2020.03.20 이전

공용 클래스

Common

uint8\_t의 재정의 UINT 정의

시스템 딜레이 공용 변수 DELAYTIME = 100

스트림 통신 주파수 BAUDRATE = 9600

추상 클래스 - 순수 추상 클래스는 아님

Device - PinManager에 핀 등록, 핀 등록을 위한 번들 반환 코드

DistanceMeasurement - 측정, 거리 반환

시스템 관리 클래스 - Singleton 방식 구현

PinManager - 핀 등록 및 중복 등록 시 정보 ‘출력’, 핀의 High, Low 설정

StreamRecorder - 공용 출력 스트림 저장, 공용 출력 스트림에 문자열 출력

구체적 구현 클래스

UltraSonicSensor - DistanceMeasurement 클래스 상속. 측정 시간 10 ms당 58 m/s의 속력.

TfLidarSensor – DistanceMeasurement 클래스 상속. TF Mini Lidar Sensor 클래스 Wrapping형 구현. 시스템과 통신을 위한 SoftwareStream을 위한 Stream 보유.

MotorDriver - power가 음수일 경우 역방향, 양수일 경우 정방향 회전. abs 255 이내 값 설정 가능. 그 이상의 값은 255로 통일.

Relay – on, off 구현. (NO-NC)에 따라 반대로 작동.

ServoMotor - 각도 설정. 각도 추가 구현. 0~180까지 설정 가능. 180 modulus 연산하여 설정.

================================================================

2020.03.20

Spread상황을 제외한 UML, Flow Chart 작성 완료

각 클래스간 인터페이스 설계 필요

================================================================

2020.03.23

통신 부분 설계 완료. -> 추상화로 작성하였으므로 상세한 인터페이스는 무시해도 될 듯.

이더넷, 와이파이, 블루투스 등 통신 플랫폼을 추상화하기 위하여 CommunicationProtocol클래스를 통해 통신. 구현된 프로토콜이 장비간에 이동할 CommunicationPacket을 채워줌. 프로토콜 내부에는 하드웨어 사이의 통신에 필요한 프로토콜 정보가 들어있음.

StateFactory는 각 하드웨어가 State별로 작동할 action을 담고있는 클래스를 반환할 Factory패턴으로 구현.

CommunicationModule클래스는 Protocol과 StateFactory를 하드웨어별로 가지고있으면서 통신 후 상태 값을 반환해준다.

================================================================

2020.03.24

수정 사항

CommunicationModule과 연결된 StateFactory를 각 하드웨어와 연결로 변경. 하드웨어는 State패턴을 이용하여 구현. CommunicationModule은 추상 클래스 Data를 주고받음.

State는 인스턴스 생성이 아닌 상태에 따른 action의 변화이므로 StateFactory는 StateConfiguration으로 변경하고 Singleton방식으로 구현. configuration은 각 인스턴스마다 생성된다.

Observer, CentralSystem, Turret은 Hardware를 상속받으며 Hardware는 StateConfiguration만 변경될 뿐, State에 따라 변화된 action을 항상 run한다.

추가 사항

Turret은 builder패턴을 통해 구현된다. Turret은 크게 Turret에 달린 카메라와 연결, 관리하는 ViewModule, 외부와의 데이터 통신을 담당하는 CommunicationModule, Turret의 조작부를 추상화한 ControlModule, Turret의 low level 조작을 담당하는 DriveModule로 구성된다. Turret의 작동 절차는 CommunicationModule을 통해 받아온 명령을 ControlModule로 넘겨 처리 후, ControlModule이 처리된 데이터를 DriveModule로 넘기는 절차를 통해 작동한다.

================================================================

2020.03.25

수정사항

Hardware와 State 클래스를 command패턴으로 구현한다. 각각의 Hardware와 State는 HardwareType을 가지고 서로 일치할 경우에만 작동 가능하다.

UML상 클래스간 연결 관계 변경.

================================================================

2020.03.26

수정사항

Data클래스에 serialize(), deserialize() 기능 추가. 디버깅 필요.

================================================================

2020.03.27

수정사항

상호 참조 문제를 해결하기 위해 Hardware와 State를 참조하는 HardwareControlSystem클래스를 추가. HardwareControlSystem은 State가 변경되고, State는 Hardware를 넘겨받아 각 State에 따른 작업을 실행한다. -> HardwareControlSystem은 추후 빌더 패턴을 인수로 받아오도록 변경.

================================================================

2020.03.28

작업내용

UML다이어그램 정리.

Hardware는 Builder패턴으로 구현하고 각 Hardware가 소유하는 Module은 Factory패턴으로 구현.

================================================================

2020.03.29

작업 내용

CommonData 클래스 Serialize 디버깅 완료. Deserializing 디버깅 필요.

