2019

ToF LIDAR Module

DEVICE SPECIFICATION

SICONTROL

개요

본 모듈은 기구 등에 고정되어 ToF 센서를 이용 점에 대한 거리를 측정하며, Pitch와 Yaw를 제어하는 두 개의 모터를 바탕으로 바라보는 방향에 대해 시야각을 지정 깊이와 위치 정보를 송신한다.

이 때 데이터의 송신은 TCP/IP 통신을 바탕으로 이루어지며, 사용자 응용 프로그램에서는 이렇게 연결된 TCP 소켓을 바탕으로 수신한 위치와 깊이 정보를 재구성해 버퍼 등에 깊이 맵을 만들 수 있다.

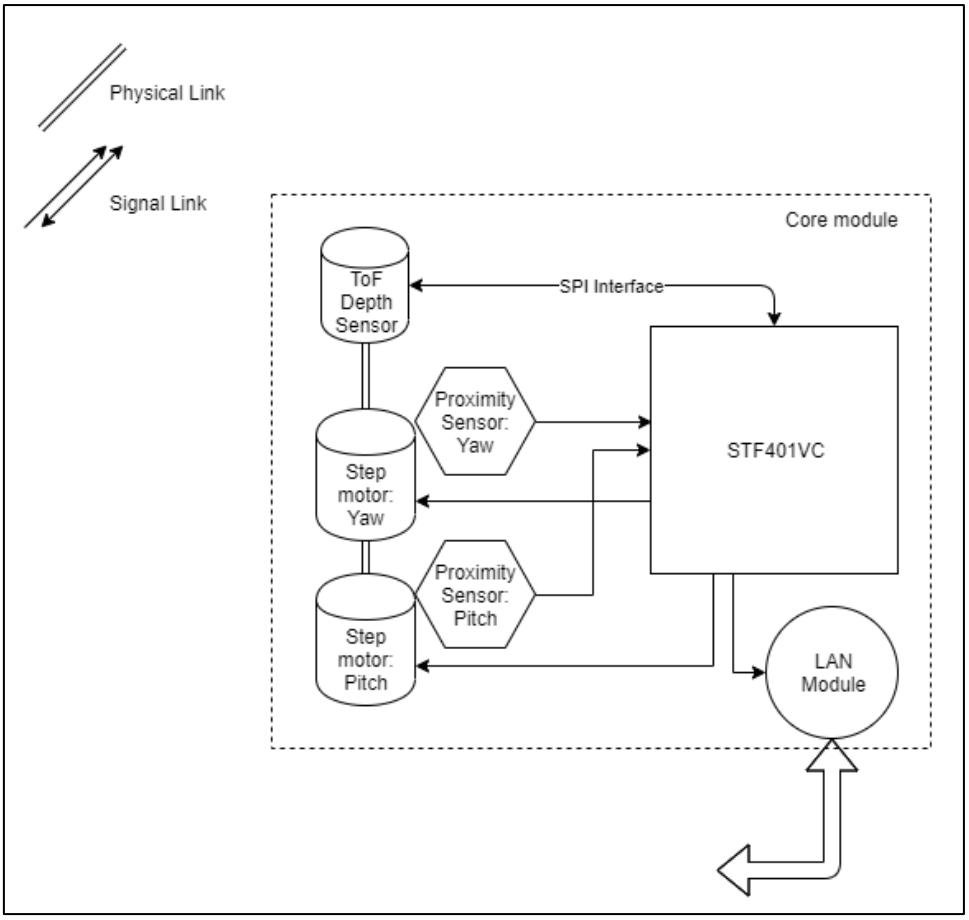

Figure 1. System layout

Hardware Specification

표 1. 핵심 부품

부품명	**모델명**	수량	기능	비고
MCU	STM32F401VC	1	컨트롤러	
ToF 깊이 센서	AFBR-S50MV85G	1	깊이 인식	
스탭 모터	**미정**	2	각각 Pitch, Yaw 좌표 지정	
근접 센서	**미정**	2	각각 Pitch, Yaw 모터 초기화	
TCP/IP 이더넷 인터페이스	Wiznet W5500 Ethernet Module	1	유선 랜 통신	

표 2. 부수 부품

카테고리	부품명	**모델명**	수량	기능	비고
깊이 센서 바이어스	-	-	1	컨트롤러	
스탭 모터 바이어스	-	-			

Port assignments

@todo.

