

기술사업화지원사업 제3차년도 최종 보고서

발 간 등 록 번 호

11-1543000-002897-01

스마트 팜 표준화를 위한 기업의 기술개발 지원(CMO)

- 최종보고서 -

2019. 11. 14.

주관연구기관 / (사)한국농식품ICT융복합산업협회
협동연구기관 / 한국전자통신연구원

농 립 축 산 식 품 부
농림식품기술기획평가원

스마트 팜 표준화를 위한 기업의 기술개발 지원(CMO) 최종보고서

2019.11.14

농림식품기술기획평가원부
농업식품기술기획평가원부

제 출 문

농림축산식품부 장관 귀하

본 보고서를 “스마트 팜 표준화를 위한 기업의 기술개발 지원(CMO)” (개발기간 : 2016. 09. 05 ~ 2019. 09. 04) 과제의 최종보고서로 제출합니다.

2019. 11. 14.

주관연구기관명 : (사)한국농식품ICT융복합산업협회 회장 박현춘 (인)
협동연구기관명 : 한국전자통신연구원 원장 김명준 (인)

주관연구책임자 : 정영환
협동연구책임자 : 허미영

국가연구개발사업의 관리 등에 관한 규정 제18조에 따라 보고서 열람에 동의합니다.

보고서 요약서

과제고유번호	816022-3	해 당 단 계 연 구 기 간	3차년도	단 계 구 분	3차년도/ 3차년도
연구사업명	단 위 사 업	농식품기술개발사업			
	사 업 명	기술사업화지원			
연구과제명	대 과 제 명	(해당 없음)			
	세부 과제명	스마트 팜 표준화를 위한 기업의 기술개발 지원(CMO)			
연구책임자	정영환	해당단계 참여연구원 수	총: 3명 내부: 3명 외부: 명	해당단계 연구개발비	정부: 467,000천원 민간: 천원 계: 467,000천원
		총 연구기간 참여연구원 수	총: 6명 내부: 3명 외부: 3명	총 연구개발비	정부: 467,000천원 민간: 천원 계: 1,200,000천원
연구기관명 및 소속부서명	(사) 한국농식품ICT융복합산업협회				
국제공동연구	(해당 없음)				
위탁연구	(해당 없음)				
※ 국내외의 기술개발 현황은 연구개발계획서에 기재한 내용으로 같음					
연구개발성과의 보안등급 및 사유	일반				

9대 성과 등록·기탁번호

구분	논문	특허	보고서 서원문	연구시설·장비	기술요약 정보	소프트 웨어	생명자원		신품종		
							화합물	생명정보	생물자원	정보	실물
등록·기탁번호	①ISSN1027-1030	①10-2034406-00-00	최종평가이후발간등록번호기재	(해당없음)	①표준화 컨버터 개발	134121-0006994	(해당없음)				
	②ISSN2233-4890				②자료 분석 장치 개발						
	③ISSN2233-3075(print)	②TTAKKO-10.1044			③스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용						

□연구의 목적 및 내용

○ 목적

표준 통신 기반 스마트팜 서비스의 상호운용·호환성을 높이기 위하여, 기존 스마트팜 업체들에게 표준의 중요성에 대한 인식을 확대하고, 레거시 제품들에 적용가능한 표준 컨버터 개발하며 업체들의 참여를 촉진함으로써 스마트팜 표준화의 확산에 기여한다.

○ 연구내용

- (1) 스마트팜 표준화 확산 지원
- (2) 표준화 컨버터 개발 : 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발
- (3) 자료 분석 장치 개발
- (4) 스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용

□ 연구개발성과

○ 본 연구개발에서는 다음과 같은 정성적 성과목표를 달성하였다.

- (1) 워크샵 및 간담회(전문가 회의) 개최
- (2) 국제·국내 농기자재 박람회 참여
- (3) “농식품ICT융복합가이드북” 발간으로 국내 스마트팜 기업홍보 및 표준인식제고

○ 본 연구개발에서는 다음과 같은 정량적 성과목표를 달성하였다.

- (1) 지식재산권 출원 : 5건 / 지식재산권 등록 : 2건
- (2) 사업화 제품화/매출액 : 3건, 국내 238백만원
- (3) 고용창출 : 3명
- (4) 논문 : 3건
- (5) 학술 발표 : 2건
- (6) 표준화 : 1건

□ 연구개발성과의 활용계획(기대효과)

가. 활용계획

○ 스마트팜 표준의 확산

- (1) “농식품ICT융복합가이드북” 을 발간하여 스마트팜 기업홍보 및 표준 인식제고
- (2) 표준의 지속적 제정

○ 스마트팜 표준 기반 제품 개발 가이드 역할 수행

- (1) 오픈소스로 표준컨버터와 드라이버 API 개발 소스 공개
- (2) 오픈소스 사이트를 지속적으로 운영하며 업체들의 표준 전환 시 기술 참조 지원

나. 기대효과

- 농업인의 스마트팜 제품 구매 시 중복투자 제거
- 4차 산업혁명 시대에 걸맞는 다양한 대농업인 서비스 개발 촉진
- 표준 기반 제품으로 글로벌 시장 진출 도모

보고서 면수
(192페이지)

<요약문>

<p>연구의 목적 및 내용</p>	<p>○ 목적 표준 통신 기반 스마트팜 서비스의 상호운용·호환성을 높이기 위하여, 기존 스마트팜 업체들에게 표준의 중요성에 대한 인식을 확대하고, 레거시 제품들에 적용 가능한 표준 컨버터를 개발하며 업체들의 참여를 촉진함으로써 스마트팜 표준화의 확산에 기여한다.</p> <p>○ 연구내용 (1) 스마트팜 표준화 확산 지원 (2) 표준화 컨버터 개발: 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발 (3) 자료 분석 장치 개발 (4) 스마트팜 내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용</p>																				
<p>연구개발성과</p>	<p>○ 본 연구개발에서는 다음과 같은 정성적 성과목표를 달성하였다.</p> <ul style="list-style-type: none"> - 표준화 추진을 위한 워크샵 및 간담회(전문가 회의) 개최 - 국제·국내 농기자재 박람회 참여 - “농식품ICT융복합가이드북”을 발간하고 국내 스마트팜 기업홍보 및 표준인식제고 <p>○ 본 연구개발에서는 다음과 같은 정량적 성과목표를 달성하였다.</p> <table border="1" data-bbox="451 952 1402 1193"> <thead> <tr> <th>지식재산권(출원)</th> <th>지식재산권(등록)</th> <th>표준화</th> <th>기술이전</th> </tr> </thead> <tbody> <tr> <td>5건</td> <td>2건 (+4건 예정)</td> <td>1건 (+1건 진행중)</td> <td>30건</td> </tr> <tr> <td>소프트웨어 등록</td> <td>사업화(제품화)</td> <td>논문</td> <td>고용창출</td> </tr> <tr> <td>1건</td> <td>3건 (국내매출 238백만원)</td> <td>3건</td> <td>3명</td> </tr> </tbody> </table>					지식재산권(출원)	지식재산권(등록)	표준화	기술이전	5건	2건 (+4건 예정)	1건 (+1건 진행중)	30건	소프트웨어 등록	사업화(제품화)	논문	고용창출	1건	3건 (국내매출 238백만원)	3건	3명
지식재산권(출원)	지식재산권(등록)	표준화	기술이전																		
5건	2건 (+4건 예정)	1건 (+1건 진행중)	30건																		
소프트웨어 등록	사업화(제품화)	논문	고용창출																		
1건	3건 (국내매출 238백만원)	3건	3명																		
<p>연구개발성과의 활용계획 (기대효과)</p>	<p>가. 활용계획</p> <ul style="list-style-type: none"> ○ 스마트팜 표준의 확산 <ul style="list-style-type: none"> - 표준화 추진을 위한 워크샵, 간담회, 농식품ICT융복합가이드북 제작 등으로 스마트팜 홍보 및 기업과 농업인의 표준에 대한 인식제고 - 표준의 지속적 제정 ○ 스마트팜 표준 기반 제품 개발을 위한 가이드 개발 <ul style="list-style-type: none"> - 오픈소스로 표준컨버터와 드라이버 API 개발소스 공개 - 오픈소스 사이트를 운영하며 업체들의 표준 전환 시 기술 개발 지원 <p>나. 기대효과</p> <ul style="list-style-type: none"> ○ 농업인의 스마트팜 제품 구매 시 중복투자 제거 ○ 4차 산업혁명 시대에 걸맞는 다양한 대농업인 서비스 개발 촉진 ○ 표준 기반 제품으로 글로벌 시장 진출 도모 																				
<p>국문핵심어 (5개 이내)</p>	스마트팜 표준화의 확산	표준 통신 기반 스마트팜 서비스	표준 컨버터 개발	스마트팜 표준 기반 제품 개발 가이드	4차 산업혁명																
<p>영문핵심어 (5개 이내)</p>	Spread of Smart Farm Standardization	Standard communication based smart farm service	Standard converter development	Smart Farm Standards-Based Product Development Guide	4th Industrial Revolution																

※ 국문으로 작성(영문 핵심어 제외)

목 차

1장 연구개발과제의 개요	9
1절 연구개발의 필요성	
1. 산업 및 정책 필요성	11
2. 기대효과	15
2절 연구개발 범위	
2장 연구수행 내용 및 결과	19
1절 연구수행 내용	
1. 연구개발 추진전략 및 추진체계	21
2. 연구개발 내용	23
가. 스마트팜 표준화 확산 지원	24
나. 표준 컨버터 개발 : 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발	45
다. 자료 분석 장치 개발	120
라. 스마트팜 내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발	125
2절 연구개발 성과	
1. 지식재산권	184
2. 논문게재 성과	184
3. 학술대회 발표성과	185
4. 기술요약정보	185
5. 소프트웨어 등록	186
6. 사업화(제품화)	186
3장 목표 달성도 및 관련 분야 기여도	187
4장 연구결과의 활용 계획 등	191
붙임. 참고 문헌	194
별첨.	
[별첨 1] 연구개발보고서 초록	197
[별첨 2] 자체평가의견서	199
[별첨 3] 연구평가활용계획서	203

1장

1장 연구개발과제의 개요

1절 연구개발의 필요성

1. 산업 및 정책 필요성
2. 기대효과

2절 연구개발 범위

1장 연구개발과제의 개요

1절 연구개발의 필요성

1. 산업 및 정책 필요성

가. 스마트팜 확산을 위한 핵심이슈

(1) 스마트팜 운영 농업인 현장 애로사항

(가) 농촌진흥청 주관으로 「스마트팜 빅데이터 활용 생산성 향상 모델 발표회 및 발전방안 간담회」를 개최하여 농업인, 스마트팜 기업체, 협회 관계자 등 150여 명이 참석한 가운데 스마트팜 운영 현장 애로사항, 향후 발전 방안 등 논의(2018.12.20.)

(나) 스마트팜을 운영하고 있는 농업인들은 기자재의 비표준화로 인한 낮은 호환성, 기자재의 낮은 품질로 인한 활용도 저하를 사용상 애로사항으로 꼽고 있음



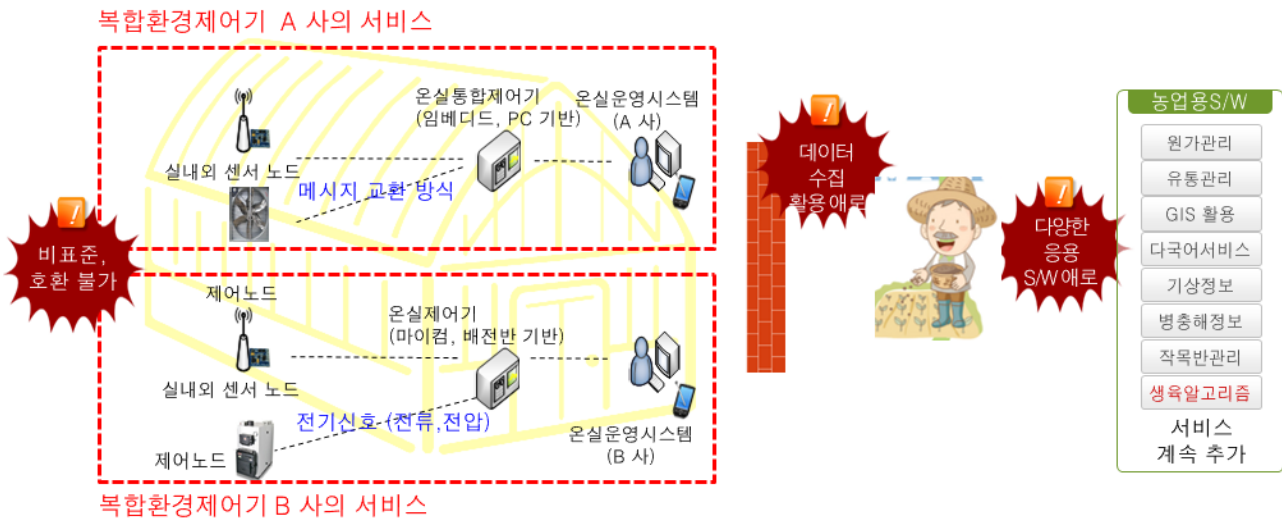
<스마트팜 빅데이터 활용 생산성 향상 모델 발표회 및 발전방안 간담회 행사 사진>

충청북도 한백베리 한현수 대표 건의사항	충청남도 주한호 대표 건의사항
<p>○ 애로사항</p> <ul style="list-style-type: none"> - 업체 회사들끼리 연계가 안 됨 - 회사마다 각 제품의 품질은 제각각이라 필요한 제품들로 골라서 설치를 하려고 해도 제품 연동이 되지 않아 한 회사에 울며 겨자 먹기로 다 구매해야함 - 또한, 자기네 회사 제품이 아니면 프로그램 자체가 맞지 않아 새로 하나 설치하려면 그 회사 제품으로 다시 구매해야 하므로 돈이 이중으로 듦 <p>○ 발전방안</p> <ul style="list-style-type: none"> - 다른 회사 제품끼리도 서로 연계해서 쓸 수 있었으면 좋겠음 - 설치 회사들의 미비한 점이 많은 것도 개선되어야 할 뿐만 아니라 부작용에 대해서도 많은 연구가 필요하다고 봄 	<p>○ 애로사항</p> <ul style="list-style-type: none"> - 편리함과 효율성은 우수하나 제어기술력이 뒷받침 되어야 함 - 스마트팜 설치 기기의 표준화 - 농가마다 기기가 달라 의견 공유가 어려움 - 컴퓨터 활용능력이나 스마트팜 제어기술 (활용능력) 부족 <p>○ 발전방안</p> <ul style="list-style-type: none"> - 스마트팜 기기 및 관리기술에 대한 체계적 교육이 필요함 - 스마트팜 도입 전이나 도입 후에도 활용능력 배양을 위한 주기적인 교육 필요

<스마트팜 빅데이터 활용 생산성 향상 모델 발표회 및 발전방안 간담회 농업인 의견 발취>

나. 4차산업혁명 시대 핵심 키워드 : 초연결(Hyper-Connected)

(1) 스마트팜을 구현하는 하드웨어와 소프트웨어간 상호 연결성 미흡



(가) 스마트팜 제공 업체들의 하드웨어(이기종 제품)들이 상호 연결 서비스가 안됨

① 이기종 제품 간 연결을 위해서는 제품 개발 단계부터 표준에 의해 개발되어야 함

(나) IoT(Internet of Things : 사물인터넷)을 구현하는 하드웨어들과 이를 이용하는 소프트웨어(영농일기, GAP인증시스템, 산지유통센터 품질관리시스템 등)간 상호 연결이 되어야 다양한 응용서비스 구현이 가능해짐

① 통신표준(기계들끼리 정보를 주고 받는 체계)이 제대로 개발되어 있지 못하여 업체들도 제각각 제품과 서비스를 만들고 있어 상호연결/상호운용 불가

다. 스마트팜 확산을 위한 정책 환경

(1) 스마트팜 관련 표준 현황

(가) TTA(정보통신기술협회)를 통해 많은 스마트팜 관련 표준들이 제정되었으나 현재 해당 표준을 지원하는 제품은 거의 없음(30여종의 표준 제정, 실제 구현은 전무한 상태)

① 제정된 대부분의 표준은 연구과제의 성과물 달성을 위해 실험실 수준에서 제정되어 실제 현실에서 제대로 적용되지 못하는 수준으로 관리되고 있음

(2) 스마트팜 확산 사업에서 표준 적용 현황

(가) 농림사업시행지침 ‘스마트팜 시설보급 사업’에서 지원 대상으로 *표준 적용 제품을 사용하도록 규정하여 표준 기자재만 정부 지원 가능

*센서와 제어장비 설치 시에는 스마트 온실을 위한 센서 인터페이스 표준(TTAK.KO-10.0903, '16.6.24)과 구동기 인터페이스 표준(TTAK.KO-10.0845)을 따라야 함

(나) 농림수산물식품교육문화정보원에서 발간한 「ICT융복합 설치규격 및 서비스 범위」

*시설원예분야*에 따르면 스마트팜 기기는 온실관리메시지 인터페이스 표준을 따를 것을 권고하고 있음

(다) 하지만, 실제 현장에서는 이들 표준을 따르는 제품이 거의 존재하지 않기 때문에 정책과 현실이 괴리되고 있음

라. 스마트팜 표준 관리를 위한 정부 정책 현황

(1) 검인증 전담부서인 농업기술실용화재단에서는 다음과 같은 검인증시행 계획을 수립하고 있음

검정범위	'19				'20				'21				'22			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
단품 (인터페이스)	[Progress bar]															
호 환 성	[Progress bar]															
통신프로토콜	[Progress bar]															

(가) 통신표준에 대한 검증은 2021년 상반기로 계획하고 있기 때문에 업체가 표준을 따른 제품을 개발하더라도 검증을 받을 방법이 현재로는 부재한 상태임

(2) 표준 확산을 위한 연구과제 진행 현황

<2019년도 「1세대 스마트팜 고도화 및 산업화」 연구개발 시행계획(2019.1월)>

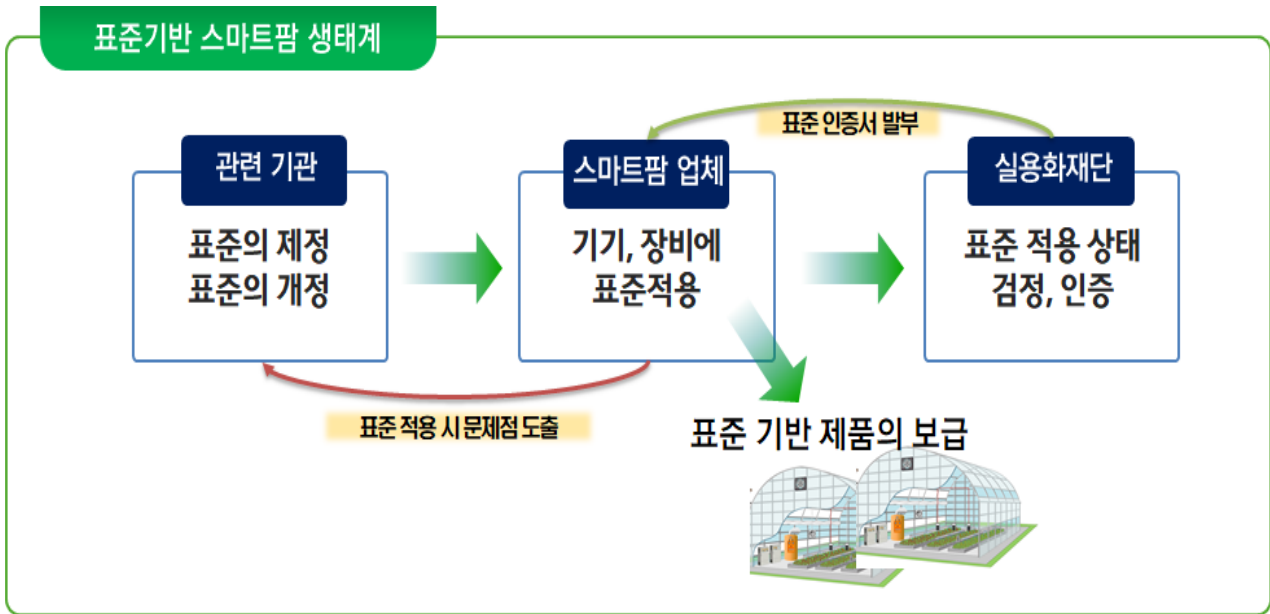
(가) 공고규모 : '19년 정부출연금 185.7억 원 이내

(나) 공고 과제 내 표준관련 과제

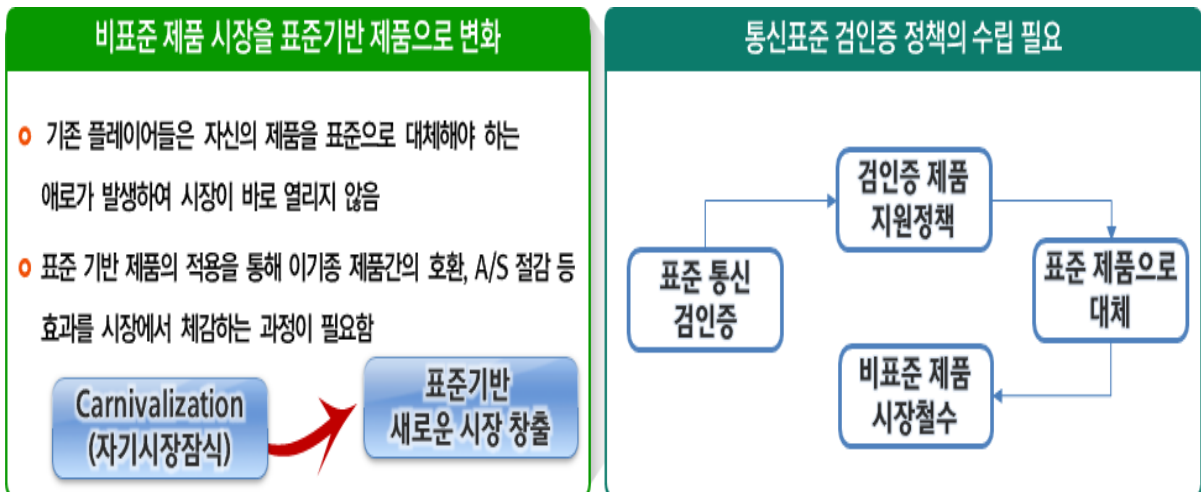
과 제 명	연구 기간	'19년 정부 출연금 (백만원)
(청) 2-1 스마트팜 2단계 표준화 로드맵 구축 및 기 제정 표준의 고도화	12개월	500
(부) 2-2 오픈소스 기반 스마트팜 개방형 제어기 고도화 및 산업화	12개월	1,000
(부) 2-3 표준기반 시설과수 적용 ICT 패키지 통합운영시스템 산업화	12개월	300
(부) 3-2 보급형 순환식 수경재배시스템 산업화	12개월	500
(부) 4-2 표준기반 약용작물 적용 ICT 패키지 통합운영시스템 산업화	12개월	300
(부) 5-2 표준기반 버섯재배사 적용 ICT 패키지 통합운영시스템 산업화	12개월	500
(부) 6-5 표준기반 스마트 공정육묘 통합관리시스템 산업화	12개월	450

*표준을 적용한 스마트팜 제품의 확산을 위해 과제 진행중(2020년 1월 목표)이나 적용표준 역시 KS표준으로 제정된 5건의 표준에 대해서만 적용을 목표로 하고 있으므로 실제 현장의 요구를 모두 수용하기에는 한계가 있음

마. 스마트팜의 활성화를 위한 생태계 프로세스



- (1) 필요한 표준을 기술의 발전에 맞게 신속히 제정하고, 제정된 표준을 스마트팜 기업들이 제품에 적용, 실용화재단에서 이를 검정 후 인증서 발급
- (2) 표준적용 제품들이 시장에 출시되어 비표준 제품과 경쟁우위를 가져가는 것이 필요
 - ※ 기존 비표준 제품들이 시장에 보급된 상태에서 바로 표준제품이 경쟁력을 가지기는 어렵기 때문에 다양한 지원정책이 뒷받침되어야 시장의 변화가 가능
- (3) 스마트팜 업체의 표준 적용을 위한 지원 정책 발굴
 - (가) 기제품을 가진 기업들은 현 시장 상황에서는 표준으로의 전환 필요성을 가지지 않고 있으며, 제품 전환 시 매우 큰 비용이 들기 때문에 자연발생적으로 표준 기반 제품의 출시를 기대하기 어려움
 - (나) 선도 기업은 표준 제품으로 상호호환성이 높아지는 것을 꺼려하는 경향 존재



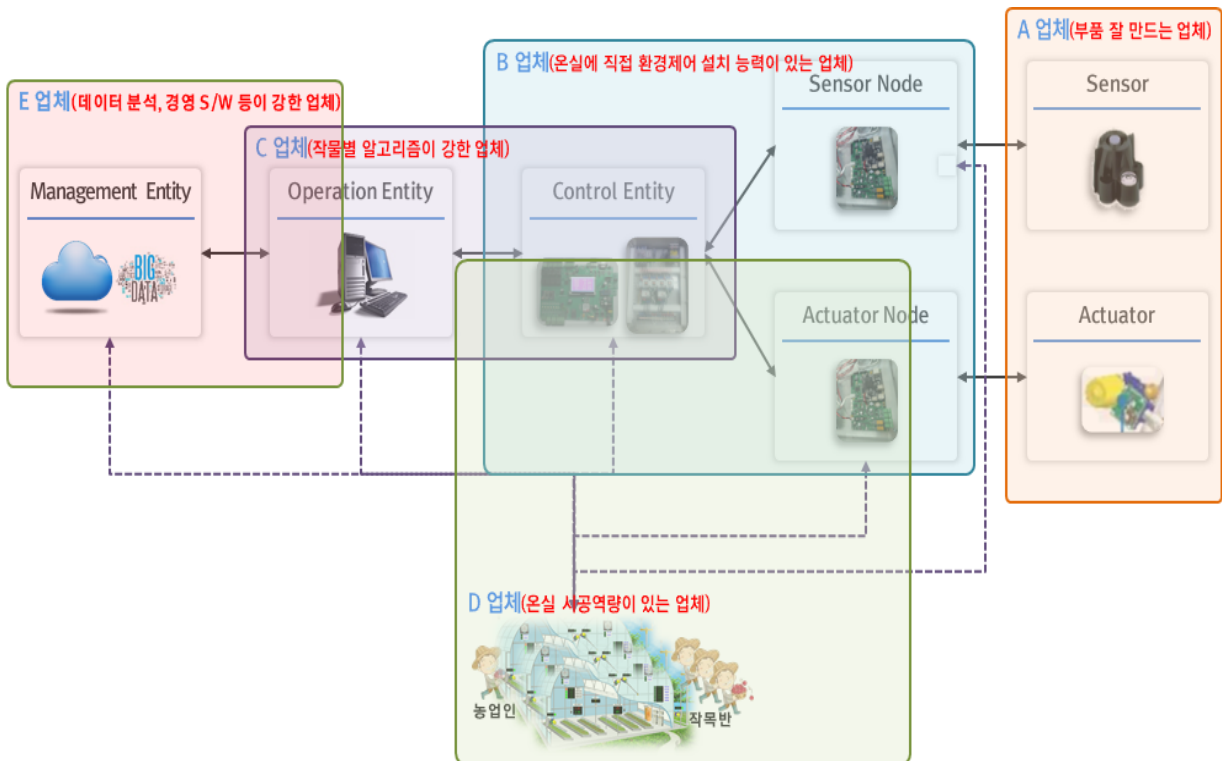
2. 기대효과

가. 농업인의 스마트팜 제품 구매 시 중복투자 제거

- (1) 표준 적용 제품을 도입한 농업인은 기능 확대, 장비 노후화 교체 시 최소의 비용으로 관리
 - (가) 예시1) 온습도 모니터링 장비 구매 후 온실 제어기능 확대를 하려고 하면 추가 센서 구매 없이 제어 서비스 부착 가능
 - (나) 예시2) 설치한 제품의 오작동, 노후화 발생 시 같은 표준을 따르는 최적의 제품을 구매하여 교체 가능 -> 기 설치 업체에 종속되는 문제 해소
- (2) 스마트팜 공급 기업 역시 A/S 비용의 획기적 절감
 - (가) 현재 A/S를 하려면 원거리 농장을 방문하는 비용도 크고, 어떤 문제인지 진단도 잘 안되기 때문에 근거리 농장에만 설치해야 하는 애로가 있음
 - (나) 표준 기반 제품 생태계가 갖춰지면 관련 업체 간 제휴를 통해 제품 개발 업체와 제품 A/S 업체 등으로 업무 분장이 가능해짐

나. 4차산업혁명 시대에 걸맞는 다양한 대농업인 서비스 개발 촉진

- (1) 현재 스마트팜 서비스는 농장자동화 수준에 머무르고 있으나 표준 기반 생태계가 갖춰지면 다양한 서비스 기업들이 경쟁력 있는 서비스로 시장에 진출할 수 있음



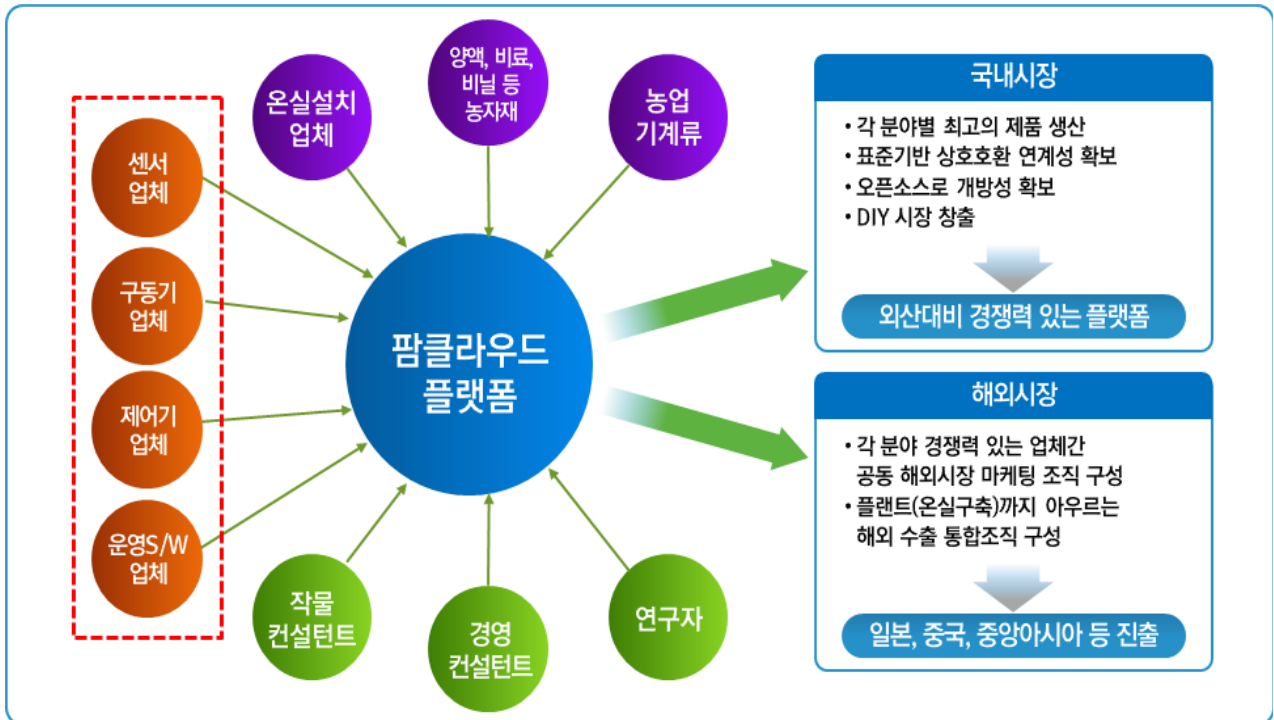
〈자료 : [농축산업 ICT기기 규격화 및 표준화 필요성, ETRI, 박주영, 2016]을 재구성〉

- (가) 스마트팜 하드웨어 업체 중심의 생태계가 컨설팅 업체, 소프트웨어업체, 알고리즘 업체 등 다양한 비즈니스 모델을 가진 기업들이 본 시장에 진출할 수 있음

- (나) 이를 통해 4차산업 기술 기반 기업들이 농업에 진출하여 고급일자리 창출이 가능해짐
- (다) 결국, 이러한 생태계는 농업인에게 유용한 서비스가 제공되어 농업경쟁력 강화로 이어짐

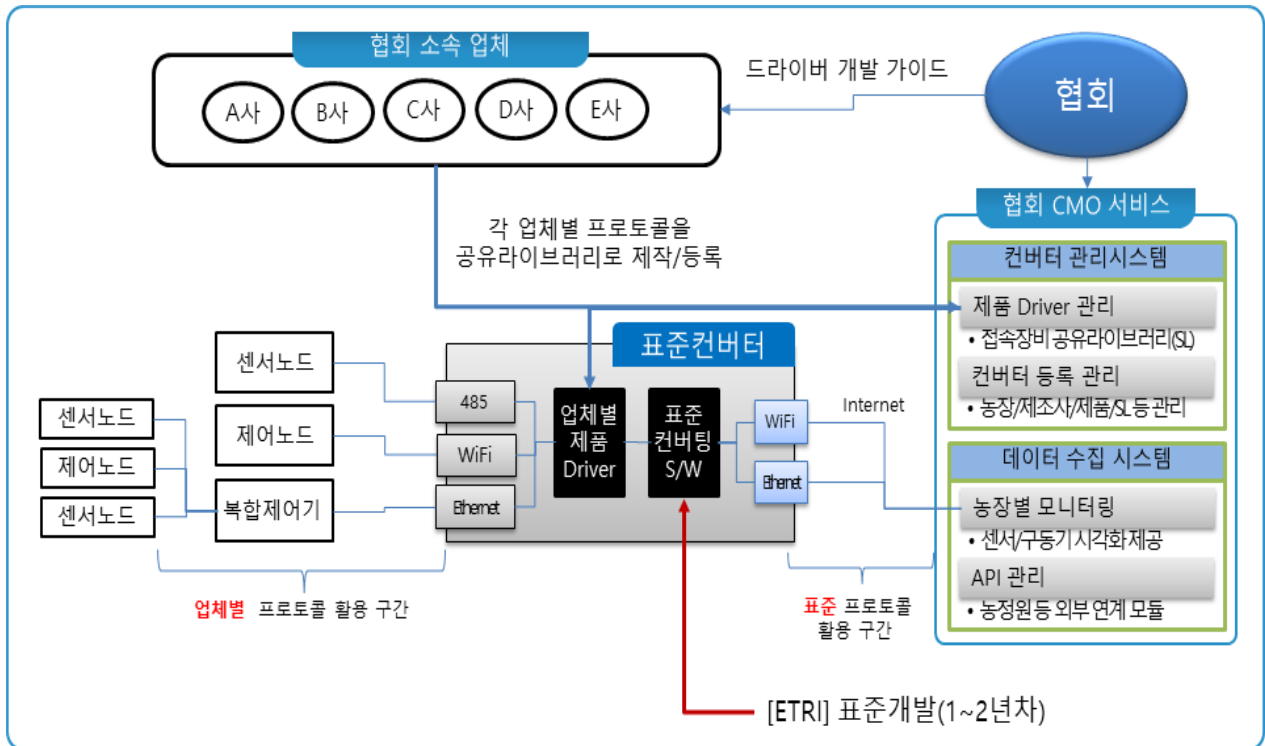
다. 표준 기반 제품으로 글로벌 시장 진출 도모

- (1) 스마트팜 기업들이 글로벌 진출하기에는 규모가 영세하여 쉽게 접근하기 어려운 상태이지만, 표준 기반 생태계가 갖춰지면 여러 기업들이 각자의 경쟁우위 서비스를 중심으로 제휴하여 글로벌 시장에 진출이 가능해짐



- (가) 국내 스마트팜 시장을 테스트베드로 하여 동남아시아, 중앙아시아, 일본, 중국 등 시설 원예 장치농업이 활성화되고 있는 시장에 경쟁력 있게 진출
- (2) 스마트팜 혁신벨리의 핵심 모델로 표준기반 생태계 시스템을 적용하여 글로벌 진출 교두보화
 - (가) 네덜란드가 PTC+처럼 후진국/개도국 농업인들을 교육하고 이들이 네덜란드의 시스템을 자국에 적용하게 함으로써 네덜란드 주 수출품이 농업자재/시스템(10조원 이상)이 되고 있음
 - (나) 스마트팜 혁신벨리의 교육대상을 국내 농업인/창업농 뿐만이 아니라 우리나라 농업을 배우고자 하는 개도국들이 농업을 배우러 오는 곳으로 서비스 개발
 - (다) 이를 이용하여 이들이 한국의 농업시스템을 적극적으로 도입할 수 있도록 할 수 있음

2절 연구개발 범위



1. 스마트팜 표준화 확산 지원

- (1) 스마트팜 장비 표준화 지원을 위한 설명회 및 간담회 개최
- (2) 농기자재 박람회 참여
- (3) 스마트팜 업체 가이드북 제작

2. 표준화 컨버터 개발 : 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발

- (1) 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발(기술이전)
- (2) 스마트팜 장비 연동용 표준 컨버터 개발(기술이전)

3. 자료 분석 장치 개발

- (1) 클라우드 기반 컨버터 관리 및 데이터수집시스템 개발

4. 스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용

- (1) RS485 MODBUS 기반의 스마트팜 통신 표준 제정
- (2) 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 신규 표준을 적용한 레퍼런스 장비 개발

2장

2장 연구수행 내용 및 결과

1절 연구수행 내용

1. 연구개발 추진전략 및 추진체계
2. 연구개발 내용

2절 연구개발 성과

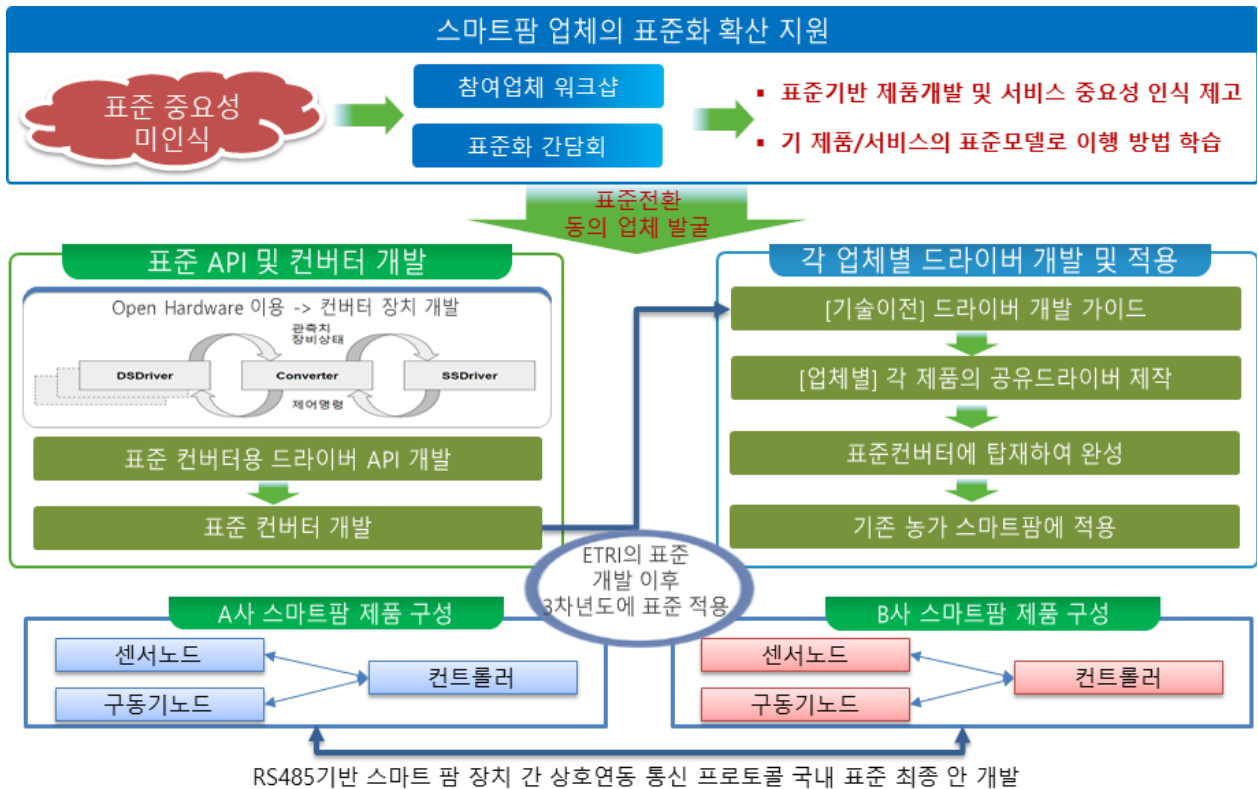
1. 지식재산권
2. 논문게재 성과
3. 학술대회 발표성과
4. 기술요약정보
5. 소프트웨어 등록
6. 사업화(제품화)

2장 연구수행 내용 및 결과

1절 연구수행 내용

1. 연구개발 추진전략 및 추진체계

가. 연구개발 추진전략



(1) 스마트펌업체의 표준화 확산 지원 전략

- (가) 농가에 보급된 환경제어장치의 표준화 미흡으로 기기 간 통신방식과 인터페이스 처리방식이 달라 상호호환이 어려운 상태이므로 표준 재개정과 표준 활용을 촉진시키기 위하여 표준컨버터를 개발하여 기업의 적극적인 참여를 유도
- (나) 스마트펌 관련된 표준의 개발은 미흡한 상태이므로 기업들이 표준을 따라 자사제품을 개발하고 제품을 출시하기에는 한계가 있음. 본 주관사는 기업들의 상황을 고려하여 과업기간 내 다양한 방법으로 소속 회원 기업들에게 표준에 대한 워크숍 및 간담회를 진행

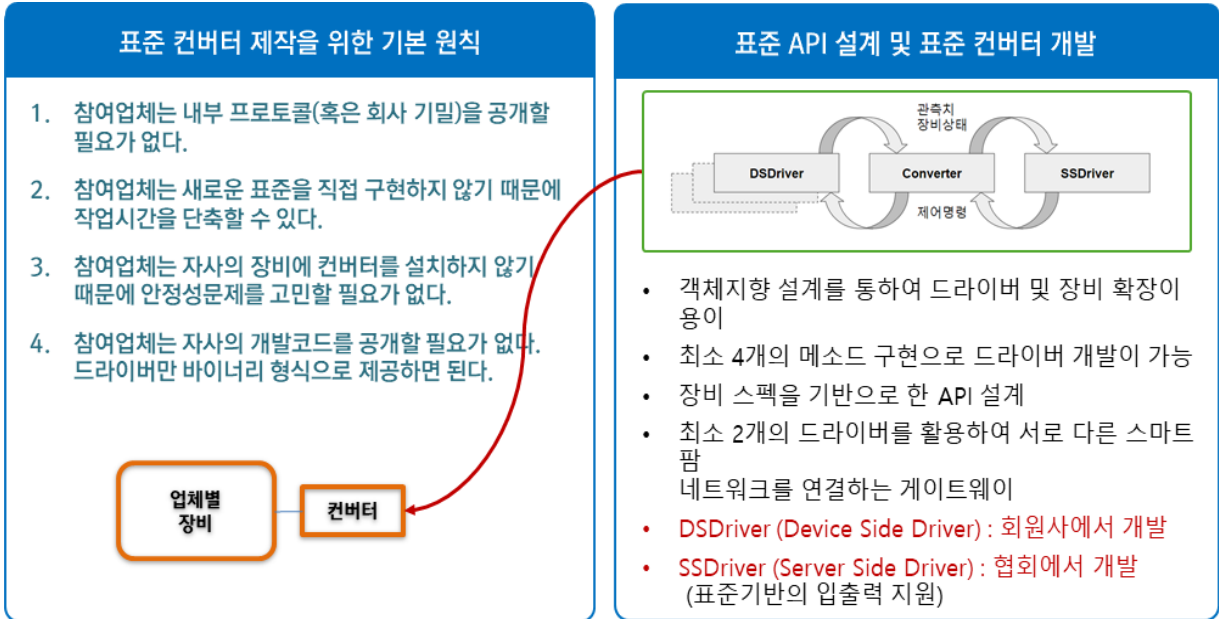
(2) 컨버터 개발 전략

- (가) 업체별로 통신프로토콜 공개에 저항이 많아 1차 년도에는 별도의 컨버터 장비를 개발하였으나 동의 받지 못함
- (나) 이를 계기로 협회가 표준가이드(API)를 만들고 이를 이용하여 각 업체별로 구현하는 원칙을 수립

