프로토콜(PROTOCOL)

프로토콜은 컴퓨터 네트워크와 통신 시스템에서 정보를 주고받는 데 사용되는 규칙과 규정의 집합입니다. 이것은 데이터가 어떻게 전송, 수신 및 처리되어야 하는지에 대한 명세를 제공하며, 서로 다른 기기나 시스템 간에 효과적인 통신을 가능하게 합니다. 프로토콜은 다양한 통신 수준에서 사용되며, 다음과 같은 몇 가지 주요 특징을 가집니다:

1. 통신규약: 프로토콜은 어떤 형식으로 데이터를 보내고 받아야 하는지에 대한 명확한 규칙을 정의합니다. 이것은 데이터의 형식, 암호화, 오류 검출 및 수정 방법, 통신 시퀀스 등을 포함할 수 있습니다.
2. 표준화: 많은 프로토콜은 산업 표준화 기구나 커뮤니티에서 표준 문서로 정의됩니다. 이렇게 하면 다른 개발자 및 제조업체가 호환성을 유지하고 상호 운용성을 확보할 수 있습니다.
3. 계층 구조: 네트워크와 통신 프로토콜은 종종 계층 구조로 설계됩니다. 이러한 구조는 각 계층이 특정한 기능과 역할을 담당하도록 분리함으로써 효율적인 관리와 개발을 가능하게 합니다. 유명한 계층 구조에는 OSI 모델과 TCP/IP 모델이 있습니다.
4. 다양한 종류: 다양한 프로토콜이 다양한 용도와 통신 환경에 맞게 개발되었습니다. 예를 들어, HTTP 는 웹 브라우징을 위한 프로토콜이며, SMTP 는 이메일 전송을 위한 프로토콜입니다.
5. 예시 프로토콜: 널리 사용되는 프로토콜에는 HTTP (Hypertext Transfer Protocol), TCP/IP (Transmission Control Protocol/Internet Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), 그리고 DNS (Domain Name System) 등이 있습니다.

프로토콜은 네트워크 및 인터넷의 핵심 요소이며, 컴퓨터와 기기 간의 통신을 원활하게 만듭니다. 따라서 프로토콜을 잘 이해하고 구현하는 것은 정보 기술 분야에서 중요한 역할을 합니다.

[CHATGPT]

간단히 설명하자면 이기종간 데이터를 주고받는 방법 및 순서 등을 정의한 것이며 여기서는 EV3(또는 Spike Prime)와 OpenMV간 주고받는 정보의 구조, 형식 의미를 정의합니다.

기초 지식(진수)

10 진수, 2 진수, 16 진수는 숫자를 나타내는 서로 다른 진법입니다. 각각의 진법은 숫자를 표현하는 방식이 다르며 다른 용도와 장점을 가지고 있습니다. 이제 각 진법에 대해 설명하겠습니다:

1. **10 진수 (Decimal):**
   - 10 진수는 일반적으로 일상 생활에서 사용하는 진법입니다.
   - 0 부터 9 까지의 10 개의 숫자를 사용하여 숫자를 표현합니다.
   - 각 자릿수는 10 의 제곱으로 증가합니다. 예를 들어, 10 의 자리, 100 의 자리, 1000 의 자리 등입니다.
2. **2 진수 (Binary):**
   - 2 진수는 0 과 1 두 개의 숫자만을 사용하여 숫자를 표현합니다.
   - 컴퓨터의 내부에서 데이터를 저장하고 처리하는 데 주로 사용됩니다.
   - 각 자릿수는 2 의 제곱으로 증가합니다. 예를 들어, 2 의 자리, 4 의 자리, 8 의 자리 등입니다.
   - 예: 10 진수 13 은 2 진수로 1101 로 표현됩니다.
3. **16 진수 (Hexadecimal):**
   - 16 진수는 0 부터 9 까지의 숫자와 A 부터 F 까지의 알파벳 문자를 사용하여 숫자를 표현합니다.
   - 컴퓨터 프로그래밍 및 디지털 시스템에서 주로 사용되며, 2 진수와 변환하기가 쉽습니다.
   - 각 자릿수는 16 의 제곱으로 증가합니다. 예를 들어, 16 의 자리, 256 의 자리, 4096 의 자리 등입니다.
   - 예: 10 진수 255 는 16 진수로 FF 로 표현됩니다.

진법 변환은 수를 다른 진법으로 변환하는 과정을 의미하며, 이를 위한 규칙과 방법이 있습니다. 예를 들어, 10 진수를 2 진수나 16 진수로 변환하거나, 그 반대로 변환할 수 있습니다. 컴퓨터 과학 및 프로그래밍 분야에서는 2 진수와 16 진수가 빈번하게 사용되며, 메모리 주소, 데이터 표현, 비트 연산 등에 활용됩니다.

