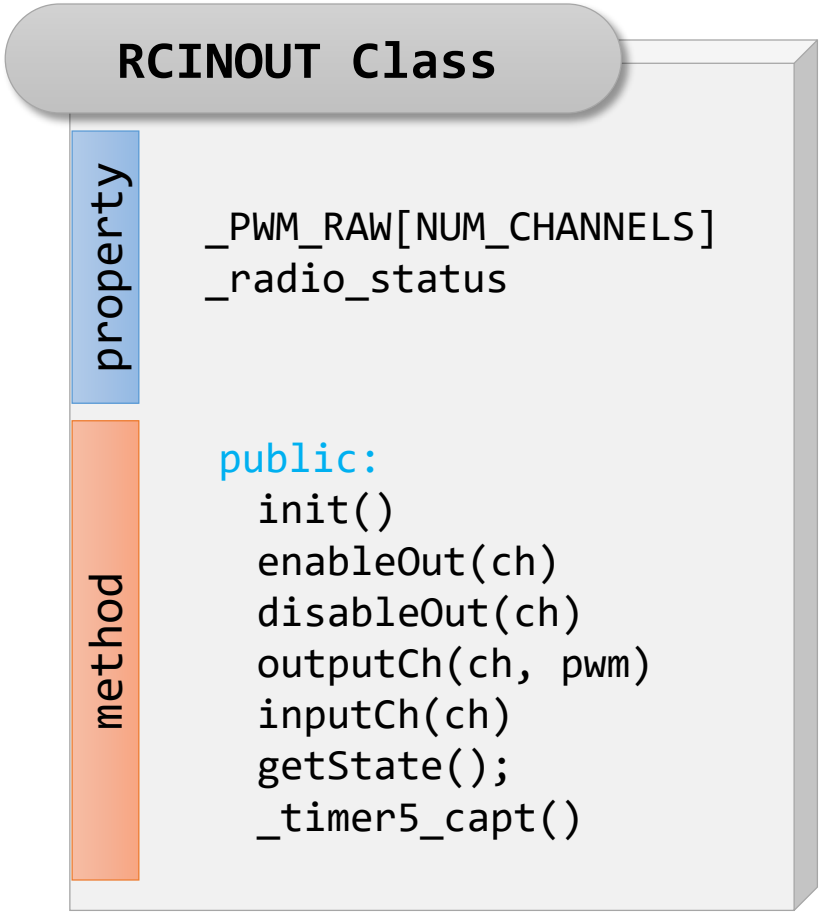

# RC Input Output Class

# RCINOUT class

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## RCINOUT Class

- Property
  - _PWM_RAW[NUM_CHANNELS]: PWM측정값을 보관
  - _radio_status: 측정 frame 완성 flag
- Method
  - init(): 타이머 등 설정
  - enableOut(ch): 해당채널의 PWM출력을 활성화
  - disableOut(ch): 해당채널의 PWM출력을 비활성화
  - inputCh(ch): 해당채널의 PWM 값 반환
  - getState(): 채널 데이터 읽기 완료면 True 반환
  - _timer5_capt(): Timer5 캡춰



# 헤더파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 주요 내용
  - 클래스 선언

rcInOut.h

```
#ifndef __RCINOUT_H__
#define __RCINOUT_H__
#define NUM_CHANNELS 8
#define MIN_PULSEWIDTH 900
#define MAX_PULSEWIDTH 2100
#include <Arduino.h>
class RCINOUT{
   volatile uint16_t _PWM_RAW[NUM_CHANNELS]
          = {2400,2400,2400,2400,2400,2400,2400,2400};
    volatile uint8_t _radio_status=0;
  public:
    void outputCh(unsigned char ch, uint16_t pwm);
    void enableOut(uint8_t ch);
    void disableOut(uint8_t ch);
    void init();
    unsigned char getState(void);
    uint16_t inputCh(unsigned char ch);
    void _timer5_capt(void);
};
#endif
```

# C++파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 주요 내용
  - outputCh()
    - PWM 출력
  - enableOut()
    - PWM 출력 활성화

rcInOut.cpp

```
void RCINOUT::outputCh(unsigned char ch, uint16_t pwm){
  pwm=constrain(pwm,MIN_PULSEWIDTH,MAX_PULSEWIDTH);
  pwm<<=1;    // pwm*2; Regiter 2000 ~ 4000
 switch(ch)   {
    case 0:  OCR1B=pwm; break;  // out1
    case 1:  OCR1A=pwm; break;  // out2
    case 2:  OCR4C=pwm; break;  // out3
    case 3:  OCR4B=pwm; break;  // out4
    case 4:  OCR4A=pwm; break;  // out5
 }
}
void RCINOUT::enableOut(uint8_t ch){
  switch(ch) {
    case 0: TCCR1A |= (1<<COM1B1); break; // CH_1 : OC1B
    case 1: TCCR1A |= (1<<COM1A1); break; // CH_2 : OC1A
    case 2: TCCR4A |= (1<<COM4C1); break; // CH_3 : OC4C
    case 3: TCCR4A |= (1<<COM4B1); break; // CH_4 : OC4B
    case 4: TCCR4A |= (1<<COM4A1); break; // CH_5 : OC4A
 }
}
```