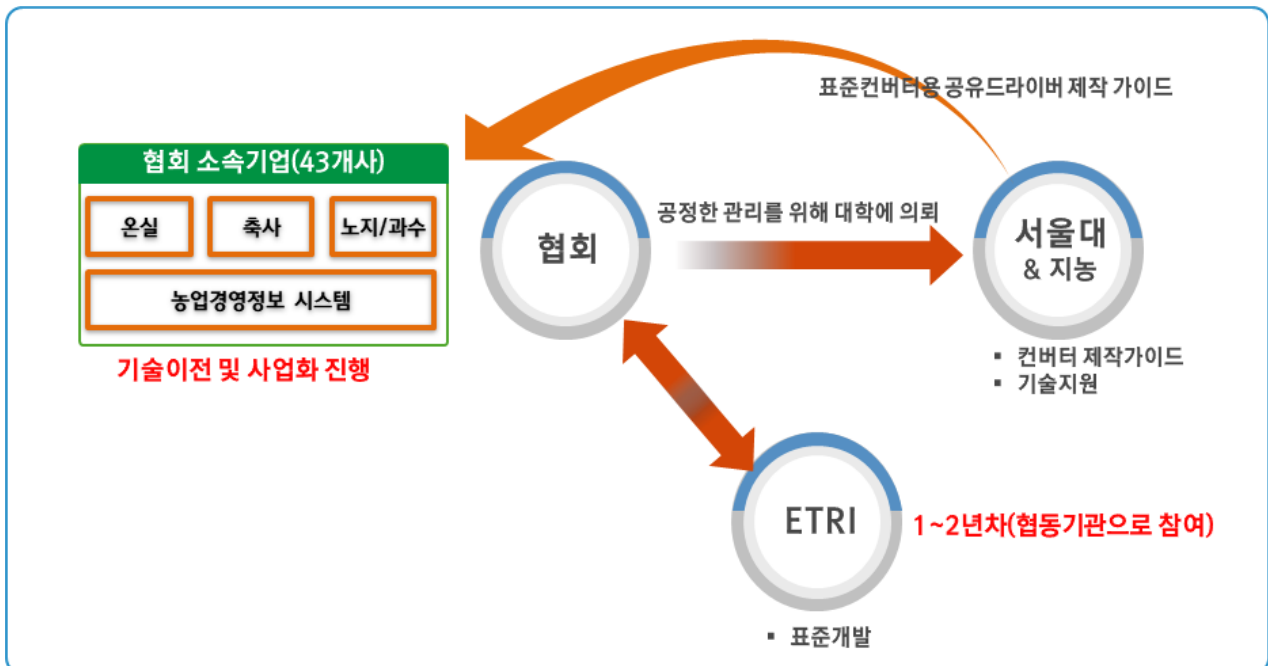
(3) 표준의 개발과 컨버터 개발을 동시 진행

(가) 아직 시장에서 스마트팜 관련 표준이 제대로 개발되어 있지 않고, 표준 기반 제품 역시 존재하지 않는 상황에서 업체들이 가장 많이 사용하고 있는 RS485 기반 Modbus 통신에 대한 표준을 개발하여 레퍼런스 장비를 만드는 전략을 동시에 적용함

(나) 또한, 업체들이 자사의 통신 프로토콜을 영업비밀로 하여 공개하기를 꺼려하는 현실이므로 표준 API 가이드를 설계하여 업체들에게 제공하고 업체들이 자사의 통신프로토콜에 맞게 이를 활용하여 공개하는 방법으로 컨버터 표준화 전략을 수립



나. 연구개발 추진체계



(1) 기업공동요구기술의 개발(CMO)에 맞게 대학과 연구기관을 활용한 공정한 추진체계 및 업무분장

2. 연구개발 내용

가. 개발된 연구내용의 결과는 아래와 같이 진행됨

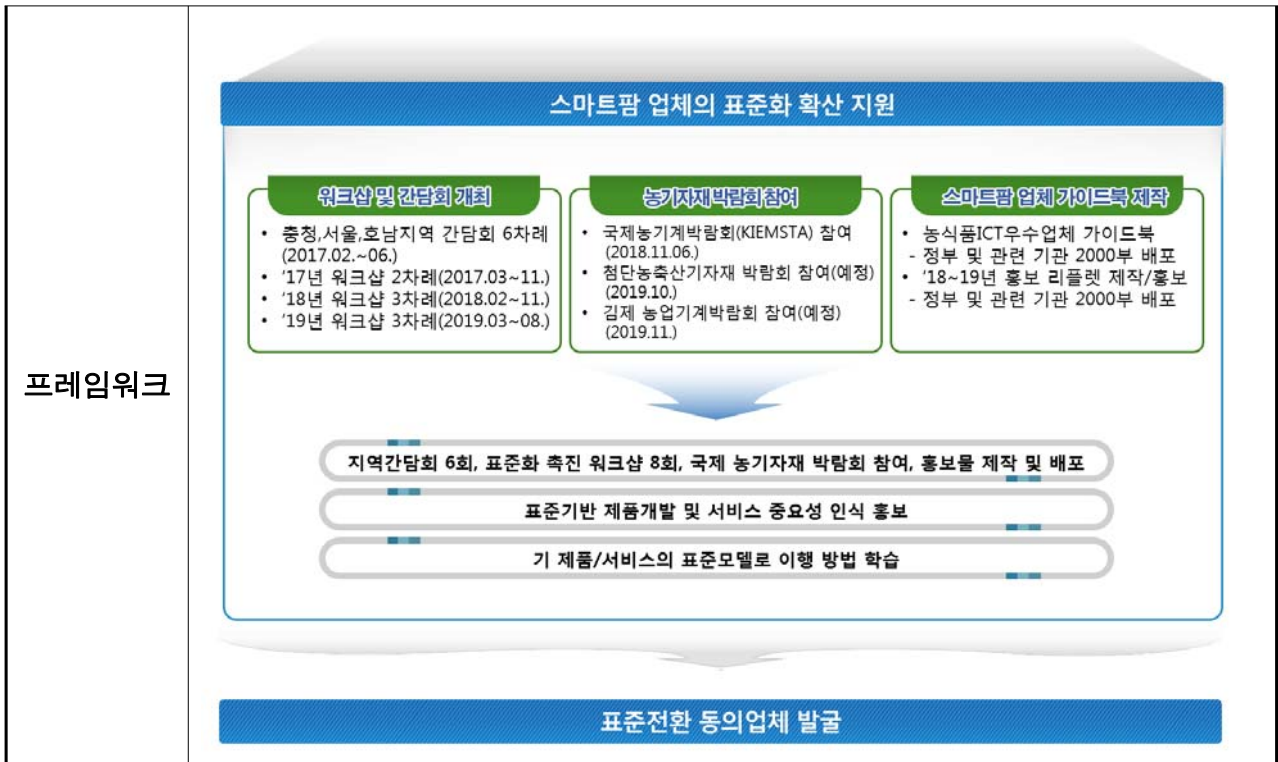
- (1) 스마트팜 표준화 확산 지원을 위해 총 8회의 워크숍과 6회의 간담회를 개최
- (2) 표준화 컨버터를 개발하여 기업들이 활용할 수 있도록 기술개발 2건을 진행하여 기술이전
- (3) 클라우드 컨버터 관리 및 데이터 수집을 위한 시스템 개발 1식
- (4) 스마트팜 내 센서/구동기-제어기 간 상호운용을 위한 표준 개발(1건 제정, 1건 진행중) 및 이를 활용하여 기업이 참조 개발할 수 있는 레퍼런스 장비 개발 1식

구분	연구개발 범위	세부 개발 내용	페이지
1	스마트팜 표준화 확산 지원	- 스마트팜 기자재 표준화 확산을 위한 워크숍 및 간담회 개최	24~44
2	표준화 컨버터 개발	- 표준 API 스펙 개발 및 오픈소스로 공개 - 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발(기술이전) - 스마트팜 장비 연동용 표준 컨버터 개발(기술이전)	45~119
3	자료 분석 장치 개발	- 클라우드 기반 컨버터 관리 및 데이터 수집시스템 개발	120~124
4	스마트팜 내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발	- RS485 MODBUS 기반의 스마트팜 통신 표준 제정 - 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 - 신규 표준을 적용한 레퍼런스 장비 개발	125~183

2. 연구개발 내용

가. 스마트팜 표준화 확산 지원

개 요	
목적	스마트팜 표준화 기반을 통한 기업의 기술개발 지원을 위하여 2016년 9월부터 2019년 9월까지 본 연구과제를 수행하였다. CMO(contract manufactur organization)방식을 농식품ICT기업의 표준화 기술개발에 적용하여 경쟁력 있는 제품을 생산할 수 있도록 촉진시키므로써 국내 농식품분야 ICT산업을 육성하고 스마트팜을 확산시키는데 목적이 있다.
배경	현재 4차 산업혁명에 의한 기술개발이 전 산업에 걸쳐 확산되고 있고 농식품 분야에도 첨단화된 기술개발이 활발하게 이루어지고 있다. 2015년 ICT기반 한국형 스마트팜 기술개발 로드맵이 수립되고, 2016년에는 스마트팜을 차세대 미래성장산업으로 선정하여 농업경쟁력을 높이기 위한 확산정책을 추진하였다. 실천방안으로써 우선 ICT기기의 표준화와 국산화를 추진하기로 하였고 스마트팜 관련기기의 표준화가 미흡하여 이종 간·장치 간 연동이 어렵고 호환이 되지않는 현상을 해소하기 위해 표준화의 촉진과 국내 산업간 국제간 기술격차 극복을 위한 방안 마련이 강조되었다.
사업범위	표준화를 위한 컨버터장치의 개발과 표준화 적용모델을 개발한다. 이를 위하여 표준화 컨버터장치를 개발하고 스마트팜 기자재 성능개선을 지원하며, 스마트팜 데이터 수집·분석과 스마트팜 표준화 적용 모델을 개발한다.
연구방향	<p>농가에 보급된 환경제어장치의 표준화 미흡으로 기기 간 통신방식과 인터페이스 처리방식이 달라 상호호환이 어려운 상태이므로 표준 재개정과 표준 활용을 촉진시키기 위하여 표준컨버터를 개발하여 기업의 적극적인 참여를 유도한다.</p> <p>스마트팜과 관련된 표준의 개발은 미흡한 상태이므로 기업들이 표준을 따라 자사제품을 개발하고 제품을 출시하기에는 한계가 있다. 본 주관사는 기업들의 상황을 고려하여 과업기간 내 다양한 방법으로 소속 회원 기업들에게 표준에 대한 워크숍 및 간담회를 진행하였다. 이러한 성과로 스마트팜 기업들이 표준에 대한 인식을 제고하고, 홍보된 내용을 바탕으로 기술공유 워크숍을 개최하고 스마트팜 표준화 촉진을 위한 <농식품ICT융복합가이드북>을 발간·배포하여 표준화 촉진활동을 하였다.</p>



프레임워크

성과요약표							
구분	수행 사항	구분	수행 내용	산출물	성과지표		
					표준화 인지	표준화 홍보	기술공유
1	워크숍 및 간담회 개최	1.1	'17년 오픈 간담회 1차	-자료집*	0		
		1.2	'17년 오픈 간담회 2차		0		
		1.3	'17년 오픈 간담회 3차		0		
		1.4	충청지역 간담회		0		
		1.5	경기·서울지역 간담회		0		
		1.6	호남지역 간담회		0		
		1.7	'17년 1차 워크숍		0	0	
		1.8	'17년 2차 워크숍		0	0	
		1.9	'18년 1차 워크숍		0	0	
		1.10	'18년 2차 워크숍		0	0	
		1.11	'18년 3차 워크숍		0	0	0
		1.12	'19년 1차 워크숍		0	0	0
		1.13	'19년 2차 워크숍		0	0	0
		1.14	'19년 3차 워크숍		0	0	0
2	농기자재 박람회 참여	2.1	국제 농기계박람회 참여	0	0		
		2.2	첨단 농축산기자재 박람회 참여(예정) 김제 농업기계박람회 참여(예정)	0	0	0	
3	스마트팜 업체 가이드북 제작	3.1	농식품ICT우수업체 가이드북 제작 및 배포	0	0		
		3.2	홍보 리플렛 제작 및 배포	0	0		

*자료집은 연구주관기관인 (사)한국농식품ICT융복합산업협회를 통해 참조가능

성과 세부내용

구분	수행사항	수행내용
1	워크숍 및 간담회 개최	<p>1.1. '17년 오픈 간담회 1차</p> <ul style="list-style-type: none"> - 일시 및 장소 : 2017.02.02.(대전) - 목적 : CMO 사업과제 1차년도 수행방안에 대한 각계 전문가 간담회 - 내용 : 표준화를 위한 기업의 기술개발지원사업에 대한 이해와 내용 사항에 관한 오픈 논의 - 참석자 : 회원사 등 관련기업체(8개사)와 정부 및 관련기관단체 등 14여명 참석 - 참고자료 : <div style="display: flex; flex-direction: column; align-items: center;">   <p><현장 사진></p> </div>

1.2. '17년 오픈 간담회 2차

- 일시 및 장소 : 2017.02.06.(대전)
- 목적 : CMO 사업과제 1차년도 수행방안에 대한 각계 전문가 간담회 2차
- 내용 : 1차 오픈 간담회에 대한 구체적 현황 논의
- 참석자 : 회원사 등 관련기업체(10개사)와 정부 및 관련기관단체 등 12여명 참석
- 참고자료 :



<현장 사진>

워크숍
및
간담회
개최

1

워크숍
및
간담회
개최

1

1.3. '17년 오픈 간담회 3차

- 일시 및 장소 : 2017.02.14.(대전)
- 목적 : CMO 사업과제 1차년도 수행방안에 대한 각계 전문가 간담회 3차
- 내용 : 협동 ETRI 표준화(통신 프로토콜과 인터페이스 사항) 방향 발표
- 참석자 : 회원사 등 관련기업체(8개사)와 정부 및 관련기관단체 등 20여명 참석
- 참고자료 :



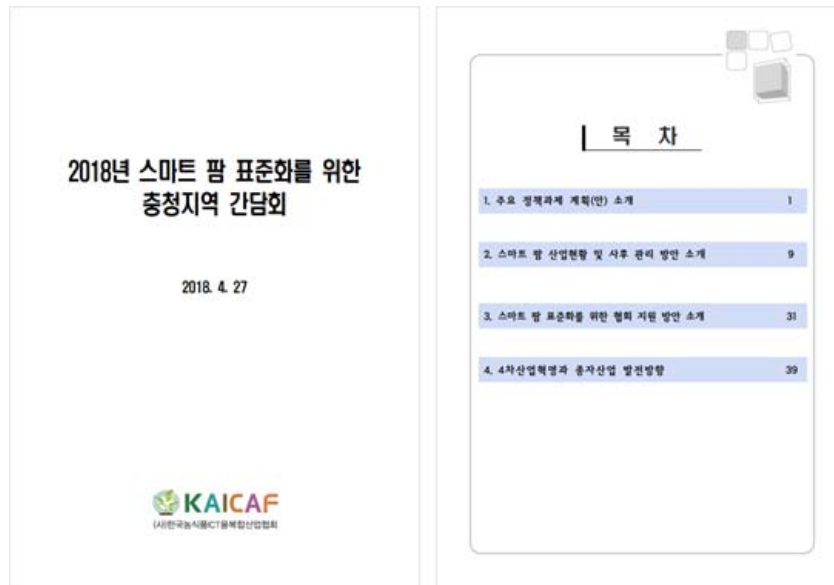
<현장 사진>

워크숍
및
간담회
개최

1

1.4. 충청지역 간담회

- 일시 및 장소 : 2018.04.27.(대전)
- 목적 : 정부의 스마트팜 확산정책과 표준화 연구용역 추진상황을 소개하고, 기업의 스마트팜 확산을 위한 제반정보의 제공
- 참석자 : 충청지역 업체 및 기관단체 등 16개 기관(25명)
- 사진자료 :



<간담회 책자>



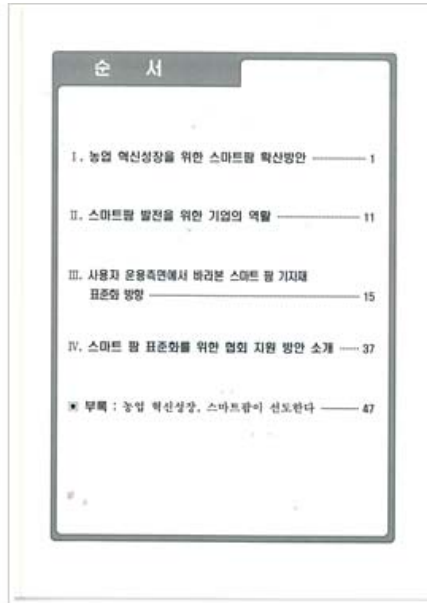
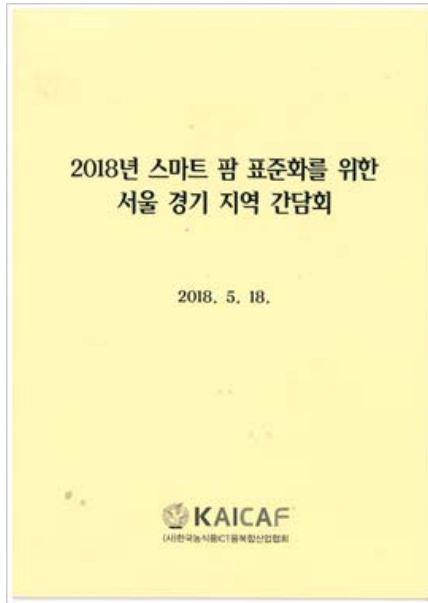
<현장 사진>

1

워크숍
및
간담회
개최

1.5. 경기·서울지역 간담회

- 일시 및 장소 : 2018.05.18.(양재)
- 목적 : 정부의 스마트팜 확산정책과 표준화 연구용역 추진상황을 소개하고, 기업의 스마트팜 확산을 위한 제반정보의 제공
- 참석자 : 수도권지역 업체 및 기관단체 등 17개 기관(21명)
- 사진자료 :



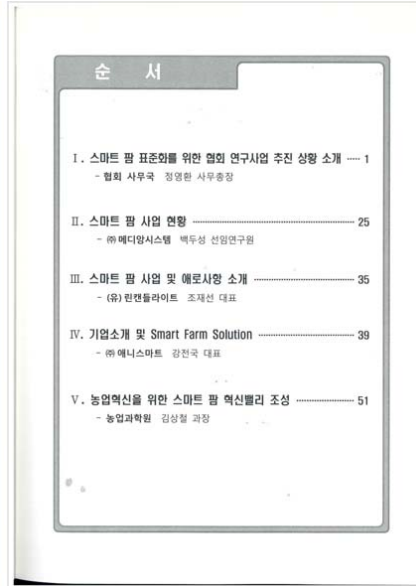
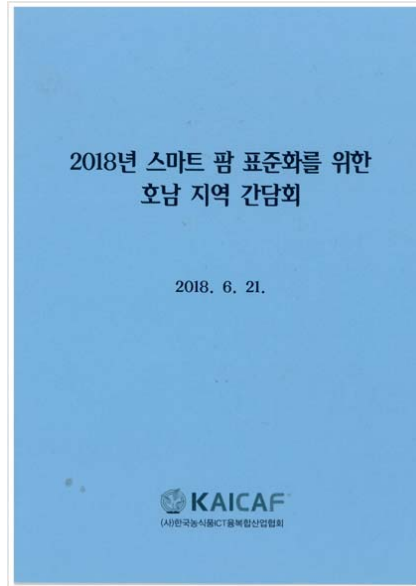
<간담회 책자>

워크숍
및
간담회
개최

1

1.6. 호남지역 간담회

- 일시 및 장소 : 2018.06.21.(전주)
- 목적 : 정부의 스마트팜 확산정책과 표준화 연구용역 추진상황을 소개하고, 기업의 스마트팜 확산을 위한 제반정보의 제공
- 참석자 : 호남지역 업체 및 기관단체 등 10개 기관(15명)
- 참고자료 :



<간담회 책자>



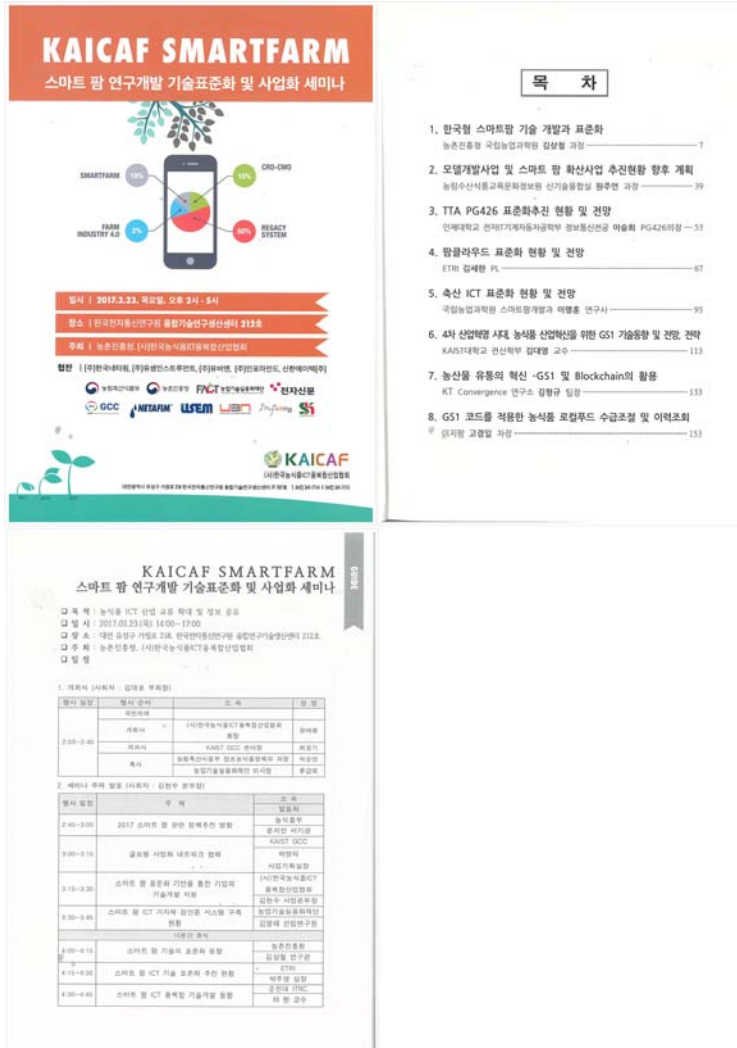
<현장 사진>

1.7. '17년 1차 워크숍

- 일시 및 장소 : 2017.03.23.(대전)
- 목적 : 농식품 ICT 산업 교류 확대 및 정보 공유
- 참석자 : 회원사 등 관련기업체(6개사)와 정부 및 관련기관단체 등 40여명 참석
- 참고자료 :

워크숍
및
간담회
개최

1



<워크숍 책자>

1.8. '17년 2차 워크숍

- 일시 및 장소 : 2017.11.16.(일산)
- 목적 : 농식품 ICT 관련 표준화 현황 및 전망 소개
- 참석자 : 스마트팜 ICT 제조업체, 협회, 학계, 연구기관 등 약 100여명
- 참고자료 :



<현장 사진>



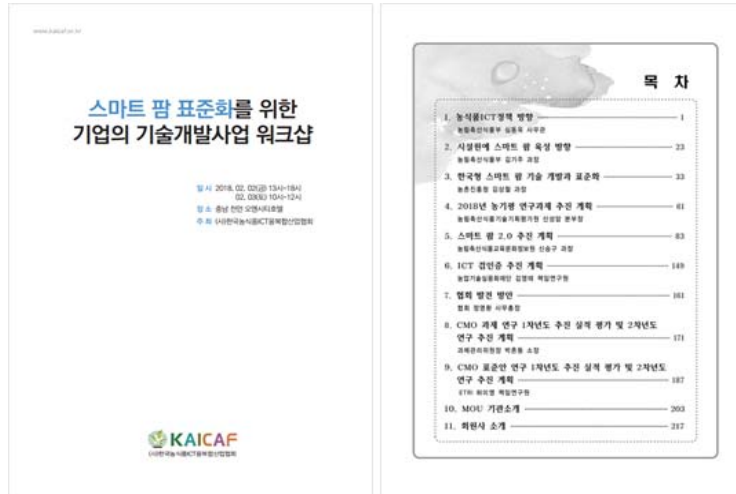
<워크숍 결과 보고서>

워크숍
및
간담회
개최

1

1.9. '18년 1차 워크숍

- 일시 및 장소 : 2018.02.02.~03.(천안)
- 목적 : 정부의 스마트팜 정책방향 공유 및 시설원예 스마트팜과 기술개발 및 검정사업 현황보고
- 참석자 : 회원사 등 관련기업체(50개사)와 정부 및 관련기관단체 등 80여명 참석
- 참고자료 :



<워크숍 책자>

워크숍
및
간담회
개최

1



<현장 사진>

워크숍
및
간담회
개최

1

1.10. '18년 2차 워크숍

- 일시 및 장소 : 2018.06.01.~02.(대전)
- 목적 : 정부의 스마트팜 확산정책과 스마트팜 표준화전략과 기술개발, 스마트팜 주요사업, 스마트팜 기자재 수출현황 등 발표 및 논의
- 참석자 : 회원사 등 관련기업체(50개사)와 정부 및 관련기관단체 등 80여명 참석
- 참고자료 :



<현장 사진>



<워크숍 책자>

1.11. '18년 3차 워크숍

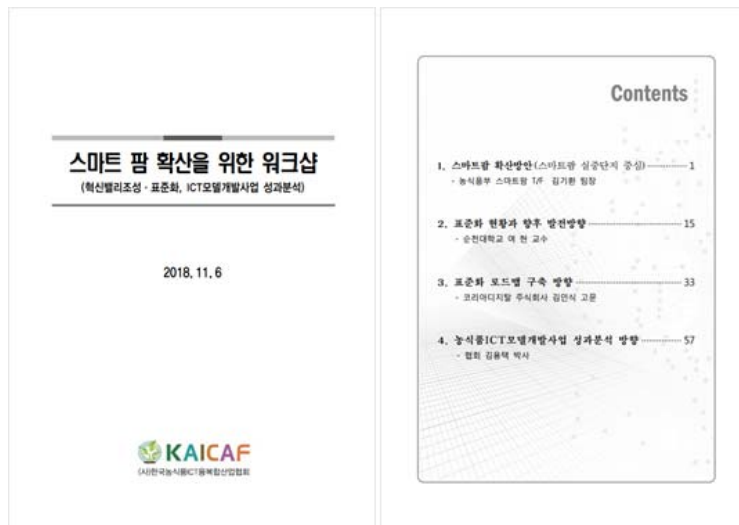
- 일시 및 장소 : 2018.11.06.(양재)
- 목적 : 혁신밸리 조성사업(실증단지 등)정책방향, 표준화 발전방향, 스마트팜 표준화 로드맵 구축방안, ICT모델개발사업 성과분석 등 발표 및 논의
- 참석자 : 회원사 등 관련기업체(50개사)와 정부 및 관련기관단체 등 80여명 참석
- 참고자료 :

워크숍
및
간담회
개최

1



<현장 사진>



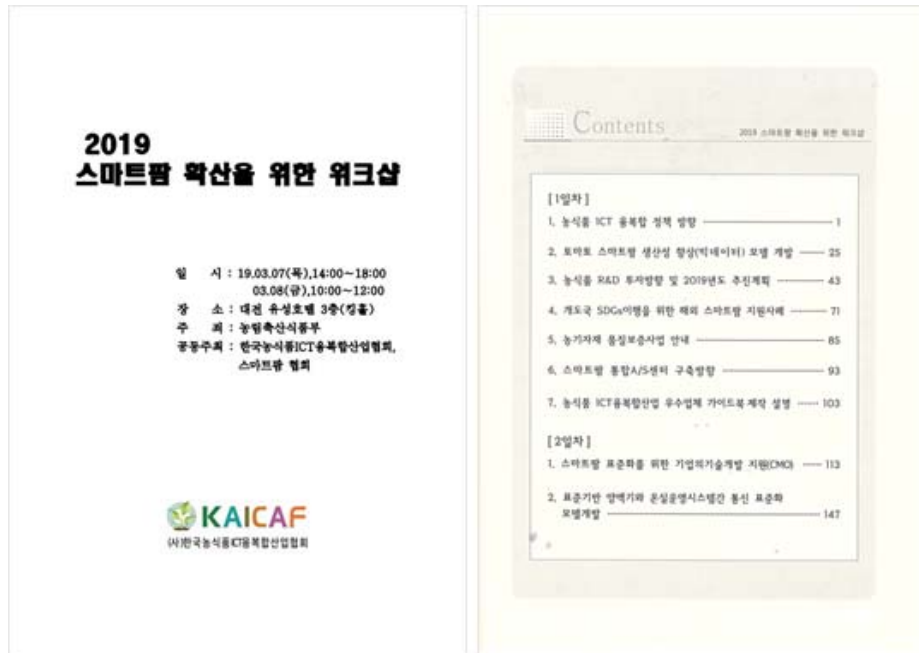
<워크숍 책자>

워크숍
및
간담회
개최

1

1.12. '19년 1차 워크숍

- 일시 및 장소 : 2019.03.07.~08(대전)
- 목적 : 스마트팜 확산을 위한 정부의 주요정책과 스마트팜 표준화 및 기술 개발, 검·인증제도와 품질보증제 등 관련 사업추진 동향을 기업인들에게 소개하고 상생협력기반 구축
- 참석자 : 회원사 등 관련기업체(60개사)와 정부 및 관련기관단체 등 80여명 참석
- 참고자료 :



<워크숍 책자>






<현장 사진>

1.13. '19년 2차 워크숍

- 일시 및 장소 : 2019.07.01.(전주)
- 목적 : 스마트팜 확산을 위하여 농진청과 협회 간 업무협력 방안을 모색하고 스마트팜 기술개발 현황 및 기업기술수요등 주요내용을 공유
- 참석자 : 스마트 팜 협회 임직원 및 회원사, 농식품부·지자체 공무원, 관련 단체 및 연구기관, 비회원 기업체 등 150여명
- 참고자료 :

워크숍
및
간담회
개최

1



농촌진흥청
& 한국스마트팜산업협회
2019. 7. 1.
공동 워크숍

농촌진흥청-한국스마트팜산업협회 공동 워크숍 세부일정

일 시(분)	내 용	비 고
14:00~14:20	20' 참석자 소개 및 주요 인사	
14:20~16:30	(1부) 발표	
14:20~14:50	30' 농업의 미래와 스마트팜	농식품부
14:50~15:20	30' 토마토 스마트팜 빅데이터 생산성 향상 모델	농업빅데이터사업자협회
15:20~15:50	30' 스마트팜 빅데이터 모델 기술이전 및 지원사업 안내	농업기술실용화재단
15:50~16:10	30' 스마트팜 ICT기재자 표준화 현황	농업기술실용화재단
16:10~16:50	40' 스마트팜 생태계 조성을 위한 상호협력 방안	한국스마트팜산업협회
16:50~17:10	20' 스마트팜 기술수출 및 교육·인증협력방안	
17:10~18:10	60' (2부) 전문가 토론	(좌장) 박현승 회장
18:10	대우리	

<워크숍 책자>



<현장 사진>

1.14. '19년 3차 워크숍

- 일시 및 장소 : 2019.08.29.(양재)
- 목적 : 스마트팜 산업표준화 확산을 위한 워크숍
- 참석자 : 스마트 팜 협회 임직원 및 회원사, 농식품부·지자체 공무원, 관련 단체 및 연구기관, 비회원 기업체 등 80여명
- 참고자료 :

시 간	내 용	주 최
10:00-10:40	현장 둘러보기 (1차)	농림축산식품부
10:40-11:10	스마트팜 정책 방향	박성호 과장
11:10-11:40	스마트팜의 수익성/농민 소득 증진 방안	손정미 과장
11:40-12:00	스마트팜 산업화 전략	최 현 과장
12:00-12:30	Lunch	
12:30-12:50	ICT 스마트팜 플랫폼 개발	박우영 과장
12:50-13:10	미래형 한국형 스마트팜	김상철 과장
13:10-13:30	스마트팜 ICT 시범 사업	김정태 과장
13:30-14:00	KASFI 스마트팜 확산 전략 및 국제교류 현황	남궁영 차장
14:00-14:30	농업데이터 활용을 위한 스마트팜 확산 방안	최우연 차장

<워크숍 책자>



워크숍
및
간담회
개최

1

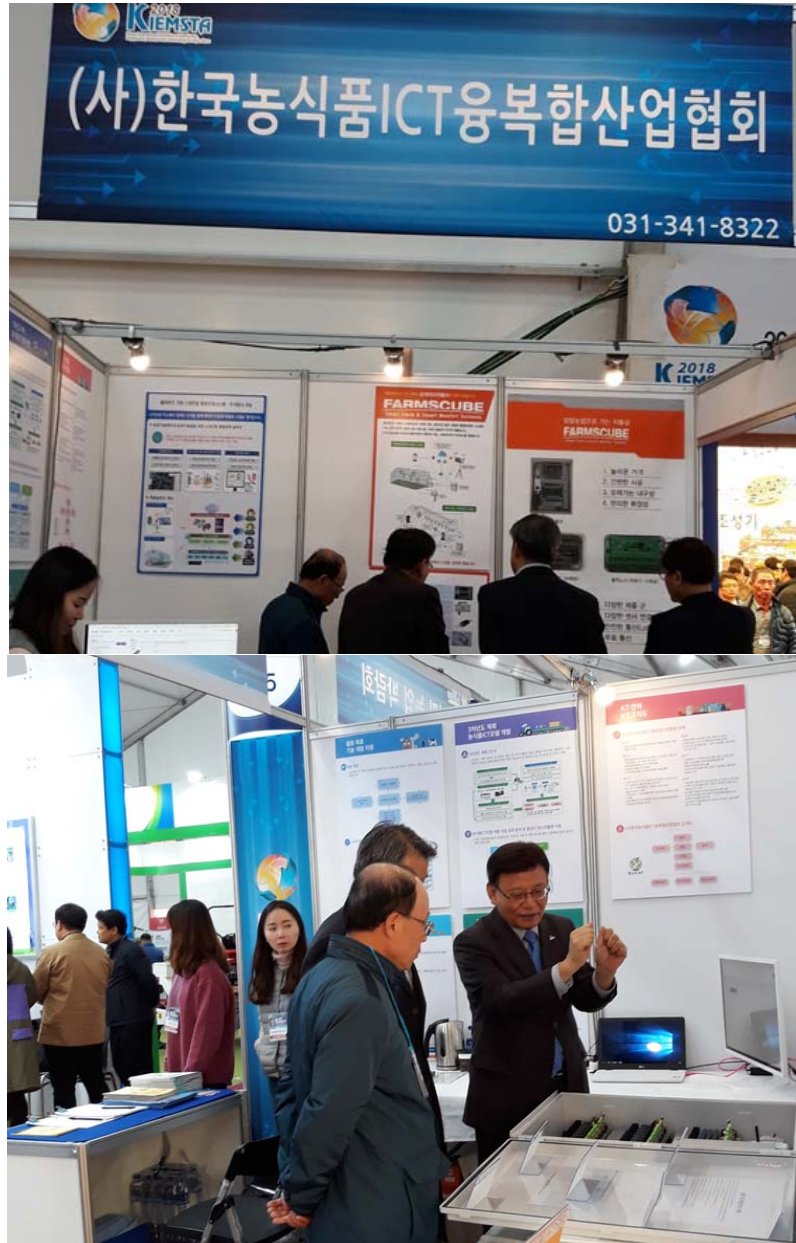


<현장 사진>

2 농기자재 박람회 참여

2.1. 국제 농기계 박람회(KIEMSTA)참여

- 일시 및 장소 : 2019.10.31.~11.03(천안)
- 목적 : 박람회에 참여하는 기업인, 농업인, 유관기관을 대상으로 협회 홍보 및 표준화 업무협의
- 참고자료 :



<현장 사진>

2.2. 향후 참여 예정

- 첨단 농축산기자재 박람회 참여(2019.10.30.~11.02. ,일산 킨텍스)(예정)
- 김제 농업기계박람회 참여(2019.11.05.~11.08 ,김제)(예정)

3.1. 농식품ICT우수업체 가이드북 제작

- 제작기간 : 2019.04.~07.
- 배포기간 : 2019.07.01.~현재(2,000부)
- 목적 : 스마트팜 표준화 촉진을 위하여 ICT 기업들의 주요 사업내용들을 상세하게 소개하는 책자를 발간, 수요자인 농업인과 관련 기관 및 단체들이 편리하게 관련 업체의 정보를 활용하고 상생할 수 있도록 함
- 참고자료 :



<가이드북 제작안내 발표>



<가이드북 원본>

회차	구분	장소	주최	주최자	주최자 연락처	주최자 주소	주최자 전화	주최자 이메일	주최자 홈페이지	주최자 소개	주최자 사업	주최자 대표	주최자 대표 연락처	주최자 대표 이메일	주최자 대표 홈페이지
1	농업	2019-07-17	041-4321-4700	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
2	농업	2019-07-17	041-4321-4711	농업	최정민	경남	010-6229-4444			농업	농업	최정민	010-6229-4444		
3	농업	2019-07-17	041-4321-4744	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
4	농업	2019-07-17	041-4321-4755	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
5	농업	2019-07-17	041-4321-4766	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
6	농업	2019-07-17	041-4321-4777	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
7	농업	2019-07-17	041-4321-4788	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
8	농업	2019-07-17	041-4321-4799	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
9	농업	2019-07-17	041-4321-4800	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
10	농업	2019-07-17	041-4321-4801	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
11	농업	2019-07-17	041-4321-4802	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
12	농업	2019-07-17	041-4321-4803	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
13	농업	2019-07-17	041-4321-4804	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
14	농업	2019-07-17	041-4321-4805	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
15	농업	2019-07-17	041-4321-4806	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
16	농업	2019-07-17	041-4321-4807	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
17	농업	2019-07-17	041-4321-4808	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
18	농업	2019-07-17	041-4321-4809	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
19	농업	2019-07-17	041-4321-4810	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
20	농업	2019-07-17	041-4321-4811	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
21	농업	2019-07-17	041-4321-4812	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
22	농업	2019-07-17	041-4321-4813	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
23	농업	2019-07-17	041-4321-4814	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
24	농업	2019-07-17	041-4321-4815	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
25	농업	2019-07-17	041-4321-4816	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
26	농업	2019-07-17	041-4321-4817	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
27	농업	2019-07-17	041-4321-4818	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
28	농업	2019-07-17	041-4321-4819	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
29	농업	2019-07-17	041-4321-4820	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
30	농업	2019-07-17	041-4321-4821	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
31	농업	2019-07-17	041-4321-4822	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
32	농업	2019-07-17	041-4321-4823	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
33	농업	2019-07-17	041-4321-4824	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
34	농업	2019-07-17	041-4321-4825	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
35	농업	2019-07-17	041-4321-4826	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
36	농업	2019-07-17	041-4321-4827	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
37	농업	2019-07-17	041-4321-4828	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
38	농업	2019-07-17	041-4321-4829	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
39	농업	2019-07-17	041-4321-4830	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
40	농업	2019-07-17	041-4321-4831	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
41	농업	2019-07-17	041-4321-4832	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
42	농업	2019-07-17	041-4321-4833	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
43	농업	2019-07-17	041-4321-4834	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
44	농업	2019-07-17	041-4321-4835	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
45	농업	2019-07-17	041-4321-4836	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
46	농업	2019-07-17	041-4321-4837	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
47	농업	2019-07-17	041-4321-4838	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
48	농업	2019-07-17	041-4321-4839	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
49	농업	2019-07-17	041-4321-4840	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		
50	농업	2019-07-17	041-4321-4841	농업	김지남	경남	010-5379-4444			농업	농업	김지남	010-5379-4444		

<가이드북 배포리스트>

스마트팜
업체
가이드북
제작

3.2. 홍보 리플렛 제작 및 배포

- 제작기간 : 2019.08.01.~30.
- 배포기간 : 2019.08.29.~현재(1,000부)
- 참고자료 :



<홍보리플렛 원본>

나. 표준화 컨버터 개발 : 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발

(1) 표준 API 스펙

표준화 컨버터	<p>○ 표준 API 스펙</p> <ul style="list-style-type: none"> - https://github.com/ebio-snu/stdcvt - https://github.com/ebio-snu/cvtdriver
---------	---

□ 오픈소스로 표준컨버터와 드라이버 API 개발 소스 공개

○ 표준컨버터 관련 소스를 모두 개발자 소스 공개 사이트인 github에 공개하여 각 개발업체들이 활용할 수 있도록 하였음

- 표준컨버터 소스 공개 : <https://github.com/ebio-snu/stdcvt>

- 컨버터 스펙을 기준으로 드라이버 개발을 돕기위한

저장소(<https://github.com/ebio-snu/cvtdriver>)

ebio-snu / stdcvt

Watch 0 Star 1 Fork 1

Code Issues 5 Pull requests 0 Projects 1 Security Insights

Join GitHub today

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

스마트팜 기기 연동을 위한 컨버터 개발

smart-farm greenhouse-control-system greenhouse-monitoring greenhouse

76 commits 1 branch 0 releases 2 contributors MIT

Branch: master New pull request Find file Clone or download

Commit	Message	Time
joonyong-jinong	nodejs 버전 수정	Latest commit 8fae630 on 17 Apr 2018
cmake/Modules	cmake 관련 설정파일 추가	2 years ago
conf	TESTUI 업데이트	2 years ago
cvtdriver @ 62f480c	cvtdriver 업데이트	2 years ago
doc	nodejs 버전 수정	2 years ago
include	설정 및 로깅등 소소한 변경	2 years ago
node	#4 샘플노드 프로토콜 및 코드 수정	2 years ago
scripts	실행 스크립트 변경	2 years ago
src	설정에서 컨버터의 타임아웃을 설정하도록 수정	2 years ago
testui	db ipaddress modified	2 years ago
.gitignore	Initial commit	2 years ago
.gitmodules	컨버터 초기버전	2 years ago
CMakeLists.txt	컨버터 초기버전	2 years ago
LICENSE	Initial commit	2 years ago
README.md	문서업데이트	2 years ago

<표준 API 스펙>

Converting Driver for SmartFarm Devices

cvtdriver

cvtdriver 는 스마트팜장비 연동을 위한 컨버터 개발 프로젝트에서 활용되는 드라이버개발을 돕기 위한 프로젝트이다. 본 프로젝트에서 드라이버 개발을 위한 API 스펙 및 샘플 드라이버를 제공하여 참여사의 드라이버 제작을 돕는다.

클래스 구조

드라이버 클래스 구조는 다음과 같다.