================================================================

2020.03.30

작업내용

CommonData클래스 deserialize 디버깅 완료.

TurretExample 구현 완료.

Observer 인터페이스와 설계는 Turret과 동일하게 구현.

================================================================

2020.04.02

수정사항

LinearData를 template형태로 변경 완료.

Observer를 구동시키는 DriveModule을 ObserverDriveModule로 독립. 기존의 DriveModule을 TurretDriveModule로 이름 변경.

ObserverDriveModule만의 함수 정의 완료.

화재 여부 감별(온도 점검, 필터링 등) 이전의 상태값 저장은 상위 클래스인 Observer에서 처리.

Factory와 Builder패턴은 실용성을 고려하여 삭제를 고민 중.

ServoMotor는 AngleMotor를 상속하도록 변경.

추가사항

Camera 클래스 정의 완료. 디버깅 필요.

================================================================

2020.04.04

수정사항

Template관련 지식 부족으로 LinearData를 int, char만을 위한 클래스로 변경.

================================================================

2020.04.07

작업사항

AMG8833 설계 완료. 디버깅 필요.

Src파일 내부의 오류가 있는 코드들을 분리하여 작동 가능한 모듈 라이브러리 추출 필요.

================================================================

2020.04.08

작업사항

amg8833 디버깅 완료.

Observer 설계 중.

HardwareStateCommon.h 생성. 이 파일에서 FSM에 필요한 상태 데이터를 저장.

Observer의 state만을 관리, 반환하는 ObserverStateOrder클래스 생성. 싱글톤 패턴으로 구현.

수정사항

FSM구현을 위해 HardwareControlSystem에 stateChange추가. State는 다음 상태 전이를 위한 값을 반환.

================================================================

2020.04.09

작업사항

메모리 누수가 예상되므로 전체 클래스에서 소멸자에 virtual 작업을 추가해야함.

함수 내부에서 static하게 생성한 메모리의 소멸에 대해 고민 필요.

Observer의 감지, 거리측정까지의 전반적인 설계 완료. 디버깅 필요.

수정사항

메모리 확보를 위해 Amg8833의 버퍼를 union형으로 수정하는 것을 고려 중.

Amg8833코드 변경. 디버깅 필요.

열원이 2개 이상일 경우 판정 방법 고민 필요.

열원이 화재지점 이상이 아니라 판단될 경우 무시할 수 있는 메모리 구조 고민 필요.

================================================================

2020.04.10

수정사항

Observer의 개략적인 디버깅 완료. 하드웨어 도착 시 조립 후 확인 필요.

================================================================

2020.04.12

작업사항

Turret관련 클래스 설계 개략 완료. 내부 구현 필요.

================================================================

2020.04.14

수정사항

Turret의 single thread를 위해 TurretState 변경을 wait->aim->check->wait or wait->aim->fire->wait로 변경.

Turret이 지상에 있고, 화재지점 역시 지상에 있을경우는 문제 없지만, Turret이 지상에 있을때 열원이 지상보다 높을경우, Turret이 지상보다 높을경우 유효한 장애물 측정이란게 무엇일지에 대한 고민이 필요.

일단은 Turret과 화재지점은 지상에 있다 가정하고, plain한 obstacle이 존재한다고 가정. Obstacle 뒤의 공간은 꽤 멀다고 가정한다.

delta curr와 prev와의 obstacle결정값은 50으로 하드코딩한다.

================================================================

2020.04.15

수정사항

Amg8833의 buffer를 union 형식으로 변경.

UML과 작업내용 동기화 완료.

Flowchart 정리 필요.

================================================================

2020.04.16

수정사항

Amg8833의 경우 카메라가 열원을 스캔을 할 때, 충분한 시간을 주지 않으면 자극이 충분히 되지 않았거나, 이전의 자극이 완전히 사라지지 않아서 겉보기에는 값을 읽어오는 것에 딜레이가 존재하는것처럼 보이는 경향을 보인다.

Amg8833의 scan이전에 delay수행. 적절한 delay시간에 대한 고민이 필요.

================================================================

2020.04.17

수정사항

Observer의 detailAiming에서 count로 나누는 부분을 삭제. 이로인해 최소 작동거리가 존재함. 현재는 1m 가정.

Observer의 거리 측정기 TF Lidar Sensor에서 출력값을 -3으로 내보내는 현상 존재. 파워 단락 후 재연결 시 정상적으로 읽어오기는 하는데 연결 문제인지, 전력 문제인지 원인을 확실히 분간할 수 없음. 추후 땜납 작업을 진행하여 연결을 확실시 한 후에도 문제가 발생한다면 전력부분을 보강하는것으로 작업 필요 및 점검 필요.

Amg8833의 최소 자극 시간은 500milli second로 계산됨.