[CHATGPT]

기초지식 (10진수, 2진수, 16진수)

10 진수, 2 진수, 16 진수 간의 값 변환은 다음과 같은 방식으로 이루어집니다. 아래에서는 각 진법 간의 변환 과정을 설명합니다.

1. **10 진수를 2 진수로 변환하기:**
   - 10 진수를 2 로 나누면서 나머지를 저장합니다.
   - 이 과정을 계속 반복하면서 나누는 수가 0 이 될 때까지 진행합니다.
   - 저장한 나머지를 역순으로 나열하면 2 진수가 됩니다.
   - 예: 10 진수 13 을 2 진수로 변환하면, 13 / 2 = 6 (나머지 1), 6 / 2 = 3 (나머지 0), 3 / 2 = 1 (나머지 1), 1 / 2 = 0 (나머지 1)입니다. 역순으로 나열하면 1101 이 됩니다.
2. **2 진수를 10 진수로 변환하기:**
   - 2 진수의 각 자릿수를 2 의 제곱에 해당하는 가중치를 곱한 후 모두 더합니다.
   - 예: 2 진수 1101 을 10 진수로 변환하면, (1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0) = 8 + 4 + 0 + 1 = 13 입니다.
3. **10 진수를 16 진수로 변환하기:**
   - 10 진수를 16 으로 나누면서 나머지를 저장합니다.
   - 10, 11, 12, 13, 14, 15 는 각각 A, B, C, D, E, F 로 표현합니다.
   - 이 과정을 계속 반복하면서 나누는 수가 0 이 될 때까지 진행합니다.
   - 저장한 나머지를 역순으로 나열하면 16 진수가 됩니다.
   - 예: 10 진수 255 를 16 진수로 변환하면, 255 / 16 = 15 (나머지 F), 15 / 16 = 0 (나머지 15 또는 F)입니다. 역순으로 나열하면 FF 가 됩니다.
4. **16 진수를 10 진수로 변환하기:**
   - 16 진수의 각 자릿수를 16 의 제곱에 해당하는 가중치를 곱한 후 모두 더합니다.
   - 예: 16 진수 FF 를 10 진수로 변환하면, (F * 16^1) + (F * 16^0) = (15 * 16) + (15 * 1) = 240 + 15 = 255 입니다.

이러한 변환 과정을 사용하여 10 진수, 2 진수, 16 진수 간의 값을 서로 변환할 수 있습니다. 이러한 변환은 컴퓨터 프로그래밍, 디지털 통신 및 하드웨어 설계와 같은 다양한 컴퓨터 관련 분야에서 중요하게 활용됩니다.

[CHATGPT]

기초 지식(16진수))

16 진수(16 진법)는 기존의 10 진수와 같은 진법 시스템 중 하나이며, 16 개의 다른 숫자를 사용하여 수를 나타내는 방법입니다. 16 진수는 컴퓨터 과학, 디지털 통신, 하드웨어 설계 등 다양한 컴퓨터와 관련된 분야에서 주로 사용됩니다. 아래는 16 진수에 대한 자세한 설명입니다:

1. **16 진수 숫자:**
   - 16 진수는 0 부터 9 까지의 숫자와 A 부터 F 까지의 알파벳 문자를 사용하여 16 개의 다른 숫자로 표현됩니다. 따라서 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F 를 사용합니다.
2. **16 진수의 자릿수:**
   - 16 진수의 각 자릿수는 16 의 거듭제곱에 해당하는 가중치를 갖습니다. 가장 오른쪽 자릿수부터 왼쪽으로 갈수록 가중치는 16 의 지수로 증가합니다. 예를 들어, 오른쪽에서부터 첫 번째 자릿수는 16^0(1), 두 번째 자릿수는 16^1(16), 세 번째 자릿수는 16^2(256), 그리고 네 번째 자릿수는 16^3(4096)의 가중치를 갖습니다.
3. **16 진수 예시:**
   - 16 진수로 숫자를 표현할 때 예를 들어 "1A3F"와 같이 사용됩니다.
   - "1A3F"를 10 진수로 변환하면 다음과 같습니다:
     - (1 * 16^3) + (10 * 16^2) + (3 * 16^1) + (15 * 16^0) = 4096 + 2560 + 48 + 15 = 6719.
4. **용도:**
   - 16 진수는 주로 컴퓨터 메모리 주소, 레지스터 값, 비트 연산 및 데이터의 표현에 사용됩니다.
   - 메모리 주소를 16 진수로 표현하면 메모리 관리가 용이하며, 레지스터 값과 비트 연산을 수행할 때 비트 단위 조작이 더 쉽습니다.
   - 또한, 컴퓨터의 디버깅 및 바이너리 데이터 편집 시에도 16 진수를 사용하여 작업하기 편리합니다.