- CvtDriver : 드라이버의 인터페이스가 되는 가상 클래스이다.
- CvtOption : 드라이버 구동을 위한 옵션을 전달하는 클래스이다.
- CvtDeviceSpec : 드라이버에서 다루는 장비의 스펙을 다루는 클래스이다. 장비종류 , 설치위치 , 장비의 작동대상 , 제조사 , 모델명을 처리한다.
- CvtDevice : 드라이버에서 다루는 장비의 인터페이스가 되는 가상 클래스이다.
- CvtSensor : 센서의 인터페이스가 되는 클래스이다. 간단한 경우라면 직접 사용이 가능하다.
- CvtActuator : 일반 구동기의 인터페이스가 되는 클래스이다. 스위치형 구동기에 사용하면 된다.
- CvtMotor : 모터형 구동기의 인터페이스가 되는 클래스로 CvtActuator 를 상속한다. 간단한 경우라면 직접 사용이 가능하다.
- CvtCommand : 구동기에서 처리하는 명령의 인터페이스가 되는 클래스이다.
- CvtRatioCommand : 구동기 전달하는 명령이 비율 (퍼센트)인 경우에 사용하는 클래스이다. 모터형 구동기에 적합한 명령클래스라고 할 수 있다.

샘플 드라이버

두가지 종류의 샘플드라이버를 제공한다.

- DSSampleDriver : 별도로 제공되는 샘플 노드와 serial 통신을 수행하는 드라이버이다. boost::asio 를 사용한다.
- SSSampleDriver : 별도로 제공되는 테스트 UI 와 연동되는 드라이버이다. (현재는 구현이 안되어있다.)

계통도 색인

[클래스 계통도]

이 상속 목록은 완전하진 않지만 알파벳순으로 대략적으로 정렬되어있습니다.:

stdcvt::CvtCommand
stdcvt::CvtRatioCommand
stdcvt::CvtRawCommand

stdcvt::CvtDevice
stdcvt::CvtActuator
ebiodriver::SSSwitch.
stdcvt::CvtMotor
ebiodriver::SSMotor

stdcvt::CvtSensor
ebiodriver::SSSensor

stdcvt::CvtDeviceFactory
stdcvt::CvtDeviceSpec
stdcvt::CvtDriver
ebiodriver::DSSampleDriver
ebiodriver::SSSampleDriver

stdcvt::CvtOption.
stdcvt::CvtSpec

데이타 구조 색인

데이타 구조

다음은 데이타 구조들입니다. (간략한 설명만을 보여줍니다) :

stdcvt::CvtActuator
stdcvt::CvtCommand

stdcvt::CvtDevice
stdcvt::CvtDeviceFactory
stdcvt::CvtDeviceSpec
stdcvt::CvtDriver
stdcvt::CvtMotor
stdcvt::CvtOption
stdcvt::CvtRatioCommand
stdcvt::CvtRawCommand
stdcvt::CvtSensor
stdcvt::CvtSpec
ebiodriver::DSSampleDriver
ebiodriver::SSMotor
ebiodriver::SSSampleDriver
ebiodriver::SSSensor
ebiodriver::SSSwitch

파일 색인

파일 목록

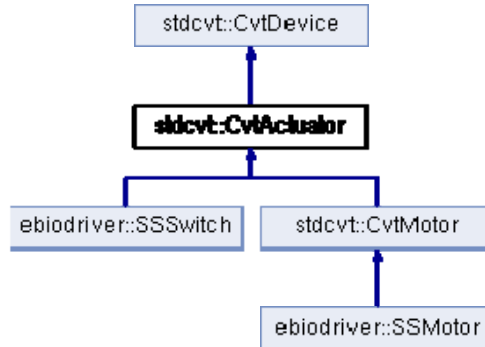
다음은 문서화된 모든 파일에 대한 목록입니다. (간략한 설명만을 보여줍니다) :

sample/dssampledriver.cpp
sample/sssampledriver.cpp
spec/cvtcode.h
spec/cvtcodedic.h
spec/cvtcommand.h
spec/cvtdevice.h
spec/cvtdevicefactory.h
spec/cvtdevicespec.h
spec/cvtdriver.h
spec/cvtoption.h
spec/cvtspec.h

데이터 구조 문서화

stdcvt::CvtActuator 클래스 참조

stdcvt::CvtActuator 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtActuator (string devid, CvtDeviceSpec *pdevspec, devstat_t devstatus)
- CvtActuator (string devid, devtype_t devtype, devsec_t section, devtarget_t target, devstat_t devstatus)
- CvtDevice * clone ()
- bool copy (CvtDevice *pdevice)
- int getlastcmdid ()
- bool turnon ()
- bool turnoff ()
- bool getonoff ()
- bool order (CvtCommand *pcmd)
- void executed (int cmdid)
- string toString ()

Protected 멤버 함수

- void setcommand (CvtCommand *pcmd)

추가로 상속된 멤버들

생성자 &소멸자 문서화

stdcvt::CvtActuator::CvtActuator (string devid , CvtDeviceSpec * pdevspec , devstat_t devstatus)[inline]

새로운 구동기를 생성한다.

매개변수 :

devid	장비의 아이디
pdevspec	장비의 스펙
devstatus	장비의 상태

stdcvt::CvtActuator::CvtActuator (string devid , devtype_t devtype , devsec_t section , devtarget_t target , devstat_t devstatus) [inline]

새로운 구동기를 생성한다.

매개변수 :

devid	장비의 아이디
devtype	장비의 종류
section	장비 설치 구역
target	장비의 대상
devstatus	장비의 상태

멤버 함수 문서화

CvtDevice* stdcvt::CvtActuator::clone () [inline], [virtual]

장비의 클론을 만든다.

반환값 :

클론의 포인터

stdcvt::CvtDevice (페이지 pagenum)를 구현.

stdcvt::CvtMotor (페이지 pagenum)에서 재구현되었습니다.

bool stdcvt::CvtActuator::copy (CvtDevice * pdevice) [inline], [virtual]

장비 정보를 복사한다.

반환값 :

복사가 성공하면 true.

stdcvt::CvtDevice (페이지 pagenum)를 구현.

stdcvt::CvtMotor (페이지 pagenum)에서 재구현되었습니다.

void stdcvt::CvtActuator::executed (int cmdid) [inline]

전달완료된 명령아이디를 세팅한다.

매개변수 :

cmdid	전달완료된 명령아이디
-------	-------------

int stdcvt::CvtActuator::getlastcmdid ()[inline]

최종 명령의 아이디를 확인한다.

반환값 :

최종 명령의 아이디. 없을경우 -1

bool stdcvt::CvtActuator::getonoff ()[inline]

장비작동 명령을 확인한다.

반환값 :

작동상태. true 면 on.

bool stdcvt::CvtActuator::order (CvtCommand * pcmd)[inline]

명령을 지시한다. 실제 실행하는 것은 아니고 내부에 명령을 저장하고 있다가 실제 장비에게 전달하는 역할을 담당한다.

매개변수 :

pcmd	명령의 포인터
------	---------

반환값 :

명령 저장 여부.

void stdcvt::CvtActuator::setcommand (CvtCommand * pcmd)[inline], [protected]

명령을 세팅한다. order 구현시 한번씩 호출해주면 최종 아이디를 기억하도록 한다.

매개변수 :

pcmd	명령에 대한 포인터
------	------------

string stdcvt::CvtActuator::tostring ()[inline]

구동기의 상태를 문자열로 내보낸다.

반환값 :

구동기의 상태 문자열

bool stdcvt::CvtActuator::turnoff ()[inline]

장비를 작동을 중지한다.

반환값 :

작동상태. true 면 on.

bool stdcvt::CvtActuator::turnon ()[inline]

장비를 작동시킨다.

반환값 :

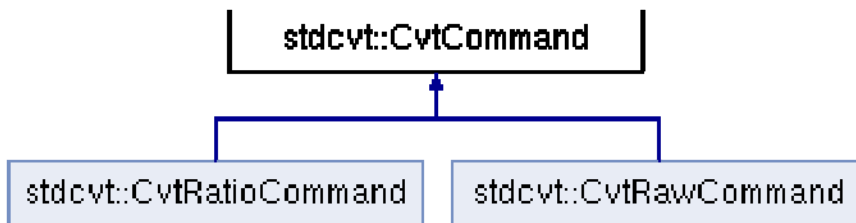
작동상태. true 면 on.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

● spec/cvtdevice.h

stdcvt::CvtCommand 클래스 참조

stdcvt::CvtCommand 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtCommand (int cmdid, CvtDeviceSpec *pdevspec, bool onoff)
- int getid ()
- int setid (int id)
- CvtDeviceSpec * getdevspec ()
- bool getonoff ()
- bool setonoff (bool onoff)

생성자 &소멸자 문서화

stdcvt::CvtCommand::CvtCommand (int cmdid , CvtDeviceSpec * pdevspec , bool onoff) [inline]

새로운 명령을 생성한다.

매개변수 :

cmdid	명령의 아이디
pdevspec	명령을 수행할 장비스펙의 포인터
onoff	작동상태

멤버 함수 문서화

CvtDeviceSpec* stdcvt::CvtCommand::getdevspec ()[inline]

명령에 부여된 장비스펙을 리턴한다.

반환값 :

명령을 실행할 장비스펙

int stdcvt::CvtCommand::getid ()[inline]

명령에 부여된 아이디를 리턴한다.

반환값 :

명령의 아이디

bool stdcvt::CvtCommand::getonoff ()[inline]

on/off 명령을 확인한다.

반환값 :

on 이면 true

int stdcvt::CvtCommand::setid (int id)[inline]

명령에 부여된 아이디를 세팅한다.

매개변수 :

id	새로운 명령 아이디
----	------------

반환값 :

새로 저장된 명령의 아이디

bool stdcvt::CvtCommand::setonoff (bool onoff)[inline]

on/off 명령을 세팅한다.

매개변수 :

onoff	on 이면 true
-------	------------

반환값 :

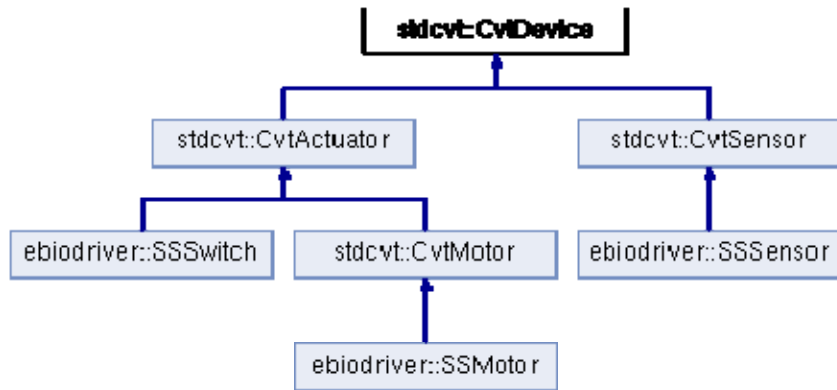
세팅된 상태.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtcommand.h

stdcvt::CvtDevice 클래스 참조

stdcvt::CvtDevice 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtDevice (string devid, CvtDeviceSpec *pdevspec, devstat_t devstatus)
- CvtDevice (string devid, devtype_t devtype, devsec_t section, devtarget_t target, devstat_t devstatus)
- string getid ()
- CvtDeviceSpec * getspec ()
- devstat_t getstatus ()
- devstat_t setstatus (devstat_t devstatus)
- string toString ()
- virtual CvtDevice * clone ()=0
- virtual bool copy (CvtDevice *pdevice)=0

정적 Public 멤버 함수

- static devgroup_t getgroup (devtype_t devtype)

Protected 멤버 함수

- void _copy (CvtDevice *pdevice)

생성자 &소멸자 문서화

`stdcvt::CvtDevice::CvtDevice (string devid , CvtDeviceSpec * pdevspec , devstat_t devstatus) [inline]`

새로운 장비를 생성한다.

매개변수 :

devid	장비의 아이디
pdevspec	장비의 스펙
devstatus	장비의 상태

stdcvt::CvtDevice::CvtDevice (string devid , devtype_t devtype , devsec_t section , devtarget_t target , devstat_t devstatus)[inline]

새로운 장비를 생성한다.

매개변수 :

devid	장비의 아이디
devtype	장비의 종류
section	장비 설치 구역
target	장비의 대상
devstatus	장비의 상태

멤버 함수 문서화

virtual CvtDevice* stdcvt::CvtDevice::clone ()[pure virtual]

장비의 클론을 만든다.

반환값 :

클론의 포인터

stdcvt::CvtMotor (페이지 pagenum), stdcvt::CvtActuator (페이지 pagenum), stdcvt::CvtSensor (페이지 pagenum)에서 구현되었습니다.

virtual bool stdcvt::CvtDevice::copy (CvtDevice * pdevice)[pure virtual]

장비 정보를 복사한다.

반환값 :

복사가 성공하면 true.

stdcvt::CvtMotor (페이지 pagenum), stdcvt::CvtActuator (페이지 pagenum), stdcvt::CvtSensor (페이지 pagenum)에서 구현되었습니다.

static devgroup_t stdcvt::CvtDevice::getgroup (devtype_t devtype)[inline], [static]

장비 그룹정보를 확인한다.

string stdcvt::CvtDevice::getid ()[inline]

장비에 부여된 아이디를 리턴한다.

반환값 :
장비의 아이디

CvtDeviceSpec* stdcvt::CvtDevice::getspec ()[inline]

장비의 스펙을 리턴한다.

반환값 :
장비 스펙의 포인터

devstat_t stdcvt::CvtDevice::getstatus ()[inline]

장비의 상태를 리턴한다.

반환값 :
장비의 상태

devstat_t stdcvt::CvtDevice::setstatus (devstat_t devstatus)[inline]

장비의 상태를 세팅한다.

매개변수 :

devstatus	새로 세팅할 장비의 상태
-----------	---------------

반환값 :
세팅된 장비의 상태

string stdcvt::CvtDevice::tostring ()[inline]

장비의 상태를 문자열로 내보낸다.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtdevice.h

stdcvt::CvtDeviceFactory 클래스 참조

Public 멤버 함수

- CvtDeviceFactory ()
- void setsensors (json sensors)
- void setmotors (json motors)
- void setswitches (json switches)
- CvtSensor * newsensor (int index)
- CvtActuator * newswitch (intindex)
- CvtMotor * newmotor (int index)

생성자 &소멸자 문서화

stdcvt::CvtDeviceFactory::CvtDeviceFactory ()[inline]

새로운 장비팩토리를 생성한다.

멤버 함수 문서화

CvtMotor* stdcvt::CvtDeviceFactory::newmotor (int index) [inline]

모터를 생성한다. 생성된 인스턴스에 대한 해제책임은 사용자에게 있다.

매개변수 :

index	0 에서 시작하는 인덱스값.
-------	-----------------

반환값 :

CvtMotor 의 포인터. nullptr 이면 설정없음.

CvtSensor* stdcvt::CvtDeviceFactory::newsensor (int index) [inline]

센서를 생성한다. 생성된 인스턴스에 대한 해제책임은 사용자에게 있다.

매개변수 :

index	0 에서 시작하는 인덱스값.
-------	-----------------

반환값 :

CvtSensor 의 포인터. nullptr 이면 설정없음.

CvtActuator* stdcvt::CvtDeviceFactory::newswitch (int index) [inline]

스위치를 생성한다. 생성된 인스턴스에 대한 해제책임은 사용자에게 있다.

매개변수 :

index	0 에서 시작하는 인덱스값.
-------	-----------------

반환값 :

CvtActuator 의 포인터. nullptr 이면 설정없음.

void stdcvt::CvtDeviceFactory::setmotors (json motors) [inline]

센서 생성을 위한 옵션을 추가한다.

매개변수 :

motors	모터를 위한 옵션. json 타입의 포인터.
--------	--------------------------

void stdcvt::CvtDeviceFactory::setsensors (json sensors) [inline]

센서 생성을 위한 옵션을 추가한다.

매개변수 :

sensors	센서를 위한 옵션. json 타입의 포인터.
---------	--------------------------

void stdcvt::CvtDeviceFactory::setswitches (json switches) [inline]

센서 생성을 위한 옵션을 추가한다.

매개변수 :

switches	스위치를 위한 옵션. json 타입의 포인터.
----------	---------------------------

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtdevicefactory.h

stdcvt::CvtDeviceSpec 클래스 참조

Public 멤버 함수

- CvtDeviceSpec ()
- CvtDeviceSpec (devtype_t devtype, devsec_t section, devtarget_t target)
- devgroup_t getgroupype ()
- devtype_t gettype ()
- devsec_t getsection ()
- devtarget_t gettarget ()
- string setmanufacturer (string manufacturer)
- string getmanufacturer ()
- string setmodel (string model)
- string getmodel ()
- bool copy (CvtDeviceSpec *pdevspec)
- string toString ()
- bool checktarget (CvtDeviceSpec *pdevspec)
- bool checktype (CvtDeviceSpec *pdevspec)
- bool checksection (CvtDeviceSpec *pdevspec)
- boolismatched (CvtDeviceSpec *pdevspec)

생성자 & 소멸자 문서화

stdcvt::CvtDeviceSpec::CvtDeviceSpec ()[inline]

새로운 장비스펙을 생성한다.

stdcvt::CvtDeviceSpec::CvtDeviceSpec (devtype_t devtype , devsec_t section , devtarget_t target) [inline]

새로운 장비스펙을 생성한다.

매개변수 :

devtype	장비의 종류
section	장비 설치 구역
target	장비의 대상

멤버 함수 문서화

bool stdcvt::CvtDeviceSpec::checksection (CvtDeviceSpec * pdevspec) [inline]

장비설치위치가 매치되는지 확인한다.

매개변수 :

pdevspec	확인할 소스 장비스펙에 대한 포인터
----------	---------------------

bool stdcvt::CvtDeviceSpec::checktarget (CvtDeviceSpec * pdevspec) [inline]

장비 작동대상이 매치되는지 확인한다.

매개변수 :

pdevspec	확인할 소스 장비스펙에 대한 포인터
----------	---------------------

bool stdcvt::CvtDeviceSpec::checktype (CvtDeviceSpec * pdevspec) [inline]

장비타입이 매치되는지 확인한다.

매개변수 :

pdevspec	확인할 소스 장비스펙에 대한 포인터
----------	---------------------

bool stdcvt::CvtDeviceSpec::copy (CvtDeviceSpec * pdevspec) [inline]

장비의 속성을 복사한다.

매개변수 :

pdevspec

복사할 소스 장비스펙에 대한 포인터

devgroup_t stdcvt::CvtDeviceSpec::getgroupype ()[inline]

장비그룹의 종류를 리턴한다.

반환값 :

장비그룹의 종류

string stdcvt::CvtDeviceSpec::getmanufacturer ()[inline]

장비의 제조사를 확인한다.

반환값 :

세팅된 장비 제조사

string stdcvt::CvtDeviceSpec::getmodel ()[inline]

장비의 모델을 확인한다.

반환값 :

세팅된 장비 모델

devsec_t stdcvt::CvtDeviceSpec::getsection ()[inline]

장비의 설치구역을 리턴한다.

반환값 :

장비의 설치구역

devtarget_t stdcvt::CvtDeviceSpec::gettarget ()[inline]

장비의 대상을 리턴한다.

반환값 :

장비의 대상

devtype_t stdcvt::CvtDeviceSpec::gettype ()[inline]

장비의 종류를 리턴한다.

반환값 :

장비의 종류

bool stdcvt::CvtDeviceSpec::ismatched (CvtDeviceSpec * pdevspec)[inline]

장비스펙이 매치되는지 확인한다.

매개변수 :

pdevspec

확인할 소스 장비스펙에 대한 포인터

string stdcvt::CvtDeviceSpec::setmanufacturer (string manufacturer) [inline]

장비의 제조사를 세팅한다.

매개변수 :

manufacturer	장비 제조사
--------------	--------

반환값 :

세팅된 장비 제조사

string stdcvt::CvtDeviceSpec::setmodel (string model) [inline]

장비의 모델을 세팅한다.

매개변수 :

model	장비 모델
-------	-------

반환값 :

세팅된 장비 모델

string stdcvt::CvtDeviceSpec::tostring () [inline]

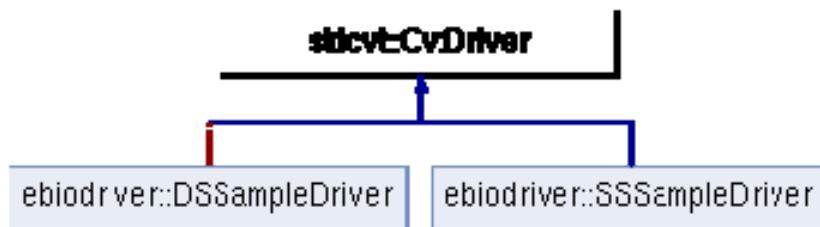
장비의 스펙을 문자열로 내보낸다.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtdevicespec.h

stdcvt::CvtDriver 클래스 참조

stdcvt::CvtDriver 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtDriver (int modelcode, int apispec)
- int getmodelcode ()
- int getapispec ()

- time_t getlastupdated ()
- void updated ()
- virtual string getversion ()=0
- virtual string getmodel ()=0
- virtual string getcompany ()=0
- virtual bool initialize (CvtOption option)=0
- virtual bool finalize ()=0
- virtual bool preprocess ()=0
- virtual bool postprocess ()=0
- virtual CvtDevice * getdevice (int index)=0
- virtual bool sharedevice (CvtDevice *pdevice)=0
- virtual CvtCommand * getcommand ()=0
- virtual bool control (CvtCommand *pcmd)=0

생성자 & 소멸자 문서화

stdcvt::CvtDriver::CvtDriver (int modelcode , int apispec) [inline]

새로운 드라이버를 생성한다.

매개변수 :

modelcode	모델 코드
apispec	API 버전

멤버 함수 문서화

virtual bool stdcvt::CvtDriver::control (CvtCommand * pcmd) [pure virtual]

다른 드라이버로부터 명령을 받아 처리한다.

매개변수 :

pcmd	명령에 대한 포인터
------	------------

반환값 :

실제 명령의 처리 여부가 아니라 명령을 수신했는지 여부이다. 해당 명령을 실행할 장비가 없다면 false 이다.

ebiodriver::DSSampleDriver (페이지/ pagenum), **ebiodriver::SSSampleDriver** (페이지/ pagenum)에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::finalize ()[pure virtual]

드라이버를 종료한다.

반환값 :

종료 성공 여부

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

int stdcvt::CvtDriver::getapispec ()[inline]

드라이버의 API 버전을 확인한다.

반환값 :

드라이버의 API 버전

virtual CvtCommand* stdcvt::CvtDriver::getcommand ()[pure virtual]

다른 드라이버가 관리하고 있는 장비를 제어하고자 할때 명령을 전달한다. 명령을 전달하지 않는 드라이버라면 그냥 NULL 을 리턴하도록 만들면 된다. NULL 이 나올때까지 반복적으로 호출한다.

반환값 :

명령의 포인터. NULL 이라면 이후에 명령이 없다는 의미이다.

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

virtual string stdcvt::CvtDriver::getcompany ()[pure virtual]

드라이버 제조사명을 확인한다. 컨버터에서는 제조사명을 로깅용도로만 사용한다.

반환값 :

문자열 형식의 제조사명

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

virtual CvtDevice* stdcvt::CvtDriver::getdevice (int index) [pure virtual]

드라이버가 관리하고 있는 장비의 포인터를 꺼내준다. 모든 장비를 꺼내주지않고 , 변경된 장비만을 꺼내주는 방식으로 효율을 높일 수 있다.

매개변수 :

index	얻고자 하는 장비의 인덱스 번호. 0 에서 시작한다.
-------	-------------------------------

반환값 :

인덱스에 해당하는 장비의 포인터. NULL 이라면 이후에 장비가 없다는 의미이다.

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

time_t stdcvt::CvtDriver::getlastupdated ()[inline]

드라이버 가장 최근 업데이트된 시간을 확인한다.

반환값 :

드라이버의 최근 업데이트 시각

virtual string stdcvt::CvtDriver::getmodel ()[pure virtual]

드라이버 제작자가 부여하는 모델번호를 확인한다. 컨버터에서는 모델코드만 확인하고, 모델번호에 대해서는 로깅용으로만 사용한다.

반환값 :

문자열 형식의 모델번호

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

int stdcvt::CvtDriver::getmodelcode ()[inline]

드라이버의 모델코드를 확인한다.

반환값 :

드라이버의 모델코드

virtual string stdcvt::CvtDriver::getversion ()[pure virtual]

드라이버 제작자가 부여하는 버전번호를 확인한다. 컨버터에서는 해당 버전을 로깅용으로만 사용한다. 문자열 비교를 통해 후순위가 더 높은 버전이 된다.

반환값 :

문자열 형식의 버전번호

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::initialize (CvtOption option) [pure virtual]

드라이버를 초기화 한다. 드라이버 동작을 위한 option 은 key-value 형식으로 전달 된다.

매개변수 :

option	드라이버동작을 위한 옵션
--------	---------------

반환값 :

초기화 성공 여부

ebiodriver::DSSampleDriver (페이지 *pagenum*), **ebiodriver::SSSampleDriver** (페이지 *pagenum*)에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::postprocess () [pure virtual]

드라이버간 상태교환이 이루어진 이후에 호출되는 메소드로 후처리를 수행한다.

반환값 :

후처리 성공 여부

ebiodriver::DSSampleDriver (페이지 *pagenum*), **ebiodriver::SSSampleDriver** (페이지 *pagenum*)에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::preprocess () [pure virtual]

드라이버간 상태교환을 하기전에 호출되는 메소드로 전처리를 수행한다.

반환값 :

전처리 성공 여부

ebiodriver::DSSampleDriver (페이지 *pagenum*), **ebiodriver::SSSampleDriver** (페이지 *pagenum*)에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::sharedevice (CvtDevice * pdevice) [pure virtual]

전달된 장비의 정보를 획득한다. 다른 드라이버의 장비정보를 입력해주기 위해 컨버터가 호출한다.

매개변수 :

pdevice	다른 드라이버의 장비 포인터
---------	-----------------

반환값 :

성공여부. 관심이 없는 장비인 경우라도 문제가 없으면 true 를 리턴한다.

ebiodriver::DSSampleDriver (*페이지 pagenum*), **ebiodriver::SSSampleDriver** (*페이지 pagenum*)에서 구현되었습니다.

void stdcvt::CvtDriver::updated ()[inline]

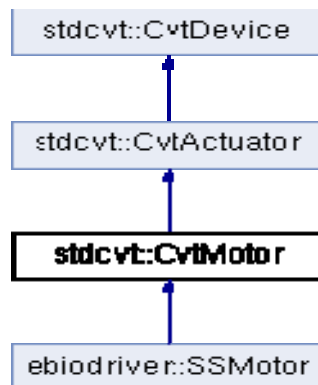
드라이버의 내용이 업데이트되면 호출한다. 관리하고 있는 장비의 데이터가 변경되면 무조건 호출해 주어야 한다. 현재는 최종업데이트 시간만을 관리한다.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtdriver.h

stdcvt::CvtMotor 클래스 참조

stdcvt::CvtMotor 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtMotor (string devid, CvtDeviceSpec *pdevspec, devstat_t devstatus)
- CvtMotor (string devid, devtype_t devtype, devsec_t section, devtarget_t target, devstat_t devstatus)
- CvtDevice * clone ()
- bool copy (CvtDevice *pdevice)
- double settarget (double ratio)
- double gettarget ()
- double setcurrent (double ratio)
- double getcurrent ()

- string toString ()
- bool order (CvtCommand *pcmd)

추가로 상속된 멤버들

생성자 &소멸자 문서화

stdcvt::CvtMotor::CvtMotor (string devid , CvtDeviceSpec * pdevspec , devstat_t devstatus)*[inline]*

새로운 모터형 구동기를 생성한다.

매개변수 :

devid	장비의 아이디
pdevspec	장비의 스펙
devstatus	장비의 상태

stdcvt::CvtMotor::CvtMotor (string devid , devtype_t devtype , devsec_t section , devtarget_t target , devstat_t devstatus)*[inline]*

새로운 모터형 구동기를 생성한다.

매개변수 :

devid	장비의 아이디
devtype	장비의 종류
section	장비 설치 구역
target	장비의 대상
devstatus	장비의 상태

멤버 함수 문서화

CvtDevice* stdcvt::CvtMotor::clone ()*[inline]*, *[virtual]*

장비의 클론을 만든다.

반환값 :

클론의 포인터

stdcvt::CvtActuator (*페이지* pagenum)(으)로부터 재구현되었습니다.

bool stdcvt::CvtMotor::copy (CvtDevice * pdevice)[inline], [virtual]

장비 정보를 복사한다.

반환값 :

복사가 성공하면 true.

stdcvt::CvtActuator (페이지 *pagenum*)(으)로부터 재구현되었습니다.

double stdcvt::CvtMotor::getcurrent ()[inline]

모터형 구동기의 현재 위치를 확인한다.

반환값 :

구동기의 위치

double stdcvt::CvtMotor::gettarget ()[inline]

모터형 구동기의 목표 위치를 확인한다.

반환값 :

구동기의 위치

bool stdcvt::CvtMotor::order (CvtCommand * pcmd)[inline]

명령을 지시한다. 실제 실행하는 것은 아니고 내부에 명령을 저장하고 있다가 실제 장비에게 전달하는 역할을 담당한다.

매개변수 :

pcmd	명령의 포인터
------	---------

반환값 :

실행명령이 저장되면 true, 실행할 명령이 아니면 false

double stdcvt::CvtMotor::setcurrent (double ratio)[inline]

모터형 구동기의 현재 위치를 세팅한다.

매개변수 :

ratio	현재 위치
-------	-------

반환값 :

현재 위치

double stdcvt::CvtMotor::settarget (double ratio)[inline]

모터형 구동기의 목표 위치를 세팅한다.

매개변수 :

ratio	세팅할 위치
-------	--------

반환값 :

세팅된 위치

string stdcvt::CvtMotor::tostring ()[inline]

모터형 구동기의 상태를 문자열로 내보낸다.

반환값 :

모터형 구동기의 상태 문자열

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtdevice.h

stdcvt::CvtOption 클래스 참조

Public 멤버 함수

- CvtOption (json option)
- json getjson (string path)
- string get (string key)
- double getdouble (string key)
- int getint (string key)
- void * getobject (string key)
- void * setobject (string key, void *object)
- CvtDeviceFactory * getdevfactory ()

생성자 &소멸자 문서화

stdcvt::CvtOption::CvtOption (json option) [inline]

새로운 옵션을 생성한다.

매개변수 :

option	특정 드라이버를 위한 옵션. json 타입의 포인터.
--------	-------------------------------

멤버 함수 문서화

string stdcvt::CvtOption::get (string key) [inline]

옵션의 값을 문자열로 리턴한다.

매개변수 :

key	옵션을 선택하기 위한 키
-----	---------------

반환값 :

옵션의 문자열 값

CvtDeviceFactory* stdcvt::CvtOption::getdevfactory () [inline]

디바이스팩토리를 얻는다.

반환값 :

디바이스팩토리의 포인터

double stdcvt::CvtOption::getdouble (string key) [inline]

옵션의 값을 실수형으로 리턴한다.

매개변수 :

key	옵션을 선택하기 위한 키
-----	---------------

반환값 :

옵션의 실수형값

int stdcvt::CvtOption::getint (string key) [inline]

옵션의 값을 정수형으로 리턴한다.

매개변수 :

key	옵션을 선택하기 위한 키
-----	---------------

반환값 :

옵션의 값

json stdcvt::CvtOption::getjson (string path) [inline]

복잡한 옵션을 사용하고자 할때에는 옵션의 값을 리턴한다.

매개변수 :

path	JSONPATH 로 나타낸 경로
------	-------------------

반환값 :

옵션의 값

void* stdcvt::CvtOption::getobject (string key) [inline]

옵션의 값을 void *로 리턴한다.

매개변수 :

key	옵션을 선택하기 위한 키
-----	---------------

반환값 :

옵션의 값

void* stdcvt::CvtOption::setobject (string key , void * object) [inline]

옵션의 값을 세팅한다.

매개변수 :

key	옵션을 선택하기 위한 키
object	옵션값

반환값 :

옵션의 값

stdcvt::CvtRatioCommand 클래스 참조

stdcvt::CvtRatioCommand 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtRatioCommand (int cmdid, CvtDeviceSpec *pdevspec, bool onoff, double ratio)
- double getratio ()

생성자 & 소멸자 문서화

stdcvt::CvtRatioCommand::CvtRatioCommand (int cmdid , CvtDeviceSpec * pdevspec , bool onoff , double ratio) [inline]

새로운 명령을 생성한다.

매개변수 :

cmdid	명령의 아이디
pdevspec	명령을 수행할 장비스펙의 포인터
onoff	작동상태
ratio	지정된 비율

멤버 함수 문서화

double stdcvt::CvtRatioCommand::getratio () [inline]

지정된 비율값을 확인한다.

반환값 :

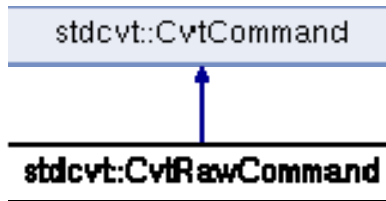
ratio 값

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtcommand.h

stdcvt::CvtRawCommand 클래스 참조

stdcvt::CvtRawCommand 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtRawCommand (int cmdid, CvtDeviceSpec *pdevspec, bool onoff, int modelcode, string devid, string arg)
- int getmodelcode ()
- string getdevid ()
- string getarg ()

생성자 & 소멸자 문서화

stdcv::CvtRawCommand::CvtRawCommand (int cmdid , CvtDeviceSpec * pdevspec , bool onoff , int modelcode , string devid , string arg) [inline]

새로운 명령을 생성한다.

매개변수 :

cmdid	명령의 아이디
pdevspec	명령을 수행할 장비스펙의 포인터
onoff	작동상태
modelcode	명령의 인자를 해석할 수 있는 드라이버의 모델코드
devid	명령을 수행할 장비의 아이디
arg	명령의 인자. 문자열 형으로 꼭 base64 인코딩 되어 있어야 함.

멤버 함수 문서화

string stdcv::CvtRawCommand::getarg () [inline]

명령 수행을 위한 인자를 확인한다.

반환값 :

명령 인자값

string stdcv::CvtRawCommand::getdevid () [inline]

명령을 수행할 장비의 아이디를 확인한다.

반환값 :

장비아이드값

int stdcv::CvtRawCommand::getmodelcode () [inline]

명령에 할당된 모델코드를 확인한다.

반환값 :

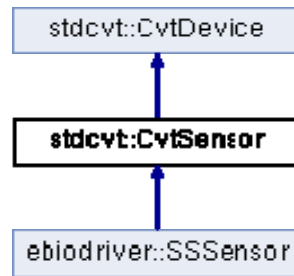
모델코드값

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtcommand.h

stdcvt::CvtSensor 클래스 참조

stdcvt::CvtSensor 에 대한 상속 다이어그램 :



Public 멤버 함수

- CvtSensor (string devid, CvtDeviceSpec *pdevspec, devstat_t devstatus, obsunit_t unit)
- CvtSensor (string devid, devtype_t devtype, devsec_t section, devtarget_t target, devstat_t devstatus, obsunit_t unit)
- CvtDevice * clone ()
- bool copy (CvtDevice *pdevice)
- obsunit_t getunit ()
- obsunit_t setunit (obsunit_t unit)
- double writeobservation (double value)
- double readobservation ()
- string toString ()

추가로 상속된 멤버들

생성자 &소멸자 문서화

`stdcvt::CvtSensor::CvtSensor (string devid , CvtDeviceSpec * pdevspec , devstat_t devstatus , obsunit_t unit) [inline]`

새로운 센서를 생성한다.

매개변수 :

devid	센서의 아이디
pdevspec	장비 스펙
devstatus	센서의 상태
unit	관측치의 단위

stdcvt::CvtSensor::CvtSensor (string devid , devtype_t devtype , devsec_t section , devtarget_t target , devstat_t devstatus , obsunit_t unit) [inline]

새로운 센서를 생성한다.

매개변수 :

devid	센서의 아이디
devtype	장비의 종류
section	장비 설치 구역
target	장비의 대상
devstatus	센서의 상태
unit	관측치의 단위

멤버 함수 문서화

CvtDevice* stdcvt::CvtSensor::clone () [inline], [virtual]

장비의 클론을 만든다.

반환값 :

클론의 포인터

stdcvt::CvtDevice (페이지 pagenum)를 구현.

bool stdcvt::CvtSensor::copy (CvtDevice * pdevice) [inline], [virtual]

장비 정보를 복사한다.

반환값 :

복사가 성공하면 true.

stdcvt::CvtDevice (페이지 pagenum)를 구현.

obsunit_t stdcvt::CvtSensor::getunit () [inline]

관측치 단위를 읽는다.

반환값 :
관측치 단위

double stdcvt::CvtSensor::readobservation ()[inline]

관측치를 읽는다.
반환값 :
관측치값

obsunit_t stdcvt::CvtSensor::setunit (obsunit_t unit)[inline]

관측치 단위를 세팅한다.
매개변수 :

unit	새로 세팅할 관측치 단위
------	---------------

반환값 :
관측치 단위

string stdcvt::CvtSensor::tostring ()[inline]

센서의 상태를 문자열로 내보낸다.

double stdcvt::CvtSensor::writeobservation (double value)[inline]

관측치를 기록한다.
매개변수 :

value	관측치
-------	-----

반환값 :
기록한 관측치값

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtdevice.h

stdcvt::CvtSpec 클래스 참조

Public 멤버 함수

- CvtSpec ()

생성자 & 소멸자 문서화

stdcvt::CvtSpec::CvtSpec () [inline]

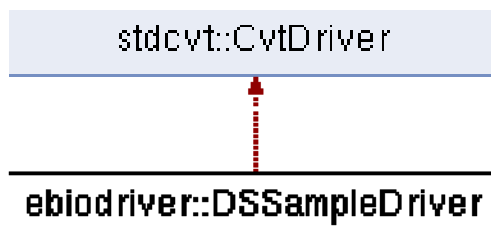
새로운 컨버터스펙을 생성한다.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- spec/cvtspec.h

ebiodriver::DSSampleDriver 클래스 참조

ebiodriver::DSSampleDriver 에 대한 상속 다이어그램 :



Public 멤버 함수

- DSSampleDriver ()
 - string getversion ()
 - string getmodel ()
 - string getcompany ()
 - bool initialize (CvtOption option)
 - bool finalize ()
 - bool preprocess ()
 - bool postprocess ()
 - CvtDevice * getdevice (int index)
 - bool sharedevice (CvtDevice *pdevice)
 - CvtCommand * getcommand ()
 - bool control (CvtCommand *pcmd)
-

생성자 & 소멸자 문서화

ebiodriver::DSSampleDriver::DSSampleDriver () [inline]

새로운 DS 드라이버를 생성한다.

멤버 함수 문서화

bool ebiodriver::DSSampleDriver::control (CvtCommand * pcmd) [inline], [virtual]

다른 드라이버로부터 명령을 받아 처리한다. 구동기를 다루어야 한다면 구현이 되어야 한다.

매개변수 :

pcmd	명령에 대한 포인터
------	------------

반환값 :

실제 명령의 처리 여부가 아니라 명령을 수신했는지 여부이다. 해당 명령을 실행할 장비가 없다면 false 이다.

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::DSSampleDriver::finalize () [inline], [virtual]

드라이버를 종료한다.

반환값 :

종료 성공 여부

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

CvtCommand* ebiodriver::DSSampleDriver::getcommand () [inline], [virtual]

다른 드라이버가 관리하고 있는 장비를 제어하고자 할때 명령을 전달한다. 명령을 전달하지 않는 드라이버라면 그냥 NULL 을 리턴하도록 만들면 된다. DSDriver 에서 는 구현할 필요가 없다.

반환값 :

명령의 포인터. NULL 이라면 이후에 명령이 없다는 의미이다.

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

string ebiodriver::DSSampleDriver::getcompany () [inline], [virtual]

드라이버 제조사명을 확인한다. 컨버터에서는 제조사명을 로깅용도로만 사용한다.

반환값 :

문자열 형식의 제조사명

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

CvtDevice* ebiodriver::DSSampleDriver::getdevice (int index)[inline], [virtual]

드라이버가 관리하고 있는 장비의 포인터를 꺼내준다.

매개변수 :

index	연고자 하는 장비의 인덱스 번호. 0 에서 시작한다.
-------	-------------------------------

반환값 :

인덱스에 해당하는 장비의 포인터. NULL 이라면 이후에 장비가 없다는 의미이다.

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

string ebiodriver::DSSampleDriver::getmodel ()[inline], [virtual]

드라이버 제작자가 부여하는 모델번호를 확인한다.

반환값 :

문자열 형식의 모델번호

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

string ebiodriver::DSSampleDriver::getversion ()[inline], [virtual]

드라이버 제작자가 부여하는 버전번호를 확인한다.

반환값 :

문자열 형식의 버전번호

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::DSSampleDriver::initialize (CvtOption option)[inline], [virtual]

드라이버를 초기화 한다. 드라이버 동작을 위한 option 은 key-value 형식으로 전달된다.

매개변수 :

option	드라이버동작을 위한 옵션
--------	---------------

반환값 :

초기화 성공 여부

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::DSSampleDriver::postprocess ()[inline], [virtual]

드라이버간 상태교환이 이루어진 이후에 호출되는 메소드로 후처리를 수행한다.

반환값 :

후처리 성공 여부

stdcvt::CvtDriver (*페이지* *pagenum*)를 구현.

bool ebiodriver::DSSampleDriver::preprocess ()[inline], [virtual]

드라이버간 상태교환을 하기전에 호출되는 메소드로 전처리를 수행한다.

반환값 :

전처리 성공 여부

stdcvt::CvtDriver (*페이지* *pagenum*)를 구현.

bool ebiodriver::DSSampleDriver::sharedevice (CvtDevice * pdevice)[inline], [virtual]

전달된 장비의 정보를 획득한다. 다른 드라이버의 장비정보를 입력해주기 위해 컨버터가 호출한다. 일반적인 업체별 드라이버에서는 특별히 구현하지 않아도 된다.

매개변수 :

pdevice	다른 드라이버의 장비 포인터
---------	-----------------

반환값 :

성공여부. 관심이 없는 장비인 경우라도 문제가 없으면 true 를 리턴한다.

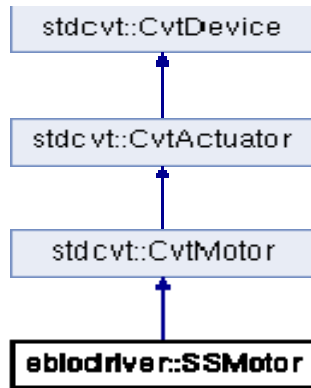
stdcvt::CvtDriver (*페이지* *pagenum*)를 구현.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

| sample/dssampledriver.cpp

ebiodriver::SSMotor 클래스 참조

ebiodriver::SSMotor 에 대한 상속 다이어그램 :



Public 멤버 함수

- SSMotor (int devid, stdcvt::devtype_t devtype, stdcvt::devsec_t section, stdcvt::devtarget_t target, stdcvt::devstat_t devstatus)

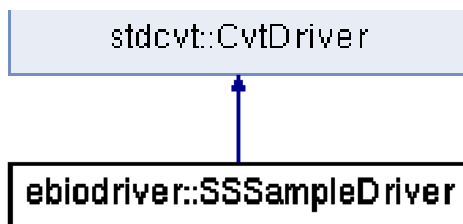
추가로 상속된 멤버들

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- sample/sssampledriver.cpp

ebiodriver::SSSampleDriver 클래스 참조

ebiodriver::SSSampleDriver 에 대한 상속 다이어그램 :



Public 멤버 함수

- SSSampleDriver ()
- string getversion ()
- string getmodel ()
- string getcompany ()
- bool initialize (CvtOption option)
- bool finalize ()
- bool preprocess ()

- bool postprocess ()
- CvtDevice * getdevice (int index)
- bool sharedevice (CvtDevice *pdevice)
- CvtCommand * getcommand ()
- bool control (CvtCommand *pcmd)

생성자 &소멸자 문서화

ebiodriver::SSSampleDriver::SSSampleDriver ()[inline]

새로운 SS 드라이버를 생성한다.