================================================================

2020.04.20

작업사항

Continuous Servo Motor, Encoder, Electronic Speed Controller초안 작성. 디버깅 필요.

설명 추가

Continuous Servo Motor의 경우 torque와 RPM사이에는 동력의 1차원적인 변환 공식이 존재하지만, datasheet를 확인해본 결과 stall상황과 no load상황에서의 current가 변하는것으로 보아 하중이 가해지면 소모 동력이 변하는것으로 짐작되어 수식적인 RPM변화량을 계산하기는 힘들것으로 짐작됨. 결국 엔코더를 추가해야 할것으로 보임.

Encoder는 interrupt한 방식으로 각도를 측정할 예정.

Electronic Speed Conroller의 경우 calibaration을 통해 BLDC에 출력을 넣을때 min값일 경우 0V를 max값일 경우 최대V를, 그 사이의 경우 scale하여 주는것으로 짐작됨. 고로 모든 ESC에는 calibration과정을 통해 통일한 출력을 낼 수 있도록 하는 과정이 필요할것으로 짐작됨.

Servo에서 지원하는 PWM은 50hz단위로 작동하는데 이는 20ms단위이다. 하지만 입력은 microseconds한 단위로 들어가는데 이럴경우 서보모터의 최대 입력이라 생각되는 2000microseconds는 2ms밖에 되지 않는다. 이러한 단위의 차이로 Servo의 PWM을 잘못 이해하는 현상이 발생하기도 할 것으로 보인다. 일부 서보 모터들 중 최대 입력 pulse가 2300으로 나오는 경우가 있는데 이가 가능한 이유는 위의 단위값때문인 것으로 보인다.

================================================================

2020.04.21

작업사항

Encoder의 pulse측정은 interrupt한 방식으로 할 예정인데, 아두이노 시스템의 특성상 클래스 멤버 함수 이내에서는 attachInterrupt()가 어렵다. 이를 위해선 세밀한 구현을 하던가, static한 선언을 하는것이 필요한데 전자는 작업시간에 매우 많이 들것으로 짐작되고, 후자는 원하는 방향의 구현이 아니므로 바로 구현하는것은 포기하고, 대신 메인 소스코드에서 원하는 함수를 wrapping하는 방식으로 구현 후 setup부분에서 attachInterrupt하도록 한다.

ESC의 경우 calibration모드로 들어가기위해 high pulse를 넣기 전에 추가적인 시간이 필요한것으로 짐작하여 일단 delay(3000)을 미리 주는 방향으로 설정. 디버깅 필요.

================================================================

2020.04.23

수정사항

ESC의 calibration은 calibration이 되는 범위가 따로 존재하는것인지, 아니면 transmitter와 통신 시 주고받는 데이터가 특정한 것인지 알 수 없으나 초기 계획한대로 calibration하는것은 불가능하다고 판단되어 instance 초기에 calibration하는 것은 포기하고 1000~2000사이의 값으로 하드코딩하는 것으로 결정.

Encoder의 경우 interrupt에 메인 phase 1개(A 또는 B 상)만을 연결하고 다른 상은 일반 digitalpin에 연결하는 방향으로 구현하는것을 추천. 메인 phase 상은 CHANGE일때 interrupt하고 Z상은 RISING할 때 interrupt. 엔코더 datasheet에는 P/R값을 제공하는데 생성 시 이 값의 2배를 넣어줘야 정상적으로 작동.

ContinuousServoMotor의 경우 정밀한 각도제어는 불가능하다 판단. Encoder와 연결하여 사용은 가능하겠지만, 독립적으로 사용하여 전압에 기반한 회전속도와 작동 시간을 결정하는 방식으로 각도를 조정하는것은 불가능.

================================================================

2020.04.26

작업사항

encoder의 interrupt 반응속도를 올리기위해 pin change interrupt를 성공했다. 하지만 여전히 고속으로 회전 시 오차를 만들어내는것을 확인할 수 있었다. 다만 이 문제가 interrupt가 실행되는 와중에 interrupt가 발생해서 벌어지는 현상인지, 아니면 다른 문제로 인한것인지는 아직 입증되지 않았다.

encoder counter라는 pulse를 측정해주는 하드웨어도 존재한다고 하니 이에 대해 고려해볼만한 것 같다. -> 정말 그런 하드웨어가 존재하는가?

pin change interrupt를 사용 시 PCINTn\_vect를 사용하는데, 이럴 경우 softwareSerial과 충돌이 발생하여 둘을 함께 사용할 수는 없다. pin change interrupt자체도 꽤 유용하기 때문에 포기하기는 쉽지 않은 기능이다. SoftwareSerial의 UART기능을 직접 구현하는 것을 고려해야 한다.