System Layout

깊이 센서의 구동에는 제조사인 Broadcom 사(社)가 제공하는 API를 이용하며, MCU와는 SPI 인터페이스를 통해 연결한다.

이더넷 모듈의 구동에는 Wiznet 사가 제공하는 API가 사용되며, 역시 SPI로 연결된다.

프로그램은 몇 개의 모듈로 이루어져 있으며, 이에 대한 설명은 위의 그림과 같다.

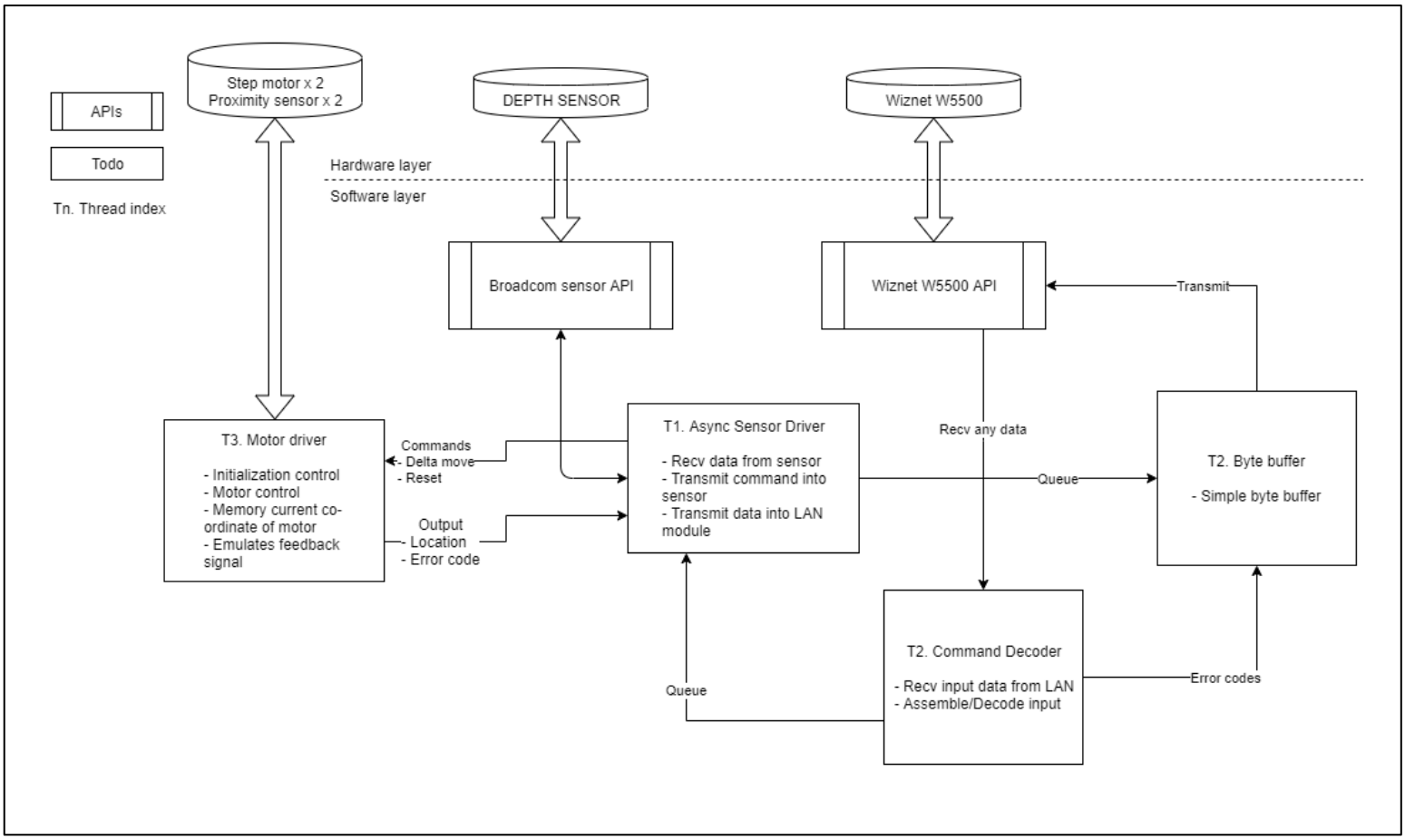

Figure 2. System Layout (2)

동작 – 통신 인터페이스

본 모듈은 고정적으로 포트 15999번을 사용하며, 항상 로컬 호스트에 클라이언트로서 연결을 시도한다.