# C++파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 주요 내용
  - disableOut()
    - PWM 출력 비활성화
  - init()
    - 타이머 설정
    - Timer1

rcInOut.cpp

```
void RCINOUT::disableOut(uint8_t ch) {
  switch(ch) {
    case 0: TCCR1A &= ~(1<<COM1B1); break; // CH_1 : OC1B
    case 1: TCCR1A &= ~(1<<COM1A1); break; // CH_2 : OC1A
    case 2: TCCR4A &= ~(1<<COM4C1); break; // CH_3 : OC4C
    case 3: TCCR4A &= ~(1<<COM4B1); break; // CH_4 : OC4B
    case 4: TCCR4A &= ~(1<<COM4A1); break; // CH_5 : OC4A
  }
}
void RCINOUT::init(){
  //TIMER1: WGM 1110, TOP=ICR1, CS=2, OUT1,OUT2
  pinMode(12,OUTPUT); // OUT1 (PB6/OC1B)
  pinMode(11,OUTPUT); // OUT2 (PB5/OC1A)
  TCCR1A =((1<<WGM11));
  TCCR1B = (1<<WGM13)|(1<<WGM12)|(1<<CS11);
  ICR1 = 40000; // 0.5us tick => 50hz freq
  OCR1A = 0xFFFF; OCR1B = 0xFFFF;//disable
```

# C++파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 주요 내용
  - init()
    - Timer4- PWM 3ch
    - Timer3- PWM 3ch

rcInOut.cpp

```
//TIMER4: WGM 1110, TOP=ICR4, CS=2, OUT3,OUT4,OUT5
 pinMode(8,OUTPUT); // OUT3 (PH5/OC4C)
 pinMode(7,OUTPUT); // OUT4 (PH4/OC4B)
 pinMode(6,OUTPUT); // OUT5 (PH3/OC4A)
 TCCR4A =((1<<WGM41));
 TCCR4B = (1<<WGM43)|(1<<WGM42)|(1<<CS41);
 OCR4A = 0xFFFF; OCR4B = 0xFFFF; OCR4C = 0xFFFF;
 ICR4 = 40000; // 0.5us tick => 50hz freq
 //TIMER3: WGM 1110, TOP=ICR3, CS=2, OUT6,OUT7,OUT8
 pinMode(3,OUTPUT); // OUT6 (PE5/OC3C)
 pinMode(2,OUTPUT); // OUT7 (PE4/OC3B)
 pinMode(5,OUTPUT); // OUT8 (PE3/OC3A)
 TCCR3A =((1<<WGM31));
 TCCR3B = (1<<WGM33)|(1<<WGM32)|(1<<CS31);
 OCR3A = 0xFFFF; OCR3B = 0xFFFF; OCR3C = 0xFFFF;
 ICR3 = 40000; // 0.5us tick => 50hz freq
```

# C++파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 주요 내용
  - init()
    - Timer5
      - PPM input
      - PWM 2ch

rcInOut.cpp

```
 //TIMER5: WGM 1111, TOP=OCR5A, CS=2, PPM,OUT10,OUT11
  pinMode(48, INPUT);  // PPM Input(PL1/ICP5)
  pinMode(45, OUTPUT); // OUT10 (PL4/OC5B)
  pinMode(44, OUTPUT); // OUT11 (PL5/OC5C)
  TCCR5A =((1<<WGM50)|(1<<WGM51));
  TCCR5B = ((1<<WGM53)|(1<<WGM52)|(1<<CS51)|(1<<ICES5)); //Rising
  OCR5A = 40000; // 0.5us tick => 50hz freq.
  TIMSK5 |= (1<<ICIE5); // Enable Input Capture interrupt
}
```

# C++파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 주요 내용

- _timer5_capt()
  - input capture interrupt
  - PPM decode

rcInOut.cpp

```
void RCINOUT::_timer5_capt(void) {
  static uint16_t prev_icr;
  static uint8_t frame_idx;
  uint16_t icr, pwidth;
  icr = ICR5;    // copy time stamp
  if ( icr < prev_icr ) {   // TOP = 40000
    pwidth = ( icr + 40000 ) - prev_icr;
  } else {
    pwidth = icr - prev_icr;  }
  if ( pwidth > 8000 ) { // if is blank time
    frame_idx=0;
  } else {
    if ( frame_idx < NUM_CHANNELS ) {
      _PWM_RAW[frame_idx++ ] = pwidth;
      if (frame_idx >= NUM_CHANNELS) {
        _radio_status = 1; // One frame finished
      }    }
  }
  prev_icr = icr; // Save icr for next call.
}
```