Atmega328p의 pin을 digitalread함수보다 직접 읽는방향도 고려해야 할 듯 하다.

전대 개발자의 경우 4000 tick/second까지가 한계라고 한다. CHANGE state로 할 경우 12000pulse가 발생하는데 확실히 이게 문제일 수도 있다. <http://www.hessmer.org/blog/2011/01/30/quadrature-encoder-too-fast-for-arduino-with-solution/>

“The microcontroller on the Arduino board is capable to count pulses and determine pulse length without software interrupt.” 이런 구문이 존재한다. 좀 더 알아봐야 한다.

<https://forum.arduino.cc/index.php?topic=158753.0>

<http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-frequency-counter-library/index.html>

<https://www.google.com/search?client=firefox-b-d&sxsrf=ALeKk01Dfa1X-qmtSxLenWCLihz3-h9R_g%3A1587914837967&ei=VailXqmqOsGNoASRwpLgAg&q=atmega328p+measure+pulse+count&oq=atmega328p+measure+pulse+count&gs_lcp=CgZwc3ktYWIQAzoECCMQJzoGCAAQDRAeUNkqWP1OYOFRaANwAHgAgAHoAYgB8ReSAQYwLjE4LjGYAQCgAQGqAQdnd3Mtd2l6&sclient=psy-ab&ved=0ahUKEwjpqObes4bpAhXBBogKHRGhBCwQ4dUDCAs&uact=5>

================================================================

2020.04.28

작업사항

기본 코드 구현. 엔코더와 BLDC모터를 연결할 방법을 찾아야한다.

================================================================

2020.04.29

작업사항

TurretDriveModule의 fire함수 구현. 터렛의 pitch축 모터가 양쪽으로 존재한다. 실 구현시 이부분 감안 필요. 양쪽으로 존재하므로 한쪽이 0~180이면 반대쪽은 180~0이다.

Turret의 readSignal과 writeSignal부분을 임시로 채워넣음. 통신 모듈 구현 시 교체 필요.

LinearMotor구현 완료.

StepMotor구현 필요.

================================================================

2020.05.01

작업사항

stepper motor의 경우 기존의 stepper클래스를 사용해도 될것으로 판단됨.

DC-Encoder Motor의 PID제어를 성공(지섭). Modulize작업 필요.

================================================================

2020.05.02

수정 사항

필요 없을것으로 예상되는 imLibrary의 PinManager, common 파일 삭제. 수정 과정 중 이상유무 점검 필요.

================================================================

2020.05.03

작업사항

DcEncoder 모듈화 작업 완료. 디버깅 필요.

================================================================

2020.05.07

작업사항

데이터 송수신 프로토콜 : 24 바이트 패킷

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start clock | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Signal type | | reserve | reserve | reserve | reserve | O2C | C2O | C2T | T2C |
| Source ID | | 1 byte | | | | | | | |
| Destination ID | | 1 byte | | | | | | | |
| Order | O2C | reserve | reserve | reserve | reserve | reserve | reserve | Fire signal | Extinguish signal |
| C2O | reserve | reserve | reserve | reserve | Init signal | reserve | reserve | Impact area check request |
| C2T | reserve | reserve | reserve | reserve | Init signal | Obstacle check request | Data for launch | Extinguish signal |
| T2C | reserve | reserve | reserve | reserve | reserve | reserve | Return obstacle check data | Launch signal |
| option | | reserve | reserve | reserve | reserve | reserve | Separate finish | separate | ACK |
| seed | | 1 byte | | | | | | | |
| data | | 4 byte \* 4 (dataType + ((alpha, beta, radius) or (x, y, z) or linearData etc)) | | | | | | | |
| Column parity | | 1 byte | | | | | | | |

O : Observer, C : Central System, T : Turret, reserve = 0, column parity : 짝수 패리티

If ACK = 1, compare seed data

Init signal send hardware serial number. It is similar as DHCP protocol.(구현 예정 없음)

2 way handshake 방식 구현 예정. Seed will transporte to compare.

Seed is randomly initialized. If packet is separated then seed will grow up.

If ID is 0b00000000, it means centralSystem address.

================================================================

2020.05.08

작업사항

CommunicationModule 전송부 완료. 디버깅 필요. 수신부 설계 필요.

================================================================

2020.05.12

작업사항

communicationModule 초안 작성 완료. 디버깅 필요.

================================================================

2020.05.13

조사내용

const 위치에 따른 상수화 결정

<https://stackoverflow.com/questions/5503352/const-before-or-const-after>

Class 내부 변수의 const static

<https://cookierobotics.com/032/>

<https://finsternis.tistory.com/167>

================================================================

2020.05.17

조사내용

delete처리를 할 경우 포인터가 nullptr를 가리키지 않고 이전의 주소값을 그대로 가진다고 한다. Delete하는 포인터들을 nullptr하도록 변경 필요.