16 진수는 주로 기술적인 분야에서 사용되며, 컴퓨터 프로그래밍과 관련된 작업에서 종종 볼 수 있습니다. 이를 통해 데이터의 표현과 조작이 더 효과적으로 이루어질 수 있습니다.

[CHATGPT]

기초 지식(비트(Bit), 니블(Nibble), 바이트(Byte))

"Bit," "nibble," 그리고 "byte"는 컴퓨터와 정보 이론 분야에서 데이터의 크기와 단위를 나타내는 용어입니다. 각각의 용어는 다음과 같이 정의됩니다:

1. **Bit (비트):**
   - "Bit"는 "binary digit"의 약어로, 컴퓨터에서 가장 작은 데이터 단위를 나타냅니다.
   - 비트는 0 또는 1 의 두 가지 값만 가질 수 있는 이진 숫자입니다.
   - 비트는 데이터를 나타내는 데 사용되며, 컴퓨터의 모든 정보는 이진 비트로 표현됩니다. 예를 들어, 전기 신호가 on 또는 off 로 나타나는 것을 비트로 표현할 수 있습니다.
2. **Nibble (니블):**
   - "Nibble"은 4 개의 연속적인 비트로 구성된 데이터 단위를 나타냅니다.
   - 따라서 하나의 nibble 은 4 비트 또는 4 개의 이진 숫자로 이루어져 있습니다.
   - Nibble 은 주로 16 진수 표기법에서 사용되며, 4 비트를 나타내기 위해 사용됩니다. 각 nibble 은 0 부터 15 까지의 값을 나타낼 수 있습니다.
3. **Byte (바이트):**
   - "Byte"는 8 개의 연속적인 비트로 구성된 데이터 단위를 나타냅니다.
   - 따라서 하나의 바이트는 8 비트 또는 8 개의 이진 숫자로 이루어져 있습니다.
   - 바이트는 컴퓨터에서 가장 일반적으로 사용되는 데이터 단위 중 하나이며, 문자, 숫자, 이미지, 오디오 등을 저장하는 데 사용됩니다.
   - 바이트는 256 가지 다른 값을 나타낼 수 있으며, 2^8 의 값 범위를 가집니다.

아래는 각 단위의 크기와 사용 사례에 대한 간단한 비교입니다:

- 1 비트: 0 또는 1 (가장 작은 데이터 단위)
- 1 nibble: 4 비트 (16 가지 값 표현, 주로 16 진수에서 사용)
- 1 바이트: 8 비트 (256 가지 값 표현, 문자, 숫자, 이미지, 오디오 등 다양한 데이터 저장)

예를 들어, 영어 알파벳 한 문자를 저장하려면 하나의 바이트가 필요하며, 이진수로는 8 비트로 표현됩니다. 따라서 "A"를 ASCII 코드로 나타내면 "01000001"입니다.

[CHATGPT]

1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Nibble				Nibble				Nibble				Nibble			
Byte <- MSB								Byte LSB ->							
Word															

기초 지식(LSB, MSB, Big Endian, Little Endian)

"LSB (Last Significant Bit)와 MSB (Most Significant Bit)는 이진수의 가장 낮은 비트와 가장 높은 비트를 나타내는 용어입니다. 이들은 데이터의 위치와 중요성을 나타냅니다. 여기에 더해, Big Endian 과 Little Endian 은 바이트 순서를 나타내는데 사용되며, 컴퓨터 아키텍처와 데이터의 저장 및 전송 방식에 영향을 미칩니다. 이제 각 용어를 자세히 설명하겠습니다:

1. **LSB (Last Significant Bit):**
    - LSB 는 이진수에서 가장 오른쪽에 있는 비트를 가리킵니다.
    - 이 비트는 가장 작은 가중치를 갖고 있으며, 데이터의 가장 낮은 자릿수를 나타냅니다.
    - 예를 들어, 이진수 "10101010"에서 가장 오른쪽 비트인 "0"은 LSB 입니다.
2. **MSB (Most Significant Bit):**
    - MSB 는 이진수에서 가장 왼쪽에 있는 비트를 가리킵니다.
    - 이 비트는 가장 큰 가중치를 갖고 있으며, 데이터의 가장 높은 자릿수를 나타냅니다.
    - 예를 들어, 이진수 "10101010"에서 가장 왼쪽 비트인 "1"은 MSB 입니다.
3. **Big Endian (빅 엔디안):**
    - Big Endian 은 데이터를 저장하거나 전송할 때 가장 큰 바이트(가장 높은 주소)가 먼저 나오는 방식을 가리킵니다.
    - 예를 들어, 16 비트 숫자 "0x1234"를 Big Endian 으로 저장하면 메모리에는 "12 34"와 같이 저장됩니다.
4. **Little Endian (리틀 엔디안):**
    - Little Endian 은 데이터를 저장하거나 전송할 때 가장 작은 바이트(가장 낮은 주소)가 먼저 나오는 방식을 가리킵니다.
    - 예를 들어, 16 비트 숫자 "0x1234"를 Little Endian 으로 저장하면 메모리에는 "34 12"와 같이 저장됩니다.

Big Endian 과 Little Endian 은 주로 컴퓨터 아키텍처와 통신 프로토콜에서 중요하며, 데이터를 올바르게 해석하려면 올바른 바이트 순서를 알아야 합니다. 예를 들어, 네트워크 프로토콜에서 데이터를 전송할 때, 어떤 바이트 순서를 사용하는지가 통신의 성공 또는 실패를 결정할 수 있습니다.

[CHATGPT]

기초 지식(정수의 표현)

"정수(Integer)형의 음수는 주로 2 의 보수(2's complement) 방식으로 표현됩니다. 2 의 보수 표현은 컴퓨터에서 음수를 효율적으로 표현하고 다루기 위한 방법 중 하나로 널리 사용됩니다. 다음은 2 의 보수 방식을 사용하여 음수를 표현하는 과정과 예시입니다:

1. **양수를 이진수로 변환:**
   - 먼저 음수의 절댓값을 이진수로 변환합니다. 예를 들어, -5 를 이진수로 변환하면 5 를 이진수로 표현한 값인 "0101"이 됩니다.
2. **비트 반전 (Bitwise Negation):**
   - 다음으로, 이진수의 모든 비트를 반전시킵니다. 0 은 1 로, 1 은 0 으로 바뀝니다. 이 과정을 통해 음수의 보수를 얻습니다. "0101"를 반전하면 "1010"이 됩니다.
3. **1 을 더함:**
   - 마지막으로, 1 을 이 보수에 더합니다. 이렇게 하면 음수가 됩니다. "1010"에 1 을 더하면 "-5"가 됩니다.

이 과정을 통해 양수를 음수로 변환할 수 있습니다. 2 의 보수 표현 방식은 음수와 양수를 덧셈과 뺄셈 연산에서 동일한 방식으로 처리할 수 있으므로, 컴퓨터에서 음수를 다루기가 편리합니다.

예를 들어, 8 비트 2 의 보수 표현에서 -5 는 "11111011"로 표현됩니다. 이진수의 가장 왼쪽 비트(최상위 비트)가 1 인 경우에 해당 숫자가 음수임을 나타내며, 나머지 비트는 해당 음수의 절댓값을 나타냅니다.

[CHATGPT]

4개의 비트로 양의 수 3을 표현하면 0011b 입니다.

-3을 표현하려면 절댓값 3 (0011b)를 반전 시킵니다 (2's complement). 값은 1100b 가 됩니다. 여기에 1을 더합니다. 1100b + 0001b = 1101b입니다.

2진수에서 -3의 값은 1101b 입니다. 여기서 최상위 비트는 수의 부호를 뜻하게 됩니다.

1101b를 10진수로 바꾸려면 일단 최상위 비트가 1 이므로 이 수는 음수입니다. 1101b에서 1을 뺍니다. 그러면 값은 1100b가 됩니다. 이 수를 반전시킵니다. 수는 0011b가 되며 이는 절댓값입니다. 앞에서 부호가 음수이므로 값은 -3이 됩니다.

1을 더하고 빼는 이유는 +0, -0 두개가 존재하지 못하게 -0의 값을 없애기 위함 입니다.

기초 지식(int와 unsigned int의 차이)

"integer"와 "unsigned integer"는 정수를 나타내는 두 가지 주요 데이터 유형입니다. 이 두 유형은 정수 값의 부호와 관련이 있으며, 컴퓨터 프로그래밍 및 데이터 저장과 처리에서 중요한 역할을 합니다.