멤버 함수 문서화

bool ebiodriver::SSSampleDriver::control (CvtCommand * pcmd)[inline], [virtual]

다른 드라이버로부터 명령을 받아 처리한다.

매개변수 :

pcmd	명령에 대한 포인터
------	------------

반환값 :

실제 명령의 처리 여부가 아니라 명령을 수신했는지 여부이다. 해당 명령을 실행할 장비가 없다면 false 이다.

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::SSSampleDriver::finalize ()[inline], [virtual]

드라이버를 종료한다.

반환값 :

종료 성공 여부

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

CvtCommand* ebiodriver::SSSampleDriver::getcommand ()[inline], [virtual]

다른 드라이버가 관리하고 있는 장비를 제어하고자 할때 명령을 전달한다. 명령을 전달하지 않는 드라이버라면 그냥 NULL 을 리턴하도록 만들면 된다.

반환값 :

인덱스에 해당하는 명령의 포인터. NULL 이라면 이후에 명령이 없다는 의미이다.
stdcvt::CvtDriver (페이지 *pagenum*)를 구현.

string ebiodriver::SSSampleDriver::getcompany ()[inline], [virtual]

드라이버 제조사명을 확인한다. 컨버터에서는 제조사명을 로깅용도로만 사용한다.

반환값 :

문자열 형식의 제조사명
stdcvt::CvtDriver (페이지 *pagenum*)를 구현.

CvtDevice* ebiodriver::SSSampleDriver::getdevice (int index)[inline], [virtual]

드라이버가 관리하고 있는 장비의 포인터를 꺼내준다.

매개변수 :

index	연고자 하는 장비의 인덱스 번호. 0 에서 시작한다.
-------	-------------------------------

반환값 :

인덱스에 해당하는 장비의 포인터. NULL 이라면 이후에 장비가 없다는 의미이다.
stdcvt::CvtDriver (페이지 *pagenum*)를 구현.

string ebiodriver::SSSampleDriver::getmodel ()[inline], [virtual]

드라이버 제작자가 부여하는 모델번호를 확인한다.

반환값 :

문자열 형식의 모델번호
stdcvt::CvtDriver (페이지 *pagenum*)를 구현.

string ebiodriver::SSSampleDriver::getversion ()[inline], [virtual]

드라이버 제작자가 부여하는 버전번호를 확인한다.

반환값 :

문자열 형식의 버전번호
stdcvt::CvtDriver (페이지 *pagenum*)를 구현.

bool ebiodriver::SSSampleDriver::initialize (CvtOption option)[inline], [virtual]

드라이버를 초기화 한다. 드라이버 동작을 위한 option 은 key-value 형식으로 전달된다.

매개변수 :

option	드라이버동작을 위한 옵션
--------	---------------

반환값 :

초기화 성공 여부

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::SSSampleDriver::postprocess ()[inline], [virtual]

드라이버간 상태교환이 이루어진 이후에 호출되는 메소드로 후처리를 수행한다.

반환값 :

후처리 성공 여부

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::SSSampleDriver::preprocess ()[inline], [virtual]

드라이버간 상태교환을 하기전에 호출되는 메소드로 전처리를 수행한다.

반환값 :

전처리 성공 여부

stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

bool ebiodriver::SSSampleDriver::sharedevice (CvtDevice * pdevice)[inline], [virtual]

전달된 장비의 정보를 획득한다. 다른 드라이버의 장비정보를 입력해주기 위해 컨버터가 호출한다.

매개변수 :

pdevice	다른 드라이버의 장비 포인터
---------	-----------------

반환값 :

성공여부. 관심이 없는 장비인 경우라도 문제가 없으면 true 를 리턴한다.

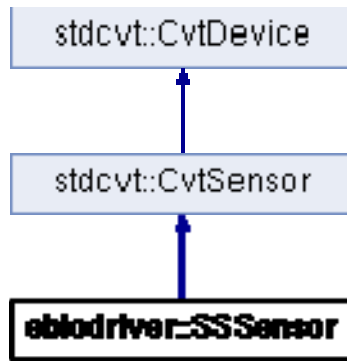
stdcvt::CvtDriver (*페이지 pagenum*)를 구현.

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- sample/sssamplerdriver.cpp

ebiodriver::SSSensor 클래스 참조

ebiodriver::SSSensor 에 대한 상속 다이어그램 :



Public 멤버 함수

- SSSensor (int devid, stdcvt::devtype_t devtype, stdcvt::devsec_t section, stdcvt::devtarget_t target, stdcvt::devstat_t devstatus, stdcvt::obsunit_t unit)

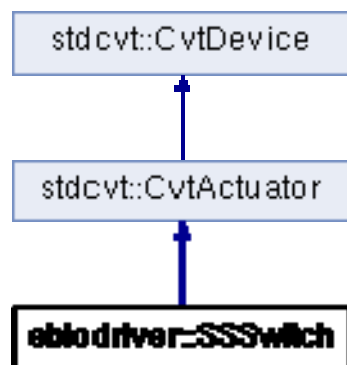
추가로 상속된 멤버들

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- sample/sssamplerdriver.cpp

ebiodriver::SSSwitch 클래스 참조

ebiodriver::SSSwitch 에 대한 상속 다이어그램 :



Public 멤버 함수

- SSSwitch (int devid, stdcvt::devtype_t devtype, stdcvt::devsec_t section, stdcvt::devtarget_t target, stdcvt::devstat_t devstatus)

추가로 상속된 멤버들

이 클래스에 대한 문서화 페이지는 다음의 파일로부터 생성되었습니다.:

- sample/sssamplerdriver.cpp

파일 문서화

sample/dssamplerdriver.cpp 파일 참조

```
#include<iostream>
#include<sstream>
#include<string>
#include<boost/asio.hpp>
#include<boost/bind.hpp>
#include<boost/config.hpp>
#include<glog/logging.h>
#include“../spec/cvtdriver.h“
```

데이터 구조

- class ebiodriver::DSSampleDriver

매크로

- #define BUFSIZE 128
- #define SMARK ``
- #define EMARK '\$'

변수

- BOOST_SYMBOL_EXPORT DSSampleDriver ebiodriver::plugin

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-02-27, JoonYong

작성자 :

Kim, JoonYong tombrad@snu.ac.kr

This file is for sample driver. refer from: <https://github.com/ebio-snu/cvtdriver>

sample/sssamplerdriver.cpp 파일 참조

```
#include<iostream>
#include<sstream>
#include<string>
#include <queue>
#include<boost/config.hpp>
#include<glog/logging.h>
#include<mysql_connection.h>
#include<cppconn/driver.h>
#include<cppconn/exception.h>
#include<cppconn/resultset.h>
#include<cppconn/statement.h>
#include<cppconn/prepared_statement.h>
#include“../spec/cvtdevice.h“
#include“../spec/cvtdriver.h“
```

데이터 구조

- class ebiodriver::SSSensor
- class ebiodriver::SSMotor
- class ebiodriver::SSSwitch
- class ebiodriver::SSSampleDriver

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-03-22, JoonYong

작성자 :

Kim, JoonYong tombrad@snu.ac.kr

This file is for server-side sample driver.refer from: <https://github.com/ebio-snu/cvtdriver>

spec/cvtcode.h 파일 참조

매크로

- #define CVT_OPTION_ASIO_SERVICE “opt_asio_service“
boost::asio::io_service 를 위한 옵션 키
- #define DL_UNKNOWN -2
설치구역 알수 없음
- #define DL_OUTSIDE -1

설치구역 외부

- #define DL_DEFAULT_ROOTZONE 10103010101

디폴트 설치구역 지하부

- #define DL_DEFAULT_PLANTZONE 10103010102

디폴트 설치구역 작물부

- #define DL_DEFAULT_ROOFZONE 10103010103

디폴드 설치구역 작물상부

타입정의

- typedef int64_t stdcvt::devsec_t

열거형 타입

- enum stdcvt::devstat_t { stdcvt::DS_DEV_ABNORMAL = 1, stdcvt::DS_SEN_NORMAL = 101, stdcvt::DS_SWC_ON = 201, stdcvt::DS_SWC_OFF = 202, stdcvt::DS_MOT_OPEN = 301, stdcvt::DS_MOT_CLOSE = 302, stdcvt::DS_MOT_STOP = 303 }

- enum stdcvt::devtarget_t { stdcvt::DO_UNKNOWN = 1, stdcvt::DO_ENV_ATMOSPHERE = 1001, stdcvt::DO_ENV_SOIL = 1002, stdcvt::DO_ENV_NUTRIENTSOLUTION = 1003, stdcvt::DO_PLANT_STEM = 2001, stdcvt::DO_PLANT_LEAF = 2002, stdcvt::DO_PLANT_FRUIT = 2003, stdcvt::DO_PLANT_ROOT = 2004, stdcvt::DO_EQUIPMENT = 3001, stdcvt::DO_EQUIPMENT_NORTH = 3002, stdcvt::DO_EQUIPMENT_SOUTH = 3003 }

enum stdcvt::devgroup_t { stdcvt::DG_UNKNOWN = 0, stdcvt::DG_SENSOR = 1, stdcvt::DG_MOTOR = 2, stdcvt::DG_SWITCH = 3 }

- enum stdcvt::devtype_t { stdcvt::DT_DEV_UNKNOWN = 0, stdcvt::DT_SEN_TEMPERATURE = 10100, stdcvt::DT_SEN_HUMIDITY = 10200, stdcvt::DT_SEN_RADIATION = 10300, stdcvt::DT_SEN_WINDDIRECTION = 10400, stdcvt::DT_SEN_WINDSPEED = 10500, stdcvt::DT_SEN_RAIN = 10600, stdcvt::DT_SEN_RAINFALL = 10700, stdcvt::DT_SEN_PHOTONFLUX = 10800, stdcvt::DT_SEN_SOILMOISTURE = 10900, stdcvt::DT_SEN_SOILMOISTURETENSION = 11000, stdcvt::DT_SEN_EC = 11100, stdcvt::DT_SEN_PH = 11200, stdcvt::DT_SEN_BATTERY = 11300, stdcvt::DT_SEN_ADC = 11400, stdcvt::DT_SEN_UNKNOWN = 19000, stdcvt::DT_MOT_SIDEWINDOW = 20100, stdcvt::DT_MOT_SIDEWINDOW_1 = 20101, stdcvt::DT_MOT_SIDEWINDOW_2 = 20102, stdcvt::DT_MOT_SIDEWINDOW_3 = 20103, stdcvt::DT_MOT_SIDEWINDOW_4 = 20104, stdcvt::DT_MOT_SIDEWINDOW_5 = 20105, stdcvt::DT_MOT_ROOFWINDOW = 20200, stdcvt::DT_MOT_ROOFWINDOW_1 = 20201, stdcvt::DT_MOT_ROOFWINDOW_2 = 20202, stdcvt::DT_MOT_ROOFWINDOW_3 = 20203, stdcvt::DT_MOT_ROOFWINDOW_4 = 20204, stdcvt::DT_MOT_ROOFWINDOW_5 = 20205, stdcvt::DT_MOT_ROOFCURTAIN

= 20400, stdcvt::DT_MOT_SIDE CURTAIN = 20500, stdcvt::DT_MOT_SHADING CURTAIN = 20600, stdcvt::DT_MOT_UNKNOWN = 29000, stdcvt::DT_SWC_FAN = 30100, stdcvt::DT_SWC_FLOWFAN = 30101, stdcvt::DT_SWC_VENTFAN = 30102, stdcvt::DT_SWC_UNKNOWN = 39000 }

- enum stdcvt::obsunit_t { stdcvt::OU_NONE = 1, stdcvt::OU_CELSIUS = 2, stdcvt::OU_PERCENT = 3, stdcvt::OU_PPM = 4, stdcvt::OU_W_PER_MSQ = 5, stdcvt::OU_DEGREE = 6, stdcvt::OU_M_PER_SEC = 7, stdcvt::OU_MMOL_PER_MSQSEC = 8, stdcvt::OU_KPA = 9, stdcvt::OU_DS_PER_M = 10, stdcvt::OU_PH = 11 }

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-02-24, JoonYong

작성자 :

Kim, JoonYong tombrad@snu.ac.kr

This file has various common codes. refer from: <https://github.com/ebio-snu/stdcvt>

타입정의 문서화

typedef int64_t stdcvt::devsec_t

장비 설치 구역 타입

열거형 타입 문서화

enum stdcvt::devgroup_t

장비 그룹 타입

열거형 멤버

DG_UNKNOWN 알수없는 장비그룹

DG_SENSOR 센서 그룹

DG_MOTOR 모터형 구동기 그룹

DG_SWITCH 스위치형 구동기 그룹

enum stdcvt::devstat_t

장비 상태 타입

열거형 멤버

DS_DEV_ABNORMAL 비정상
DS_SEN_NORMAL 센서 정상 상태
DS_SWC_ON 스위치 켜짐 (정상)
DS_SWC_OFF 스위치 꺼짐 (정상)
DS_MOT_OPEN 모터 열리는 중 (정상)
DS_MOT_CLOSE 모터 닫히는 중 (정상)
DS_MOT_STOP 모터 중지상태 (정상)

enum stdcvt::devtarget_t

장비 작동대상 타입

열거형 멤버

DO_UNKNOWN 대상을 알지 못함
DO_ENV_ATMOSPHERE 대기를 대상으로 함
DO_ENV_SOIL 토양을 대상으로 함
DO_ENV_NUTRIENTSOLUTION 양액을 대상으로 함
DO_PLANT_STEM 작물의 줄기를 대상으로 함
DO_PLANT_LEAF 작물의 잎을 대상으로 함
DO_PLANT_FRUIT 작물의 과실을 대상으로 함
DO_PLANT_ROOT 작물의 뿌리를 대상으로 함
DO_EQUIPMENT 농장의 장비를 대상으로 함
DO_EQUIPMENT_NORTH 농장의 장비가 좌우대칭인 경우 북쪽에 가까운 장비를 대상으로 함
DO_EQUIPMENT_SOUTH 농장의 장비가 좌우대칭인 경우 남쪽에 가까운 장비를 대상으로 함

enum stdcvt::devtype_t

장비 종류 타입

열거형 멤버

DT_DEV_UNKNOWN 알수없는 장비
DT_SEN_TEMPERATURE 온도센서
DT_SEN_HUMIDITY 습도센서
DT_SEN_RADIATION 일사센서
DT_SEN_WINDDIRECTION 풍향센서
DT_SEN_WINDSPEED 풍속센서
DT_SEN_RAIN 감우센서
DT_SEN_RAINFALL 강우량센서
DT_SEN_PHOTONFLUX PAR 센서

DT_SEN_SOILMOISTURE 토양함수율센서
 DT_SEN_SOILMOISTURETENSION 토양수분장력센서
 DT_SEN_EC EC 센서
 DT_SEN_PH pH 센서
 DT_SEN_BATTERY 배터리센서
 DT_SEN_ADC ADC 변환된 Raw 값 센서 (?)
 DT_SEN_UNKNOWN 알수없는 센서
 DT_MOT_SIDEWINDOW 측창 구동기
 DT_MOT_SIDEWINDOW_1 1 중 측창 구동기
 DT_MOT_SIDEWINDOW_2 2 중 측창 구동기
 DT_MOT_SIDEWINDOW_3 3 중 측창 구동기
 DT_MOT_SIDEWINDOW_4 4 중 측창 구동기
 DT_MOT_SIDEWINDOW_5 5 중 측창 구동기
 DT_MOT_ROOFWINDOW 천창 구동기
 DT_MOT_ROOFWINDOW_1 1 중 천창 구동기
 DT_MOT_ROOFWINDOW_2 2 중 천창 구동기
 DT_MOT_ROOFWINDOW_3 3 중 천창 구동기
 DT_MOT_ROOFWINDOW_4 4 중 천창 구동기
 DT_MOT_ROOFWINDOW_5 5 중 천창 구동기
 DT_MOT_ROOFCURTAIN 천장보온커튼 구동기
 DT_MOT_SIDECURTAIN 측면보온커튼 구동기
 DT_MOT_SHADINGCURTAIN 차광커튼 구동기
 DT_MOT_UNKNOWN 알수없는 모터형 구동기
 DT_SWC_FAN 팬
 DT_SWC_FLOWFAN 유동팬
 DT_SWC_VENTFAN 환기팬
 DT_SWC_UNKNOWN 알수없는 스위치형 구동기

enum stdcvt::obsunit_t

센서 관측치 단위 타입

열거형 멤버

OU_NONE 단위가 없는 무차원 값

OU_CELSIUS °C

OU_PERCENT %

OU_PPM ppm

OU_W_PER_MSQ W/m²

OU_DEGREE °

OU_M_PER_SEC m/s

OU_MMOL_PER_MSQSEC μ mol/m²/s.

OU_KPA kPa

OU_DS_PER_M dS/m

OU_PH pH

spec/cvtcodedic.h 파일 참조

변수

- map< string, int64_t >stdcvt::_codedic
-

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-03-20, JoonYong

작성자 :

Kim, JoonYong tombrad@snu.ac.kr

This file is for code dictionary. referfrom: <https://github.com/ebio-snu/stdcvt>

spec/cvtcommand.h 파일 참조

#include "cvtdevicespec.h"

데이터 구조

- class stdcvt::CvtCommand
 - class stdcvt::CvtRatioCommand
 - class stdcvt::CvtRawCommand
-

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-02-24, JoonYong

작성자 :

Kim, JoonYong tombrad@snu.ac.kr

This file is a template of CvtCommand. referfrom: <https://github.com/ebio-snu/stdcvt>

spec/cvtdevice.h 파일 참조

```
#include<iostream>
#include<string>
#include<glog/logging.h>
#include“cvtcode.h“
#include“cvtdevicespec.h“
#include“cvtcommand.h“
```

데이터 구조

- class stdcvt::CvtDevice
- class stdcvt::CvtSensor
- class stdcvt::CvtActuator
- class stdcvt::CvtMotor

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-02-24, JoonYong

작성자 :

Kim, JoonYong tombrad@snu.ac.kr

This file is template of CvtDevice. referfrom: <https://github.com/ebio-snu/stdcvt>

spec/cvtdevicefactory.h 파일 참조

```
#include<jsoncons/json.hpp>
#include<jsoncons_ext/jsonpath/json_query.hpp>
#include“cvtdevice.h“
#include“cvtcodedic.h“
```

데이터 구조

- class stdcvt::CvtDeviceFactory

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-03-20, JoonYong

작성자 :

Kim, JoonYong tombraid@snu.ac.kr

This file is for CvtDeviceFactory referfrom: <https://github.com/ebio-snu/stdcvt>

spec/cvtdevicespec.h 파일 참조

```
#include<iostream>
```

```
#include<string>
```

```
#include<glog/logging.h>
```

```
#include“cvtcode.h“
```

데이타 구조

- class stdcvt::CvtDeviceSpec

매크로

- #define _DS_MAX 6

열거형 타입

- enum _dsiv_t { _DS_TX = 0, _DS_TY = 1, _DS_TZ = 2, _DS_PX = 3, _DS_PY = 4, _DS_PZ = 5 }

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-02-24, JoonYong

작성자 :

Kim, JoonYong tombraid@snu.ac.kr

This file is template of CvtDeviceSpec.refer from: <https://github.com/ebio-snu/stdcvt>

spec/cvtdriver.h 파일 참조

```
#include<ctime>
```

```
#include “cvtdevice.h“
```

```
#include“cvtoption.h“
```

데이타 구조

- class stdcvt::CvtDriver
-

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-02-24, JoonYong

작성자 :

Kim, JoonYong tombruid@snu.ac.kr

This file is template of CvtDriver. referfrom: <https://github.com/ebio-snu/stdcvt>

spec/cvtoption.h 파일 참조

```
#include<jsoncons/json.hpp>
```

```
#include<jsoncons_ext/jsonpath/json_query.hpp>
```

```
#include“cvtdevice.h“
```

```
#include“cvtdevicefactory.h“
```

데이터 구조

- class stdcvt::CvtOption

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :

2018-03-03, JoonYong

작성자 :

Kim, JoonYong tombruid@snu.ac.kr

This file is for CvtOption. refer from: <https://github.com/ebio-snu/stdcvt>

spec/cvtspec.h 파일 참조

```
#include<iostream>
```

```
#include<string>
```

```
#include<glog/logging.h>
```

```
#include“cvtcode.h“
```

```
#include“cvtoption.h“
```

데이터 구조

- class stdcvt::CvtSpec
-

상세한 설명

Copyright 2018 ebio lab. SNU. All Rights Reserved.

날짜 :


2018-03-05, JoonYong

작성자 :

Kim, JoonYong tombruid@snu.ac.kr

This file is template of CvtSpec. referfrom: <https://github.com/ebio-snu/stdcvt>

(2) 스마트팜 장비 연동용 표준 컨버터 개발(기술이전)

<p>표준화 컨버터</p>	<p>○ 스마트팜 기기 연동을 위한 표준 컨버터 개발 기술 - 개발자 매뉴얼 및 기술개발문서를 제작하여 업체들에게 기술이전 실시</p>
<p>스마트팜 기기 연동을 위한 표준 컨버터 개발 기술</p>	
<div style="text-align: right; margin-bottom: 20px;"> <p>기술이전문서 Ver. 1.0</p> </div> <div style="text-align: center; margin-bottom: 20px;">  </div> <div style="text-align: center;"> <p>Copyright © KAICAF</p> <p>KAICAF 의 사전 승인 없이 본 내용의 전부 또는 일부에 대한 복사, 전재, 배포, 사용을 금합니다.</p> </div>	

개 정 이 력

버전	변경일	변경사유	변경내용	작성자	승인자
1.0	2019.08.19	최초제정			

1. 개요

한국 농림수산물식품부에서는 스마트팜 기자재 표준화 및 검정제 도입을 준비하고 있다. 현 시점에서 국내에서 개발되어 판매되는 스마트팜 기기들의 표준화는 아직 요원한 상황이다. 이에 (사) 한국농식품 ICT 융복합 산업 협회에서는 호준화 장비를 중심으로한 주요 스마트팜 기자재의 성능 개선 지원 및 현장생산관리 정보수집 및 활용을 위해 본 프로젝트를 진행하고 있다.

본 프로젝트는 다양한 스마트팜 기기들을 함께 운용하기 위한 드라이버를 설계 / 개발하는 것을 1차 목적으로 한다. 부차적으로 해당 드라이버를 실행하기 위한 컨버터, 드라이버를 테스트하기 위한 테스트 UI, 컨버터로부터 데이터를 수집할 수 있는 데이터 수집기를 개발한다.

전체 시스템 구성은 아래와 같다. 협회에서 드라이버 제작을 위한 API 스펙을 공개하면 개별 업체에서 자사의 기기와 통신할 수 있는 공유 라이브러리 형식의 드라이버를 개발하여 제공한다. 컨버터는 제공된 드라이버를 활용하여 서로 다른 업체간 통신을 돕고, 나아가 협회 서비스로 데이터를 전송하는 기능을 수행하게 된다.

1.1 드라이버 개발 방식의 장점

: 드라이버 방식으로 개발하는 경우 다음과 같은 장점이 있다.

- 1) 참여업체는 내부 프로토콜 (혹은 회사 기밀)을 공개할 필요가 없다.
- 2) 참여업체는 새로운 표준을 직접 구현하지 않기 때문에 작업시간을 단축할 수 있다.
- 3) 참여업체는 자사의 장비에 컨버터를 설치하지 않기 때문에 안정성문제를 고민할 필요가 없다.
- 4) 참여업체는 자사의 개발코드를 공개할 필요가 없다 . 드라이버만 바이너리 형식으로 제공하면 된다.

이런 장점을 가진 드라이버 개발 방식을 통해 참여업체의 부담을 최소화 하고 스마트팜 기기간의 연동을 이끌어 낼 수 있다.

1.2 개발관련 공통사항

- 라이브러리
 - google glog : Google 에서 공개한 로깅라이브러리
 - jsoncons : JSON 라이브러리
 - Boost : asio 와 dll 을 주로 사용
 - 개별 드라이버 개발을 위해 필요한 라이브러리를 사용할 수 있다 .
- 개발환경
 - g++ : 기본 설정으로 g++을 사용 . VC 에서 테스트되지는 않았지만 큰 문제는 없을것으로 기대
 - cmake : 빌드환경
 - nodejs : Test UI 를 위한 환경
 - mysql-5.5 : (사용여부 불확실) 데이터저장소를 위한 용도로 활용 예정
- 개발문서
 - doxygen : 코드 문서화를 위해 활용

2. 스마트팜 기기 연동을 위한 컨버터

스마트팜 기기연동을 위한 컨버터는 최소 2 개의 드라이버를 활용하여 서로 다른 네트워크를 연결하는 게이트웨이 역할을 수행하는 장치이다. 드라이버는 센서노드, 제어노드, 컨트롤러와 직접 연결되는 장비측(DeviceSide) 드라이버와 데이터 수집기와 연결되는 서버측(Servier Side) 드라이버로

구성된다. 각 드라이버를 DSDriver, SSDriver 라고 한다.

DS 드라이버와 SS 드라이버는 동일한 드라이버 API 에 의해서 연동되며 , 컨버터는 두 드라이버가 가지고 있는 장비의 상태를 교환하는 방식으로 작동한다 .

2.1. 장비의 개념

: 장비는 다음과 같이 구분된다 .

- 센서 : 하나의 관측치를 제공하는 장비. cf) 온습도 센서는 온도 센서, 습도 센서로 구분되어야 한다.
 - 구분될만한 센서가 있다면 추후 하위 카테고리가 생길 수 있음.
- 구동기 : 명령에 의해 어떤 동작을 수행하는 장비 .
 - 모터형 구동기 : 열림 /닫힘의 방향을 가지고 구동될 수 있는 구동기. % 단위의 제어를 수행함.
 - 스위치형 구동기 : 켜거나 끌 수 있는 구동기.
 - 추후에 양액기 등이 추가될 수 있음.

실제 구성에서는 센서노드, 제어노드, 컨트롤러 등이 DS 드라이버와 연결되어 데이터 교환을 하게 되는데, 컨버터는 이를 추상화하여 노드나 컨트롤러의 존재 여부에는 관심이 없다.

2.2. 컨버터의 작동시퀀스

: 컨버터는 SS 드라이버와 DS 드라이버 사이의 상태 교환을 수행하는데, 개별 드라이버의 구동을 최대한 방해하지 않는 방향으로 작동한다. 컨버터의 작동 프로세스는 다음과 같다.

컨버터는 실행되면 설정파일에 기반해서 SS 드라이버와 DS 드라이버를 생성하고, 각각에 부여된 옵션 (설정)정보를 initialize 메소드에 전달한다. 반대로 종료시에는 finalize 메소드를 호출한다.

실제 연동작업은 프로세싱 과정을 통해서 이루어진다. 다양한 DS 드라이버 구현이 있을 수 있기때문에 프로세싱과정도 다양하게 나타날 수 있다. 컨버터는 비효율적인 방법이지만 일관성 있는 방법으로 이를 처리하고자 한다.

- 1) 컨버터는 DS 드라이버와 SS 드라이버의 상태에 전혀 관여하지 않고, 상태의 교환만을 수행한다.
- 2) DS 드라이버와 실장비간의 싱크는 DS 드라이버 내에서 모두 처리한다.
- 3) SS 드라이버와 데이터 수집 기간의 싱크는 SS 드라이버 내에서 모두 처리한다.
- 4) 상태의 교환이 이루어지기 전 preprocess 메소드가 호출되고, 상태의 교환이 완료된 이후에 postprocess 메소드가 호출된다.

5) 컨버터에서는 비동기 IO를 지원하기 위해 Boost::asio 를 활용한다.

6) (중요) Boost::asio 를 활용하지 않아도 되지만, 드라이버에서 대기모드로 진입해서는 안된다. (sleep, select 등)

7) (논의) 개별 드라이버가 내부적으로 별도의 스레드를 돌리거나 통신을 수행하는 별도의 프로세스와 내부통신 (공유메모리 등)을 수행한다면 해당 스레드나 프로세스에서 대기모드 진입은 가능하다.

2.3. 컨버터의 설치

: 컨버터는 최대한 특정 OS 에 종속되지 않도록 개발하려고 한다. 다만, 현 상태에서 사용성, 개별편의 등을 고려하여 라즈베리파이3를 기본 하드웨어로 하고 Raspbian Stretch를 기본 OS로 하여 개발을 진행하고 있다. 컨버터의 설치를 위해 라즈베리파이3 및 Raspbian은 설치가 되어 있다고 가정한다.

- 초기 패키지 업데이트
 - sudo apt update

- sudo apt upgrade
- 필요한 패키지의 설치
 - 소스 코드의 획득
 - sudo apt install git
 - git clone https://github.com/ebio-snu/stdcvr.git
 - cd stdcvr
 - git submodule init
 - git submodule update
 - Test UI 를 위한 NodeJS
 - curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
 - sudo apt-get install -y nodejs
 - 빌드를 위한 패키지 설치
 - sudo apt install build-essential cmake libgoogle-glog-dev libboost-all-dev
 - 소스코드 업데이트 및 빌드
 - cd stdcvr
 - git pull
 - git submodule update
 - mkdir build
 - cd testui
 - npm install
 - cd ../build
 - cmake ..
 - make

2.4. 컨버터의 설정

: 컨버터는 json 파일로 된 설정 파일에 의해서 설정이 이루어진다. 설정 파일은 ssdriver와 dsdriver로 나뉘어지는데 기본적인 구조는 동일하다.

```
{
  "ssdriver": [{
    "driver": "libsssample.so",
    "option": {
      "value": "value.json",
      "command": "command.json"
    }
  }],
  "dsdriver": [{
    "driver": "libdsssample.so",
    "option": {
      "port": "/dev/ttyUSB0",
      "baudrate": 115200
    }
  }]
```



```

    }
}

```

하나의 드라이버에 대한 설정은 드라이버 파일명과 해당 드라이버 구동을 위한 옵션으로 구성된다. 드라이버 파일명은 추후 협회의 시스템이 구축될 때 내부적인 규칙에 따라 정리될 예정이다. 드라이버의 옵션은 해당 드라이버에 맞게 개발자가 설정하면 된다.

3. 컨버터 사용법

3.1. 컨버터 설치

:라즈베리파이를 기본으로 하여 설치하는 방법을 설명한다. 설치 절차는 다음과 같다.

- 1) raspbian 최신 버전을 설치한다. 데스크탑 버전도 상관없으나 LITE 버전을 추천한다.
- 2) 설치 후 패스워드 변경, SSH 서버 구동, 로케일 설정, 타임존 설정 등을 수행한다. 이와 관련해서는 많은 자료가 인터넷 상에 있기때문에 따로 다루지 않는다.
- 3) 다음의 명령을 실행하여 필요한 패키지를 설치한다.

```

- sudo apt update
- sudo apt upgrade -y
- curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
- sudo apt install -y git buildessential cmake libgoogle-glog-dev libboost-all-dev nodejs
libmysqlcppconn-dev mysql-server

```

- 4) 다음의 명령으로 컨버터 소스를 받는다.

```

- git clone https://github.com/ebio-snu/stdcvt.git
- cd stdcvt
- git submodule init
- git submodule update

```

- 5) 다음의 명령으로 샘플 드라이버를 컴파일 한다.

```

- cd cvtdriver
- mkdir build
- cd build
- cmake ..
- make

```

- 6) make test 명령을 통해 샘플 드라이버를 테스트 할 수 있다 .

- 7) 다음의 명령으로 컨버터를 컴파일 한다 .

```

- cd ../..
- mkdir build
- cd build
- cmake ..
- make

```

3.2. 컨버터의 실행

: 실제 컨버터는 데몬으로 동작하게 되겠지만 드라이버 개발을 위해서 사용하는 경우에는 자주 반복해서 실행해야한다. 이때 다음의 명령으로 실행한다.

```

- GLOG_logtostderr=1 ./stdcvt

```

3.3. 테스트용 UI 사용법

: 테스트용 UI 는 nodejs 를 기반으로 작동하며 외부에서 웹 브라우저를 통해서 사용할 수 있도록 되어있다 . 동작에 필요한 패키지를 설치하기 위해서 testui 폴더에서 다음의 명령을 실행한다 .

- npm install
- sudo npm install -g cross-env

테스트용 UI를 실행하기 위해 다음의 명령을 실행한다.

- npm run start

테스트용 8880 포트로 접근하면 다음의 화면을 볼 수 있다. 화면은 driver 와 device로 구분되는데, driver에서는 드라이버 설정을 확인할 수 있고, device에서는 DSDriver 에서 제공하는 장비 정보를 확인하거나 명령을 전송할 수 있다.

장비 정보는 센서, 개폐기, 스위치로 구성되어 있다. 개폐기와 스위치에는 명령을 전달할 수 있는데, 스위치는 단순히 작동/정지를 할 수 있고, 개폐기의 경우 개폐율 (소수점 값)을 입력하여 작동할 수 있다. 예를 들어 10%를 열고자 하면 0.1을 입력하고 작동을 누르면 된다.

4. 샘플 노드

: 샘플노드는 아두이노 나노, LCD, 온습도센서로 구성된다.

4.1. 샘플노드 기능

: 다음과 같은 기능을 가지고 있다.

- 1) 컨버터와 USB serial 통신을 한다.
- 2) 습도 센서 , 온도 센서, 그리고 모터와 스위치가 각 두개씩 있다.
- 3) 모터와 스위치는 가상으로 동작한다.
- 4) 습도 센서에서 습도값을 읽어서 전송한다.
- 5) 온도 센서에서 온도값을 읽어서 전송한다.
- 6) 모터는 약 1초에 1%씩 움직인다.

4.2. 샘플노드 디스플레이

: 샘플노드 LCD 는 2 줄로 구성되어 있다. 첫번째 줄에는 습도값, 온도값, 스위치 두개의 상태가 표시된다. 스위치의 상태는 T/F로 표시된다. 두번째 줄에는 두개의 모터 상태가 표시된다. 현재 위치 (%), 작동상태(O/C/S)로 표시된다. O : Opening, C : Closing, S : Stopping

4.3. 샘플노드 프로토콜

: 샘플노드를 만들기위해 간단한 프로토콜을 설계하였다.

- 7) 프로토콜은 ^로 시작해서 \$로 끝난다.
- 8) s, m, w 중 하나로 시작한다. s는 센서, m은 모터, w는 스위치를 의미한다.
- 9) 각각의 값은 스페이스로 구분된다.
- 10) node -> downdriver로 전송되는 메시지는 다음 중 하나이다. devid 는 장비아이디, obs는 관측치, status는 장비상태, cmdid는 명령의 아이디, cur의 모터의 현재 위치, target은 모터의 목표 위치를 의미한다.
 - ^s devid obs status\$
 - ^m devid cmdid cur target status\$
 - ^w devid cmdid status\$

11) downdriver->node 로 전송되는 메시지는 다음중 하나이다 .


- ^m devid cmdid target\$
- ^w devid cmdid status\$

12) 메시지를 받으면 확인을 했다는 의미로 ^\$를 전송한다 .

4.3.1 예시

- 센서 메시지 예시 : 10번 센서의 값이 19.6 이고, 정상이다.
 - ^s 10 19.6 1\$
- 모터 메시지 예시 : 20번 모터의 현재 포지션은 10%이고, 20%를 향해서 열리고 있다. 마지막 명령 아이디는 3이다.
 - ^m 20 3 10 20 1\$
- 스위치 메시지 예시 : 30번 스위치는 꺼져있다. 마지막 명령 아이디는 2이다.
 - ^w 30 2 0\$
- 모터명령 예시 : 20번 모터의 작동을 중지해라. 명령 아이디는 1이다.
 - ^m 20 1 -1\$
- 모터명령 예시 : 20 번 모터를 10%만큼 열어라 . 명령아이디는 1 이다 .
 - ^m 20 1 10\$
- 스위치명령 예시 : 30 번 스위치를 켜라 . 명령아이디는 2 이다 .
 - ^w 30 2 1\$

(3) 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발

<p>표준화 컨버터</p>	<p>○ 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발 - 개발자 매뉴얼 및 기술개발문서를 제작하여 업체들에게 기술이전 실시</p>
<p>표준 컨버터용 디바이스 드라이버 개발기술</p>	
<div style="text-align: right;"> <p>기술이전문서 Ver. 1.0</p> </div> <div style="text-align: center; margin-top: 200px;">  </div> <div style="text-align: center; margin-top: 100px;"> <p>Copyright © KAICAF</p> <p>KAICAF 의 사전 승인 없이 본 내용의 전부 또는 일부에 대한 복사, 전재, 배포, 사용을 금합니다.</p> </div>	

개정이력

버전	변경일	변경사유	변경내용	작성자	승인자
1.0	2019.08.19	최초제정			

1. 개요

: 드라이버는 크게 2종류로 구분된다. 센서노드, 제어노드, 컨트롤러와 직접 연결되는 장비측(Device Side) 드라이버와 데이터 수집기와 연결되는 서버측(Servier Side)드라이버로 구성된다. 각 드라이버를 DSDriver, SSDriver라고 한다.

드라이버 API는 기초연동용과 고급연동용으로 구분될 계획이다. 현 단계에서 드라이버는 현 상황을 충실히 포함하는 쪽으로 설계하는 것을 목표로 하고 있다. 추후에 활성화가 되어 추상화 레벨을 높여서 더 많은 범위를 커버할 수 있기를 바란다.

4-1. 드라이버 스펙 문서

:현 시점에서 기초 연동을 위한 API 개발이 완료되었다. 추후 참여사의 개발지원 및 미팅을 통해 API가 업데이트 될 수 있다.

4-2. DeviceID, Device Spec, Device Index

: 컨버터와 드라이버의 이해를 위해서 ID, Spec, Index를 구분하는 것이 필요하다.
 Device ID는 개발사에서 관리하는 ID이며, 다양한 방식의 아이디를 고려하여 문자열 형이다. 컨버터에서는 ID를 기반한 작업이 없지만, CvtRawCommand를 사용해서 특정 ID를 가진 장비에게 명령을 전송할 수는 있다.
 컨버터에서는 Index와 Spec에는 관심이 많다. Index는 컨버터에서 ID와 같이 사용되는 값이다. 드라이버는 getdevice 메소드로 장비를 요청할 때 동일한 index에 대해서 동일한 장비를 리턴해야 한다.
 DeviceSpec은 서로 다른 드라이버 사이에서 정보를 교환할 때 기준이 되는 값이다. 서로 다른 드라이버는 서로 다른 개발사에서 개발이 되기 때문에 서로 ID를 공유할 수 없을 뿐만 아니라 해당 장비의 종류, 설치위치 등을 알 수 없다. 각 개발사에서는 해당 장비의 Spec을 정확하게 기록해야 한다. 특정 장비의 Spec을 공통 코드로 표현할 수 없을 때에는 즉각 이를 알려서 공통 코드를 추가하거나 변경할 수 있도록 하여야 한다. 경우에 따라 공통 코드의 대대적인 변경이 필요할 경우에는 UNKNOWN을 활용하도록 한다.

1.3. 드라이버가 가진 장비 데이터의 공유

: 2개 이상의 DSDriver를 사용하는 경우 개별 DSDriver는 서로 다른 장비를 관리해야한다. 하나의 DSDriver에서 관리되는 장비에 대한 정보는 다른 장비로 전달될 수 있다.
 드라이버로부터 장비 정보를 획득하기 위해서는 CvtDevice*getdevice(int index); 메소드를 활용한다. index는 0부터 시작하고, 값을 하나씩 올라가면서 장비를 꺼낼 수 있다. 이때 리턴값이 nullptr(NULL)인 경우 관리하는 장비가 더 이상 없다는 의미이다.
 getdevice 메소드를 통해서 얻은 포인터는 DSDriver 내부에서 관리하고 있는 장비에 대한 것이기 때문에 외부에서 delete를 수행하지 않는다.
 장비의 정보를 다른 드라이버로 전달하기 위해서 boolsharedevice (CvtDevice *pdevice); 메소드를 활용한다. sharedevice 메소드 안에서는 pdevice로부터 정보만을 추출하거나, pdevice->clone()을 이용해서 사본을 만들어서 활용한다.

1.4. 다른 드라이버로 제어명령 전달

: 장비에 대한 명령도 다른 드라이버로 전달될 수 있다. 다만 초기 버전에서는 SSDriver가 하나만 존재하고, SSDriver에서만 명령을 전달할 수 있는 것으로 한다.
 하나의 드라이버(SSDriver)가 전달하고자 하는 명령은 CvtCommand *getcommand(); 메소드를 이용해 얻을 수 있다. 반복적으로 호출하여 명령을 꺼낼 수 있다. 이때 리턴값이 nullptr(NULL)인 경우 더 이상의 명령이 없다는 의미이다.
 획득한 명령을 다른 드라이버로 전달하기 위해서 boolcontrol(CvtCommand *pcmd); 메소드를 활용한다.

control 메소드에 의해서 전달받은 pcmd는 명령 정보만을 추출해 사용한다.

2. 드라이버 개발을 위한 코드테이블

: 스마트팜 기기의 원활한 연동을 위해서 별도의 문서를 통해 공통 코드를 공유한다. 해당 문서는 지속적으로 업데이트 될 예정이며, 하위 호환성을 최대한 유지할 계획이다. 공통코드는 장비의 상태, 장비의 종류, 장비의 설치위치, 관측치의 단위 등이다.

2.1. 장비 상태

: 장비상태는 devstat_t 로 표현된다.

이름	번호	대상장비	설명
DS_DEV_ABNORMAL	1	모든장비	장비가 정상적으로 동작하지 않음. 연결이 끊어지는 경우 포함.
DS_SEN_NORMAL	101	센서	센서가 정상적으로 동작함.
DS_SWC_ON	201	스위치	스위치가 작동중임.
DS_SWC_OFF	202	스위치	스위치가 중지 상태임.
DS_MOT_OPEN	301	모터	모터가 열리는 방향으로 작동중임.
DS_MOT_CLOSE	302	모터	모터가 닫히는 방향으로 작동중임.
DS_MOT_STOP	303	모터	모터가 중지 상태임.

```
typedef enum {
    DS_DEV_ABNORMAL = 1,
    DS_SEN_NORMAL = 101,
    DS_SWC_ON = 201,
    DS_SWC_OFF = 202,
    DS_MOT_OPEN = 301,
    DS_MOT_CLOSE = 302,
    DS_MOT_STOP = 303
} devstat_t;
```

2.2. 장비 종류

: 장비 종류는 devtype_t로 표현된다. 장비 종류는 크게 5자리 unsigned int로 구분되는데, 장비의 분류(1자리), 장비 종류(2자리), 장비 하위 구분(2자리)로 구성된다. 장비 하위 구분은 구동기의 경우에 많이 보이는데, 온실의 측창이나 천창이 다중창인 경우가 대표적인 예라고 할 수 있다.

작명법은 센서의 경우 DT_SEN으로 시작하고, 모터형 구동기의 경우 DT_MOT로, 스위치형 구동기의 경우 DT_SWC 로 시작한다 . 이후의 명칭은 장비가 작동하는 대상이 된다. 온도센서의 경우 DT_SEN_TEMPERATURE, 측창의 경우 DT_MOT_SIDEWINDOW 가 된다.

아래의 센서 종류는 TTAK.KO-10.0903을 참고하여 작성되었다.