Startup(리셋 등) 시점에 외부 스위치의 값을 읽어, UDP 를 사용할지 TCP/IP 를 사용할지 결정할 수 있다. (예정)

기본적으로 TCP Stream connection 을 가정하므로, 아래와 같은 별도의 데이터 프로토콜을 사용하게 된다. (바이트 단위)

모든 통신의 첫 4바이트는 식별자로서 반드시 0x1E3F5CD9가 되어야 하며, 만약 패킷의 첫 4바이트가 이 값이 아닌 경우 마이크로칩은 abort 를, PC 는 Disconnection 을 발생시킨다.

헤더		데이터
0 ... 3	4 ... 5	6 ...
0x1E3F5CD9	OpCode	Data

16비트 OpCode는 아래와 같이 구성된다. 각각의 OpCode마다 Data 영역이 서로 다른 구조체로 해석되며, 이는 차후 C++로 작성된 Host Interface Library에서 추상화한다.

모듈 기준 방향	Op Code	Name	Data align [in bytes] 데이터 영역의 바이트 정렬	Description
In/Out	0x0000	Test	[0..1] Data Length [2..] Data	Test packet
In	0x0001	Set Resolution	[0..1] Vertical Resolution [2..3] Horizontal -	Set resolution. Unit is Capture-per-degree. Core factor of capturing speed.
In	0x0002	Set FOV: Vertical	[0..1] FOV in integer	Set vertical FOV
In	0x0003	Set FOV: Horizontal	[0..1] FOV in integer	Set horizontal FOV
In	0x0004	Stop	-	Reset head location to init point. Stop capturing
In	0x0005	Start	-	Start capture
In	0x0006	Pause	-	Pause capture
In	0x0007	Report		Request report
Out	0x0001	Status	[0..1] Head Pitch Coord [2..3] Head Yaw Coord [4..x] Reserved	Output report
Out	0x0002	Log	[0..1] Error code [2..3] String Length [4..5] String	Send log
Out	0x0003	Row Data	[0..1] Row Index [2..3] Column Count [4..4*n] Floats	Module stores single row data into internal data buffer, and transmits data when capturing for each row is done.
Out	0x0004	Frame done	-	Every single frame capture.

소켓에 대한 입출력은 바이너리를 낮은 주소부터 그대로 byte stream 에 r/w 하므로, little endian 을 사용한다.

일단 모듈을 Start 명령을 통해 가동하면 시작 위치에서 수평으로 LIDAR 를 수행하게 되며, 한 행을 완성할 때마다 수직으로 지정된 해상도에 따라 일정 각도만큼 내려가며 캡쳐를 진행한다. 이 때, 열과 행은 끝에 다다를 때마다 방향을 바꿔 지그재그로 화면을 스캔, Idling 타임을 최소화한다.

이 때, 역방향으로 Row 스캔을 할 때에도 내부 버퍼는 정방향으로 정렬하여, 수신자가 보기에는 항상 정렬된 것처럼 보이게끔 하는 것이 중요하다.

Circuit

@todo.

Artwork

@todo.

Milestone

아래는 모듈의 개발 마일스톤이다.

Sequence	Category	Name	Description
1	ARM 프로세서	개발 환경 구성	STM32F4xx 칩을 컴파일하고 프로그램하기 위한 개발 환경을 구성한다.
		기능 파악	STM32F4xx 가 지원하는 기능을 대강 파악한다.
		인터럽트 연구	STM32F4xx 에서 인터럽트 핸들러를 작성하는 방법을 파악한다.
		타이머 작성법	STM32F4xx 에서 타이머를 사용하는 방법을 파악한다.
2		SPI - 기본	SPI 의 기본적인 사용법과 개념을 파악한다.
		SPI – 심화	다수의 Slave 를 컨트롤하는 방법을 파악한다.
	네트워크	장비 구성	W5500 모듈과 STM32F401VCD 키트를 연결한다.
		라이브러리 임포트	W5500 라이브러리의 기능을 파악하고, SPI 통신을 구성한다.