1. **Integer (정수):**
    - "Integer"는 정수 데이터 유형을 나타내는 용어로, 양수와 음수 정수 값을 포함합니다.
    - 정수는 소수점 이하 부분이 없는 숫자로, 정수형 변수에 저장됩니다.
    - 예를 들어, -1, 0, 1, 100, -500 등 모든 정수 값은 integer 로 표현될 수 있습니다.
    - 정수 데이터 유형은 컴퓨터 메모리에서 일반적으로 2 의 보수 표현 방식을 사용하여 표현됩니다. 이 방식은 양수와 음수를 효과적으로 나타내기 위해 사용됩니다.
2. **Unsigned Integer (부호 없는 정수):**
    - "Unsigned integer" 또는 "unsigned"는 부호 없는 정수 데이터 유형을 나타내는 용어입니다.
    - 부호 없는 정수는 양수 값만을 나타내며, 음수 값을 표현하지 않습니다.
    - 따라서 0 과 양의 정수 값만을 가집니다.
    - 예를 들어, 부호 없는 8 비트 정수는 0 부터 255 까지의 값을 나타낼 수 있습니다.
    - 부호 없는 정수는 양수 값의 범위를 효과적으로 나타낼 때 사용되며, 데이터가 항상 양수임을 보장할 때 유용합니다.

정수와 부호 없는 정수는 프로그래밍에서 데이터를 저장하고 처리하는 데 중요한 역할을 합니다. 정수는 보통 "int" 또는 "signed int"로 표기되며, 부호 없는 정수는 "unsigned int" 또는 "unsigned"로 표기됩니다. 이들 데이터 유형을 사용하여 수학적 연산, 배열 인덱싱, 데이터베이스 저장 등 다양한 프로그래밍 작업을 수행할 수 있습니다.

[CHATGPT]

Integer형은 통상 2 Byte로 표현하며 int형은 -32768 ~ 0 ~ +32767 까지 unsigned int형의 경우 0 ~ 65536 까지 표현됩니다.

signed int 형에서 32767 + 1 의 결과는?

signed int 형의 값 32767 에 1을 더하면 값은 표현 범위를 벗어나 Overflow가 발생되며 32768이 아닌 -32768의 결과가 나옵니다. EV3와 OpenMV간 통신에서 데이터는 2Byte signed integer형의 값을 Little Endian 형태로 주고 받기에 값의 범위에 주의하여야 합니다.

기초 지식(Exclusive OR : XOR)

XOR(Exclusive OR)은 논리 연산자로, 두 개의 논리 값 또는 비트를 비교하고, 두 값 중 하나만 참(1)일 때 결과가 참(1)이 되는 연산을 나타냅니다. XOR 연산은 다음과 같은 진리표로 나타낼 수 있습니다:

입력 A	입력 B	XOR 결과
0	0	0
0	1	1
1	0	1
1	1	0

XOR 연산은 다음과 같은 특징을 가지고 있습니다:

1. **배타적인 참(Exclusive True):** 두 입력 중 하나만 참(1)일 때 결과가 참(1)이 됩니다. 즉, 두 입력 값이 같으면 결과는 거짓(0)이 되고, 두 입력 값이 다르면 결과는 참(1)이 됩니다.
2. **논리적으로 부정(Negation):** XOR 연산은 논리적으로 부정 연산을 수행합니다. 두 입력 값이 같으면 결과는 거짓(0)이며, 두 입력 값이 다르면 결과는 참(1)입니다.
3. **암호학 및 데이터 복구에 활용:** XOR 연산은 데이터의 비트 수준에서 변경, 암호화, 복구, 오류 검출 및 수정과 같은 작업에 사용됩니다. XOR 연산은 비트 단위 XOR 마스크를 사용하여 데이터를 암호화하거나 해독하는 데도 사용됩니다.
4. **집합 이론과 논리 회로에서 사용:** XOR 연산은 집합 이론, 논리 회로 설계, 그리고 컴퓨터 과학의 다양한 분야에서 중요한 개념으로 사용됩니다.

XOR 연산은 두 비트나 논리 값 간의 관계를 분석하고 다루는 데 매우 유용하며, 컴퓨터 과학, 전자공학, 암호학 등 다양한 분야에서 빈번하게 활용됩니다.

[CHATGPT]

기초 지식(CHECK SUM)

체크 섬(Checksum)은 데이터의 무결성을 확인하고 오류를 감지하는 데 사용되는 검사 합입니다. 주로 데이터 전송 중에 오류를 검출하기 위해 사용되며, 데이터 블록의 합계 또는 해시 값을 계산하여 생성됩니다. 데이터 수신 측에서도 체크 섬을 다시 계산하고, 송신 측에서 계산한 체크 섬 값과 비교하여 데이터가 손상되었는지를 확인할 수 있습니다.