작업내용

패킷의 패키징 디패키징 작업 완료.

================================================================

2020.05.19

작업내용

Server 초안 작성 중.

================================================================

2020.06.01

작업내용

Request계열 함수 작업 필요.

Observer\_Turret\_LinkedList 클래스 작업 필요.

================================================================

2020.06.02

작업내용

Turret에게 Obstacle Check를 요청할 경우 Shperical한 데이터로 통신. Radius = shortest distance

Call by pointer보다 call by reference형식으로 바꾸는것이 좋지 않을까 고민중.

Server초안 작성 완료. 디버깅 필요. 적절한 포구초속과 장애물에 걸리는지 여부를 결정할 수 있는 알고리즘 필요.

================================================================

2020.06.03

작업내용

CommonMath 구현 필요. CommonMath와 관련된 Server코드 주석 처리.

추상 클래스 CommunicationModule로 인한 Server생성자 부분에 주석처리.

Test Code를 먼저 작성하여 인터페이스를 먼저 정의한 후 코드를 구현하는 방식으로 구현 고민 중.

================================================================

2020.06.04

작업내용

#if defined방식을 이용하여 하나의 소스코드를 다수의 컴파일을 이용하여 테스트 코드의 유연성을 확보하는 방식 적용을 고민중.

Server 디버깅 완료. 테스팅 필요.

================================================================

2020.06.05

수정사항

CommunicationModule을 debuging 모드를 추가하여 CommunicationWithConsole이 제대로 작동하는지 확인 필요. 확인 완료.

Server Debuggin모드 추가하고 테스팅 필요. 추가 완료.

================================================================

2020.06.08

수정사항

CommunicationModule관련 수정사항 발생. Test코드 수정 필요. 수정 완료.

Server debuging작업 중.

================================================================

2020.06.09

작업사항

Server의 디버깅 결과 일단은 정상적으로 작동하는것으로 보인다. 확실한건 정확한 실험 데이터 설정이 필요할것으로 보임.

atan 출력 각도가 1도씩 모자르게 출력됨. 디지털 계산상의 오차로 인한것으로 짐작됨. 확인필요. -> 이유를 알 수 없이 문제가 해결됐다...

================================================================

2020.06.10

작업사항

ManualCommunicationModule작업 필요. 작성 완료 시 turret, observer, server에 장착하여 테스트 작업 필요.

StepMotor 작성 필요.

================================================================

2020.06.11

작업사항

ManualCommunicationModule response 처리 작업 필요.

================================================================

2020.06.12

확인사항

통신 중 오류가 있는 데이터 수신 시 재송신을 요청 또는 대기하는 구문이 존재하는가?

================================================================

2020.06.13

작업사항

ManualCommunicationModule을 server에 적용하여 만족스럽게 작동하는것을 확인. 구현방식이 동일하므로 Observer와 Turret에서도 잘 작동할것으로 예상됨. 다만, 하드코딩으로 넣은 내용으로 인해 메모리 용량이 초과하는 현상이 발생. 이에 대한 고민 필요.

================================================================

2020.06.16

작업사항

Observer에 ManualCommunicationModule을 연결 후 테스트를 진행하려 했지만 무한 인스턴스 생성 현상 발생. 메모리 초과로 인한 Arduino의 reset이 원인. 문자열을 극단적으로 줄여 작동은 가능하게 작성완료. 다만 실 테스트 시 가독성이 어떻게 발생할지는 관찰 후 재작업이 필요할 듯.

================================================================

2020.06.17

작업사항

Observer 테스팅 작업을 수행 중 ServoMotor가 매우 심하게 진동하는 것을 확인. 조사결과 소모전류량의 증가에 따른 전압 강하로 인해 발생하는 것을 확인. 이에 대한 보완책으로 regulator 또는 step motor가 가능한 것으로 보임.

완전하게 PC와 독립된 상태로 통신을 수행할 경우 MCU 및 모터 이외의 센서들 역시 외부 전원으로부터 연결되야 하므로 regulator는 기본적으로 필요할 것으로 예상됨.

다만 그러한 상황 발생 시 사용할 regulator와 현재 servo motor에 필요한 regulator의 전압이 서로 상이할 수 있으므로 굳이 지금 당장 regulator에 대해 고민할 필요는 없을 것으로 보임.

또한 이전부터 step motor로의 교체가 예정되어 있던 상황이므로 지금 도입하는 것이 굳이 나쁘지는 않다고 판단됨. 물론 현재 존재하는 보급형 step motor로 테스트해본 결과 역시 소모 전류량이 증가하면 전압강하 현상이 발생하므로 regulator는 여전히 필요할 것으로 짐작됨.