	SPI	멀티-슬레이브 큐	프로그램 내 다수의 모듈이 SPI 통신의 마스터로 동작하게끔 소프트웨어 버스를 작성한다.
			Example:
			`struct SPITransmitQueue {`
			`  uint16_t head;`
			`  uint16_t tail;`
			`  char buff[BUFFER_MAX];`
			`} SPITransmitQueue;`
			
			`enum SPISLAVE {`
			`  SPISLAVE_SENS,`
			`  SPISLAVE_ETH,`
			`  SPISLAVE_MAX`
			`};`
			`SPITransmitQueue QUEUE[SPISLAVE_MAX];`

	네트워크	에코 프로그램	PC에서 간단한 로컬호스트 서버 프로그램을 작성, W5500이 연결된 STM32F401VCD 키트에서 받은 데이터를 표시한다.
		프로토콜 디코더 작성	예의 프로토콜을 바탕으로, PC와 모듈에서 프로토콜 파서를 작성한다.
		Logging 구성	용이한 개발을 위해, STM32의 _write함수를 네트워크로 redirect, 표준 출력을 PC에서 받게끔 한다.

3	**깊이 센서**	연결	SPI로 MCU와 연결. 프로그램 내부적으로는, W5500과 더불어 두 개의 슬레이브를 사용해야 하므로, 내부 SPI 버스에서 센서 API를 감싸는 래퍼를 작성한다.
		테스트	지속적으로 센서에서 값을 읽어 이를 로그로 보고하는 테스트 프로그램을 작성한다. 이후 이 값을 평가 보드와 대조, 거리를 Meter 단위로 변환하는 로직을 작성한다.
	모터	스태핑 모터 연결	스태핑 모터를 연결하여, 간단한 구동 테스트를 한다. 특히, 고정밀을 요하므로 타이머와 마이크로스태핑을 도입, 일정 주기마다 일정 거리만큼 이동해 한 번씩 깊이를 캡쳐할 수 있게끔 인터페이스를 구성한다.
		센서 연결	스태핑 모터의 초기화가 가능하게끔, Pitch와 Yaw를 담당하는 두 개의 모터에 각각 근접 센서를 하나씩 연결하고, 초기화 시 정상적으로 모터가 영점에 도달하는지 확인한다. 이 때, 모터의 영점은 좌상단, (0, 0)으로 한다.
		센서 결합	센서와 두 개의 모터를 각각 결합한다. (하드웨어) 모터가 정지하는 시점과 센서가 계측하는 시점이 조밀하게 접하도록 Calibration을 수행한다.
4	**시스템**	코어	네트워크를 통해 전송받은 명령어를 바탕으로, 모터와 센서를 조작하는 코어 클래스를 작성한다.
		펌웨어 완성	위의 프로토콜과 명령어 세트를 모두 구현한다.
		호스트 라이브러리	호스트에서 사용할 수 있는 라이브러리를 작성한다. CMake를 이용하여 프로젝트를 구성, 크로스 플랫폼에 대비하고, 일단 Win32용 바이너리만을 빌드한다.
		호스트 어플리케이션	C++ MFC를 이용, 모듈로부터 깊이 데이터를 전달받아 화면에 그리는 간단한 어플리케이션을 작성한다.
5	**하드웨어**	회로도 작성	위의 개발 정보를 바탕으로 회로도를 작성한다. OrCAD 이용.
		Artwork 작성	위의 회로도를 바탕으로 아트웍을 작성한다. Allegro 이용.
		디버깅	PCB 기판에 부품을 실장하고, 모터 등의 하드웨어를 연결한다. 이후 개발 키트와 동일하게 작동하는지 검증한다.

기본적으로 Sequence 는 1+ Week 이다.