이름	분류 번호	번호	분류	장비	설명
DT_DEV_UNKNOWN	0	0	기타	알수없는	코드에 없는 장비

			장비	장비	인 경우.
DT_SEN_TEMPERATURE	1	10100	센서	온도센서	온도를 측정하는 센서.
DT_SEN_HUMIDITY	1	10200	센서	습도센서	습도를 측정하는 센서.
DT_SEN_RADIATION	1	10300	센서	일사센서	일사량을 측정하는 센서.
DT_SEN_WINDDIRECTION	1	10400	센서	풍향센서	풍향을 측정하는 센서.
DT_SEN_WINDSPEED	1	10500	센서	풍속센서	풍속을 측정하는 센서.
DT_SEN_RAIN	1	10600	센서	강우센서	비가 오는지 측정하는 센서.
DT_SEN_RAINFALL	1	10700	센서	강우량센서	강우량을 측정하는 센서.
DT_SEN_PHOTONFLUX	1	10800	센서	PAR 센서	유효광량을 측정하는 센서.
DT_SEN_SOILMOISTURE	1	10900	센서	토양함수율센서	토양 함수율을 측정하는 센서.
DT_SEN_SOILMOISTURETENSION	1	11000	센서	토양수분장력센서	토양수분장력을 측정하는 센서.
DT_SEN_EC	1	11100	센서	EC 센서	EC 를 측정하는 센서.
DT_SEN_PH	1	11200	센서	pH 센서	pH 를 측정하는 센서.
DT_SEN_BATTERY	1	11300	센서	배터리센서	배터리의 상태를 확인하는 센서.
DT_SEN_ADC	1	11400	센서	ADC	센서는 아니지만 아날로그값을 디지털로 전환한 상태의 값을 전달.
DT_SEN_UNKNOWN	1	19000	센서	알수없는 센서	코드에 없는 센서인 경우.
DT_MOT_SIDEWINDOW	2	20100	모터형 구동기	측창 구동기	측창을 구동하는 모터형 구동기.
DT_MOT_SIDEWINDOW_1	2	20101	모터형 구동기	1 중 측창 구동기	측창을 구동하는 모터형 구동기.
DT_MOT_SIDEWINDOW_2	2	20102	모터형 구동기	2 중 측창 구동기	측창을 구동하는 모터형 구동기.
DT_MOT_SIDEWINDOW_3	2	20103	모터형 구동기	3 중 측창 구동기	측창을 구동하는 모터형 구동기.
DT_MOT_SIDEWINDOW_4	2	20104	모터형	4 중 측창	측창을 구동하는

			구동기	구동기	모터형 구동기.
DT_MOT_SIDEWINDOW_5	2	20105	모터형 구동기	5 중 측창 구동기	측창을 구동하는 모터형 구동기.
DT_MOT_ROOFWINDOW	2	20200	모터형 구동기	천창 구동기	천창을 구동하는 모터형 구동기.
DT_MOT_ROOFWINDOW_1	2	20201	모터형 구동기	1 중 천창 구동기	천창을 구동하는 모터형 구동기.
DT_MOT_ROOFWINDOW_2	2	20202	모터형 구동기	2 중 천창 구동기	천창을 구동하는 모터형 구동기.
DT_MOT_ROOFWINDOW_3	2	20203	모터형 구동기	3 중 천창 구동기	천창을 구동하는 모터형 구동기.
DT_MOT_ROOFWINDOW_4	2	20204	모터형 구동기	4 중 천창 구동기	천창을 구동하는 모터형 구동기.
DT_MOT_ROOFWINDOW_5	2	20205	모터형 구동기	5 중 천창 구동기	천창을 구동하는 모터형 구동기.
DT_MOT_ROOFCURTAIN	2	20400	모터형 구동기	천장보온 커튼 구동기	천장보온커튼을 구동하는 모터형 구동기.
DT_MOT_SIDECURTAIN	2	20500	모터형 구동기	측면보온 커튼 구동기	측면보온커튼을 구동하는 모터형 구동기.
DT_MOT_SHADINGCURTAIN	2	20600	모터형 구동기	차광커튼 구동기	차광커튼을 구동하는 모터형 구동기.
DT_MOT_UNKNOWN	2	29000	모터형 구동기	알수없는 모터형 구동기	코드에 없는 모터형 구동기인 경우.
DT_SWC_FAN	3	30100	스위치형 구동기	팬	팬
DT_SWC_FLOWFAN	3	30101	스위치형 구동기	유동팬	내부 공기 유동을 위한 팬
DT_SWC_VENTFAN	3	30102	스위치형 구동기	환기팬	환기를 위한 팬
DT_SWC_UNKNOWN	3	39000	스위치형 구동기	알수없는 스위치형 구동기	코드에 없는 스위치형 구동기인 경우.

알수없는 센서, 구동기, 장비의 경우 9000번 이후의 번호는 개별 구현에서 충돌하지 않도록 사용할 수 있다. 이 경우 개발사 번호와 매칭시켜서 판단한다.

```
typedef enum {
    DG_UNKNOWN = 0,        ///< 알수없는 장비그룹
    DG_SENSOR = 1,        ///< 센서 그룹
    DG_MOTOR = 2,         ///< 모터형 구동기 그룹
```

```

DG_SWITCH = 3,          ///< 스위치형 구동기 그룹
} devgroup_t;
typedef enum {
    DT_DEV_UNKNOWN = 0, ///< 알수없는 장비
    DT_SEN_TEMPERATURE = 10100, ///< 온도센서
    DT_SEN_HUMIDITY = 10200, ///< 습도센서
    DT_SEN_RADIATION = 10300, ///< 일사센서
    DT_SEN_WINDDIRECTION = 10400, ///< 풍향센서
    DT_SEN_WINDSPEED = 10500, ///< 풍속센서
    DT_SEN_RAIN = 10600, ///< 감우센서
    DT_SEN_RAINFALL = 10700, ///< 강우량센서
    DT_SEN_PHOTONFLUX = 10800, ///< PAR 센서
    DT_SEN_SOILMOISTURE = 10900, ///< 토양함수율센서
    DT_SEN_SOILMOISTURETENSION = 11000, ///< 토양수분장력센서
    DT_SEN_EC = 11100, ///< EC 센서
    DT_SEN_PH = 11200, ///< pH 센서
    DT_SEN_BATTERY = 11300, ///< 배터리센서
    DT_SEN_ADC = 11400, ///< ADC 변환된 Raw 값 센서 (?)
    DT_SEN_UNKNOWN = 19000, ///< 알수없는 센서
    DT_MOT_SIDEWINDOW = 20100, ///< 측창 구동기
    DT_MOT_SIDEWINDOW_1 = 20101, ///< 1 중 측창 구동기
    DT_MOT_SIDEWINDOW_2 = 20102, ///< 2 중 측창 구동기
    DT_MOT_SIDEWINDOW_3 = 20103, ///< 3 중 측창 구동기
    DT_MOT_SIDEWINDOW_4 = 20104, ///< 4 중 측창 구동기
    DT_MOT_SIDEWINDOW_5 = 20105, ///< 5 중 측창 구동기
    DT_MOT_ROOFWINDOW = 20200, ///< 천창 구동기
    DT_MOT_ROOFWINDOW_1 = 20201, ///< 1 중 천창 구동기
    DT_MOT_ROOFWINDOW_2 = 20202, ///< 2 중 천창 구동기
    DT_MOT_ROOFWINDOW_3 = 20203, ///< 3 중 천창 구동기
    DT_MOT_ROOFWINDOW_4 = 20204, ///< 4 중 천창 구동기
    DT_MOT_ROOFWINDOW_5 = 20205, ///< 5 중 천창 구동기
    DT_MOT_ROOFCURTAIN = 20400, ///< 천장보온커튼 구동기
    DT_MOT_SIDECURTAIN = 20500, ///< 측면보온커튼 구동기
    DT_MOT_SHADINGCURTAIN = 20600, ///< 차광커튼 구동기
    DT_MOT_UNKNOWN = 29000, ///< 알수없는 모터형 구동기
    DT_SWC_FAN = 30100, ///< 팬
    DT_SWC_FLOWFAN = 30101, ///< 유동팬
    DT_SWC_VENTFAN = 30102, ///< 환기팬
    DT_SWC_UNKNOWN = 39000, ///< 알수없는 스위치형 구동기
} devtype_t;

```

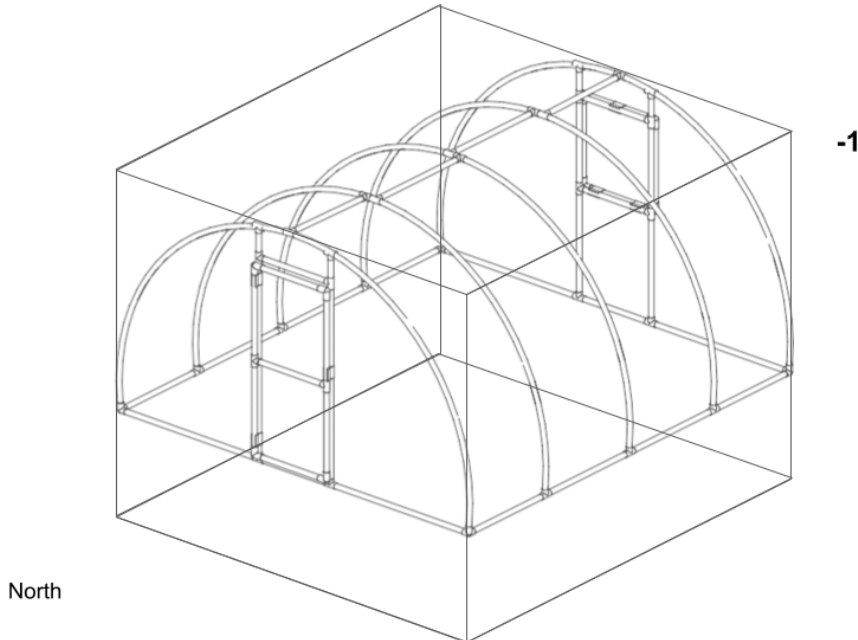
2.3. 장비 설치 구역

: 장비 설치 구역은 devsec_t로 표현된다. 장비 설치 구역은 농장을 3차원으로 구분한 구조로

구성된다. 조금 복잡할 수 있다.

농장 구역에 대한 구분 값으로 농장 외부는 항상 -1이며, 농장 내부는 0 이상의 값이 된다. 이 값은 농장을 x, y, z로 나눈 영역을 기반으로 한다. 각각을 1로 하는 경우 농장 전체가 하나의 영역 (1x1x1)이 된다. 각각을 3 등분하는 경우 농장을 27등분으로 나누는 것이 된다 (ex. cubic).

농장 구역의 최소 구분은 높이를 3 등분하는 것이다. 높이 방향으로 지하부, 작물주변, 지상부 (작물높이이상)의 3단계로 구분하고, 폭과 길이를 각각 1로 하는 것이 최소 구역 구분이다.



온실을 예로 설명하면 일반적인 단동 온실의 경우 센서를 하나만 설치하는 경우가 일반적이기 때문에 온실 내부를 최소 구분단위보다 세분하는 것이 의미가 없다. 이 경우 외부기상대는 구역 구분 값이 -1, 온실 내부 온도 센서는 작물 주변의 온도를 측정해야 하기 때문에 구역 구분값을 2로 한다. 단동 온실을 폭, 길이, 높이를 모두 3구역으로 나눈다면 총 27구역이 된다. 이때 구역 구분값은 다음과 같이 처리된다.

번호는 북쪽에 가까운 폭방향의 왼쪽부터 순차적으로 주어진다. 위 예에서는 왼쪽 아래 방향이 북쪽방향에 해당한다.

실제적으로 단동 온실을 여러 구역으로 나누는 경우는 거의 없을 것이다. 연동 온실의 경우 몇 개의 센서를 두고 관리하는 경우가 있는데, 이 경우 센서로 구분하고 싶은 구역에 따라서 x, y, z 값을 선정할 수 있다. 3 연동 온실에 동별로 센서를 설치한다면 x, y, z 값으로 3, 1, 3을 설정할 수 있을 것이다. 장비 위치는 long int 타입으로 총 12자리 정수를 사용한다. 자리수는 다음과 같다.

X(폭)	Y(길이)	Z(높이)	위치번호
2	2	2	6

다음은 단동 온실의 예시이다.

- 1) 단동 온실 내부구역을 구분하지 않아 최소 구역 단위를 사용한다. (X: 1, Y: 1, Z: 3)
- 2) 단동 온실 외부에 외부기온 측정용 센서가 있다. (위치 번호 : -1)
- 3) 단동 온실 토양에 토양수분 측정용 센서가 있다. (위치 번호 : 000001)
- 4) 단동 온실에 내부기온 측정용 센서가 있다. (위치 번호 : 010102)

5) 단동 온실에 천창이 있다. (위치번호 : 010103)

X(폭)	Y(길이)	Z(높이)	위치 번호	장비 설치 구역 코드	비고
01	01	03	-1	-1	외부 기온 센서 . 외부인 경우 X, Y, Z 는 무시.
01	01	03	010101	010103010101	토양 수분 센서
01	01	03	010102	010103010102	내부 기온 센서
01	01	03	010103	010103010103	천창 위치

```
typedef long devsec_t;
```

설치 구역정보를 용이하게 다루기 위해서 다음과 같은 미리 정의된 값들이 있다.

```
#define DL_UNKNOWN          -2          ///< 설치구역 알수 없음
#define DL_OUTSIDE         -1          ///< 설치구역 외부
#define DL_DEFAULT_ROOTZONE 10103010101  ///< 디폴트 설치구역 지하부
#define DL_DEFAULT_PLANTZONE 10103010102  ///< 디폴트 설치구역 작물부
#define DL_DEFAULT_ROOFZONE 10103010103  ///< 디폴트 설치구역 작물상부
```

2.4. 장비 작동 대상

: 장비가 작동하는 대상을 구분하는 것은 중요하다. 예를 들어 온도 센서의 경우 대상을 땅으로 하면 지온이 되고, 대기로 하면 기온이 되고, 작물로 하면 작물 온도가 된다. 따라서 해당 센서가 작동하는 대상을 지정할 수 있어야 한다.

장비가 작동하는 대상 번호는 다음과 같다.

이름	그룹	번호	대상	설명
DO_UNKNOWN	없음	1	없음	장비의 대상이 없는 경우 혹은 특정하기 어려운 경우
DO_ENV_ATMOSPHERE	환경	1001	대기	대기를 대상으로 하는 경우 (ex. 온도, 습도)
DO_ENV_SOIL	환경	1002	토양	토양을 대상으로 하는 경우 (ex. 지온, 지습, EC)
DO_ENV_NUTRIENTSOLUTION	환경	1003	양액	양액을 대상으로 하는 경우 (ex. EC, pH)
DO_PLANT_STEM	작물	2001	줄기	작물의 줄기를 대상으로 하는 경우
DO_PLANT_LEAF	작물	2002	잎	작물의 잎을 대상으로 하는 경우

DO_PLANT_FRUIT	작물	2003	과실	작물의 과실을 대상으로 하는 경우
DO_PLANT_ROOT	작물	2004	뿌리	작물의 뿌리를 대상으로 하는 경우
DO_EQUIPMENT	설비	3001	설비	설비를 대상으로 하는 경우 (ex. 창(방향 구분이 없는 창), 팬)
DO_EQUIPMENT_NORTH	설비	3002	북측설비	북쪽을 기준으로 시계 방향으로 봤을 때 가까운 설비를 대상으로 하는 경우 (ex. 우측창)
DO_EQUIPMENT_SOUTH	설비	3003	남측설비	북쪽을 기준으로 시계 방향으로 봤을 때 먼 설비를 대상으로 하는 경우 (ex. 좌측창)

```
typedef enum {
    DO_UNKNOWN =1,                //< 대상을 알지 못함
    DO_ENV_ATMOSPHERE= 1001,      //< 대기를 대상으로 함
    DO_ENV_SOIL =1002,            //< 토양을 대상으로 함
    DO_ENV_NUTRIENTSOLUTION = 1003, //< 양액을 대상으로 함
    DO_PLANT_STEM =2001,          //< 작물의 줄기를 대상으로 함
    DO_PLANT_LEAF =2002,         //< 작물의 잎을 대상으로 함
    DO_PLANT_FRUIT =2003,        //< 작물의 과실을 대상으로 함
    DO_PLANT_ROOT =2004,         //< 작물의 뿌리를 대상으로 함
    DO_EQUIPMENT =3001,          //< 농장의 장비를 대상으로 함
    DO_EQUIPMENT_NORTH = 3002,    //< 장비가 좌우대칭인 경우 북쪽에 가까운 장비를 대상으로 함
    DO_EQUIPMENT_SOUTH = 3003    //< 장비가 좌우대칭인 경우 남쪽에 가까운 장비를 대상으로 함
} devtarget_t;
```

2.5. 관측치 단위

관측치 단위는 obsunit_t로 표현된다. 사용되는 모든 단위는 이 문서에 등록되어야 한다. 단위를 조합하여 사용하는 것은 불가능하다.

아래의 관측치 단위는 TTAK.KO-10.0903 을 참고하여 작성되었다.

이름	번호	단위	설명
OU_NONE	1		단위가 없는 무차원 값
OU_CELSIUS	2	℃	섭씨온도단위
OU_PERCENT	3	%	퍼센트
OU_PPM	4	ppm	농도
OU_W_PER_MSQ	5	W/m ²	평방미터당 와트
OU_DEGREE	6	°	각도
OU_M_PER_SEC	7	m/s	속도
OU_MMOL_PER_MSQSEC	8	μ mol/m ² /s	유효광량
OU_KPA	9	kPa	압력 (토양수분장력)
OU_DS_PER_M	10	dS/m	전기전도도
OU_PH	11	pH	수소이온농도

```
typedef enum {
    OU_NONE = 1, //< 단위가 없는 무차원 값
    OU_CELSIUS = 2, //< ℃
    OU_PERCENT = 3, //< %
    OU_PPM = 4, //< ppm
    OU_W_PER_MSQ = 5, //< W/m²
    OU_DEGREE = 6, //< °
    OU_M_PER_SEC = 7, //< m/s
    OU_MMOL_PER_MSQSEC = 8, //< μ mol/m²/s
    OU_KPA = 9, //< kPa
    OU_DS_PER_M = 10, //< dS/m
    OU_PH = 11, //< pH
} obsunit_t;
```

2.6. 개발사번호

모든 드라이버 개발사는 개발사 번호를 부여받는다. 개발사 번호는 관리를 위한 번호일 뿐 다른 의미를 갖지 않는다. 개발사 번호는 천 단위의 번호이며, 이하의 숫자는 개별 제품의 종류 (드라이버의 종류)에 따라서 부여할 수 있다. 예를 들어, (사)한국농식품 ICT 융복합산업협회의 개발사 번호는 1000번이며, 1001번은 (사)한국농식품 ICT 융복합산업협회의 첫번째 드라이버 번호가 된다. 1999번까지 드라이버 번호를 할당할 수 있다. 드라이버 번호는 드라이버의 버전과는 무관하다. 개발사 번호는 협회에서 부여하며 번호를 부여하는 기준은 협회의 내규를 따른다.

3. 드라이버 설정

드라이버의 설정은 conf/cvtdriver.json에 기록되어 있다. 설정 파일은 ssdriver와 dsdriver로 나뉘어지는데 기본적인 구조는 동일하다. ssdriver와 dsdriver 모두 배열로 구성되어 있지만 실제로 ssdriver는 이번 버전에서는 하나의 아이템만이 의미가 있다. dsdriver는 여러 개의 아이템을 사용할 수 있다.

3.1. 드라이버 설정 예시

```
{
  "ssdriver": [{
    "driver": "libsssample.so",
    "option": {
      "host": "tcp://127.0.0.1:3306",
      "user": "ssdriver",
      "pass": "sssample",
      "db": "sssample"
    }
  }],
  "dsdriver": [{
    "driver": "libdsssample.so",
    "option": {
      "port": "/dev/ttyUSB0",
      "baudrate": 115200
    }
  }]
}
```

3.2. 개별 드라이버 설정

: 개별 드라이버에 대한 설정은 드라이버 파일명("driver")과 해당 드라이버 구동을 위한 옵션("option")으로 구성된다. 드라이버 파일명은 추후 협회의 시스템이 구축될 때 내부적인 규칙에 따라 정리될 예정이다. 드라이버의 옵션은 해당 드라이버에 맞게 개발자가 설정하면 된다.

위에서 ssdriver 는 파일에 정보를 기록하고, 파일로부터 명령을 읽는 구조를 가지고 있어 해당 파일 명들이 옵션에 기록되고, dsdriver 는 시리얼 통신을 통해 샘플 노드와 통신을 하는 드라이버이기 때문에 옵션으로 port와 baudrate를 가지고 있다.

3.3. 장비스펙을 반영한 설정

장비 스펙을 설정하기 위해서 코드 테이블 을 이해하는 것이 중요하다. 코드 테이블을 이해하더라도 실제 제조사의 제품에 장비 스펙에 대한 기록이 없는 경우 동적으로 이를 반영하는 것이 어렵다. 이런 경우 쉽게 장비 스펙을 반영하기 위한 옵션을 제공한다. 아래의 예에서 dsdriver의 option을 보면 , _sensors, _motors, _switches 가 있는 것을 확인할 수 있다. 이렇게 옵션을 넣어두는 경우 CvtDeviceFactory를 사용해서 쉽게 장비 스펙을 반영한 장비를 생성할 수 있다.

```
{
```

```
“ssdriver“: [{
  “driver“: “libsssample.so“,
  “option“: {
    “value“: “value.json“,
    “command“: “command.json“
  }
}],
“dsdriver“: [{
  “driver“: “libdsssample.so“,
  “option“: {
    “port“: “/dev/ttyUSB0“,
    “baudrate“: 115200,
    “_sensors“: [{
      “id“: “10“,
      “type“: “DT_SEN_HUMIDITY“,
      “section“: “DL_DEFAULT_PLANTZONE“,
      “target“: “DO_ENV_ATMOSPHERE“,
      “status“: “DS_SEN_NORMAL“,
      “unit“: “OU_PERCENT“
    }, {
      “id“: “11“,
      “type“: “DT_SEN_TEMPERATURE“,
      “section“: “DL_DEFAULT_PLANTZONE“,
      “target“: “DO_ENV_ATMOSPHERE“,
      “status“: “DS_SEN_NORMAL“,
      “unit“: “OU_CELSIUS“
    }
  ]
}, {
  “_switches“: [{
    “id“: “30“,
    “type“: “DT_SWC_FAN“,
    “section“: “DL_DEFAULT_PLANTZONE“,
    “target“: “DO_EQUIPMENT“,
    “status“: “DS_SWC_OFF“
  }, {
    “id“: “31“,
    “type“: “DT_SWC_FAN“,
    “section“: “DL_DEFAULT_PLANTZONE“,
    “target“: “DO_EQUIPMENT“,
    “status“: “DS_SWC_OFF“
  }
}],
  “_motors“: [{
    “id“: “20“,
```



```

        "type": "DT_MOT_SIDEWINDOW",
        "section": "DL_DEFAULT_PLANTZONE",
        "target": "DO_EQUIPMENT",
        "status": "DS_MOT_STOP"
    }, {
        "id": "21",
        "type": "DT_MOT_SIDEWINDOW",
        "section": "DL_DEFAULT_PLANTZONE",
        "target": "DO_EQUIPMENT",
        "status": "DS_MOT_STOP"
    }
}
}
}
}

```

4. 샘플드라이버

본 프로젝트에서는 두 가지 종류의 샘플 드라이버를 제공한다. 하나는 DSSampleDriver이고, 다른 하나는 SSSampleDriver이다. DSSampleDriver는 스마트팜 장비와 직접 통신을 위한 DSDriver의 예로 제공되는 샘플이고, SSSampleDriver는 스마트팜 장비를 핸들링할 소프트웨어와의 통신을 위한 SSDriver의 예로 제공된다 .

4.1. DSSampleDriver

소스코드에 구현되어 있다. 컨버터와 함께 제공되는 샘플 노드와 통신을 수행한다. 샘플 노드는 시리얼 통신을 통해 컨버터와 통신을 한다. 통신 프로토콜과 관련된 내용은 샘플노드의 프로토콜을 참고한다. 다수의 센서와 구동기를 다루는 예로 구현되었다. RS232 혹은 RS485 등의 통신방법을 사용하고 제조사별 프로토콜을 가지고 있다면 이 샘플을 참고하면 쉽게 드라이버 개발이 가능할 것이다.

4.2. SSSampleDriver

소스코드에 구현되어 있다. 테스트용 UI와 함께 동작하는 것으로 목표로 개발되었다. 테스트용 UI와 데이터를 공유하기 위해 mysql을 사용한다. 샘플로 제공되는 드라이버이기 때문에 디비를 사용하지만 스키마는 아래와 같이 간단하다.

```

create table devices (
    id varchar(30),
    devtype integer,
    section long,
    target integer,
    status integer,
    value float,
    unit integer

```

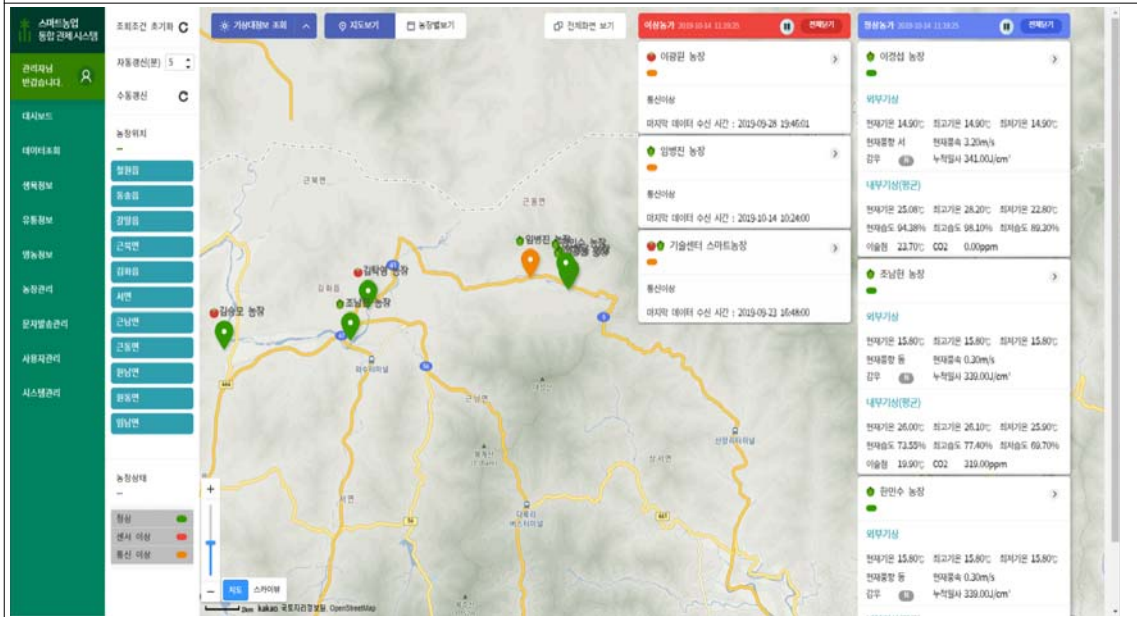
```
);  
  
create table commands (  
    id integer primary key,  
    devtype integer,  
    section long,  
    target integer,  
    onoff integer,  
    ratio float  
);
```

다. 자료 분석 장치 개발

자료분석 장치	<p>○ 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발</p> <p>- 개발자 메뉴얼 및 기술개발문서를 제작하여 업체들에게 기술이전 실시</p>
<p>(1) 클라우드 기반 컨버터 관리 및 데이터 수집시스템 목표</p> <p>표준통신기반 스마트팜 서비스의 상호운용 및 호환성을 높이기 위해 스마트팜 관련 주요 업체들의 레거시 제품에 적용할 수 있는 통신용 표준컨버터(표준드라이버)를 시스템에 적용함으로써 스마트팜 표준화 확산에 기여</p>	
<p>(2) 클라우드 기반 컨버터 관리 및 데이터 수집시스템 개발 내용</p>	

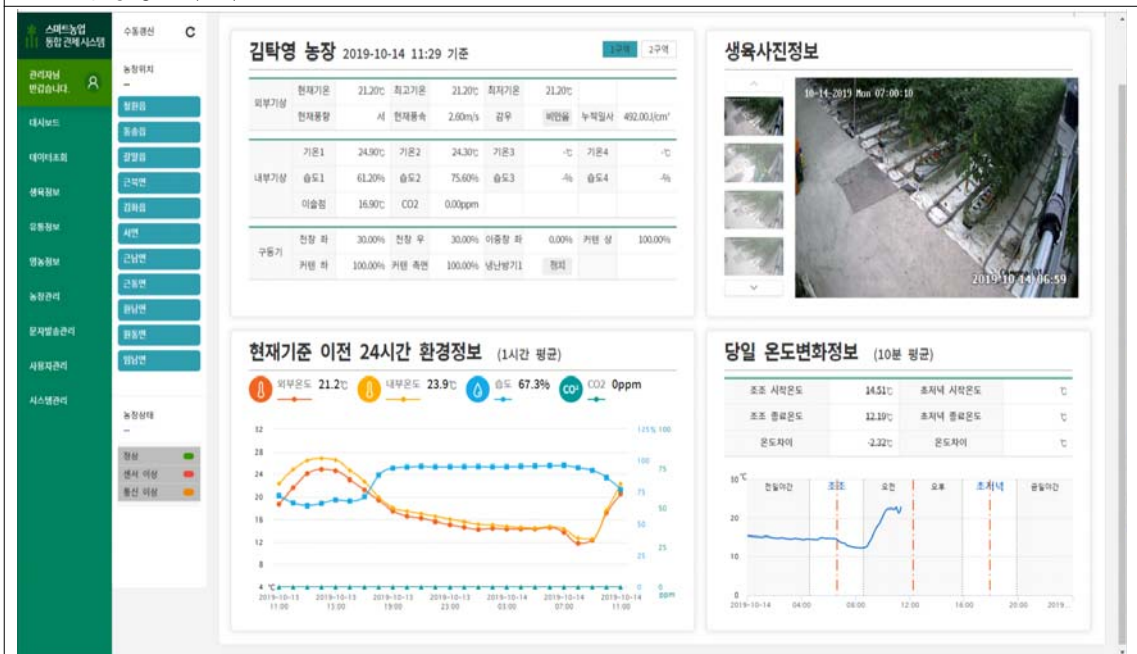
대시보드 (지도보기)

- 시스템에 연결된 전체 농장의 운영현황을 모니터링
- 농장의 주요 환경정보를 조회하며, 이상농가와 정상농가를 구분하여 조회



대시보드 (농장별보기)

- 시스템에 연결된 개별 농장의 운영현황을 모니터링
- 개별 농장의 환경정보, 제어정보를 상세하게 모니터링할 수 있으며, 설정에 따라 각각의 농장정보가 자동으로 스크롤



농장별 일간 온도변화

- 개별 농장에 대해 하루의 온도변화를 구간별로 조회 (전일야간, 조조, 오전, 오후, 초저녁, 금일야간)



농장별 환경정보 조회

- 농장을 기준으로 하나의 농장에 대해 다수의 환경정보를 조회
- 표와 차트의 형태로 조회할 수 있으며, 표로 조회했을 경우 해당 정보를 엑셀로 다운로드 받을 수 있다



농장별 제어환경정보 조회

- 환경정보를 기준으로 다수의 농장에 대해 환경정보를 상호 비교 조회
- 표와 차트의 형태로 조회할 수 있으며, 표로 조회했을 경우 해당 정보를 엑셀로 다운로드 받을 수 있다

The interface for searching environmental information by farm includes a sidebar with navigation options like '대시보드', '데이터조회', '생육정보', '유동정보', '영농정보', '농장관리', '문자발송관리', '사용자관리', and '시스템관리'. The main content area is titled '환경정보별농장조회' and contains a search filter section with date and time pickers, a search button, and a list of farm types to filter by. Below the filters, there are two tabs: '표' (Table) and '그래프' (Graph). The '표' tab displays a table with columns for '측정일시' (Measurement Time), '습도(%)' (Humidity), '온도(°C)' (Temperature), and '습도(%)' (Humidity) with corresponding farm names. The '그래프' tab displays a line chart showing the trends of these variables over time.

농장정보관리

- 시스템에 연결할 농가, 농장, 구역정보를 관리

The '농장정보관리' interface features a sidebar with navigation options and a main content area titled '농장관리'. It includes a '농가 목록' (Farm List) section and a '농가 정보' (Farm Information) form. The form contains fields for farm name, address, phone number, and email. Below this, there are sections for '농장 정보' (Farm Details), '습도농장의 온실 정보' (Greenhouse Information for Humidity Farm), and '습도농장의 구역 정보' (Zone Information for Humidity Farm), each with a '추가' (Add) button.

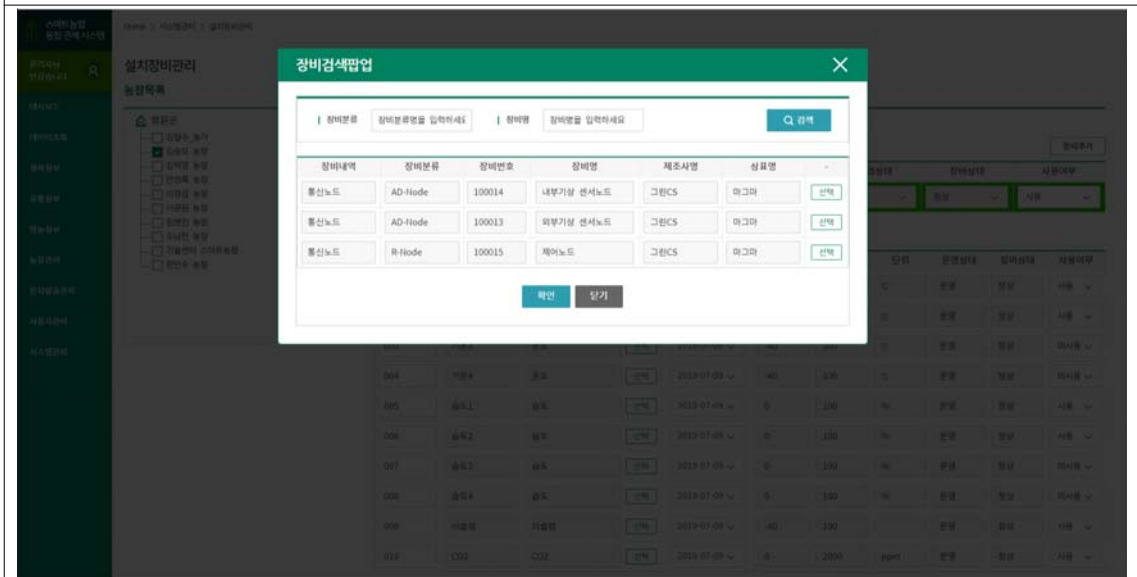
기준장비관리

- 시스템에서 연결할 수 있는 각종 센서, 구동기, 통신노드에 대한 기준장비정보를 관리
- 업체장비 등록 시 사용할 수 있는 해당 업체의 표준드라이버를 선택



설치장비관리

- 실제 농장에 설치된 각종장비를 관리 (센서, 구동기, 통신노드 등)
- 기준장비관리에서 등록한 장비를 선택
- 기준장비등록 시 해당 업체에서 사용하는 표준드라이버는 선택된 상태이며, 설치장비등록과 동시에 해당 드라이버를 통해 시스템과 장비 간 통신 연결




(3) 활용방안

- 표준컨버터로부터 산출되는 센서데이터의 시각화 서비스로 모니터링 시스템을 개발
- 표준 기반 제품 개발 시 데이터 수집상황 관제용으로 활용

라. 스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발

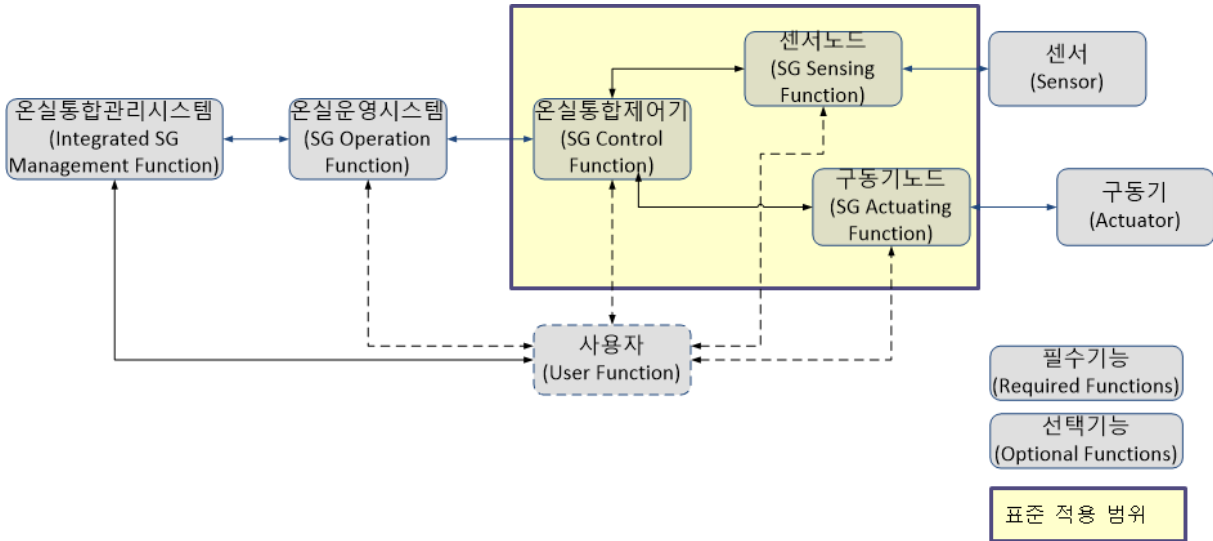
(1) RS485 MODBUS 기반의 스마트팜 통신 표준 제정

<p>표준 제정</p>	<p>○ RS485 MODBUS 기반의 스마트팜 통신 표준 제정</p> <ul style="list-style-type: none"> - TTA(한국정보통신기술협회)의 단체표준으로 제정(2018년) - 이 표준을 근거로 2019년에 KS표준이 제정됨
<div style="border: 1px solid black; padding: 10px;"> <div style="display: flex; align-items: center;"> <div style="writing-mode: vertical-rl; text-orientation: upright; font-size: 2em; font-weight: bold; margin-right: 10px;">TTA Standard</div> <div style="flex-grow: 1;"> <p>정보통신단체표준(국문표준) TTAK.KO-10.1044 제정일: 2018년 06월 27일</p> <div style="border: 1px solid black; padding: 5px; text-align: center; margin: 10px auto; width: 80%;"> <p>스마트 온실 센서/구동기 및 제어기 간 RS485 기반 모드버스(MODBUS) 인터페이스</p> <p>RS485 based MODBUS Interface between Sensor/Actuator and Controller In Smart Greenhouse</p> </div> <div style="text-align: center; margin-top: 20px;">  <p>한국정보통신기술협회 Telecommunications Technology Association</p> </div> </div> </div> </div>	

1. 표준 최종안 개발

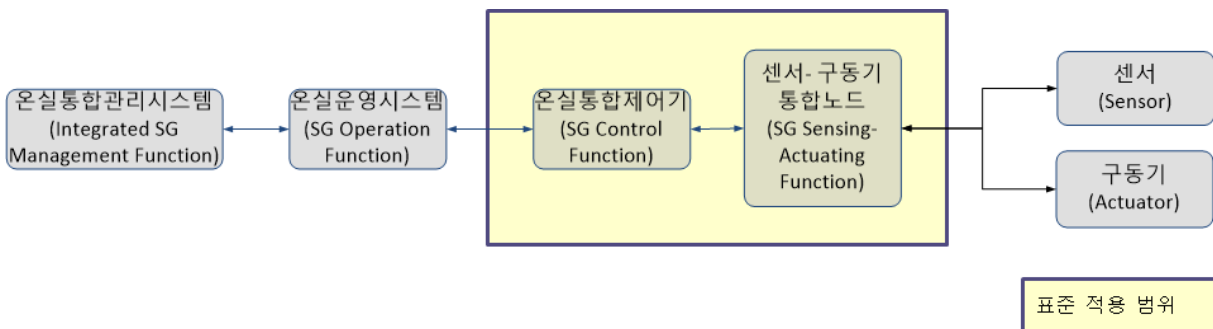
가. 기존 표준과의 관계

제안하는 표준에서 정의하는 범위는 (그림 1)과 같다.



(그림 1) 센서/구동기 노드와 온실 통합 제어기 간 RS485 기반 MODBUS 인터페이스 범위

센서 노드와 구동기 노드가 통합 노드로 존재하는 경우, 표준의 범위는 (그림 2)와 같다.



(그림 2) 센서-구동기 통합 노드와 온실 통합 제어기 간 RS485 기반 MODBUS 인터페이스 범위

제안하는 표준에서는 센서 노드/구동기 노드/센서-구동기 통합 노드와 온실 통합 제어기 간에 제공되는 기능과 각 기능을 위해 상호 교환되는 메시지 인터페이스, 전달되는 데이터에 대하여 기술한다.

나. 노드, 디바이스 ID와 레지스터 주소 매핑 방안 결정

RS485 MODBUS RTU 방식으로 통신 시 상호 연동을 위해 사전에 확보해야 할 정보는 다음과 같이 매핑된다.

- 온실 통합 제어기의 ID

- 센서 노드, 구동기 노드, 센서-구동기 통합 노드의 ID
- 센서, 구동기, 센서-구동기 디바이스의 ID

확보된 제어기, 노드, 디바이스 등의 ID는 RS485 MODBUS RTU에서 다음과 같은 주소 정보와 매핑된다
 <표 1> ID와 RS485 MODBUS 주소 매핑

노드 및 디바이스 ID	RS 485 MODBUS 주소
온실 통합 제어기의 ID	마스터 주소(Master Address)
센서 노드, 구동기 노드, 센서-구동기 통합 노드의 ID	슬레이브 주소(Slave Address)
센서, 구동기, 센서-구동기 디바이스의 ID	레지스터 주소(Register Address)

o 센서, 구동기, 센서-구동기 디바이스 ID 매핑

RS485 MODBUS에서는 특정 센서(예: 온도 센서)의 센싱 값을 얻고자 할 때, 해당 센서가 어느 레지스터 주소에 있는지를 사전에 알고 있어야 한다. 마찬가지로 특정 구동기(예: 천창)를 동작시키고자 할 때에도 해당 구동기의 레지스터 주소를 사전에 알고 있어야 한다.

확보된 센서, 구동기, 센서-구동기 디바이스의 ID는 해당 온실의 노드상에서 할당된 레지스터 주소에 매핑된다. 관련 표준에서 각 디바이스 ID의 기술 위치는 다음과 같다.

<표 2> RS485 MODBUS 레지스터 주소를 위한 디바이스 ID 기술 위치

관련 표준	디바이스 ID	주소
온실관제시스템 - 제1부 센서노드와 온실통합제어기 간 인터페이스 [TTAK.KO-06.0288-Part1_R1]	Sensor/Actuator + Type + Sensor ID (6.2.1절 참조)	RS485 MODBUS
온실관제시스템 - 제2부 제어노드와 온실통합제어기 간 인터페이스 [TTAK.KO-06.0288-Part2-R1]	Sensor/Actuator + Actuator ID (6.2.2절 참조)	레지스터 주소 (Register Address)
스마트팜 온실통합제어기와 센서-구동기 통합노드 간 통신 프로토콜 [TTAK.KO-10.0943]	DeviceType + DeviceCode + DeviceID (6.2.3절 참조)	

o 센서 노드와 온실 통합 제어기 간 인터페이스에서의 센싱 디바이스 ID

온실관제시스템 - 제1부 센서노드와 온실통합제어기 간 인터페이스 [TTAK.KO-06.0288-Part1_R1] 표준에서는 ‘Sensor/Actuator + Type + Sensor ID’ 를 통하여 특정 센싱 디바이스를 구분한다. 각 필드는

<표 3>에 기술되어 있다.