# C++파일

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

## 주요 내용

- getState()
  - PPM decode 상태
    - 1: 완료 0: 진행중
- inputCh()
  - PWM 값읽기
- ISR()
  - input capture interrupt

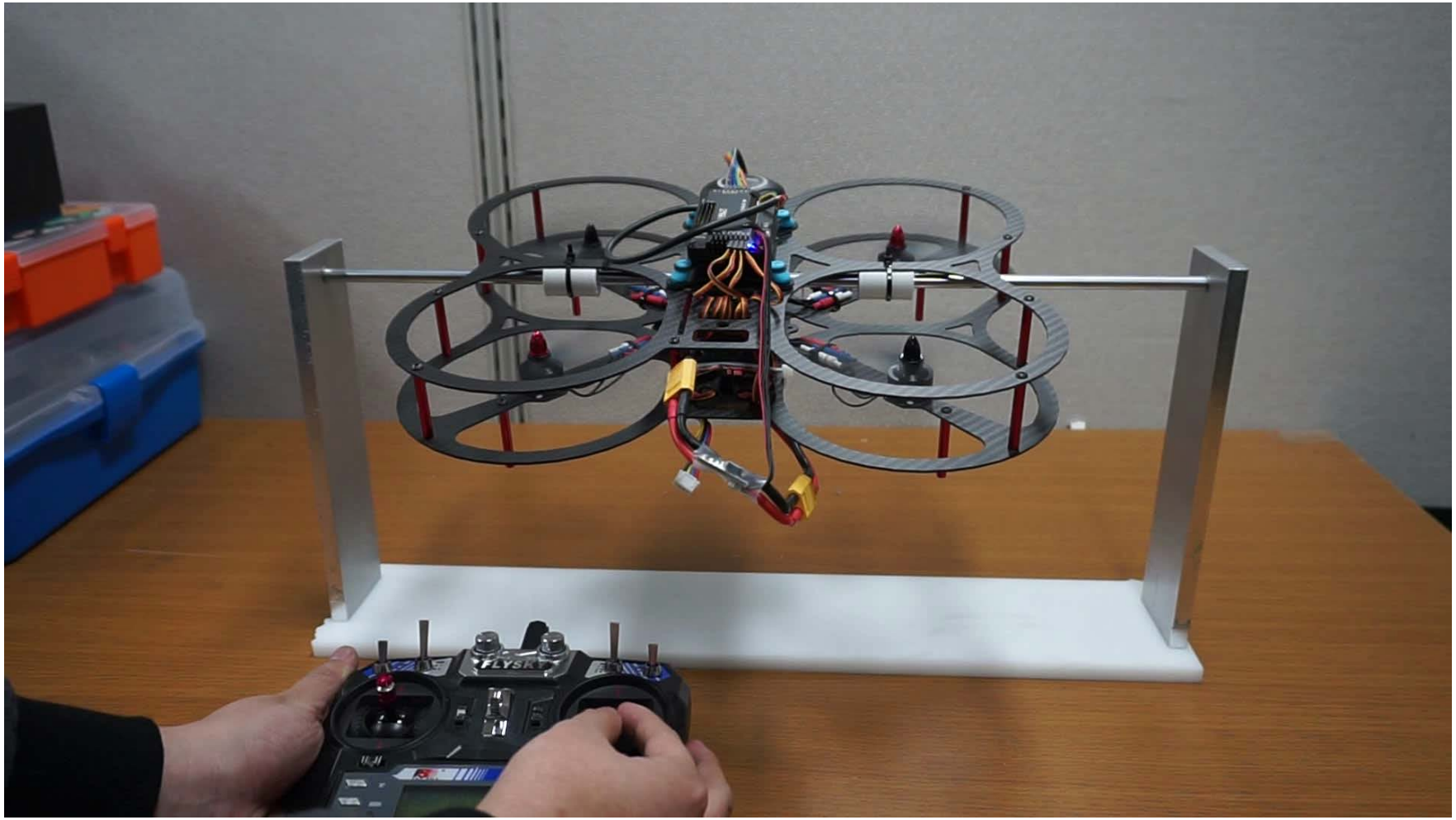
rcInOut.cpp

```
unsigned char RCINOUT::getState(void){
  return _radio_status;
}
uint16_t RCINOUT::inputCh(unsigned char ch){
  uint16_t result;
  cli();    //disable all interrupts
  result = _PWM_RAW[ch];
  sei();  //enabal all interrupts again
  result >>= 1; // convert to us unit
  result = constrain(result,MIN_PULSEWIDTH,MAX_PULSEWIDTH);
  _radio_status = 0; // Radio channel read
  return result;
}
ISR(TIMER5_CAPT_vect) {
    radio._timer5_capt();
}
```

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Dept. of Mechanical System Design, Seoul National University of Science and Technology.



# **THANK** YOU

Powerpoint is a complete presentation graphic package it gives you everything you need to produce a professional-looking presentation