체크 섬은 주로 다음과 같은 목적으로 사용됩니다:

1. **데이터 무결성 확인:** 데이터가 전송되거나 저장되는 동안 변경되거나 손상되는 경우가 있습니다. 체크 섬은 데이터의 무결성을 보호하고 손상된 데이터를 식별하는 데 도움을 줍니다.
2. **오류 검출:** 데이터 전송 중에 발생하는 오류(예: 전송 중에 비트 뒤집힘)를 검출합니다. 데이터가 올바른 지 확인하고 손상된 데이터를 다시 요청할 수 있게 합니다.
3. **보안:** 데이터의 무결성을 확인하여 데이터 변조 또는 침입을 탐지할 수 있습니다. 이것은 암호화된 데이터나 네트워크 통신에서 중요한 역할을 합니다.

체크 섬의 계산 방법은 데이터의 종류와 용도에 따라 다양합니다. 가장 간단한 형태의 체크 섬은 각 바이트의 합 또는 XOR 연산 결과를 사용하는 것이며, 더 복잡한 해시 함수나 알고리즘을 사용하여 체크 섬을 생성할 수도 있습니다. 일반적으로 체크 섬은 데이터의 끝에 추가되거나 헤더에 포함되어 송신 측과 수신측 모두에서 검사할 수 있도록 합니다.

체크 섬은 데이터 무결성을 보호하고 오류를 검출하는 데 중요한 도구로 사용되며, 네트워크 통신, 파일 전송, 데이터베이스 관리, 디스크 저장 및 기타 다양한 응용 분야에서 활용됩니다.

[CHATGPT]

기초 지식(XOR를 이용한 CHECK SUM)

패킷에서 XOR 를 사용하여 체크 섬(Checksum)을 계산하는 방법은 데이터 무결성을 확인하고 패킷의 오류를 감지하는 간단한 방법 중 하나입니다. 이 방법은 패킷의 모든 비트를 XOR 하여 체크 섬 값을 생성하는 것으로, 송신 측에서 데이터를 보내기 전에 체크 섬을 계산하고, 수신 측에서는 동일한 방법으로 체크 섬을 다시 계산하여 송신 측과 비교하여 데이터의 무결성을 확인합니다. 아래는 이 과정을 단계별로 설명한 것입니다:

1. **패킷 데이터 준비:**
   - 송신 측에서 패킷 데이터를 준비합니다. 이 데이터는 메시지, 파일, 네트워크 패킷 또는 기타 형식의 정보일 수 있습니다.
2. **XOR 로 체크 섬 계산:**
   - 패킷 데이터의 모든 비트를 XOR(Exclusive OR)하여 체크 섬 값을 계산합니다.
   - 데이터의 모든 비트를 순서대로 XOR 하여 하나의 값으로 만듭니다.
3. **체크 섬 추가:**
   - 계산한 체크 섬 값을 패킷에 추가합니다. 체크 섬 값은 데이터와 함께 전송됩니다.
4. **데이터 전송:**
   - 패킷 데이터와 함께 체크 섬이 수신 측으로 전송됩니다.
5. **수신 측에서 체크 섬 검사:**
   - 수신 측에서 패킷을 수신한 후, 수신한 데이터와 함께 체크 섬 값을 계산합니다.
6. **체크 섬 비교:**
   - 수신 측에서 계산한 체크 섬 값과 패킷에 포함된 체크 섬 값을 비교합니다.
   - 두 체크 섬 값이 일치하면 데이터가 손상되지 않았음을 나타냅니다.
   - 두 체크 섬 값이 일치하지 않으면 데이터가 손상되었을 가능성이 있으므로, 에러를 감지하고 처리할 수 있습니다.

이 방법은 간단하고 오류 감지에 유용하지만, 보다 복잡한 체크 섬 알고리즘과 비교하여 오류를 더 잘 감지하지 못할 수 있습니다. 더 강력한 오류 감지 기능을 제공하는 체크 섬 알고리즘(예: CRC, Adler-32 등)을 사용하는 경우가 많습니다.

[CHATGPT]

송신 측에서는 보내는 모든 데이터의 배타적 논리합(XOR)을 구해 PACKET 끝에 추가합니다.

수신측 에서는 수신되는 모든 데이터(CHECKSUM 포함)의 배타적 논리합을 구했을 때 0이되면 정상적인 데이터를 수신한 겁니다. 그렇지 않다면 이는 체크 섬 에러로서 신뢰할 수 없는 데이터입니다.

기초 지식(PACKING과 PARSING)

"Packing"과 "parsing"은 데이터 처리와 관련된 용어입니다. 각각은 데이터를 다루는 과정에서 특정한 역할을 수행하는데 사용됩니다.