<표 3> 센싱 디바이스 ID 세부 구성 요소와 해당 표준 기술 위치 간 매핑

의미	관련 표준 위치
센서와 구동기 구분	TTAK.KO-06.0288-Part1_R1] 6장 메시지 포맷의 ‘프레임 제어 정보(frame control)’ 필드 내 ‘Sensor/Actuator’ 필드(1 bit)
온도, 습도, 풍향, 풍속 등 센서의 타입 결정	[TTAK.KO-06.0288-Part1_R1] 6.7.1절 Request 메시지의 ‘Request Command’ 필드가 ‘PASSIVE_MODE’ 인 경우, ‘Sensor Node ID’ 필드의 ‘Type’ (1 byte) 필드의 값
특정 센싱 디바이스 결정	[TTAK.KO-06.0288-Part1_R1] 6.7.1절 Request 메시지의 ‘Request Command’ 필드가 ‘PASSIVE_MODE’ 인 경우, ‘Sensor Node ID’ 필드의 ‘Sensor ID’ 필드

o 구동기 노드와 온실 통합 제어기 간 인터페이스에서의 구동기 디바이스 ID

온실관제시스템 - 제2부 제어노드와 온실통합제어기 간 인터페이스 [TTAK.KO-06.0288-Part2-R1] 표준
에서는 ‘Sensor/Actuator + Actuator ID’ 를 통하여 특정 구동기 디바이스를 구분한다. 각 필드는 <표
4>에 기술되어 있다.

<표 4> 구동기 디바이스 ID 세부 구성 요소와 해당 표준 기술 위치 간 매핑

의미	관련 표준 위치
센서와 구동기 구분	[TTAK.KO-06.0288-Part2-R1] 6장 메시지 포맷의 ‘프레임 제어 정보 (frame control) “ 필드 내 ‘Sensor/Actuator’ 필드(1 bit)
천창, 측창 등 구동기의 세부 타입 결정	-
특정 구동기 디바이스 결정	[TTAK.KO-06.0288-Part2-R1] 6.7.1절 Request 메시지의 ‘Request-Command-Type’ 필드가 ‘ACTUATOR_SET’ 인 경우, ACTUATOR_SET 페이로드의 ‘Control Value’ 필드의 ‘Actuator ID’ 필드

o 센서-구동기 통합 노드와 온실 통합 제어기 간 인터페이스에서의(센싱/구동기) 디바이스 ID

스마트팜 온실통합제어기와 센서-구동기 통합노드 간 통신 프로토콜 [TTAK.KO-10.0943] 표준에서는
‘DeviceType + DeviceCode + DeviceID’ 를 통하여 특정 센싱 디바이스나 구동기 디바이스를 구분한
다. 각 필드는 <표 5>에 기술되어 있다.

<표 5> 센서-구동기 통합 디바이스 ID 세부 구성 요소와 해당 표준 기술 위치 간 매핑

의미	관련 표준 위치
센서와 구동기 구분	[TTAK.KO-10.0943] 7.3.2.2절 디바이스 주 속성의 'Type' 필드
온도, 습도, 풍향, 풍속 등의 센싱 디바이스와 천창, 측창 등 구동기 디바이스의 세부 타입 결정	[TTAK.KO-10.0943] 7.3.2.2절 디바이스 주 속성의 'Subtype' 필드 주) 온실관제 데이터 규격 [RUCFS-0009] 표준의 <표 19> EQUIPMENT_TYPE 및 <표 35> SENSING_TYPE 개체 값 참조
특정 센서 및 구동기 디바이스 결정	[TTAK.KO-10.0943] 7.3.2.2절 디바이스 주 속성에서 'Device ID' 필드

다. 디바이스 ID와 레지스터 주소, 데이터 바이트 수 확보방안 결정

각 디바이스 ID와 레지스터 주소를 확보하기 위한 방안으로 다음과 같은 방법이 가능하다.

- 환경 설정(configuration)을 통해 사전에 확보한다.
- 매핑 테이블의 상호 교환을 통해 사전에 인지한다.
- 웹을 통해 사전에 확보한다.

이외 각 ID와 주소를 확보하기 위한 다른 방법이 존재할 수 있으며, 제안하는 표준에서는 상호 연동을 위해 사전에 알고 있다고 가정한다.

센서, 구동기, 센서-구동기 디바이스 ID는 RS485 MODBUS RTU에서 레지스터 주소와 매핑된다. 한편, RS485 MODBUS RTU에서 각 레지스터 주소는 2 바이트를 기본으로 가정하여 설계되어 있다. 그러나, 각 센서, 구동기, 센서-구동기 디바이스의 데이터 값이 2바이트 이상으로 표현될 수 있다. 따라서 각 디바이스의 ID별 레지스터 시작 주소와 함께 해당 데이터의 바이트 수를 파악하고 있어야 한다.

각 센서, 구동기, 센서-구동기 디바이스의 타입별 데이터 바이트 수는 스마트 온실 센서 메타데이터 표준과 스마트 온실 구동기 메타데이터 표준에서 정의한 메타데이터 정보를 참조한다.

<표 6> ID와 RS485 MODBUS 주소 매핑

센서, 구동기, 센서-구동기 디바이스 ID	레지스터 주소	바이트 수
센서 ID	레지스터 시작 주소	센서 데이터 바이트 수
구동기 ID	레지스터 시작 주소	구동기 데이터 바이트 수
센서-구동기 디바이스 ID	레지스터 시작 주소	디바이스 데이터 바이트 수

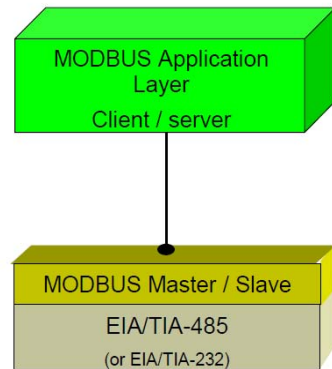
특정 센서(예: 온도 센서)에서 측정된 센싱 값 정보 및 특정 구동기(예: 천창)를 제어하기 위한 제어 명령 정보는 각 온실의 기기종 디바이스와 상관없이 동일한 정보가 동일한 형태로 표현되고 해석되어야 기기종 장치 간 상호 연동이 가능하다. 이를 위하여 각 디바이스별 데이터 타입, 범위, 단위, 소수 표현 방안, 디바이스별 속성 값 등에 대한 메타데이터 표준이 필요하다.

제안하는 표준에서는 이러한 정보를 기술하고 있는 스마트 온실 센서 메타데이터 표준과 스마트 온실 구동기 메타데이터 표준에서 정의한 내용을 참조하여 구체적인 정보 표현 방법은 제안하는 표준에서 기술하도록 한다.

라. RS485 시리얼라인 MODBUS 지원 방식 결정

MODBUS 표준은 다양한 타입의 버스나 네트워크상에 연결된 장치 사이에 클라이언트-서버 통신을 제공하는 OSI 모델의 7 계층에 해당하는 응용 계층의 메시징 프로토콜이다.

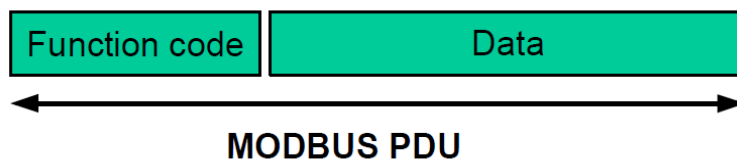
Layer	ISO/OSI Model	
7	Application	MODBUS Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	MODBUS Serial Line Protocol
1	Physical	EIA/TIA-485 (or EIA/TIA-232)



(그림 3) MODBUS 프로토콜 계층

MODBUS 응용 프로토콜은 하부의 통신 계층에 독립적인 MODBUS PDU를 정의한다. MODBUS PDU는 Function code와 Data 로 구성된다.

- Function Code: 서버가 어떤 종류의 액션을 수행해야 하는지를 표시한다. 코드 값은 1-255가 가능하나 코드 값 128-255는 예외 응답을 위해 사용되도록 예약되어 있다. 0은 사용하지 않는다.
- Data: 요청이나 응답 메시지 파라미터를 기술한다. 서버가 function code에 정의된 동작을 수행하기 위해 사용하는 부가적인 정보들이다. 레지스터 주소, 다루어야 할 항목의 양, 실 데이터 바이트의 수 등을 포함할 수 있다.



(그림 4) MODBUS PDU

데이터 인코딩 방법은 ‘big-endian’ 표현법을 사용한다. 주소가 데이터 항목의 수치 값이 한 바이트 이상이 전송될 때, 가장 의미 있는 바이트를 우선 전송한다.

예를 들어, 레지스터 크기가 16 비트이고 그 값이 ‘0x1234’ 인 경우, 0x12부터 먼저 보내고 0x34는 다음에 전송한다.

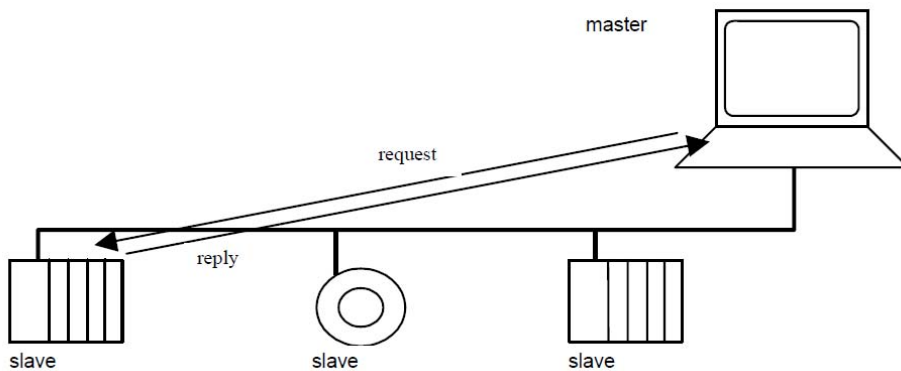
MODBUS 시리얼 라인 프로토콜은 마스터와 하나 이상의 슬레이브 사이에 MODBUS 요청 메시지를 교

환하기 위한 시리얼 라인상의 프로토콜이다. 하부의 물리적인 인터페이스로는 RS485 2선 인터페이스가 사용된다.

MODBUS 시리얼 라인 프로토콜은 Master-Slave 방식을 사용한다.

이는 버스상에 오직 하나의 마스터(master) 노드가 존재하고 최대 247개의 슬레이브(slave) 노드가 연결되어 있다. 마스터(master) 노드가 슬레이브(slave) 노드 중 하나에게 요청을 하고, 이에 대한 응답을 처리한다. 슬레이브(slave) 노드는 마스터(master) 노드로부터의 요청 없이 데이터를 전송할 수 없고, 다른 슬레이브(slave) 노드와 통신할 수도 없다. 마스터(master) 노드는 동시에 하나의 트랜잭션을 발생시킨다.

MODBUS 시리얼 라인 프로토콜은 유니캐스트 모드와 브로드캐스트 모드가 있으며, 제안하는 표준에서는 유니캐스트 모드를 사용한다.



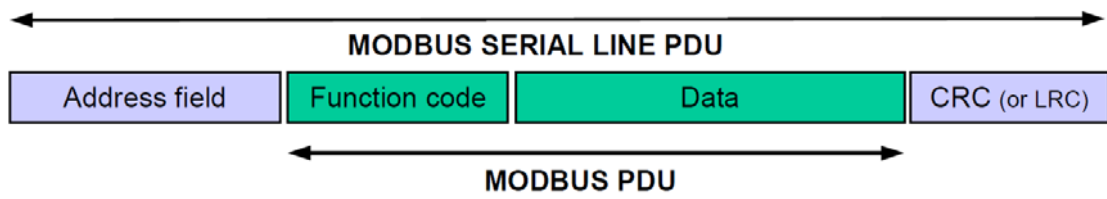
(그림 5) Master-Slave 방식

마스터(master) 노드는 특정 슬레이브(slave) 노드를 지정하여 요청 메시지를 보내고, 해당 슬레이브(slave) 노드는 요청 메시지를 받아 처리한 후 이에 대한 응답 메시지를 보낸다. 즉, MODBUS 트랜잭션은 요청과 응답의 두 가지 메시지로 구성된다. 개별 슬레이브(slave) 노드 주소는 1에서 247이 가능하다. 주소 0은 모든 슬레이브(slave) 노드에 요청 메시지를 보내는 브로드캐스트 주소로 제안하는 표준에서는 사용하지 않는다.

시리얼 라인상에서 메시지 전송 모드는 RTU(Remote Terminal Unit) 모드와 ASCII 모드가 있으며, 제안하는 표준에서는 RTU 모드를 사용한다.

각 장치가 RTU 모드를 사용하여 MODBUS 시리얼 라인으로 통신할 때 데이터 형식은 4-bit 16진수 데이터 포맷을 사용한다. 이는 문자 밀집도를 제공함으로써 ASCII 모드보다 더 좋은 처리량(throughput)을 제공한다.

MODBUS PDU를 시리얼 라인으로 전달하기 위해서는 통신 PDU로 MODBUS 시리얼 라인 PDU를 생성한다. MODBUS 시리얼 라인 PDU는 Modbus PDU의 앞에 Address Field를 추가하고, MODBUS PDU 뒤에 CRC를 추가함으로써 생성된다.



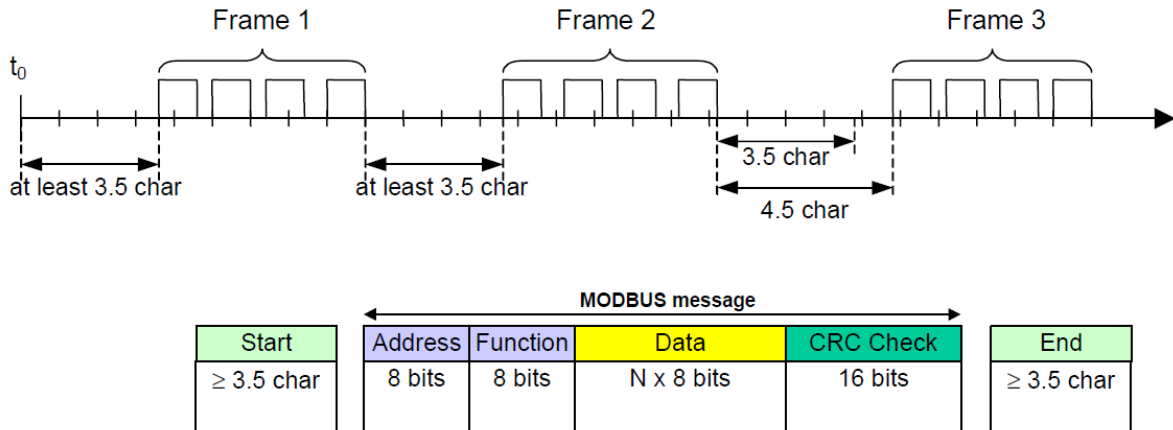
(그림 6) MODBUS 메시지 프레임과 MODBUS PDU와의 관계

MODBUS 시리얼 라인상에서 Address Field는 Slave Address만을 포함한다. 따라서, 시리얼 라인 상의 MODBUS RTU 메시지 프레임 구성은 다음과 같이 구성된다. MODBUS RTU 프레임의 최대 크기는 256 바이트이다.

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low CRC Hi

(그림 7) MODBUS 메시지 프레임 구성

MODBUS 메시지는 전송 장치가 시리얼 라인상에 프레임 형태로 놓음으로써 전달된다. 전체 메시지 프레임은 문자의 연속적인 스트림으로 전송되어야 하며, MODBUS 메시지 프레임의 전송 직전이나 직후에 3.5 문자(character) 이상의 시간 공백을 유지하도록 한다. 만약, 두 문자 사이에 1.5 문자 이상의 조용한 간격(silent interval)이 존재하면 그 메시지 프레임은 완전하지 않은 것으로 간주하여 수신자가 버리도록 한다.



(그림 8) MODBUS 메시지 전송

패킷 구조는 다음과 같은 세가지로 분류할 수 있으며 제안되는 표준에서 적용을 위하여 다음과 같이 사용된다.

- o 요청(request) 패킷

<표 7> 요청 패킷의 구조

Slave Address	Function Code	Data	CRC
1 Byte	1 Byte	0-252 Bytes	2 Bytes

- Slave address: 센서 노드나 구동기 노드의 주소가 해당됨. 1-247이 가능하며, 0은 브로드 캐스트 주소로 되어 있으나, 본 표준에서는 브로드 캐스트 주소를 사용하지 않음.
- Function Code: 지정된 슬레이브(slave)가 어떤 동작을 수행해야 할지를 알려주는 값. 마스터(Master)에서 슬레이브(slave)로 요청하는 요청 값(request code).
- Data: 슬레이브 노드가 Function Code에 정의된 동작을 수행하기 위해 사용하는 추가적인 정보들이다. 레지스터 주소, 다루어야 할 항목의 양, 실 데이터 바이트의 수 등을 포함할 수 있으며, Function Code별로 구체적인 필드가 정해짐.
- CRC: CRC-16 알고리즘 오류 체크, 수신 장치는 에러 체크를 위한 CRC 필드를 제외한 나머지 필드 값으로 CRC 계산

o 정상적인 응답(normal response) 패킷

<표 8> 정상 응답 패킷의 구조

Slave Address	Function Code	Data	CRC
1 Byte	1 Byte	0-252 Bytes	2 Bytes

- Slave address: 응답하는 센서 노드나 구동기 노드의 주소
- Function Code: 마스터(master)로부터 수신한 Function Code 값 복사
- Data: 슬레이브 노드가 Function Code에 정의된 동작을 수행한 후 응답하는 데 사용하는 추가적인 정보들이다. 레지스터 주소, 실 데이터 바이트의 수 등을 포함할 수 있으며, Function Code별로 구체적인 필드가 정해짐.
- CRC: 에러 체크 값을 표현하는 16 비트 수

o 예외 응답(exception response) 패킷

<표 9> 예외 응답 패킷의 구조

Slave Address	Function Code	Error Code	CRC
1 Byte	1 Byte	N Bytes	2 Bytes

- Slave address: 응답하는 센서 노드나 구동기 노드의 주소
- Function Code: 마스터에서 수신한 Function Code 값에 0x80 추가하여 수정
 . 마스터에서 수신한 Function Code: 0x03 → 예외 응답의 Function Code: 0x83

- . 마스터에서 수신한 Function Code: 0x06 → 예외 응답의 Function Code: 0x86
- Error Code: 예외 발생 내역에 대한 상세 코드 값
 - . 0x01(illegal function): 제품에서 지원되지 않는 function code
 - . 0x02(illegal data address): 유효하지 않은 주소를 접근하고자 할 때
 - . 0x03(illegal data value): 지원되지 않는 데이터 값으로 지정하고자 할 때
 - . 0x04(slave device failure): 디바이스 문제 발생
- CRC: 에러 체크 값을 표현하는 16 비트 수

마. 온실통합제어기와 센서노드, 구동기노드, 통합노드간 제공기능 결정

온실 통합 제어기와 센서 노드/구동기 노드/센서-구동기 통합 노드 간에 제공되는 기능은 크게 다음과 같은 두가지 기능이 제공된다. 크게 센싱 정보 획득, 구동기 상태 정보 획득 등의 데이터 확보 기능(GetData)과 구동기 제어 명령 등의 데이터 지정 기능(SetData)으로 분류된다.

이를 위하여 온실 통합 제어기는 센서 노드/구동기 노드/센서-구동기 통합 노드 내에 각 센싱 디바이스나 구동기 디바이스에 대한 정보를 알고 있다고 가정한다.

1) 데이터 확보 기능(GetData)

온실 통합 제어기는 마스터로 동작하고, 센서 노드 또는 구동기 노드는 슬레이브로 동작한다. 온실 통합 제어기(마스터)는 데이터를 읽어 올 레지스터 주소와 읽어 올 해당 레지스터 수를 포함한 Function Code 0x03(Read Holding Register) 요청 메시지를 센서 노드 또는 구동기 노드(슬레이브)에게 보낸다.

센서 노드 또는 구동기 노드(슬레이브)에서는 온실 통합 제어기(마스터)로 해당 레지스터 주소의 값과 바이트 수를 포함한 응답 메시지를 온실 통합 제어기(마스터)에게 전송한다.

이때, 온실 통합 제어기는 해당 센서 노드에 연결된 각 센서의 레지스터 주소와 데이터 값의 크기를 사전에 알고 있어야 한다. 마찬가지로 온실 통합 제어기는 해당 구동기 노드에 연결된 각 구동기의 레지스터 주소와 데이터 값의 크기를 사전에 알고 있어야 한다.

MODBUS 프로토콜에서 레지스터 데이터 값은 2 바이트를 가정하고 있다. 따라서 센서나 구동기의 데이터 값이 2 바이트를 초과하는 경우, 해당 레지스터의 시작 주소부터 연속해서 해당 데이터 값을 포함하는 크기를 요청하도록 한다.

예를 들어, 만약 온도 센서 A의 값이 4 바이트로 표현이 되는 경우, MODBUS 메시지는 온도 센서 A로 매핑된 레지스터 주소와 레지스터의 수로 2를 포함한 0x03(Read Holding Registers) 요청 메시지를 전송한다.

2) 데이터 지정 기능(SetData) - 단일 레지스터 (데이터가 2바이트 이내인 경우)

온실 통합 제어기는 마스터로 동작하고, 구동기 노드는 슬레이브로 동작한다. 온실 통합 제어기(마스터)는 데이터를 쓸 레지스터 주소와 레지스터 값을 포함한 Function Code(0x06, Write Holding Register) 요청 메시지를 구동기 노드(슬레이브)에게 보낸다.

구동기 노드(슬레이브)에서는 온실 통합 제어기(마스터)로 레지스터 주소 정보와 해당 레지스터에 기

록한 값을 포함한 응답 메시지를 온실 통합 제어기(마스터)에게 전송한다. 정상적인 응답 메시지는 요청 메시지를 그대로 표현하게 된다.

이때, 온실 통합 제어기는 해당 구동기 노드에 연결된 각 구동기의 레지스터 주소와 레지스터에 쓰고자 하는 제어 명령 데이터 값의 크기를 사전에 알고 있어야 한다.

MODBUS 프로토콜에서 레지스터 데이터 값은 2 바이트를 가정하고 있다. 따라서 구동기의 제어 명령 데이터 값이 2 바이트를 초과하지 않는 경우 사용할 수 있다. 구동기의 제어 명령 데이터 값이 2 바이트를 초과하는 경우 8.3절을 활용한다.

3) 데이터 지정 기능(SetData) - 복수 레지스터 (데이터가 2바이트를 초과하는 경우)

온실 통합 제어기는 마스터로 동작하고, 구동기 노드는 슬레이브로 동작한다. 온실 통합 제어기(마스터)는 데이터를 쓸 레지스터의 시작 주소와 레지스터 수, 레지스터 데이터 값의 바이트 수, 레지스터 값을 포함한 Function Code(0x10, Write Multiple Register) 요청 메시지를 구동기 노드(슬레이브)에게 보낸다.

구동기 노드(슬레이브)에서는 온실 통합 제어기(마스터)로 레지스터 시작 주소 정보와 레지스터 수에 대한 정보를 포함한 응답 메시지를 온실 통합 제어기(마스터)에게 전송한다.

이때, 온실 통합 제어기는 해당 구동기 노드에 연결된 각 구동기의 레지스터 주소와 레지스터에 쓰고자 하는 제어 명령 데이터 값의 크기를 사전에 알고 있어야 한다.

MODBUS 프로토콜에서 레지스터 데이터 값은 2 바이트를 가정하고 있다. 따라서 본 기능은 구동기의 제어 명령 데이터 값이 2 바이트를 초과하는 경우 사용할 수 있다. 구동기의 제어 명령 데이터 값이 2 바이트를 초과하지 않는 경우 8.2절을 활용한다.

예를 들어, 만약 천창 A에 대한 제어 명령의 값이 4 바이트로 표현이 되는 경우, MODBUS 메시지는 천창 A로 매핑된 레지스터 시작 주소와 레지스터의 수로 2, 제어 명령에 해당하는 레지스터 값의 바이트 수, 제어 명령 레지스터 값을 포함한 0x10(Write Multiple Registers) 요청 메시지를 전송한다.

또한 스마트 온실에서 각 구동기 제어 명령에 대한 데이터 포맷은 제안되는 표준에서 정의한다.

바. 센싱 정보 확보 기능, 구동기 타입별 동작 상태 확보 기능 결정

센서의 센싱 정보 및 구동기의 동작 상태는 응답 패킷의 Register Value에 포함된다. 해당 데이터 포맷은 다음과 같다.

1) 센서 데이터 포맷

스마트 온실의 각 센서에서 센싱한 데이터 값에 대한 데이터 포맷은 스마트 온실 센서 메타데이터 표준에서 정의한 내용을 따르며, 본 표준에서는 이를 다음과 같이 적용한다.

- 데이터 전송 대역폭이 적은 관계로 바이너리 인코딩 방식을 사용한다.
- 각 센서에서 센싱한 값(Sensing Value)만 전달하도록 한다.

센서에서 센싱하는 값과 관련된 속성 정보는 <10>과 같다.

<표 10> 센서의 센싱 정보 및 관련 속성 정보

구분	분류	M/O	데이터 값
상태 정보	센싱 값(Sensing Value)	M	각 센서 타입별 센싱 값

<표 8-3>에서 응답 패킷의 Register Value에 포함되는 내용은 센싱 값(value)만 포함하며, 센싱 값에 대한 바이너리 인코딩 포맷은 <표 11>와 같다.

<표 11> 센싱 디바이스 레지스터 값 표현 바이너리 포맷

필드	Bits	Type	설명	비고
Sensing Value	32	값 타입에 따라 다름	디바이스 타입 및 분류에 해당하는 디바이스 센싱 값 - 센싱 값이 float로 표현되는 경우, 4 바이트로 표현되며 IEEE-754 표준을 따른다.	<표 8-5> 참조

센싱 디바이스는 디바이스의 타입에 따라 데이터 측정 범위가 다르다. <표 12>는 각 센싱 디바이스 종류별 데이터 범위, 데이터 타입을 제공한다. 이는 스마트온실 센서 메타데이터 표준을 따른다.

<표 12> 센싱 디바이스 종류별 데이터 범위 및 데이터 타입

번호	분류	데이터 측정 범위	데이터 단위	타입
1	온도	-20~80	oC	float
2	습도	0~100	%	float
3	CO2	0-3000	ppm	float
4	일사	0-2000	W/m2	float
5	풍향	0-360	o(방향 각)	float
6	풍속	0~40	m/s	float
7	감우	on/off	-	integer
8	광양자	0-2000	umol/m2s	float

9	토양함수율	0~50	% vol.	float
10	토양수분장력	0~100	kPa	float
11	EC	0-10	dS/m	float
12	PH	2~12	pH	float
13	지온	-20~80	oC	float

2) 구동기 데이터 포맷

스마트 온실에서 각 구동기 상태에 대한 데이터 포맷은 스마트 온실 구동기 메타데이터 표준에서 정의한 내용을 따르며, 제안하는 표준에서는 이를 다음과 같이 적용한다.

- 데이터 전송 대역폭이 적은 관계로 바이너리 인코딩 방식을 사용한다.
- 구동기 타입을 스위치형과 모터형으로 구분하며, 각 타입에 따른 동작 상태와 동작 시간, 동작 상태와 동작 위치 등의 속성 정보를 함께 전달하도록 한다.
- 스위치형 구동기의 동작 기간 속성 정보와 모터형 구동기의 동작 속도 속성 정보는 적용하지 않는다.

구동기 디바이스는 구동기 타입에 따라 상태 정보가 다르다. <표 13>은 각 구동기 디바이스 종류별 구동기 타입 및 가능한 상태 정보를 제공한다. 이는 스마트온실 구동기 메타데이터 표준을 따른다.

<표 13> 구동기 디바이스 종류별 타입 구분 및 가능한 상태 정보

번호	분류	구동기 타입	상태 정보
1	천창	모터형	OPENING(여는 중)/CLOSING(닫는 중)/ OPEN(열림)/CLOSED(닫힘)
2	측창	모터형	OPENING/CLOSING/OPEN/CLOSED
3	보온재	모터형	OPENING/CLOSING/OPEN/CLOSED
4	커튼	모터형	OPENING/CLOSING/OPEN/CLOSED
5	환풍기	스위치형	ON(작동)/OFF(정지)
6	유동팬	스위치형	ON(작동)/OFF(정지)
7	관수모터	스위치형	ON(작동)/OFF(정지)
8	관수밸브	스위치형	ON(작동)/OFF(정지)
9	냉난방기	스위치형	ON(작동)/OFF(정지)

스위치형 구동기와 관련된 상태 정보 및 상태 관련 속성 정보는 <표 14>과 같이 제공한다. 이는 스마트온실 구동기 메타데이터 표준을 따른다.

<표 14> 스위치형 구동기의 상태 정보 및 관련 속성 정보

구분	분류	M/O	데이터 값
상태 정보	동작 상태 (Operation Status)	M	ON(작동)/OFF(정지)
	동작 기간 (Operation Duration)	O	동작 유지 기간 특정 일자와 특정 시간까지를 기술
	동작 시간 (Operation Time)	O	동작 유지 시간 시(hour), 분(minute), 초(second) 정보를 기술

<표 8-7>에서 응답 패킷의 Register Value에 포함되는 내용은 구동기 동작 상태 부분으로, 동작 상태에 대한 바이너리 인코딩 포맷은 <표 15>과 같다.

<표 15> 스위치형 구동기 디바이스 레지스터 값 표현 바이너리 포맷

필드	Bits	Type	설명	비고
Operation Status	8	unsigned int	구동기 동작 상태 정보 - 0x00: ON (작동) - 0x01: OFF (정지)	스위치형 구동기는 ON/OFF
Operation Time	24	unsigned int	동작 시간 - 시(hour), 분(minute), 초(second) 순서로 1 바이트씩 기술	

모터형 구동기와 관련된 상태 정보 및 상태 관련 속성 정보는 <표 16>와 같이 제공한다. 이는 스마트온실 구동기 메타데이터 표준을 따른다.

<표 816> 모터형 구동기의 상태 정보 및 관련 속성 정보

구분	분류	M/O	데이터 값
상태 정보	동작 상태 (Operation Status)	M	OPENING(여는 중)/CLOSING(닫는 중) /OPEN(열림)/CLOSED(닫힘)
	동작 위치 (Operation Position)	O	현재 위치 정보, 0 ~ 100 - 0 %는 완전히 닫힌 상태 - 100 %는 완전히 열린 상태
	동작 속도 (Operation Speed)	O	동작 속도

<표 8-9>에서 응답 패킷의 Register Value에 포함되는 내용은 구동기 동작 상태 부분으로, 동작 상태에 대한 바이너리 인코딩 포맷은 <표 17>과 같다.

<표 17> 모터형 구동기 디바이스 레지스터 값 표현 바이너리 포맷

필드	Bits	Type	설명	비고
Operation Status	8	unsigned int	구동기 상태 정보 - 0x10 : OPENING(열리는 중) - 0x11 : CLOSING(닫히는 중) - 0x12 : OPEN(열림) - 0x13 : CLOSED(닫힘)	모터형 구동기는 OPENING/CLOSING/OPEN/CLOSED
Operation Position	8	unsigned int	동작 위치 정보, 비율로 표시 - 0 ~ 100	모터형 구동기에 사용 0 %는 완전히 닫힌 상태 100 %는 완전히 열린 상태

사. 구동기 동작 제어 명령 지정 기능 결정

구동기의 동작 제어 명령 정보는 요청 패킷이나 응답 패킷의 Register Value에 포함된다. 해당 데이터 포맷은 다음과 같다.

스마트 온실에서 각 구동기 제어 명령에 대한 데이터 포맷은 스마트 온실 구동기 메타데이터 표준에서 정의한 내용을 따르며, 본 표준에서는 이를 다음과 같이 적용한다.

- 데이터 전송 대역폭이 적은 관계로 바이너리 인코딩 방식을 사용한다.
- 구동기 타입을 스위치형과 모터형으로 구분하며, 각 타입에 따른 제어 명령과 동작 시간, 제어 명령과 동작 위치 등의 속성 정보를 함께 전달하도록 한다.
- 스위치형 구동기의 동작 기간 속성 정보와 모터형 구동기의 동작 속도 속성 정보는 적용하지 않는다.

스위치형 구동기와 관련된 제어 명령 및 제어 명령 관련 속성 정보는 <표 18>와 같이 제공한다. 이는 스마트온실 구동기 메타데이터 표준을 따른다.

<표 18> 스위치형 구동기의 제어 명령 및 관련 속성 정보

구분	분류	M/O	데이터 값
제어 정보	동작 명령 (Operation Command)	M	ON(작동)/OFF(정지)
	동작 기간 (Operation Duration)	O	동작 유지 기간 특정 일자와 특정 시간까지를 기술
	동작 시간 (Operation Time)	O	동작 유지 시간 시(hour), 분(minute), 초(second) 정보를 기술

<표 8-5>에서 Register Value에 포함되는 내용은 구동기 동작 제어 명령 부분으로, 스위치형 구동기의 레지스터 값에 대한 바이너리 인코딩 포맷은 <표 19>과 같다.

<표 19> 스위치형 구동기 디바이스 레지스터 값 표현 바이너리 포맷

필드	Bits	Type	설명	비고
Operation Command	8	unsigned int	구동기 제어 명령 - 0x00: ON(작동) - 0x01: OFF(정지)	스위치형 구동기는 ON/OFF
Operation Time	24	unsigned int	동작 시간, 시(hour), 분(minute), 초(second) 순서로 1 바이트씩 기술	

모터형 구동기와 관련된 제어 명령 및 제어 명령 관련 속성 정보는 <표 20>과 같이 제공한다. 이는 스마트온실 구동기 메타데이터 표준을 따른다.

<표 20> 모터형 구동기의 제어 명령 및 관련 속성 정보

구분	분류	M/O	데이터 값
제어 정보	동작 명령 (Operation Command)	M	OPEN(열어라)/CLOSE(닫아라)
	동작 위치 (Operation Position)	O	원하는 동작 위치 정보, 0 ~ 100 - 0 %는 완전히 닫힌 상태 - 100 %는 완전히 열린 상태

<표 8-7>에서 Register Value에 포함되는 내용은 구동기 동작 상태 부분으로, 모터형 구동기의 레지스터 값에 대한 바이너리 인코딩 포맷은 <표 21>과 같다.

<표 21> 모터형 구동기 디바이스 레지스터 값 표현 바이너리 포맷

필드	Bits	Type	설명	비고
Operation Command	8	unsigned int	구동기 제어 명령 - 0x00: OPEN - 0x01: CLOSE	모터형 구동기는 OPEN/CLOSE
Operation Position	8	unsigned int	동작 위치 정보, 비율로 표시 - 0 ~ 100	모터형 구동기에 사용 0 %는 완전히 닫힌 상태 100 %는 완전히 열린 상태

아. 메시지 및 전송 데이터 포맷 결정

1) 데이터 확보 기능(GetData)

1.1) 요청 패킷

요청 메시지의 Function Code로 0x03을 사용하며, 시작 주소(Start Address)로 시작하는 레지스터부터 시작하여 레지스터의 수(Quantity of Registers) 만큼의 레지스터 값을 읽는 것을 요청한다.

<표 22> 요청 패킷의 구조

Function Code	Starting Address	Quantity of Registers
1 Byte	2 Bytes	2 Bytes

- Function Code: 0x03(Read Holding Registers) 요청
- Starting Address: 데이터를 읽어 올 대상 레지스터의 시작 주소, 각 센서나 구동기에 매핑되는 레지스터 시작 주소
- Quantity of Registers: 데이터를 읽어 올 대상 레지스터의 수, 대상 레지스터에 표현된 데이터 값의 크기

1.2) 응답 패킷

Function Code 0x03의 레지스터 읽기 요청에 대하여 해당 레지스터로부터 읽어 온 데이터 값의 바이트 수와 레지스터 데이터 값을 포함하여 응답한다.

<표 23> 응답 패킷의 구조

Function Code	Byte Count	Register Value
1Byte	1Byte	N* 2Bytes(N: Quantity of Registers)

- Function Code: 0x03(Read Holding Registers) 요청에 대한 응답
- Byte Count: 레지스터 값들을 구성하는 부분의 길이(2*N), 데이터를 표현하는 완전한 바이트의 수
- Register Value: 각 레지스터 값으로, 각 레지스터별 2 byte를 가정한다. 응답 패킷의 Register Value에 포함되는 데이터 포맷은 바. 센싱 정보 확보 기능, 구동기 타입별 동작 상태 확보 기능을 따른다.

1.3) 사용 예

a) 레지스터 100-103의 값을 읽어오기 위한 요청 메시지 구성

<표 24> 요청 메시지 사용 예

구분	필드 명	Hex 값
요청 메시지	Function Code	03
	Starting Address Hi	00
	Starting Address Lo	63
	Quantity of Registers Hi	00
	Quantity of Registers Lo	04

b) 레지스터 100-103의 값 읽기 요청에 대한 응답 메시지 구성

- 1) 레지스터 100에는 0x02 34
- 2) 레지스터 101에는 0x03 0A
- 3) 레지스터 102에는 0x63 00
- 4) 레지스터 103에는 0x00 0B 값이 있다고 가정한다.

<표 25> 응답 메시지 사용 예

구분	필드 명	Hex 값
응답 메시지	Function Code	03
	Byte Count	08
	Register Value Hi(100번 레지스터 주소)	02
	Register Value Lo(100번 레지스터 주소)	34
	Register Value Hi(101번 레지스터 주소)	03
	Register Value Lo(101번 레지스터 주소)	0A
	Register Value Hi(102번 레지스터 주소)	63
	Register Value Lo(102번 레지스터 주소)	00
	Register Value Hi(103번 레지스터 주소)	00
	Register Value Lo(103번 레지스터 주소)	0B

2) 데이터 지정 기능(SetData) - 단일 레지스터 (데이터가 2바이트 이내인 경우)

2.1) 요청 패킷

요청 메시지의 Function Code로 0x06를 사용하며, Register Address로 시작하는 레지스터에 Register Value 값을 쓰는(write) 것을 요청한다.

<표 26> 요청 패킷의 구조

Function Code	Register Address	Register Value
1 Byte	2 Bytes	2 Bytes

- Function Code: 0x06(Write Holding Registers)
- Register Address: 데이터를 기록할 대상 레지스터 주소, 각 구동기에 매핑되는 레지스터 주소
- Register Value: 데이터를 기록할 대상 레지스터 값

2.2) 응답 패킷

Function Code 0x06의 레지스터 기록 요청에 대하여 해당 레지스터의 주소와 해당 레지스터에 기록한 데이터 값을 포함하여 응답한다.

<표 27> 응답 패킷의 구조

Function Code	Register Address	Register Value
1Byte	2Bytes	2Bytes

- Function Code: 0x06(Read Holding Registers)
- Register Address: 데이터를 기록한 대상 레지스터 주소
- Register Value: 데이터를 기록한 대상 레지스터 값, 응답 패킷의 Register Value에 포함되는 데이터 포맷은 사. 구동기 동작 제어 명령 지정 기능을 따른다.

2.3) 사용 예

a) 레지스터 02에 0x 03 00의 값을 기록하기 위한 요청 메시지 구성

<표 28> 요청 메시지 사용 예

구분	필드 명	Hex 값
요청 메시지	Function Code	06
	Register Address Hi	00
	Register Address Lo	01
	Register Value Hi	03
	Register Value Lo	00

b) 레지스터 02에 0x 03 00의 값을 기록하는 요청에 대한 응답 메시지 구성

<표 29> 응답 메시지 사용 예

구분	필드 명	Hex 값
응답 메시지	Function Code	06
	Register Address Hi	00
	Register Address Lo	01
	Register Value Hi	03
	Register Value Lo	00

3) 데이터 지정 기능(SetData) - 복수 레지스터 (데이터가 2바이트를 초과하는 경우)

3.1) 요청 패킷

요청 메시지의 Function Code로 16(0x10)을 사용하며, 레지스터 시작 주소(Starting Address)로 시작하여 레지스터 수(Quantity of Registers)에 해당하는 레지스터에 Register Value 값을 Byte Count 만큼 쓰는(write) 것을 요청한다.

MODBUS 프로토콜에서 레지스터는 2 byte를 가정하고 있다. 따라서 특정 디바이스의 데이터 값이 2 byte를 초과하는 경우, 해당 레지스터의 시작 주소부터 연속해서 해당 데이터 값을 포함하는 크기를 기록하도록 한다.

또한, 특정 디바이스들을 연속한 주소로 할당하고, 각 디바이스에 대한 데이터 값의 크기를 아는 경우, 한 명령어로 두 개 이상의 디바이스에 대한 제어 명령을 전송할 수 있다.

<표 30> 요청 패킷의 구조

Function Code	Starting Address	Quantity of Registers	Byte Count	Register Value
1 Byte	2 Bytes	2 Bytes	1 Byte	(2 * N) Bytes N = Quantity of Registers

- Function Code: 16(0x10, Write Multiple Registers)
- Starting Address: 데이터를 기록할 대상 레지스터의 시작 주소
- Quantity of Registers: 대상 레지스터의 수(N)
- Byte Count: 레지스터 데이터 값의 바이트 수
- Register Value: 데이터를 기록할 대상 레지스터 데이터 값

3.2) 응답 패킷

Function Code 16(0x10)의 여러 개의 레지스터에 레지스터 기록 요청에 대하여 해당 레지스터의 시작 주소와 레지스터 수를 포함하여 응답한다.

<표 31> 요청 패킷의 구조

Function Code	Starting Address	Quantity of Registers
1 Byte	2 Bytes	2 Bytes

- Function Code: 16(0x10, Write Multiple Registers)
- Starting Address: 데이터를 기록한 대상 레지스터의 시작 주소
- Quantity of Registers: 데이터를 기록한 레지스터의 수, 응답 패킷의 Register Value에 포함되는 데이터 포맷은 사. 구동기 동작 제어 명령 지정 기능을 따른다.

3.3) 사용 예

a) 02로 시작하는 두 레지스터에 0x 03 00과 0x 01 05 값을 기록하기 위한 요청 메시지 구성

<표 32> 요청 메시지 사용 예


구분	필드 명	Hex 값
요청 메시지	Function Code	10
	Starting Address Hi	00
	Starting Address Lo	01
	Quantity of Registers Hi	00
	Quantity of Registers Lo	02
	Byte Count	04
	Register Value Hi	03
	Register Value Lo	00
	Register Value Hi	01
	Register Value Lo	05

b) 02로 시작하는 두 레지스터에 값을 기록하는 요청에 대한 응답 메시지 구성

<표 33> 응답 메시지 사용 예

구분	필드 명	Hex 값
응답 메시지	Function Code	10
	Starting Address Hi	00
	Starting Address Lo	01
	Quantity of Registers Hi	00
	Quantity of Registers Lo	02

- (2) 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스(TTA 표준)
 - (가) 2019년도에 추가로 TTA 단체 표준 제정을 목표로 개발
 - (나) 현재 심의 진행중임

표준 제정	○ 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스
TTA Standard	<div style="border: 1px solid black; padding: 5px;"> <p>정보통신단체표준(국문표준) <small>*()안에는 국문표준/영문표준/잠정표준/기술규격 등 기술</small> TTAx.xx-xx.xxxx/R1 <small>(올림, 12포인트, 굵게, 앞쪽정렬)</small> TTAx.xx-xx.xxxx/R1-Cor1</p> <p style="text-align: right;">제(개)정일: 20xx년 xx월 xx일 <small>*개정 표준의 경우 최종 개정일만 기재</small> 오류정정일: 20xx년 xx월 xx일 <small>*오류정정일의 경우 최종 정정일만 기재</small></p> </div> <div style="border: 1px solid black; padding: 10px; text-align: center; margin: 10px 0;"> <p>스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스</p> <hr style="border-top: 1px dotted black;"/> <p>Application Programming Interface of Device Driver for Interaction between Smart Farm Devices</p> </div> <p style="text-align: center; color: green;">(앞 표지)</p> <div style="text-align: center; margin-top: 20px;">  <p>한국정보통신기술협회 Telecommunications Technology Association</p> </div>

표준초안 검토 위원회 스마트농업 프로젝트그룹(PG426)

표준안 심의 위원회 정보기술 융합 기술위원회(TC4)

	성명	소속	직위	위원회 및 직위	표준번호
표준(과제) 제안	김준용	서울대학교	선임연구원		
표준 초안 작성자	김준용	서울대학교	선임연구원		
	박수현	한국과학기술연구원 강릉분원	선임연구원		
	이세용	주식회사 지능	실장		
	박주영	한국전자통신연구원	실장	PG426 부의장	
	허미영	한국전자통신연구원	책임연구원	PG426 위원	
	현욱	한국전자통신연구원	책임연구원	PG426 위원	

사무국 담당

(※ ‘표준번호’ 는 제정 또는 개정 시의 표준번호를 기입한다.)