이러한 이유로 현재 존재하는 보급형 step motor를 기반으로 새 하드웨어를 출력 후 테스팅을 진행한 후, 문제 발생 시 대체할만한 step motor를 찾아보기로 결정.

CommunicationModule과 상태 절차를 변경한 Observer의 디버깅은 하드웨어 출력까지 대기 상태.

Step Motor를 위한 library 작성 완료.

Step Motor Driver를 위한 library 작성 필요.

Turret에 CommunicationModule 설치 및 전체 테스팅 작업 필요.

================================================================

2020.06.18

작업사항

스텝모터도 전압강하시 고속으로 돌지 못하는 경우가 존재한다. 레귤레이터는 역시 필요하다.

스텝모터, 라이더, 열감지센서의 작동 확인 완료.

리니어, bldc 작동 확인 필요.

Step motor driver 작업 필요

================================================================

2020.06.19

작업사항

LidarSensor 2가지 유형 모두 호환 되는 것을 확인.

리니어 모터, ESC작동 확인.

Step Motor 재확인 필요.

================================================================

2020.06.20

작업사항

step motor, driver작업 완료.

================================================================

2020.06.24

작업사항

현재 발견한 블루투스 모델이 bluetooth 4.0으로 broadcasting이 가능하지만 pairing을 위해서는 작업이 필요함. 확장성을 생각하면 broadcasting기능이 유용할 것으로 판단됨. 이를 위해 아두이노에서 지원하는 라이브러리를 검토해본 결과 uno보드를 사용시 매우 많은 메모리를 차지하여 기존의 코드와 병합시 메모리가 부족한 것이 확인됨.

이를 위한 해결책으로 첫번째는 메가보드를 사용하는 것인데, 이럴 경우 uno보드를 기반으로 작성해둔 코드가 어떻게 작동할지 예상할 수 없음. 두번째로 직접 bluetooth의 패킷을 채워서 SoftwareSerial로 송수신하는 방법이 있는데 이는 작업에 필요 이상의 시간이 소모될 것으로 예상됨. 세번째 방법은 BLE의 기본 송수신 모드인 beacon에 데이터를 쓰는 방식을 사용하는 것인데, 현재로서는 불가능하다 판단됨, 추가적인 조사 필요. 마지막으로 기존 bluetooth 2.0방식과 동일한 pairing작업을 기본으로 하여 말단 하드웨어들은 Server의 pairing데이터만 가지고있고, Server는 관리하는 모든 말단 하드웨어들의 pairing 정보를 저장한 후, 이 pairing 작업을 연결-해제 하는 방식으로 통신하는 방법이 존재. 현재 이 방법이 가장 구현 가능성이 높은 것으로 판단됨.

하드웨어 완성을 기다리기보다는 모듈단위의 테스트를 먼저 시행해야 할것으로 생각됨. 블루투스 연구 중단 고민 중.

Observer는 이미 한번 검증한 코드를 기반으로 짜였기 때문에 module단위 테스트가 필요 없어 보임. 다만 step motor적용 시 detailAiming구문에서 최소 회전 각도보다 적은 각도 이동을 요구할 경우 회전하지 못하고 반복문을 탈출하지 못할 가능성이 존재. 총 조립 완료 후 정상 작동 검증 필요.

Turret의 check, launch state 점검 필요. OptimumPathAlgorithm이 추가될경우 driveModule에서 setPower부분 update 필요.

Server의 좌표계 변경 함수 점검 필요. OptimumPathAlgorithm update필요.

================================================================

2020.06.25

작업사항

Uno에서 작업 시 메모리 부족현상이 발생, 코드수정을 통해 해결하는 것은 불가능할 것으로 판단되어 mega로 변경 결정.

Mega를 사용하여 observer의 각 센서들 작동 여부 테스트 결과 정상 작동 확인. Turret에서 사용하는 센서들 역시 mega에서 문제가 생길 부분은 공통적이라 판단했으므로 Turret에서도 mega의 사용이 가능할 것으로 예상됨. 이에 따라 이전에 판단을 보류했던 ArduinoBLE라이브러리의 사용 역시 가능할 것으로 판단됨. 다만 선 접속에 문제가 있는것으로 짐작되어 이 문제가 해결 시 observer의 전체 알고리즘 테스트.

Server에서 좌표계 변환 작업을 독립 모듈로 결정. CustomMath 파일에서 ThirdDimensionMath namespace에서 연산. 컴파일 완료. 정상 작동 확인 필요.

================================================================

2020.06.26

작업사항

Server optimum Velocity and pitchAngle 이외의 fire 구문 연산 정상 작동 확인. 다만 int단위로 데이터가 오가느라 약간의 오차 존재.

Observer 문제 해결. 전체 디버깅 과정 남음.