1. **Packing (패킹):**
    - 패킹은 데이터를 더 작은 메모리 또는 데이터 구조에 저장하거나 표현하는 프로세스를 가리킵니다.
    - 일반적으로 데이터를 패킹하는 이유는 메모리를 절약하거나 데이터를 전송하기 위해 데이터를 조밀하게 저장하려는 경우입니다.
    - 예를 들어, 여러 가지 데이터 유형 (예: 정수, 부동 소수점, 문자열)을 이진 데이터로 패킹하여 파일에 저장하거나 네트워크를 통해 전송할 수 있습니다. 이러한 작업은 데이터 직렬화(Serialization)의 한 형태입니다.
2. **Parsing (파싱):**
    - 파싱은 데이터를 분석하고 그 의미를 이해하는 과정을 나타냅니다.
    - 주로 텍스트 데이터나 구조화된 데이터 (예: XML, JSON)를 해석하고 필요한 정보를 추출하는 데 사용됩니다.
    - 예를 들어, 웹 페이지의 HTML 코드를 파싱하여 원하는 정보 (예: 제목, 링크, 본문)를 추출하거나, JSON 형식의 데이터를 파싱하여 웹 애플리케이션에서 사용할 수 있는 데이터로 변환하는 작업이 파싱의 예입니다.

둘 다 데이터 처리와 관련된 중요한 작업이며, 서로 보완적으로 사용됩니다. 데이터를 패킹하고 전송한 다음, 수신 측에서 데이터를 받아서 파싱하여 의미 있는 정보를 추출하고 처리할 수 있습니다. 이것은 데이터 통신, 파일 입출력, 데이터베이스 관리 및 다양한 응용 분야에서 사용되는 일반적인 작업입니다.

[CHATGPT]

기초 지식(structure module format string)

Format String (Byte Order, Size, and Alignment)

Character	Byte order	Size	Alignment
@	native	native	native
=	native	standard	none
<	**little-endian**	**standard**	**none**
>	big-endian	standard	none
!	network (= big-endian)	standard	none
Format	**C Type**	**Python type**	**Standard size**
x	pad byte	no value	
c	char	bytes of length 1	1
b	**signed char**	**integer**	**1**
B	**unsigned char**	**integer**	**1**
?	_Bool	bool	1
h	**short**	**integer**	**2**
H	**unsigned short**	**integer**	**2**
i	int	integer	4
I	unsigned int	integer	4
l	long	integer	4
L	unsigned long	integer	4
q	long long	integer	8
Q	unsigned long long	integer	8
n	ssize_t	integer	
N	size_t	integer	
f	**float**	**float**	**4**
d	double	float	8
s	char[]	bytes	
p	char[]	bytes	
P	void *	integer	

[OpneMV Document]

기본적으로 1 바이트 정수, 2 바이트 정수형 또는 4 바이트 소수형으로 구성할 예정입니다.

Little Endian 방식('<')으로 전송할 것이며 Format 은 '<b', <B', '<h', '<H', '<f' 형식이 사용될 것입니다.

EV3 - OpenMV 간 통신 프로토콜

```
# PACKET FORMAT (Unsigned Integer 2 Byte 는 LSB, MSB 순으로 되어 있으며 Little Endian 형태로 전송됨.)
# ------------------------------------------------------------------------------------------------------------------
# PACKET HEADER, PACKET SeqNo, ULTRA, TOF, YAW, ROLL, PITCH, Object Count n, OBJECT[0]...OBJECT[n-1], CHECKSUM
# ------------------------------------------------------------------------------------------------------------------
# 1. PACKET HEADER  : 2 Byte (0xAA , 0xCC)              -- 0XAA 와 0xCC 가 연달아 들어오면 PCKET 의 시작임.
# 2. PACKET SeqNo   : Unsigned Integer 1 Byte (0 ~ 127) -- PACKET 송신시마다 1 씩 증가함. (수신측에서 이값이
바껴야 새로운 PACKET 으로 인식)
# 3. ULTRA  : Unsigned Integer 1 Byte   -- HC-SR04 를 이용한 초음파 거리 측정 값.(Format:'<B')
# 4. TOF    : Unsigned Integer 2 Byte   -- VL53L0X 를 이용한 TOF 거리 측정 값.(Format:'<H')
# 5. YAW    : Float 4 Byte              -- BNO055 를 이용한 YAW 값.(Format:'<f')
# 6. ROLL   : Float 4 Byte              -- BNO055 를 이용한 ROLL 값.(Format:'<f')
# 7. PITCH  : Float 4 Byte              -- BNO055 를 이용한 PITCH 값.(Format:'<f')
# 8. OBJECT COUNT n  : Unsigned Integer 1 Byte       -- 포함하고 있는 Object 의 갯수
# 9. OBJECT[n]       : n 개의 Object 정보. 구조는 아래와 같음. (OBJECT 당 10 Byte)
#       ----------------------------------------
#       : Object Type, X, Y, Width, Height
#       ----------------------------------------
#       : Object Type   : Unsigned Integer 2 Byte  -- Object 의 종류 (예: 1 = Silver Ball, 2 = Black Ball ...)
#       : X             : Unsigned Integer 2 Byte
#       : Y             : Unsigned Integer 2 Byte
#       : R             : Unsigned Integer 2 Byte
#       : MAGNITUDE     : Unsigned Integer 2 Byte
#       ----------------------------------------
# ------------------------------------------------------------------------------------------------------------------
# 10. CHECKSUM : PACKET HEADER 를 포함한 모든 수신데이터의 Exclusive Or 값. (chksum ^= all recieve data)
# ------------------------------------------------------------------------------------------------------------------
```