(※ 개정된 표준일 경우, 공헌자를 제정 및 개정 표준별로 구분하여 병기할 수 있다.)

본 문서에 대한 저작권은 TTA에 있으며, TTA와 사전 협의 없이 이 문서의 전체 또는 일부를 상업적 목적으로 복제 또는 배포해서는 안 됩니다.

본 표준 발간 이전에 접수된 지식재산권 협약서 정보는 본 표준의 ‘부록(지식재산권 협약서 정보)’ 에 명시하고 있으며, 이후 접수된 지식재산권 협약서는 TTA 웹사이트에서 확인할 수 있습니다.

본 표준과 관련하여 접수된 협약서 외의 지식재산권이 존재할 수 있습니다.

발행인 : 한국정보통신기술협회 회장

발행처 : 한국정보통신기술협회

13591, 경기도 성남시 분당구 분당로 47

Tel : 031-724-0114, Fax : 031-724-0109

발행일 : 20xx.xx

서 문

1 표준의 목적

본 표준의 목적은 스마트농장에서 사용되는 장비 연동을 위한 어플리케이션 프로그래밍 인터페이스(API)를 정의하고 상세내용을 기술한다.

2 주요 내용 요약

본 표준에서는 스마트팜에서 사용되는 장비 중 본 표준의 범위에 해당하는 장비를 선별하고, 대상이 되는 장비를 연동하기 위한 API 및 사용되는 자료구조를 정의한다.

3 인용 표준과의 비교

3.1 인용 표준과의 관련성

이 표준은 온실 관제 시스템 요구사항 프로파일(TTAK.KO-06.0286)에서 제안한 온실 관제 시스템의 구성 요소를 기반한다. 본 표준의 내용은 온실 관제 시스템 센서노드와 온실 통합제 어기간 인터페이스 (TTAK.KO-06.0288-Part1/R1)와 팜클라우드와 클라우드 장치간 데이터 전송 프로토콜 (TTAK.KO-10.1007)을 참고하여 구성되었다. 본 표준 활용을 위해 스마트 온실용 센서/구동기 I/O 인터페이스 추상화 모듈 (TTAK.KO-10.1086)을 활용할 수 있다.

3.2 인용 표준과 본 표준의 비교표

	TTAK.KO -06.0286	TTAK.KO -06.0288 -Part1/R1	TTAK.KO -10.1086	비고
1. 적용 범위	일부	동일	일부	

Preface

1 Purpose

The purpose of this standard is to define the Application Programming Interface (API) for devices interworking in smart farms and to describe the details.

2 Summary

This standard sorts out devices used for smart farming and defines APIs and the data structures to utilize the selected devices.

3 Relationship to Reference Standards

3.1 Relevance to citation standards

This standard is based on the components of the greenhouse control system proposed in the Greenhouse Control System Requirements Profile (TTAK.KO-06.0286). The contents of this standard are based on the reference of the greenhouse control system sensor node and greenhouse integration control period interface (TTAK.KO-06.0288-Part1 / R1) and the data transmission protocol between farm cloud and cloud devices (TTAK.KO-10.1007).

3.2 Comparison table between the reference standard and this standard

	TTAK.KO -06.0286	TTAK.KO -06.0288 -Part1/R1	TTAK.KO -10.1086	Remarks
1. Coverage	Partial	Equal	Partial	

목 차

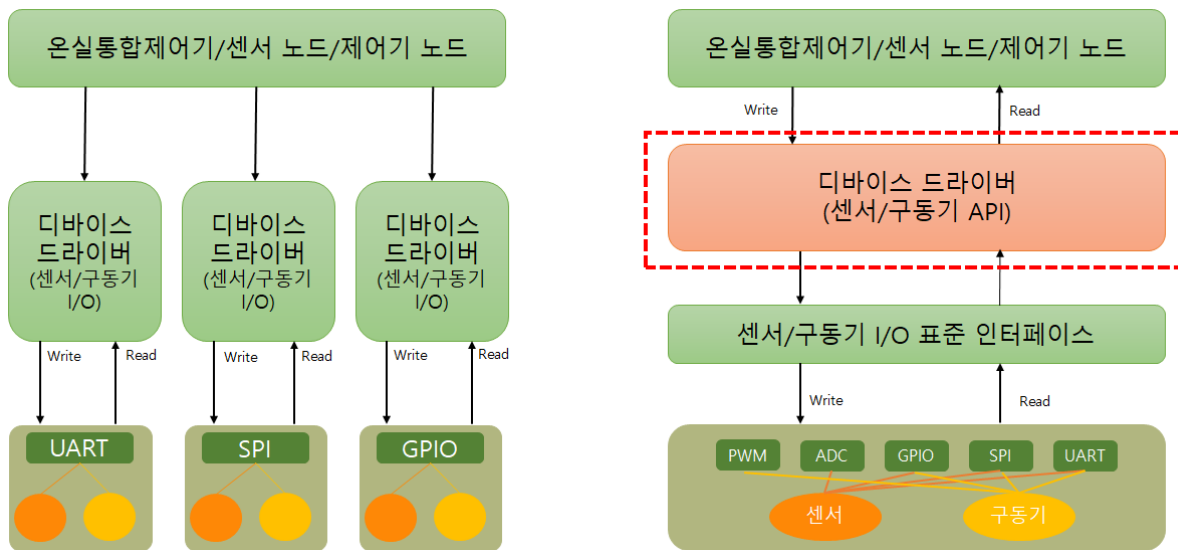
1 적용 범위	1
2 인용 표준	2
3 용어 정의	2
4 약어	2
5 일반사항	2
6 어플리케이션 프로그래밍 인터페이스	3
6.1 전체 클래스 구조	3
6.2 주요 클래스	4

7 상수 정의	5
7.1 장비 상태	5
7.2 장비 종류	5
7.3 장비 설치 구역	5
7.4 장비 작동 대상	5
7.5 관측치 단위	5
부록 I -1 지식재산권 협약서 정보	7
부록 I -2 참고문헌	8
부록 I -3 API Reference	9
부록 I -4 상수 정의 예시	29

스마트팜 장비 연동을 위한 디바이스 드라이버
어플리케이션 프로그래밍 인터페이스
(Device Driver Application Programming Interface
for Smart Farm Devices)

1. 적용 범위

본 표준은 스마트농장에서 사용되는 장비 연동을 위한 어플리케이션 프로그래밍 인터페이스를 정의하고, 이를 활용하기 위한 자료구조의 상세내용을 기술한다. 본 표준에서 제공하는 어플리케이션 프로그래밍 인터페이스(API)를 통하여 온실 통합 제어기에서 센서나 구동기에 접근할 수 있다.



(그림 1-1) 현행 장비연동 방식과 표준 API를 활용한 연동 방식

그림 1-1의 좌측과 같이 공통 API를 사용하지 않는 경우, 온실통합제어기에서 센서나 구동기 등의 장비에 접근하기 위한 인터페이스가 제품마다 다르게 되고, 이를 위한 개발비 상승 및 호환성 문제가 발생할 수 있다. 반면, 공통 API를 사용하게 되면 온실통합제어기/센서노드/구동기노드를 개발하는 경우 하위 장비의 호환성 문제를 고려하지 않고 개발을 진행할 수 있는 장점이 생긴다.

2. 인용 표준

- TTA.KO-06.0286, 온실 관제 시스템 요구사항 프로파일
- TTA.KO-06.0288-Part 1, 온실 관제 시스템 - 제1부 센서 노드와 온실 통합 제어기 간 인터페이스
- TTA.KO-10.1086, 스마트 온실용 센서/구동기 I/O 인터페이스 추상화 모듈

3. 용어 정의

가. 3.1 센서 (Sensor) [TTAK.KO-06.0286]

온실 내외의 환경 정보를 수집하는 장치

3.2 센서 노드 (Sensor Node) [TTAK.KO-06.0286]

센서와 통신모듈이 결합된 구조로서 측정된 센싱값을 온실 통합 제어기에 전달하는 장치

3.3 온실통합제어기 (GCG, Greenhouse Control Gateway) [TTAK.KO-06.0286]

온실운영시스템과 노드의 중간 매개체로서, 센서 노드로부터 받은 환경 정보를 온실운영시스템으로 전달해 주는 장치

나. 3.4 객체 식별자 (OID, Object Identifier) [TTAK.KO-10.0885]

객체를 유일하게 식별하는 식별자로 객체 식별자 트리 구조에서 특정 노드를 유일하게 식별하기 위한 루트부터 특정 노드까지의 1차 정수 값의 목록

다. 3.5 사물인터넷(IoT, Internet of Things) [TTAK.KO-10.0885]

ICT(Information and Communication Technologies)를 기반으로 다양한 물리적(physical) 및 가상(virtual)의 사물들을 연결하여 진보된 서비스를 제공하기 위한 글로벌 서비스 인프라

4. 약어

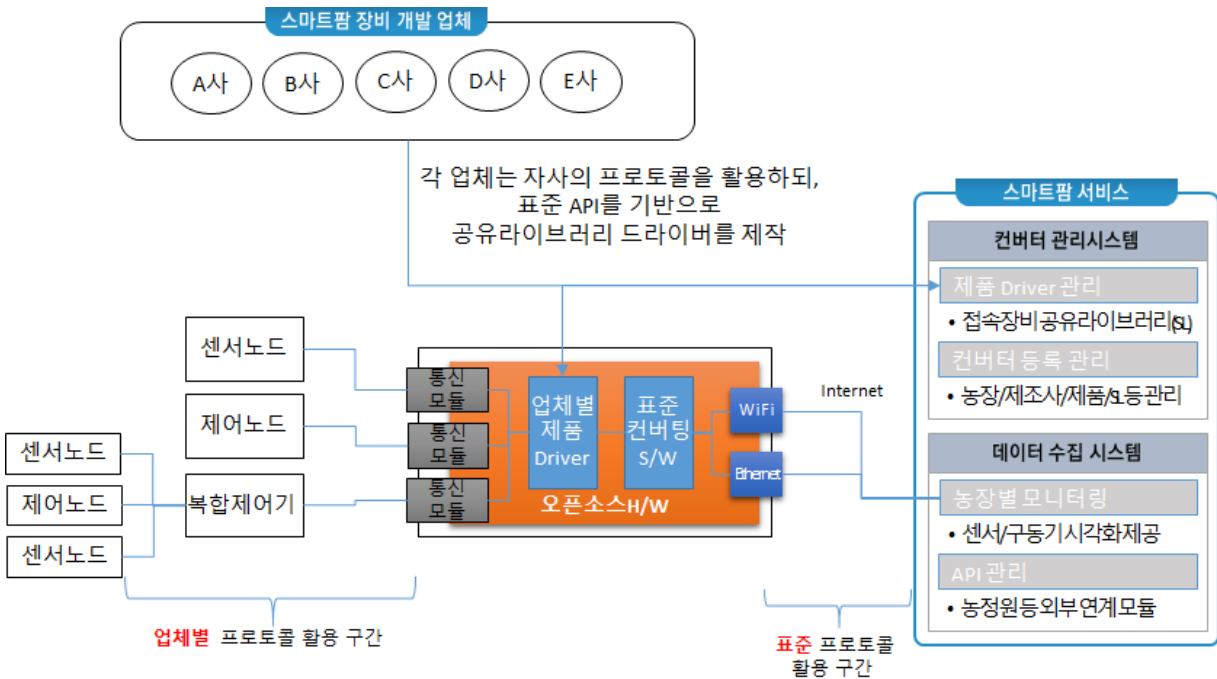
GCG Greenhouse Control Gateway

ID Identifier

ACK Acknowledgement Message

5. 일반 사항

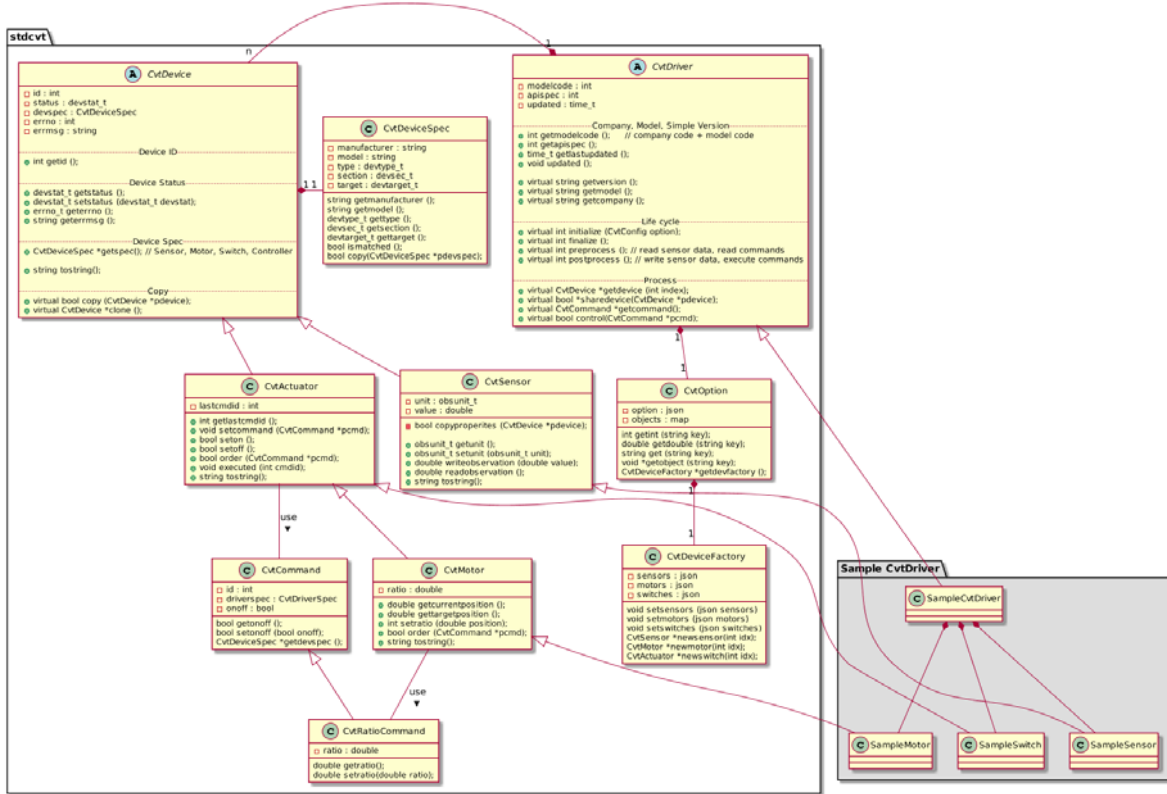
본 표준은 스마트팜에서 활용되는 장비를 드라이버형태로 연동할 수 있도록 하는 어플리케이션 프로그래밍 인터페이스를 다룬다. 이러한 형태로 장비를 연동하게 되면 스마트팜 장비 개발 업체는 표준 통신 프로토콜을 활용하는 것이 아니라 자사의 프로토콜을 그대로 활용하여 API에 맞는 드라이버를 개발할 수 있다. 이렇게 만들어진 드라이버를 교체하는 방식으로 다양한 스마트팜 장비를 연동할 수 있게 된다.



(그림 5-1) 전체 시스템 구성도

6. 어플리케이션 프로그래밍 인터페이스(API)

6.1 전체 클래스 구조



(그림 6-1) 전체 클래스 다이어그램

6.2 주요 클래스

6.2.1 CvtDriver 클래스

CvtDriver 클래스는 API를 활용할 사용자가 다루게 되는 인터페이스 클래스이다. 이를 기반으로 하여 드라이버가 제작된다.

6.2.2 CvtOption 클래스

CvtDriver에게 전달되는 설정값을 다루는 클래스이다. 사용자는 이 클래스를 통해 설정파일에 있는 설정값을 손쉽게 접근할 수 있다.

6.2.3 CvtDeviceSpec 클래스

장비의 규격을 정의하는 클래스로, 장비의 종류, 장비 설치 구역, 장비의 작동대상을 다룬다.

6.2.4 CvtDevice 클래스

장비에 대한 추상클래스로, 센서, 구동기는 이 클래스를 상속받아서 작성된다.

6.2.4.1 CvtSensor 클래스

센서를 다루는 클래스로 CvtDevice를 상속한다.

6.5.2 CvtActuator 클래스

구동기를 다루는 클래스로 스위치형 구동기를 기본으로 한다. 역시 CvtDevice를 상속한다.

6.5.3 CvtMotor 클래스

개폐기형 구동기를 다루는 클래스로 CvtActuator를 상속한다.

6.6 CvtCommand 클래스

구동기에게 전달되는 명령을 다루는 클래스로 스위치형 구동기를 위한 명령이라고 보면 된다.

6.6.1 CvtRatioCommand 클래스

개폐기형 구동기에게 전달되는 명령으로 CvtCommand를 상속한다.

7 상수 정의

본 장에서는 스마트팜 장비연동에 사용되는 API 활용을 위한 상수 정의를 다룬다.

7.1 장비 상태

장비와 센서, 스위치, 모터의 상태에 따라 상수를 사용한다. 부록 I-3의 예시를 참고한다.

7.2 장비 종류

장비종류를 표현하기 위해 상수를 사용한다. 장비의 대분류, 중분류, 소분류로 나누어 구성할 수도 있다. 장비하위구분은 구동기의 경우에 많이 보이는데, 온실의 측창이나 천창이 다 중창인 경우가 대표적인 예라고 할 수 있다. 부록 I-3 2번의 센서종류는 TTA.KO-10.0903을 참고하여 작성되었다.

알수없는 센서, 구동기, 장비의 경우 특정 번호 이후의 값을 사용하여 개별 구현에서 충돌하지 않도록 사용할 수 있다. 이 경우 개발사 번호와 매칭시켜서 판단할 수 있다.

7.3 장비 설치 구역

장비 설치 구역을 표현하기 위해 상수를 사용한다. 장비 설치 구역은 농장을 3차원으로 구분한 구조로 구성할 수 있다. 부록 I-3의 3에 있는 설치구역을 표현하기 위한 상수 구성의 예시를 참고할 수 있다.

7.4 장비 작동 대상

장비가 작동하는 대상을 구분하는 것은 중요하다. 예를 들어 온도센서의 경우 대상을 땅으

로 하면 지온이 되고, 대기로 하면 기온이 되고, 작물로 하면 작물 온도가 된다. 따라서 해당 센서가 작동하는 대상을 지정할 수 있어야 한다. 장비가 작동하는 대상번호는 부록 I-3의 4와 같다.

7.5 관측치 단위

관측치 단위를 표현하기 위해 상수를 사용한다. 단위를 조합하여 사용하는 것은 불가능하다. 관측치 단위는 TTA.KO-10.0903 을 참고하여 작성되었다.

부 록 I-1 지식재산권 요약서 정보

해당 사항 없음

부 록 I-2

참고 문헌

- [1] TTA.KO-06.0286, “온실 관제 시스템 요구사항 프로파일”, 2012.
- [2] TTA.KO-06.0288-Part 1, “온실 관제 시스템 - 제1부 센서 노드와 온실 통합 제어기 간 인터페이스”, 2015.
- [3] TTA.KO-10.1007, “팜클라우드와 클라우드 장치간 데이터 전송 프로토콜”, 2017.
- [4] TTA.KO-10.0885, “사물인터넷을 위한 객체 식별자 할당 체계”, 2015.
- [5] TTA.KO-06.0365/R1, “객체 식별자 기반 사물인터넷 디바이스 식별 체계”, 2015.
- [6] TTA.KO-10.1086, 스마트 온실용 센서/구동기 I/O 인터페이스 추상화 모듈, 2018.

부 록 I-3

API Reference

1. CvtDriver 클래스

CvtDriver 클래스는 API를 활용할 사용자가 다루게 되는 인터페이스 클래스이다.

Public 멤버 함수

생성자 & 소멸자 문서화

```
stdcvt::CvtDriver::CvtDriver ( int modelcode,  
                               int apispec  
                               )
```

새로운 드라이버를 생성한다.

매개변수

modelcode 모델코드

apispec API 버전

멤버 함수 문서화

```
virtual bool stdcvt::CvtDriver::control ( CvtCommand * pcmd )
```

다른 드라이버로부터 명령을 받아 처리한다.

매개변수

pcmd 명령에 대한 포인터

반환값

실제 명령의 처리 여부가 아니라 명령을 수신했는지 여부이다. 해당 명령을 실행할 장비가 없다면 false이다.

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

```
virtual bool stdcvt::CvtDriver::finalize ( )
```

드라이버를 종료한다.

반환값

종료 성공 여부

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

```
int stdcvt::CvtDriver::getapispec ( )
```

드라이버의 API 버전을 확인한다.

반환값

드라이버의 API 버전

```
virtual CvtCommand* stdcvt::CvtDriver::getcommand ( )
```

다른 드라이버가 관리하고 있는 장비를 제어하고자 할때 명령을 전달한다. 명령을 전달하지 않는 드라이버라면 그냥 NULL을 리턴하도록 만들면 된다. NULL이 나올때까지 반복적으로 호출한다.

반환값

명령의 포인터. NULL 이라면 이후에 명령이 없다는 의미이다.
ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

virtual string stdcvt::CvtDriver::getcompany ()

드라이버 제조사명을 확인한다. 컨버터에서는 제조사명을 로깅용도로만 사용한다.
반환값

문자열 형식의 제조사명
ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

virtual CvtDevice* stdcvt::CvtDriver::getdevice (int index)

드라이버가 관리하고 있는 장비의 포인터를 꺼내준다. 모든 장비를 꺼내주지않고, 변경된 장비만을 꺼내주는 방식으로 효율을 높일 수 있다.

매개변수

index 얻고자 하는 장비의 인덱스 번호. 0에서 시작한다.

반환값

인덱스에 해당하는 장비의 포인터. NULL 이라면 이후에 장비가 없다는 의미이다.
ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

time_t stdcvt::CvtDriver::getlastupdated ()

드라이버 가장 최근 업데이트된 시간을 확인한다.

반환값

드라이버의 최근 업데이트 시각

virtual string stdcvt::CvtDriver::getmodel ()

드라이버 제작자가 부여하는 모델번호를 확인한다. 컨버터에서는 모델코드만 확인하고, 모델번호에 대해서는 로깅용도로만 사용한다.

반환값

문자열 형식의 모델번호
ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

int stdcvt::CvtDriver::getmodelcode ()

드라이버의 모델코드를 확인한다.

반환값

드라이버의 모델코드

virtual string stdcvt::CvtDriver::getversion ()

드라이버 제작자가 부여하는 버전번호를 확인한다. 컨버터에서는 해당 버전을 로깅용도로만 사용한다. 문자열 비교를 통해 후순위가 더 높은 버전이 된다.

반환값

문자열 형식의 버전번호

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::initialize (CvtOption option)

드라이버를 초기화 한다. 드라이버 동작을 위한 option 은 key-value 형식으로 전달된다.

매개변수

option 드라이버동작을 위한 옵션

반환값

초기화 성공 여부

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::postprocess ()

드라이버간 상태교환이 이루어진 이후에 호출되는 메소드로 후처리를 수행한다.

반환값

후처리 성공 여부

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::preprocess ()

드라이버간 상태교환을 하기전에 호출되는 메소드로 전처리를 수행한다.

반환값

전처리 성공 여부

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

virtual bool stdcvt::CvtDriver::sharedevice (CvtDevice * pdevice)

전달된 장비의 정보를 획득한다. 다른 드라이버의 장비정보를 입력해주기 위해 컨버터가 호출한다.

매개변수

pdevice 다른 드라이버의 장비 포인터

반환값

성공여부. 관심이 없는 장비인 경우라도 문제가 없으면 true를 리턴한다.

ebiodriver::DSSampleDriver, ebiodriver::SSSampleDriver에서 구현되었습니다.

void stdcvt::CvtDriver::updated ()

드라이버의 내용이 업데이트되면 호출한다. 관리하고 있는 장비의 데이터가 변경되면 무조

건 호출해 주어야 한다. 현재는 최종업데이트 시간만을 관리한다.

2. CvtOption 클래스

생성자 & 소멸자 문서화

`stdcvt::CvtOption::CvtOption (json option)`

새로운 옵션을 생성한다.

매개변수

`option` 특정 드라이버를 위한 옵션. json 타입의 포인터.

멤버 함수 문서화

`string stdcvt::CvtOption::get (string key)`

옵션의 값을 문자열로 리턴한다.

매개변수

`key` 옵션을 선택하기 위한 키

반환값

옵션의 문자열 값

`CvtDeviceFactory* stdcvt::CvtOption::getdevfactory ()`

디바이스팩토리를 얻는다.

반환값

디바이스팩토리의 포인터

`double stdcvt::CvtOption::getdouble (string key)`

옵션의 값을 실수형으로 리턴한다.

매개변수

`key` 옵션을 선택하기 위한 키

반환값

옵션의 실수형값

`int stdcvt::CvtOption::getint (string key)`

옵션의 값을 정수형으로 리턴한다.

매개변수

`key` 옵션을 선택하기 위한 키

반환값

옵션의 값

json stdcvt::CvtOption::getjson (string path)

복잡한 옵션을 사용하고자 할때에는 옵션의 값을 리턴한다.

매개변수

path JSONPATH로 나타낸 경로

반환값

옵션의 값

void* stdcvt::CvtOption::getobject (string key)

옵션의 값을 void *로 리턴한다.

매개변수

key 옵션을 선택하기 위한 키

반환값

옵션의 값

void* stdcvt::CvtOption::setobject (string key,
void * object
)

옵션의 값을 세팅한다.

매개변수

key 옵션을 선택하기 위한 키

object 옵션값

반환값

옵션의 값

3. CvtDeviceSpec 클래스

생성자 & 소멸자 문서화

stdcvt::CvtDeviceSpec::CvtDeviceSpec ()

새로운 장비스펙을 생성한다.

stdcvt::CvtDeviceSpec::CvtDeviceSpec (devtype_t devtype,
devsec_t section,
devtarget_t target
)

새로운 장비스펙을 생성한다.

매개변수

devtype 장비의 종류

section 장비 설치 구역

target 장비의 대상

멤버 함수 문서화

bool stdcvt::CvtDeviceSpec::checksection (CvtDeviceSpec * pdevspec)

장비설치위치가 매치되는지 확인한다.

매개변수

pdevspec 확인할 소스 장비스펙에 대한 포인터

bool stdcvt::CvtDeviceSpec::checktarget (CvtDeviceSpec * pdevspec)

장비 작동대상이 매치되는지 확인한다.

매개변수

pdevspec 확인할 소스 장비스펙에 대한 포인터

bool stdcvt::CvtDeviceSpec::checktype (CvtDeviceSpec * pdevspec)

장비타입이 매치되는지 확인한다.

매개변수

pdevspec 확인할 소스 장비스펙에 대한 포인터

bool stdcvt::CvtDeviceSpec::copy (CvtDeviceSpec * pdevspec)

장비의 속성을 복사한다.

매개변수

pdevspe 복사할 소스 장비스펙에 대한 포인터

devgroup_t stdcvt::CvtDeviceSpec::getgrouptype ()

장비그룹의 종류를 리턴한다.

반환값

장비그룹의 종류

string stdcvt::CvtDeviceSpec::getmanufacturer ()

장비의 제조사를 확인한다.

반환값

세팅된 장비 제조사

string stdcvt::CvtDeviceSpec::getmodel ()

장비의 모델을 확인한다.

반환값

세팅된 장비 모델

devsec_t stdcvt::CvtDeviceSpec::getsection ()

장비의 설치구역을 리턴한다.

반환값

장비의 설치구역

devtarget_t stdcvt::CvtDeviceSpec::gettarget ()

장비의 대상을 리턴한다.

반환값

장비의 대상

devtype_t stdcvt::CvtDeviceSpec::gettype ()

장비의 종류를 리턴한다.

반환값

장비의 종류

bool stdcvt::CvtDeviceSpec::ismatched (CvtDeviceSpec * pdevspec)

장비스펙이 매치되는지 확인한다.

매개변수

pdevspec 확인할 소스 장비스펙에 대한 포인터

string stdcvt::CvtDeviceSpec::setmanufacturer (string manufacturer)

장비의 제조사를 세팅한다.

매개변수

manufacturer 장비 제조사

반환값

세팅된 장비 제조사

string stdcvt::CvtDeviceSpec::setmodel (string model)

장비의 모델을 세팅한다.

매개변수

model 장비 모델

반환값

세팅된 장비 모델

```
string stdcvt::CvtDeviceSpec::tostring ( )  
    장비의 스펙을 문자열로 내보낸다.
```

4. CvtDevice 클래스

생성자 & 소멸자 문서화

```
stdcvt::CvtDevice::CvtDevice ( string devid,  
                               CvtDeviceSpec * pdevspec,  
                               devstat_t devstatus  
                               )
```

새로운 장비를 생성한다.

매개변수

devid 장비의 아이디
pdevspec 장비의 스펙
devstatus 장비의 상태

```
stdcvt::CvtDevice::CvtDevice ( string devid,  
                               devtype_t devtype,  
                               devsec_t section,  
                               devtarget_t target,  
                               devstat_t devstatus  
                               )
```

새로운 장비를 생성한다.

매개변수

devid 장비의 아이디
devtype 장비의 종류
section 장비 설치 구역
target 장비의 대상
devstatus 장비의 상태

멤버 함수 문서화

virtual CvtDevice* stdcvt::CvtDevice::clone ()

장비의 클론을 만든다.

반환값

클론의 포인터

stdcvt::CvtMotor, stdcvt::CvtActuator, stdcvt::CvtSensor에서 구현되었습니다.

virtual bool stdcvt::CvtDevice::copy (CvtDevice * pdevice)

장비 정보를 복사한다.

반환값

복사가 성공하면 true.

stdcvt::CvtMotor, stdcvt::CvtActuator, stdcvt::CvtSensor에서 구현되었습니다.

static devgroup_t stdcvt::CvtDevice::getgroup (devtype_t devtype)

장비 그룹정보를 확인한다.

string stdcvt::CvtDevice::getid ()

장비에 부여된 아이디를 리턴한다.

반환값

장비의 아이디

CvtDeviceSpec* stdcvt::CvtDevice::getspec ()

장비의 스펙을 리턴한다.

반환값

장비 스펙의 포인터

devstat_t stdcvt::CvtDevice::getstatus ()

장비의 상태를 리턴한다.

반환값

장비의 상태

devstat_t stdcvt::CvtDevice::setstatus (devstat_t devstatus)

장비의 상태를 세팅한다.

매개변수

devstatus 새로 세팅할 장비의 상태

반환값

세팅된 장비의 상태

string stdcvt::CvtDevice::tostring ()

장비의 상태를 문자열로 내보낸다.

4.1 CvtSensor 클래스

생성자 & 소멸자 문서화

```
stdcvt::CvtSensor::CvtSensor ( string  devid,  
                               CvtDeviceSpec * pdevspec,  
                               devstat_t devstatus,  
                               obsunit_t unit  
                               )
```

새로운 센서를 생성한다.

매개변수

devid 센서의 아이디
pdevspec 장비 스펙
devstatus 센서의 상태
unit 관측치의 단위

```
stdcvt::CvtSensor::CvtSensor  ( string  devid,  
                                devtype_t  devtype,  
                                devsec_t  section,  
                                devtarget_t  target,  
                                devstat_t  devstatus,  
                                obsunit_t  unit  
                                )
```

새로운 센서를 생성한다.

매개변수

devid 센서의 아이디
devtype 장비의 종류
section 장비 설치 구역
target 장비의 대상
devstatus 센서의 상태
unit 관측치의 단위

멤버 함수 문서화

`CvtDevice* stdcvt::CvtSensor::clone ()`

장비의 클론을 만든다.

반환값

클론의 포인터

`stdcvt::CvtDevice`를 구현.

`bool stdcvt::CvtSensor::copy (CvtDevice * pdevice)`

장비 정보를 복사한다.

반환값

복사가 성공하면 `true`.

`stdcvt::CvtDevice`를 구현.

`obsunit_t stdcvt::CvtSensor::getunit ()`

관측치 단위를 읽는다.

반환값

관측치 단위

`double stdcvt::CvtSensor::readobservation ()`

관측치를 읽는다.

반환값

관측치값

`obsunit_t stdcvt::CvtSensor::setunit (obsunit_t unit)`

관측치 단위를 세팅한다.

매개변수

`unit` 새로 세팅할 관측치 단위

반환값

관측치 단위

`string stdcvt::CvtSensor::tostring ()`

센서의 상태를 문자열로 내보낸다.

`double stdcvt::CvtSensor::writeobservation (double value)`

관측치를 기록한다.

매개변수

`value` 관측치

반환값

기록한 관측치값

4.2 CvtActuator 클래스

생성자 & 소멸자 문서화

```
stdcvt::CvtActuator::CvtActuator ( string devid,  
                                   CvtDeviceSpec * pdevspec,  
                                   devstat_t devstatus  
                                   )
```

새로운 구동기를 생성한다.

매개변수

devid	장비의 아이디
pdevspec	장비의 스펙
devstatus	장비의 상태

```
stdcvt::CvtActuator::CvtActuator ( string devid,  
                                   devtype_t devtype,  
                                   devsec_t section,  
                                   devtarget_t target,  
                                   devstat_t devstatus  
                                   )
```

새로운 구동기를 생성한다.

매개변수

devid	장비의 아이디
devtype	장비의 종류
section	장비 설치 구역
target	장비의 대상
devstatus	장비의 상태

멤버 함수 문서화

```
CvtDevice* stdcvt::CvtActuator::clone ( )
```

장비의 클론을 만든다.

반환값

클론의 포인터

stdcvt::CvtDevice를 구현.

stdcvt::CvtMotor에서 재구현되었습니다.

bool stdcvt::CvtActuator::copy (CvtDevice * pdevice)

장비 정보를 복사한다.

반환값

복사가 성공하면 true.

stdcvt::CvtDevice를 구현.

stdcvt::CvtMotor에서 재구현되었습니다.

void stdcvt::CvtActuator::executed (int cmdid)

전달완료된 명령아이디를 세팅한다.

매개변수

cmdid 전달완료된 명령아이디

int stdcvt::CvtActuator::getlastcmdid ()

최종 명령의 아이디를 확인한다.

반환값

최종 명령의 아이디. 없을경우 -1

bool stdcvt::CvtActuator::getonoff ()

장비작동명령을 확인한다.

반환값

작동상태. true 면 on.

bool stdcvt::CvtActuator::order (CvtCommand * pcmd)

명령을 지시한다. 실제 실행하는 것은 아니고 내부에 명령을 저장하고 있다가 실제 장비에게 전달하는 역할을 담당한다.

매개변수

pcmd 명령의 포인터

반환값

명령 저장 여부.

void stdcvt::CvtActuator::setcommand (CvtCommand * pcmd)

명령을 세팅한다. order 구현시 한번씩 호출해주면 최종 아이디를 기억하도록 한다.

매개변수

pcmd 명령에 대한 포인터

string stdcvt::CvtActuator::toString ()

구동기의 상태를 문자열로 내보낸다.

반환값

구동기의 상태 문자열

`bool stdcvt::CvtActuator::turnoff ()`

장비를 작동을 중지한다.

반환값

작동상태. true 면 on.

`bool stdcvt::CvtActuator::turnon ()`

장비를 작동시킨다.

반환값

작동상태. true 면 on.

4.3 CvtMotor 클래스

생성자 & 소멸자 문서화

```
stdcvt::CvtMotor::CvtMotor ( string devid,  
                             CvtDeviceSpec * pdevspec,  
                             devstat_t devstatus  
                             )
```

새로운 모터형 구동기를 생성한다.

매개변수

<code>devid</code>	장비의 아이디
<code>pdevspec</code>	장비의 스펙
<code>devstatus</code>	장비의 상태

```
stdcvt::CvtMotor::CvtMotor ( string devid,  
                             devtype_t devtype,  
                             devsec_t section,  
                             devtarget_t target,  
                             devstat_t devstatus  
                             )
```

새로운 모터형 구동기를 생성한다.

매개변수

<code>devid</code>	장비의 아이디
<code>devtype</code>	장비의 종류

section 장비 설치 구역
target 장비의 대상
devstatus 장비의 상태

멤버 함수 문서화

CvtDevice* stdcvt::CvtMotor::clone ()

장비의 클론을 만든다.

반환값

클론의 포인터

stdcvt::CvtActuator(으)로부터 재구현되었습니다.

bool stdcvt::CvtMotor::copy (CvtDevice * pdevice)

장비 정보를 복사한다.

반환값

복사가 성공하면 true.

stdcvt::CvtActuator(으)로부터 재구현되었습니다.

double stdcvt::CvtMotor::getcurrent ()

모터형 구동기의 현재 위치를 확인한다.

반환값

구동기의 위치

double stdcvt::CvtMotor::gettarget ()

모터형 구동기의 목표 위치를 확인한다.

반환값

구동기의 위치

bool stdcvt::CvtMotor::order (CvtCommand * pcmd)

명령을 지시한다. 실제 실행하는 것은 아니고 내부에 명령을 저장하고 있다가 실제 장비에게 전달하는 역할을 담당한다.

매개변수

pcmd 명령의 포인터

반환값

실행명령이 저장되면 true, 실행할 명령이 아니면 false

double stdcvt::CvtMotor::setcurrent (double ratio)

모터형 구동기의 현재 위치를 세팅한다.

매개변수

ratio 현재 위치

반환값

현재 위치

double stdcvt::CvtMotor::settarget (double ratio)

모터형 구동기의 목표 위치를 세팅한다.

매개변수

ratio 세팅할 위치

반환값

세팅된 위치

string stdcvt::CvtMotor::tostring ()

모터형 구동기의 상태를 문자열로 내보낸다.

반환값

모터형 구동기의 상태 문자열

5. CvtCommand 클래스

생성자 & 소멸자 문서화

```
stdcvt::CvtCommand::CvtCommand ( int cmdid,  
                                  CvtDeviceSpec * pdevspec,  
                                  bool onoff  
                                  )
```

새로운 명령을 생성한다.

매개변수

cmdid 명령의 아이디

pdevspec 명령을 수행할 장비스펙의 포인터

onoff 작동상태

멤버 함수 문서화

```
CvtDeviceSpec* stdcvt::CvtCommand::getdevspec ( )
```

명령에 부여된 장비스펙을 리턴한다.

반환값

명령을 실행할 장비스펙

```
int stdcvt::CvtCommand::getid ( )
```

명령에 부여된 아이디를 리턴한다.

반환값

명령의 아이디

```
bool stdcvt::CvtCommand::getonoff ( )
```

on/off 명령을 확인한다.

반환값

on이면 true

```
int stdcvt::CvtCommand::setid ( int id )
```

명령에 부여된 아이디를 세팅한다.

매개변수

id 새로운 명령 아이디

반환값

새로 저장된 명령의 아이디

```
bool stdcvt::CvtCommand::setonoff ( bool onoff )
```

on/off 명령을 세팅한다.

매개변수

onoff on이면 true

반환값

세팅된 상태.

5.1 CvtRatioCommand 클래스

생성자 & 소멸자 문서화

```
stdcvt::CvtRatioCommand::CvtRatioCommand ( int cmdid,  
                                             CvtDeviceSpec * pdevspec,  
                                             bool onoff,  
                                             double ratio  
                                             )
```

새로운 명령을 생성한다.

매개변수

cmdid 명령의 아이디
 pdevspec 명령을 수행할 장비스펙의 포인터
 onoff 작동상태
 ratio 지정된 비율

멤버 함수 문서화

```
double stdcvt::CvtRatioCommand::getratio ( )
지정된 비율값을 확인한다.
반환값
ratio 값
```

부 록 I -4

상수 정의 예시

1. 장비 상태

작명법은 DS_로 시작하고, 대상 장비의 종류에 따라 DEV_, SEN_, SWC_, MOT_ 등을 사용할 수 있다.

<표 I.3-1> 장비 상태

이름	번호	대상장비	설명
DS_DEV_ABNORMAL	1	모든장비	장비가 정상적으로 동작하지 않음. 연결이 끊어지는 경우 포함.
DS_SEN_NORMAL	101	센서	센서가 정상적으로 동작함.
DS_SWC_ON	201	스위치	스위치가 작동중임.
DS_SWC_OFF	202	스위치	스위치가 중지상태임.
DS_MOT_OPEN	301	모터	모터가 열리는 방향으로 작동중임.
DS_MOT_CLOSE	302	모터	모터가 닫히는 방향으로 작동중임.
DS_MOT_STOP	303	모터	모터가 중지상태임.

2. 장비 종류

작명법은 센서의 경우 DT_SEN 으로 시작하고, 모터형 구동기의 경우 DT_MOT 로, 스위치형 구동기의 경우 DT_SWC로 시작한다. 이후의 명칭은 장비가 작동하는 대상이 된다. 온도센서의 경우 DT_SEN_TEMPERATURE, 측창의 경우 DT_MOT_SIDEWINDOW 가 된다.