아두이노간 블루투스를 통한 통신 성공. Master-slave간 명령어가 다르다는 점을 주의해야 한다.

================================================================

2020.06.30

작업사항

Observer의 전체 절차가 정상 작동하는 것을 확인. Manual한 packet의 데이터 입출력을 읽을 수 있는 상황. 정밀 테스트가 가능한 것으로 확인됨.

================================================================

2020.07.01

작업사항

Bluetooth의 재연결 대신 relay와 연결하여 재부팅 가능성 확인. Bluetooth의 ground와 relay의 common을 연결하고 다시 NC를 공용 ground와 연결해줘야 정상 작동됨.

Observer 작동 절차 확인 완료.

Turret 작동 절차 확인 필요.

================================================================

2020.07.02

작업사항

Turret 절차 정상 작동 확인.

================================================================

2020.07.03

작업사항

Server 최적 포구 초속과 최적 각도 계산 식 이식 완료.

다만 소수점 버림으로 인한 오차 존재. 계산이 누적될 경우 오차가 점점 커지는 현상 존재.

================================================================

2020.07.13

고려사항

Bluetooth에서 slave 모드일 때, at+type1을 적용해야 pin이 일치할 경우만 연결 가능. -> 불가능.

Bluetooth에서 master모드일 때, at+cona [para]에서 mac주소를 기반으로 연결 가능. Mac주소는 0x부터 시작해야 한다.

작업가정

초기화를 위한 패킷 설계, 패킷 검증 절차 작성이 불필요하게 많은 시간이 소요될것으로 짐작된다.

Bluetooth를 이용한 통신은 mac주소를 미리 알고있다고 가정한다.

Hardware간의 초기화 작업은 생략한다.

master, slave 등 at명령어를 통한 설정은 라이브러리 작업 이전에 미리 설정한 후 이용한다.

작업 예정

step motor에서 0으로 초기화 할 경우 정확히 0으로 가지 않는 현상 존재.

소수점 버림으로 인한 오차 누적 최소화 작업 필요.

================================================================

2020.07.14

작업사항

Bluetooth 작성 완료.

================================================================

2020.07.15

작업사항

블루투스 firmware업데이트 방법 확인.

V540으로 업데이트 시 명령어는 항상 대문자이어야하고, line ending없음 상태로 입력해야 한다.

================================================================

2020.07.20

작업사항

HM10 구버전은 Pin connect자체가 불가능. Pin 통신을 하고자 한다면 firmware update를 실시해야 함.

작업절차

각 블루투스 모듈은 미리 설정을 해놓아야 한다.

Slave

AT+ROLE0 //slave 모드.

AT+TYPE1 // pin을 통한 연결 설정.

AT+PASS\*\*\*\*\*\* // master와 같은 값이어야 연결 가능.

AT+RESET // reset해야 설정 적용.

Master

AT+IMME1 // 미설정 시 파워가 인가될때마다 자동으로 이전 연결되기 때문에 이를 막기위해 설정.

AT+ROLE1 // master 모드.

AT+PASS\*\*\*\*\*\* // slave와 같은 값이어야 연결 가능.

AT+AFTC3FF // STATE, EN 핀을 통해 연결 여부 확인 가능.

AT+RESET // reset해야 설정 적용.

비밀번호 : 191225

연결 절차

Master에서 AT+CON[MAC ADDR]입력 시 연결.

연결 해제

AT 입력 후 AT+RESET 바로 입력.

주의사항

AT command로 넘어간 명령에 대한 응답은 buffer에 쌓인다. 이를 무시할 수 있는 구문이 필요.

==========================================================

2020.07.21

작업사항

BluetoothCommunicationModule초안 작성 완료.

Sender와 receiver가 나눠지는 이중 단방향 통신 형태로 구현 예정.

Receiver List

Observer : 6CC374FE0372, BLUE

Server : 6CC374FE967F, RED

Turret : 6CC374FE9E52, GREEN

==========================================================

2020.07.22

고려사항

Keyestudio의 블루투스의 경우 pin pairing이 가능하지만, firmware를 업데이트한 기존의 블루투스는 pin pairing이 불가능하다. Firmware 버전이 부족해서 그런거일 수 있지만 현재로선 업데이트가 힘드므로 keyestudio의 블루투스를 slave하게 사용해야 할 듯 하다.

Keyestudio의 블루투스에서 연결을 해제할 경우 state핀이 원상복귀가 되지 않는 현상 존재. zs 040에서 연결을 해제하면 정상적으로 핀상태가 변경된다.

또한 zs 040을 slave로 할 경우 pairing이 되어도 양방향 통신이 불가능했지만, keyestudio를 slave로 설정하면 양방향 통신까지 가능하다.