상기 프로토콜은 영상내 오브젝트 정보와 5 개의 센서 값을(초음파, TOF, IMU(YAW, ROLL, PITCH)) 송신하기 위한 것입니다.

테스트를 위해 총 15 Bytes 의 센서 값을 송 수신 합니다. (Unsigned Integer 1 Byte, Unsigned Integer 2 Bytes, Float 4 Bytes, Float 4 Bytes, Float 4 Bytes.)

소수점 형태의 데이터를 보낼 때 4byte Float 형태 말고 소숫점 이하 자릿수만큼 곱해 정수화해서 보내고 수신 측에서 수신데이터에서 나누기 해 주는 방법이 있습니다. (예: 123.45678 -> X 100 -> 12345.678 -> 정수화 -> 12345 -> 송신, 수신 -> 12345 -> / 100 -> 123.45)

곱하기 했을 때 Data 형에 따른 범위 안에 들어오는지 주의해야 합니다. (예 123456 <- 이 값은 2 byte 로 표현할 수 없습니다. 이 값은 16 진수로 0x01E240 이며 2Byte 데이터형에 담으면 0xE240 만 담기게 됩니다 이는 10 진수로 57920 입니다.

송 수신 하는 센서정보에 따라 (3 ~ 7) 번의 구조를 바꿔야 합니다.

EV3 - OpenMV 간 통신 테스트

OpenMV Test Program : Codes/OpenMv/EV3 at main · hakyungi/Codes (github.com)

EV3 Test Program : Codes/EV3/OpenMV at main · hakyungi/Codes (github.com)

OpenMV 가 송신한 DATA

ULTRA : 164

TOF : 200

IMU : [69.125, 6.5625, -76.75]

{"x":120, "y":112, "r":20, "magnitude":5100}

{"x":120, "y":144, "r":20, "magnitude":5458}

send 40

Objects 2

AA CC 25 A4 C8 0 0 40 8A 42 0 0 D2 40 0 80 99 C2 2 1 0 78 0 70 0 14 0 EC 13 1 0 78 0 90 0 14 0 52 15 B4

EV3 가 수신 한 결과

ULTRA : 164

TOF : 200

YAW : 69.12500000000001

ROLL : 6.5625

PITCH : -76.75000000000001

[(1, 120, 112, 20, 5100), (1, 120, 144, 20, 5458)]

PACKET 해석 (16 진수)

AA CC 25 A4 C8 0 0 40 8A 42 0 0 D2 40 0 80 99 C2 2 1 0 78 0 70 0 14 0 EC 13 1 0 78 0 90 0 14 0 52 15 B4

PACKET HEADER : AA CC

PACKET SEQ. NO. : 25 (37)

초음파 : A4 (164)

TOF : C8 00 (Little Indian 형태 본래 값은 00 C8 (200))

YAW : 00 40 8A 42 (4 bytes float)

ROLL : 00 00 D2 40 (4 bytes float)

PITCH : 00 80 99 C2 (4 bytes floa t)

Object Count : 02 (2)

Object 정보 (2 개 X 10 Bytes)

첫번째 Object : 1 0 78 0 70 0 14 0 EC 13

- 타잎: 01 00 (0001h:1), X: 78 00 (0078h:120), Y: 70 00 (0070h:112), R: 14 00(0014h:20), Mag: EC 13(13ECh:5100)

두번째 Object : 1 0 78 0 90 0 14 0 52 15

- 타잎: 01 00 (0001h:1), X: 78 00 (0078h:120), Y: 90 00 (0090h:144), R: 14 00(0014h:20), Mag: 52 15(1552h:5458)

CHECK SUM : B4