==========================================================

2020.07.23

작업사항

Serialize과정에서 문제 발생.

bluetooth pairing과정에서 데이터 자체는 정상적으로 전송되는 것을 확인.

START\_CLOCK ^ c != 0b00000000은 왜 참이 아닌가?

Response는 제대로 작동하는가?

==========================================================

2020.07.25

작업사항

SoftwareSerial은 두개가 동시에 작동하지 않는다.

<https://www.arduino.cc/en/Tutorial/TwoPortReceive>

==========================================================

2020.07.27

작업사항

블루투스간의 통신 완료.

listen 뿐만 아니라 AT명령어간 적절한 delay가 있어야 연결 여부 반영이 가능.

직렬화 과정에서 발생하는 문제 해결. 아두이노는 16비트 프로세서로 int의 사이즈도 2바이트이고, vtable의 크기도 2바이트이다. 직렬화 과정에서 vtable의 크기를 제거하려 할때, 기존의 프로세서 환경으로 짐작하여 4바이트를 줄인것으로 인해 직렬화된 데이터의 마지막 2바이트가 오염된 것으로인해 발생한 현상.

StepMotor구조 변경. StepMotorDriver 작동 확인 필요. 개선된 사항은 없음.

==========================================================

2020.07.28

작업사항

StepMotorDriver정상 작동 확인.

Turret과 Turret Muzzle부위는 I2C방식으로 통신 예정. 코드 작성 완료. 테스팅 필요.

==========================================================

2020.08.11

작업사항

엔코더 코드 분석 및 초벌 작성이 무난히 적용되는 듯 함. 다만 BLDC 모터에서 발생시키는 최대 interrupt가 초당 66,666회정도 일것으로 예상되는데 이 경우 아두이노가 정지할 가능성이 있음. 이 때 최대 처리 가능 명령어 수가 약 240개 정도 되는데, 이런 상황이 발생할 경우 asm으로 코드를 작성하는 것을 고려해야 함. -> change가 아닌 rise나 falling일때 interrupt하여 횟수를 절반으로 줄이면 무난하게 적용이 가능할 듯 함.

추가적인 조사결과 softwareSerial과 interrupt의 충돌은 pcint일때 발생하는 것으로 생각되어 일단은 external interrupt만 사용하여 기존의 엔코더 라이브러리를 적용하는 것을 고민 중.

UV, SoundModule의 Module화 초벌 작업 완료. 디버깅 필요.

진화가 작성한 코드와 병합 과정 필요.

==========================================================

2020.08.12

작업사항

진화가 작성한 코드와 통합 완료. 정상 작동 확인 필요.

==========================================================

2020.08.13

수정사항

장애물 점검 절차 삭제.

================================================================

2020.08.19

작업사항

step motor와 encoder를 함께 사용할때 interrupt가 너무 많이 발생하면 step motor가 정상적으로 작동하지 않는다.

Stepper motor, encoder에서 올림처리 되는 것에 대한 작업 필요.

================================================================

2020.08.20

작업사항

encoder, stepper motor 반올림 처리 작업 완료.

================================================================

2020.08.21

작업사항

Turret과 Server간의 통신 확인 완료.

================================================================

2020.08.24

작업사항

Server 메인 연산 점검 완료. 이 상황에서 이제 문제가 발생 시 옵저버에서 던지는 데이터의 정합성을 먼저 확인해봐야 할듯.

================================================================

2020.08.26

작업사항

Observer에서 impact check 요청이 들어올 시 거리 측정이 안되는 현상 발생. SoftwareSerial의 문제인 것으로 짐작되어 listen구문 추가함. 제대로 작동하는지 점검 필요,

Server에서 Obserer와 Turret간의 통신을 할때 블루투스 모드로 할 경우 observer로 impact area check요청이 안들어가는 현상 발생.

블루투스간 통신 시 연결을 해제할때 slave에서 연결을 해제하는 것에 대해 고민이 필요할 듯.

================================================================

2020.08.27

작업사항

26일 작업으로부터 branch 됨

observer, turret, Server의 send실패 시 재송신 구문 추가.

BluetoothCommunicationModule의 연결-해제 절차 변경. Receiver에서 연결 해제 요청.

Bluetooth의 read에서 count이상의 시간이 걸릴경우 0을 반환하도록 변경.

CommunicationModule에서 전송 실패 시 3회 시도하도록 변경.

================================================================

2020.08.28

작업사항

26일자 통신 문제 해결. Disconnect후 충분한 시간이 주어지지 않아서 자신의 state가 업데이트 되기 이전에 새 통신을 요청하는 과정에서 state핀이 high이므로 새 블루투스로 연결하는 단계를 건너뜀으로서 발생한 문제. Connect이전에 delay시간을 줌으로써 해결.