<표 I.3-2> 장비 종류

이름	분류 번호	번호	분류	장비	설명
DT_DEV_UNKNOWN	0	0	기타장비	알수없는 장비	코드에 없는 장비인 경우.
DT_SEN_TEMPERATURE	1	10100	센서	온도센서	온도를 측정하는 센서
DT_SEN_HUMIDTY	1	10200	센서	습도센서	습도를 측정하는 센서
DT_SEN_RADIATION	1	10300	센서	일사센서	일사량을 측정하는 센서
DT_SEN_WINDIRECTION	1	10400	센서	풍향센서	풍향을 측정하는 센서
DT_SEN_WINDSPEED	1	10500	센서	풍속센서	풍속을 측정하는 센서
DT_SEN_RAIN	1	10600	센서	감우센서	비가 오는지 측정하는 센서
DT_SEN_RAINFALL	1	10700	센서	강우량센서	강우량을 측정하는 센서
DT_SEN_PHOTONFLUX	1	10800	센서	PAR센서	유해광량을 측정하는 센서
DT_SEN_SOILMOISTURE	1	10900	센서	토양습수율센서	토양 습수율을 측정하는 센서
DT_SEN_SOILMOISTURESENS ION	1	11000	센서	토양수분장력센서	토양수분장력을 측정하는 센서
DT_SEN_EC	1	11100	센서	EC센서	EC를 측정하는 센서
DT_SEN_PH	1	11200	센서	pH센서	pH를 측정하는 센서
DT_SEN_BATTERY	1	11300	센서	배터리센서	배터리의 상태를 확인하는 센서
DT_SEN_ADC	1	11400	센서	ADC	센서는 아니지만 아날로그값을 디지털로 전환한 상태의 값을 전달
DT_SEN_CO2	1	11500	센서	이산화탄소센서	이산화탄소의 농도를 측정하는 센서
DT_SEN_UNKNOWN	1	19000	센서	알수없는 센서	코드에 없는 센서인 경우.
DT_MOT_SIDEWINDOW	2	20100	모터형 구동기	측창 구동기	측창을 구동하는 모터형 구동기
DT_MOT_SIDEWINDOW_1	2	20101	모터형 구동기	1중 측창 구동기	측창을 구동하는 모터형 구동기
DT_MOT_SIDEWINDOW_2	2	20102	모터형 구동기	2중 측창 구동기	측창을 구동하는 모터형 구동기
DT_MOT_SIDEWINDOW_3	2	20103	모터형 구동기	3중 측창 구동기	측창을 구동하는 모터형 구동기
DT_MOT_SIDEWINDOW_4	2	20104	모터형 구동기	4중 측창 구동기	측창을 구동하는 모터형 구동기
DT_MOT_SIDEWINDOW_5	2	20105	모터형 구동기	5중 측창 구동기	측창을 구동하는 모터형 구동기
DT_MOT_ROOFWINDOW	2	20200	모터형 구동기	천창 구동기	천창을 구동하는 모터형 구동기
DT_MOT_ROOFWINDOW_1	2	20201	모터형 구동기	1중 천창 구동기	천창을 구동하는 모터형 구동기
DT_MOT_ROOFWINDOW_2	2	20202	모터형 구동기	2중 천창 구동기	천창을 구동하는 모터형 구동기
DT_MOT_ROOFWINDOW_3	2	20203	모터형 구동기	3중 천창 구동기	천창을 구동하는 모터형 구동기

DT_MOT_ROOFWINDOW_4	2	20204	모터형 구동기	4중 천창 구동기	천창을 구동하는 모터형 구동기
DT_MOT_ROOFWINDOW_5	2	20205	모터형 구동기	5중 천창 구동기	천창을 구동하는 모터형 구동기
DT_MOT_ROOFCURTAIN	2	20400	모터형 구동기	천장보온커튼 구동기	천장보온커튼을 구동하는 모터형 구동기
DT_MOT_SIDECURTAIN	2	20500	모터형 구동기	측면보온커튼 구동기	측면보온커튼을 구동하는 모터형 구동기
DT_MOT_SHADINGCURTAIN	2	20600	모터형 구동기	차광커튼 구동기	차광커튼을 구동하는 모터형 구동기
DT_MOT_UNKNOWN	2	29000	모터형 구동기	알수없는 모터형 구동기	코드에 없는 모터형 구동기인 경우
DT_SVC_FAN	3	30100	스위치형 구동기	팬	팬
DT_SVC_FLOWFAN	3	30101	스위치형 구동기	유동팬	내부 공기 유동을 위한 팬
DT_SVC_VENTFAN	3	30102	스위치형 구동기	환기팬	환기를 위한 팬
DT_SVC_UNKNOWN	3	39000	스위치형 구동기	알수없는 스위치형 구동기	코드에 없는 스위치형 구동기인 경우

3. 장비 설치 위치

다음은 장비 설치 위치의 예시이다.

1. 단동온실 내부구역을 구분하지 않아 최소구역단위를 사용한다. (X : 1, Y : 1, Z : 3)
2. 단동온실 외부에 외부기온 측정용 센서가 있다. (위치번호 : -1)
3. 단동온실 토양에 토양수분 측정용 센서가 있다. (위치번호 : 000001)
4. 단동온실에 내부기온 측정용 센서가 있다. (위치번호 : 010102)
5. 단동온실에 천창이 있다. (위치번호 : 010103)

<표 1.3-3> 장비 설치 위치

X(폭)	Y(길이)	Z(높이)	위치번호	장비설치구역코드	비고
01	01	03	-1	-1	외부기온센서. 외부인경우 X, Y, Z는 무시.
01	01	03	010101	010103010101	토양수분센서
01	01	03	010102	010103010102	내부기온센서
01	01	03	010103	010103010103	천창위치

4. 장비 작동 대상번호

<표 I.3-4> 장비 작동 대상번호

이름	그룹	번호	대상	설명
DO_UNKNOWN	없음	1	없음	장비의 대상이 없는 경우 혹은 특정하기 어려운 경우
DO_ENV_ATMOSPHERE	환경	1001	대기	대기를 대상으로 하는 경우 ex. 온도, 습도
DO_ENV_SOIL	환경	1002	토양	토양을 대상으로 하는 경우 ex. 지온, 지습, EC
DO_ENV_NUTRIENT SOLUTION	환경	1003	양액	양액을 대상으로 하는 경우 ex. EC, pH
DO_PLANT_STEM	작물	2001	줄기	작물의 줄기를 대상으로 하는 경우
DO_PLANT_LEAF	작물	2002	잎	작물의 잎을 대상으로 하는 경우
DO_PLANT_FRUIT	작물	2003	과실	작물의 과실을 대상으로 하는 경우
DO_PLANT_ROOT	작물	2004	뿌리	작물의 뿌리를 대상으로 하는 경우
DO_EQUIPMENT	설비	3001	설비	설비를 대상으로 하는 경우 ex. 창(방향구분이 없는 창), 팬
DO_EQUIPMENT_NORTH	설비	3002	북측설비	북쪽을 기준으로 시계방향으로 봤을때 가까운 설비를 대상으로 하는 경우 ex. 우측창
DO_EQUIPMENT_SOUTH	설비	3003	남측설비	북쪽을 기준으로 시계방향으로 봤을때 먼 설비를 대상으로 하는 경우 ex. 좌측창

5. 관측치의 단위

<표 I.3-5> 관측치의 단위

이름	번호	단위	설명
OU_NONE	1		단위가 없는 무차원 값
OU_CELSIUS	2	℃	섭씨온도단위
OU_PERCENT	3	%	퍼센트
OU_PPM	4	ppm	농도
OU_W_PER_MSQ	5	W/m ²	평방미터당 와트
OU_DEGREE	6	°	각도
OU_M_PER_SEC	7	m/s	속도
OU_MMOL_PER_MSQSEC	8	μ mol/m ² /s	유효광량
OU_KPA	9	kPa	압력(토양수분장력)
OU_DS_PER_M	10	dS/m	전기전도도

(3) 신규 표준을 적용한 레퍼런스 장비 개발

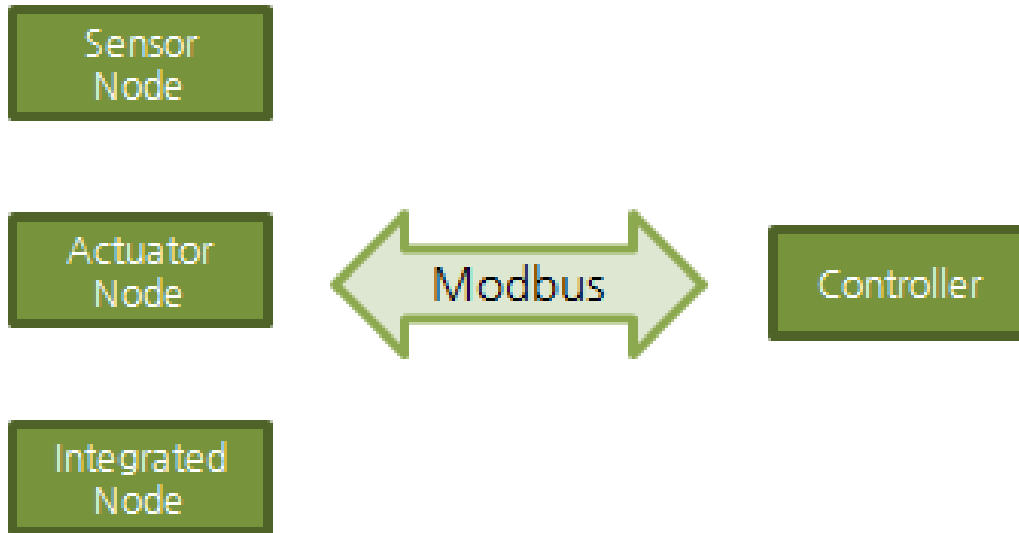
표준개발	<ul style="list-style-type: none"> ○ 스마트온실 센서/구동기 및 제어기간 표준 레퍼런스 모델 개발 <ul style="list-style-type: none"> - 제정된 TTA표준(TTAK.KO-10.1044)을 적용하여 실제 작동되는 모형 온실과 구동기노드/센서노드를 제작 공개
-------------	--

□ 연구개발의 목표

CMO 과제에서 제정한 TTAK.KO-10.1044* 표준을 활용한 레퍼런스 모델을 개발하고, 표준 컨버터와 연결한다.

* 스마트 온실 센서/구동기 및 제어기 간 RS485 기반 모드버스(MODBUS) 인터페이스

○ 목표시스템 구성도



가. TTAK.KO-10.1044 표준 분석 및 시스템 설계

- (1) TTAK.KO-10.1044 표준 분석
- (2) 신규표준적용을 위한 시스템 설계

나. 신규 표준 적용 드라이버 개발

- (1) 신규 표준 라이브러리 설계 및 개발
- (2) 신규 표준용 DSDriver 설계 및 개발

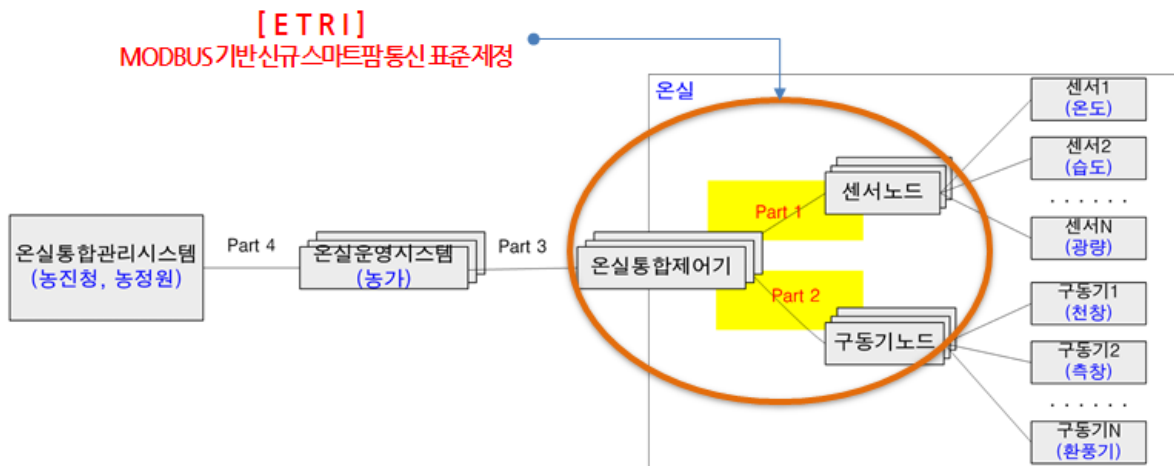
다. 신규 표준 드라이버를 적용한 레퍼런스 장비 개발

- (1) 신규 표준 적용 센서/구동기 복합노드 설계
- (1) 신규 표준 적용 센서/구동기 복합노드 개발

□ 연구개발의 내용

가) TTAK.KO-10.1044 표준 분석 및 시스템 설계

- TTAK.KO-10.1044 표준은 온실통합제어기와 센서/구동기 노드 사이의 통신을 다룬다. RS-485 위에서 동작하는 MODBUS를 기반으로 하고 있다.



- 본 절에서는 온실 통합 제어기와 센서 노드/구동기 노드/센서-구동기 통합 노드 간에 제공되는 기능을 기술한다. 크게 센싱 정보 획득, 구동기 상태 정보 획득 등의 데이터 확보 기능(GetData)과 구동기 제어 명령 등의 데이터 지정 기능(SetData)으로 분류된다. 이를 위하여 온실 통합 제어기는 센서노드/구동기 노드/센서-구동기 통합 노드 내에 각 센서나 구동기에 대한 정보를 알고 있다고 가정한다.
- 데이터 확보 기능은 온실 통합 제어기는 마스터로 동작하고, 센서 노드 또는 구동기 노드는 슬레이브로 동작한다. 온실 통합 제어기(마스터)는 데이터를 읽어 올 레지스터 주소와 읽어 올 해당 레지스터 수를 포함한 기능코드 0x03(Read Holding Register) 요청 메시지를 센서 노드 또는 구동기 노드(슬레이브)에게 보낸다. 센서 노드 또는 구동기 노드(슬레이브)는 온실 통합 제어기(마스터)로 해당 레지스터 주소의 값과 바이트 수를 포함한 응답 메시지를 온실 통합 제어기(마스터)에 전송한다.
- 데이터 지정 기능은 온실 통합 제어기는 마스터로 동작하고, 구동기 노드는 슬레이브로 동작한다. 온실 통합 제어기(마스터)는 데이터를 쓸 레지스터 주소와 레지스터 값을 포함한 기능 코드(0x06, Write Holding Register) 요청 메시지를 구동기 노드(슬레이브)에 보낸다. 구동기 노드(슬레이브)에서는 온실 통합 제어기(마스터)로 레지스터 주소 정보와 해당 레지스터에 기록한 값을 포함한 응답 메시지를 온실 통합 제어기(마스터)에게 전송한다. 정상적인 응답 메시지는 요청 메시지를 그대로 표현하게 된다.

나) 신규 표준 적용 드라이버 개발

- 표준 컨버터는 표준 API를 사용해서 스마트팜 장비와 통신하고 그 콘텐츠를 표준 통신프로토콜로 외부로 전달하는 게이트웨이 역할을 수행하게 된다.
- 신규 표준을 적용한 센서/구동기 복합노드는 해당 표준을 활용하는 표준 컨버터와 통신을 할 수 있다. 나아가 통합제어기에서 직접 해당 노드와 통신을 수행하는 것도 가능하다.
- 이때 통합제어기는 신규 표준을 적용하기 위한 라이브러리가 필요하다. 본 과제에서는 과제의 통일성을 위해 표준컨버터에 활용될 수 있는 신규표준용 DSDriver를 개발하는 방법을 사용하였다.
- 모형 온실의 복합노드와 통신을 위해 구현된 드라이버의 소스의 주요한 부분은 다음과 같다. 표준을 통해 장비에서 데이터를 수신하기 위해 구현된 코드이다.

```
# 레지스터에서 읽은 값을 파싱한다.
def parseregisters(self, names, values):
    idx = 0
    ret = {}
    for nm in names:
        (size, vtype) = KSX3267Mate._KEYWORDS[nm]
        if vtype == "float":
            val = struct.unpack('f', struct.pack('HH', values[idx], values[idx+1]))[0]
        elif vtype == "int":
            val = struct.unpack('i', struct.pack('HH', values[idx], values[idx+1]))[0]
        else:
            val = values[idx]
        ret[nm] = val
        idx = idx + size
    print "parsed", ret
    return ret

# 하나의 장비에 대한 정보를 읽는다.
def readinfofromdev(self, dev):
    size = self.getsize(self.getdk(dev, 2))
    for _ in range(3):
        res = self.readregister(self.getdk(dev, 1), size, self.getdk(dev, 0))
```

```

        if res.isError():
            self._logger.info("retry to get status from " + str(dev['dk']) + " " +
str(res))
            continue
        else:
            if len(res.registers) == size:
                return self.parseregisters(self.getdk(dev, 2), res.registers)
            else:
                self._logger.info("retry to get data since size of data is not matched.
" + str(size) + " " + str(len(res.registers)))
            self._logger.warn("fail to get status from " + str(dev['dk']))
            return None

# 노드의 정보를 읽는다. readinfofromdev 메소드를 활용한다.
def readnodeinfo(self, node):
    ret = {"id" : node["id"], "sen" : {}, "act" : {}}
    info = self.readinfofromdev(node)
    if info:
        ret["nd"] = info
    else:
        self._logger.warn("fail to read node info : " + str(node))
        #return None

    for dev in node['children']:
        info = self.readinfofromdev(dev)
        if info:
            if DevType.issensor(dev["dt"]):
                ret["sen"][dev["id"]] = info
            else:
                ret["act"][dev["id"]] = info
        else:
            self._logger.warn("fail to read dev info : " + str(dev) + " however
continue to read other device")
    return ret

```

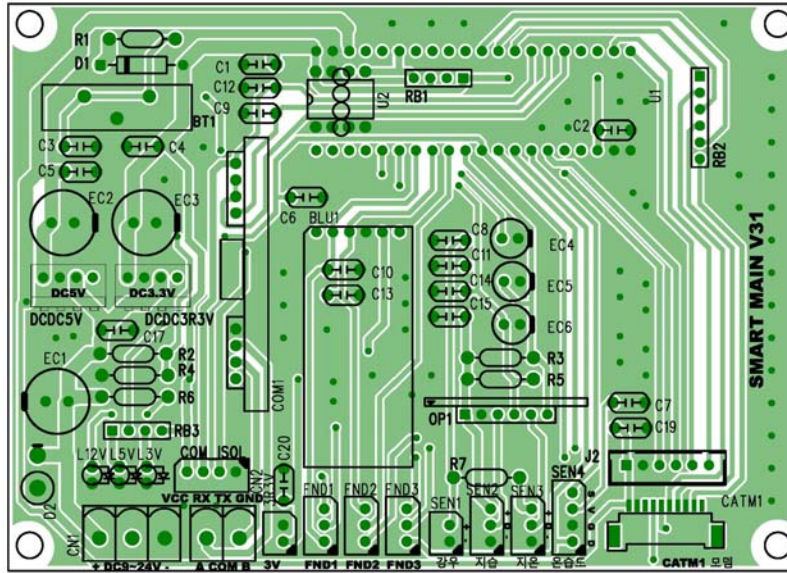

다) 신규 표준 드라이버를 적용한 레퍼런스 장비(센서/구동기 복합 노드) 개발

- 신규 표준에서는 센서노드, 구동기노드 또한 센서/구동기 복합노드가 활용될 수 있다.
- 센서/구동기 복합노드는 센서노드와 구동기노드의 역할을 동시에 수행할 수 있기 때문에 센서노드, 구동기노드로 분리하여 개발하는 것보다 센서/구동기 복합노드로 레퍼런스 장비를 개발하는 것이 유리하다고 판단하였다.
- 온실에서 사용될 복합노드의 모드버스맵은 다음과 같이 결정하였다.

장비정보	2	3	4	5	6	7													
정보	회사코드	타입/노드/양역	제플코드	프로토콜버전	연결장비수	구역수													
Type	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit													
Value	7	3	1	101	16	0													
연결제플코드	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116			
정보	장치코드#1	장치코드#2	장치코드#3	장치코드#4	장치코드#5	장치코드#6	장치코드#7	장치코드#8	장치코드#9	장치코드#10	장치코드#11	장치코드#12	장치코드#13	장치코드#14	장치코드#15	장치코드#16			
Type	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit	Uint 16bit			
Value	101	202	303	704	905	806	607	108	209	6210	6211	6212	6213	5214	5215	5216			
노드상태	40201	40202	-																
정보	opid#1	노드상태																	
Type	Uint 16bit	Uint 16bit																	
센서정보	40211	40212	40213	-	40221	40222	40223	-	40231	40232	40233	-	40241	40242	40243	-			
정보	온도센서값	온도센서상태			습도센서값	습도센서상태			이슬점센서값	이슬점센서상태			일사량값	일사량계 상태					
Type	Float 32bit	Uint 16bit			Float 32bit	Uint 16bit			Float 32bit	Uint 16bit			Float 32bit	Uint 16bit					
센서정보	40251	40252	40253	-	40261	40262	40263	-	40271	40272	40273	-	40281	40282	40283	-			
정보	풍향센서값	풍향계상태			풍속센서값	풍속계상태			감우센서값	감우센서상태			온도센서값(온실내)	온도센서상태					
Type	Float 32bit	Uint 16bit			Float 32bit	Uint 16bit			Float 32bit	Uint 16bit			Float 32bit	Uint 16bit					
센서정보	40291	40292	40293	-															
정보	습도센서값(온실내)	습도센서상태																	
Type	Float 32bit	Uint 16bit																	
구동기정보	40301	40302	40303	40304	-	40311	40312	40313	40314	-									
정보	opid#2	작동상태	작동남은시간			opid#3	우회상태	작동남은시간											
Type	Uint 16bit	Uint 16bit	Uint32bit			Uint 16bit	Uint 16bit	Uint32bit											
구동기정보	40321	40322	40323	40324	-	40331	40332	40333	40334	-									
정보	opid#4	좌측상태	작동남은시간			opid#5	우측상태	작동남은시간											
Type	Uint 16bit	Uint 16bit	Uint32bit			Uint 16bit	Uint 16bit	Uint32bit											
구동기정보	40341	40342	40343	40344	-	40351	40352	40353	40354	-	40361	40362	40363	40364	-				
정보	opid#6	커플상태	작동남은시간			opid#7	유동판상태	작동남은시간			opid#8	보광동상태	작동남은시간						
Type	Uint 16bit	Uint 16bit	Uint32bit			Uint 16bit	Uint 16bit	Uint32bit			Uint 16bit	Uint 16bit	Uint32bit						
노드제어명령	45001	45002	-																
정보	명령코드	opid#1																	
Type	Uint16bit	Uint16bit																	
좌우전장제어명	45011	45012	45013	45014	-	45021	45022	45023	45024	-									
정보	좌전장/명령코드	opid#2	작동시간			우전장/명령코드	opid#3	작동시간											
Type	Uint16bit	Uint16bit	Uint32bit			Uint16bit	Uint16bit	Uint32bit											
좌우전장제어명	45031	45032	45033	45034	-	45041	45042	45043	45044	-									
정보	좌측장/명령코드	opid#4	작동시간			우측장/명령코드	opid#5	작동시간											
Type	Uint16bit	Uint16bit	Uint32bit			Uint16bit	Uint16bit	Uint32bit											
좌우전장제어명	45051	45052	45053	45054	-	45061	45062	45063	45064	-	45071	45072	45073	45074	-				
정보	커플/명령코드	opid#6	작동시간			유동판/명령코드	opid#7	작동시간			보광동/명령코드	opid#8	작동시간						
Type	Uint16bit	Uint16bit	Uint32bit			Uint16bit	Uint16bit	Uint32bit			Uint16bit	Uint16bit	Uint32bit						

- 복합노드를 실체화 하기 위해서 동일한 센서구성과 구동기구성을 갖는 모형 온실을 제작하였다. 모형온실을 위해 제작된 PCB는 다음과 같다.

SMART MAIN V31_ing-3.pc - 1ue Oct 01 15:07:25 2019



(COMPONENT SIDE) FR4 : 121mm x 87mm x 1.6T

- 구현된 모형 온실은 다음과 같다. 시연이 가능하도록 좌측에 터치 모니터를 설치하였고, 온실 아래쪽에 공간을 만들어 PC와 표준 컨버터를 설치할 수 있도록 하였다.



2절. 연구개발 성과

1. 지식재산권 성과

1-1. 지식재산권(출원)		출원번호	출원일자	출원인
1-11	ISG에 설치된 IoT 디바이스간 협업을 위한 리소스 할당 방법 및 협업 방법	10-2017-0095335	2017-07-27	ETRI
1-12	스마트팜 비표준 센서의 표준화 컨버터장치	10-2019-0039360	2019-04-04	KAICAF외 1명
1-13	스마트팜용 비표준센서장치의 데이터측정장치	10-2019-0039365	2019-04-04	KAICAF외 1명
1-14	스마트팜 안전 작업시스템	10-2019-0039362	2019-04-04	KAICAF외 1명
1-15	스마트팜 작업량 측정장치	10-2019-0039356	2019-04-04	KAICAF외 1명
1-2. 지식재산권(등록)		등록번호	등록일자	등록인
1-21	스마트온실에서이종센서/구동기및제어기간상호연동을 위한 RS485기반 modbus 인터페이스에대한 TTA 단체표준 과제 제안	TTAK.KO-10.1044	2018-06-27	허미영
1-22	스마트팜용 비표준 센서장치의 데이터측정장치	10-2034406-00-00	2019-10-11	KAICAF외 1명
1-23	ISG에 설치된 IoT 디바이스간 협업을 위한 리소스 할당 방법 및 협업 방법	2019.11월 등록예정	-	허미영
1-24	스마트팜 비표준 센서의 표준화 컨버터장치	2019.11월 등록예정	-	KAICAF외 1명
1-25	스마트팜 안전 작업시스템	2019.11월 등록예정	-	KAICAF외 1명
1-26	스마트팜 작업량 측정장치	2019.11월 등록예정	-	KAICAF외 1명

2. 논문게재 성과

논문	구분	게재 연도	저자명	학술지명	논문등록 번호 (ISSN)	비고	
2-1	스마트농업 국제 표준화 동향 (Vol. 34_No. 1_2017)	비SCI	2016	박주영 허미영	한국통신학 회지	1027-1030	국내
2-2	한국형 시설원에 스마트 팜 평가 기준 개발을 위한 모델 연구 (Vol. 8_No. 8_2017)	비SCI	2017	김태형 김대호	한국융합학 회논문지	2233-4890	국내
2-3	지역별 주요 작목의 재배면적 변화와 농업 소득 간 관계에 대한 탐색적 연구 -비교 분석에서 최적 포트폴리오 분석까지 (REPORT_Vol. 11_2019)	비SCI	2019	장익훈 김연진 최도형 최영찬 정구현	농식품IT전문 보고서_AIM REPORT _vol.11_2019	2234-3075 (Print) 2288-7806 (Online)	국내

3. 학술대회 발표성과

발표제목		학술회의명	인쇄물명	발표일	발표자	비고
3-1	스마트온실의 IoT 디바이스간 협업을 위한 리소스 할당 기법	제 27회 통신정보 합동학술대회	JCCi2017	2017-04-27	허미영 박주영	국내
3-2	시설원예용 스마트 팜 평가지표 개발에 관한 연구	학문의 융합과 상호 발전	(사)인문사회과학 기술융합학회	2017-07-07	김태형 김대호	국내

4. 기술요약정보

기술실시(이전)		
4-1. 표준 컨버터용 디바이스 드라이버 개발 기술		이전일자
4-1.1	그리심산업	2018-04-20
4-1.2	그린씨에스(주)	2018-04-20
4-1.3	린캔들라이트	2018-04-20
4-1.4	멀틱스	2018-04-20
4-1.5	메디앙시스템	2018-04-20
4-1.6	유샘 인스트루먼트	2018-04-20
4-1.7	제닉스시스템	2018-04-20
4-1.8	주식회사 지능	2018-04-20
4-1.9	코리아디지털	2018-04-20
4-1.10	한국시설원예ICT협동조합	2018-04-20
4-1.11	(주)에이투지시스템	2019-06-24
4-1.12	(주)인지시스템	2018-05-31
4-1.13	인포벨리코리아	2019-06-24
4-1.14	우림인포텍	2019-06-24
4-1.15	애니스마트	2019-06-24
4-2. 스마트팜 기기 연동을 위한 표준 컨버터 개발 기술		이전일자
4-2.1	그리심산업	2019-06-24
4-2.2	그린씨에스(주)	2019-06-24
4-2.3	린캔들라이트	2019-06-24
4-2.4	멀틱스	2019-06-24
4-2.5	메디앙시스템	2019-06-24
4-2.6	유샘 인스트루먼트	2019-06-24
4-2.7	제닉스시스템	2019-06-24
4-2.8	주식회사 지능	2019-06-24
4-2.9	코리아디지털	2019-06-24
4-2.10	한국시설원예ICT협동조합	2019-06-24
4-2.11	(주)에이투지시스템	2019-06-24
4-2.12	(주)인지시스템	2019-06-24
4-2.13	인포벨리코리아	2019-06-24
4-2.14	우림인포텍	2019-06-24
4-2.15	애니스마트	2019-06-24

5. 소프트웨어 등록

5	저작권(sw)	등록번호	등록일자
5-1	스마트팜 기기연동을 위한 표준컨버터	134121-0006994	2019-05-10

6. 사업화(제품화)

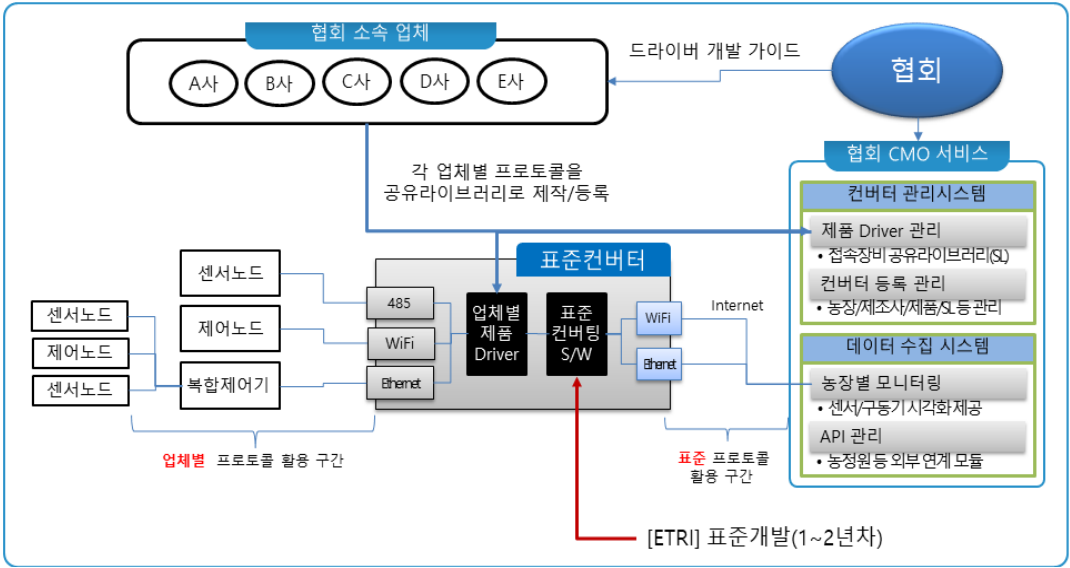
기술명		기술실시기관	기술료	제품명	제품출시일	매출액 기여율	총 매출액 (vat별도)
6-1	표준 컨버터용 다바이스 드라이버 개발기술	에스엔제이	무상	화성 스마트팜 시스템	2018-12	20%	6,255,000원
6-2	표준 컨버터용 다바이스 드라이버 개발기술	주식회사 지능	무상	스마트팜 관제용 시스템	2018-10	40%	163,215,000원
				스마트농업 관제시스템 기반조성 관제시스템	2019-03	20%	
6-3	표준 컨버터용 다바이스 드라이버 개발기술	코리아디지털(주)	무상	센서 스테이션	2019-08	20%	68,430,000원
합계							237,900,000원

3장

3장 목표 달성도 및 관련 분야 기여도

3장 목표 달성도 및 관련 분야 기여도

가. 목표

구분	내용
<p>최종목표</p>	<p>표준 통신 기반 스마트팜 서비스의 상호운용 및 호환성을 높이기 위하여, 기존 스마트팜 업체들에게 표준의 중요성에 대한 인식을 확대하고, 업체들의 레거시 제품들에 적용가능한 표준 컨버터 개발하며 업체들의 참여를 촉진함으로써 스마트팜 표준화의 확산에 기여한다.</p> 
<p>세부목표</p>	<p>○ 본 과제의 기술 개발 목표를 중심으로 다음 4개의 핵심 기술 개발을 진행함</p> <ol style="list-style-type: none"> (1) 스마트팜 표준화 확산 지원 <ul style="list-style-type: none"> - 스마트팜 장비 표준화 지원을 위한 설명회 및 간담회 개최 - 농기자재 박람회 참여 - 스마트팜 업체 가이드북 제작 (2) 표준화 컨버터 개발 : 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발 <ul style="list-style-type: none"> - 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발(기술이전) - 스마트팜 장비 연동용 표준 컨버터 개발(기술이전) (3) 자료 분석 장치 개발 <ul style="list-style-type: none"> - 클라우드 기반 컨버터 관리 및 데이터수집시스템 개발 (4) 스마트팜내 센서구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용 <ul style="list-style-type: none"> - RS485 MODBUS 기반의 스마트팜 통신 표준 제정 - 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 표준 개발 - 신규 표준을 적용한 레퍼런스 장비 개발

나. 목표 달성여부

구분	항목	목표치	달성치	달성도
1	스마트팜 표준화 확산 지원	15회	19회	127%
2	표준화 컨버터 개발	1건	1건	100%
3	자료 분석 장치 개발	1건	1건	100%
4	표준 개발	2건	2건	100%

다. 목표 미달성 시 원인(사유) 및 차후대책(후속연구의 필요성 등)

(1) 스마트팜 표준화 확산 지원

(가) 본 과제 초기 스마트팜 장비업체들의 표준화에 대한 이해도는 매우 낮은 상태이었으나 본 과제를 통해 다양한 워크숍과 간담회를 진행하여 표준에 대한 중요성과 인식을 제고

(나) 표준화를 위한 API 개발 가이드를 제공하여 기술 이전함으로써 표준전환의 계기를 마련

(2) 후속 연구를 통해 지속적인 표준 인식 제고 필요

(가) 초기 단계인 스마트팜 표준 기반 개발을 보다 촉진하기 위하여 주관사인 “한국농식품ICT융복합산업협회”의 지속적인 업체 교육 진행 필요

4장

4장 연구결과의 활용 계획 등

4장 연구결과의 활용 계획 등

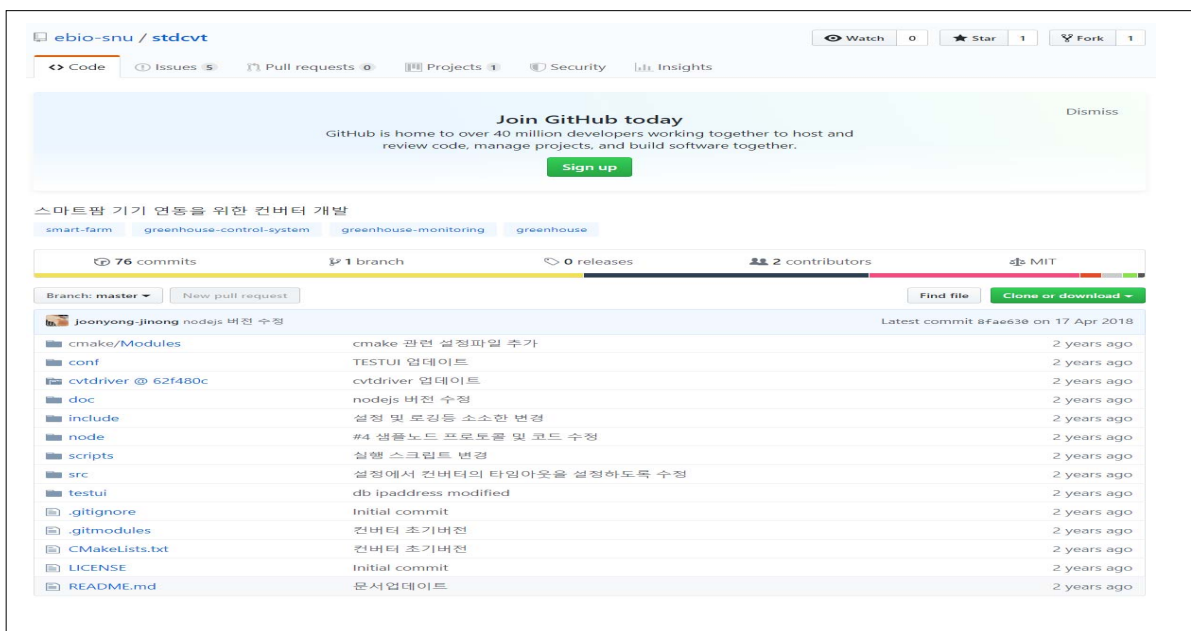
가. 스마트팜 표준의 확산

- (1) 본 과제에서 제작한 “농식품ICT가이드북” 를 발간하여 스마트팜 기업 홍보 및 표준 인식 제고 활동
- (2) 표준의 지속적 제정
 - (가) “스마트온실에서 이기종센서/구동기 및 제어기간상호연동을 위한 RS485기반 modbus 인터페이스(TTAK.KO-10.1044)” 가 2019년 KS표준으로 제정되었음
 - (나) 현재, 본 과제 내에서 “스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스” 추가 표준을 진행중임

나. 스마트팜 표준 기반 제품 개발 가이드의 지속 활용

(1) 오픈소스로 표준컨버터와 드라이버 API 개발 소스 공개

- (가) 표준컨버터 관련 소스를 모두 개발자 소스 공개 사이트인 github에 공개하여 각 개발업체가 활용할 수 있도록 하였음
 - ① 표준 컨버터 소스 공개 : <https://github.com/ebio-snu/stdcvt>
 - ② 컨버터 스펙을 기준으로 드라이버 개발을 돕기 위한 저장소 : <https://github.com/ebio-snu/cvtdriver>
 - ③ 협회에서는 본 사이트를 지속적으로 운영하여 소속 회원사들의 표준 전환 시 기술 참조가 가능하도록 지원



붙임. 참고문헌
(해당사항 없음)

별첨

[별첨 1] 연구개발보고서 초록

[별첨 2] 자체평가의견서

[별첨 3] 연구평가활용계획서

연구개발보고서 초록

과 제 명	(국문) 스마트 팜 표준화를 위한 기업의 기술 개발 지원(CMO)				
	(영문) Technology support corporate standards for Smart Farm(CMO)				
주관연구기관	사) 한국농식품ICT융복합산업협회		주 관 연 구	(소속)사) 한국농식품ICT융복합산업협회	
참 여 기 업	한국전자통신연구원 (1차년도 참여)		책 임 자	(성명) 정영환	
총연구개발비 (1,200,000천원)	계	1,200,000천원	총 연구 기간	2016-09-05 ~ 2019-09-04(3년)	
	정부출연 연구개발비	1,200,000천원	총 참 여 연 구 원 수	총 인 원	6명
	기업부담금	(해당없음)		내부인원	3명
	연구기관부담금	(해당없음)		외부인원	3명(1차년도 참여)

○ 연구개발 목표 및 성과

표준 통신 기반 스마트팜 서비스의 상호운용 및 호환성을 높이기 위하여, 기존 스마트팜 업체들에게 표준의 중요성에 대한 인식을 확대하고, 업체들의 레거시 제품들에 적용 가능한 표준 컨버터 개발하며 업체들의 참여를 촉진함으로써 스마트팜 표준화의 확산에 기여한다.

○ 연구내용 및 결과

(1) 스마트팜 표준화 확산 지원

- 스마트팜 장비 표준화 지원을 위한 설명회 및 간담회 개최
- 국제 농기자재 박람회 참여
- 스마트팜 업체 가이드 개발

(2) 표준화 컨버터 개발 : 스마트 팜 기기 연동을 위한 표준 API 설계 및 표준 컨버터 개발

- 스마트팜 장비 연동용 표준 컨버터에서 사용되는 드라이버 API 개발(기술이전)
- 스마트팜 장비 연동용 표준 컨버터 개발(기술이전)

(3) 자료 분석 장치 개발

- 클라우드 기반 컨버터 관리 및 데이터수집시스템 개발

(4) 스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용

- RS485 MODBUS 기반의 스마트팜 통신 표준 제정
- 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 표준 개발
- 신규 표준을 적용한 레퍼런스 장비 개발

○ 연구성과 활용실적 및 계획

(1) 연구성과 활용실적

지식재산권(출원)	지식재산권(등록)	표준화	기술이전
5건	2건 (+4건 예정)	1건 (+1건 진행중)	30건
소프트웨어 등록	사업화(제품화)	논문	고용창출
1건	3건 (매출액/국내 238백만원)	3건	3명

(2) 연구성과 활용계획

- 오픈소스로 표준컨버터와 드라이버 API 개발 소스 공개 (<https://github.com/ebio-snu/stdcvt>, <https://github.com/ebio-snu/cvtdriver>)
- 협회에서는 본 사이트를 지속적으로 운영하여 소속 회원사들의 표준 전환 시 기술 참조가 가능하도록 지원

자체평가의견서

1. 과제현황

		과제번호	816022-03		
사업구분	기술사업화사업				
연구분야	농림식품 기계·시스템 > 농업기계·시스템 > 농업기계·시스템		과제구분	단위	
사업명	기술사업화사업			주관	
총괄과제	스마트 팜 표준화를 위한 기업의 기술 개발 지원(CMO)		총괄책임자	정영환	
과제명	스마트 팜 표준화를 위한 기업의 기술 개발 지원(CMO)		과제유형	개발	
연구기관	(사)한국농식품ICT융복합산업협회		연구책임자	정영환	
연구기간	연차	기간	정부	민간	계
	1차연도	2016.09.05.-2017.09.04	400,000	-	400,000
연구비 (천원)	2차연도	2017.09.05.-2018.09.04	333,000	-	333,000
	3차연도	2018.09.05.-2019.09.04	467,000	-	467,000
	계	3년	1,200,000		1,200,000
참여기업	한국전자통신연구원(1차년도 참여)				
상대국	(해당없음)	상대국연구기관	(해당없음)		

※ 총 연구기간이 5차연도 이상인 경우 셀을 추가하여 작성 요망

2. 평가일 : 2019.10.22.

3. 평가자(연구책임자) :

소속	직위	성명
한국농식품ICT융복합산업협회	사무총장	정영환

4. 평가자(연구책임자) 확인 :

본인은 평가대상 과제에 대한 연구결과에 대하여 객관적으로 기술하였으며, 공정하게 평가하였음을 확약하며, 본 자료가 전문가 및 전문기관 평가 시에 기초자료로 활용되기를 바랍니다.

확약	
-----------	--

I. 연구개발실적

※ 다음 각 평가항목에 따라 자체평가한 등급 및 실적을 간략하게 기술(200자 이내)

1. 연구개발결과의 우수성/창의성

■ 등급 : (아주우수, 우수, √보통, 미흡, 불량)

○ 스마트팜 통신 표준의 개발

- 과제 초기 각 스마트팜 기업들의 표준에 대한 거부감을 극복하고 다양한 워크샵과 간담회를 통해 업계가 가장 많이 적용하고 있는 486 Modbus 통신에 대한 TTA 표준제정

2. 연구개발결과의 파급효과

■ 등급 : (아주우수, √우수, 보통, 미흡, 불량)

○ 오픈소스로 표준컨버터와 드라이버 API 소스를 공개

- <https://github.com/ebio-snu/stdcvt>, <https://github.com/ebio-snu/cvtdriver>

- 협회에서는 본 사이트를 지속적으로 운영하여 소속 회원사들의 표준 전환 시 기술 참조가 가능하도록 지원

○ 개발된 표준의 KS 제정

- 본 과제에서 개발한 TTA 표준(TTAK.KO-10.1044)이 2019년 12월 KS표준으로 제정됨 (ETRI에서 별도 연구로 진행)

3. 연구개발결과에 대한 활용가능성

■ 등급 : (아주우수, 우수, √보통, 미흡, 불량)

○ 농식품부의 표준화 정책 기여

- 표준화 컨버터 방식에서 나아가 현재 농식품부는 스마트팜 장비에 직접 표준을 적용하는 방식으로 정책과제를 진행중임

- 본 과제에서 개발된 표준화 API 가이드를 활용하여 업체들의 스마트팜 표준 적용 시 활용 가능 (일부 수정이 필요함)

4. 연구개발 수행노력의 성실도

■ 등급 : (아주우수, √우수, 보통, 미흡, 불량)

○ 협약시 제시한 연구개발 내용을 100% 충실히 이행함

○ 정량적 성과목표를 100% 이상 초과 달성함

- 초과달성 내역 : 특허출원 4건 → 5건, 학술발표 0건 → 2건(조기 달성), 홍보전시 1건 → 2건, 정책활용 2건 → 8건

5. 공개발표된 연구개발성과(논문, 지적소유권, 발표회 개최 등)

■ 등급 : (아주우수, √우수, 보통, 미흡, 불량)

- 특허 출원 : 5건, 특허 등록 : 2건 등록 (4건 예정)

- 표준화 : 1건 제정, 1건 진행중 / - 기술이전 : 30건

- 소프트웨어 등록 : 1건 / - 사업화 제품화/매출액 : 3건, 국내 238백만원

- 학술발표 : 2건 / - 논문 : 3건

II. 연구목표 달성도

구분	세부연구목표 (연구계획서상의 목표)	비중 (%)	달성도 (%)	자체평가
1	스마트팜 표준화 확산 지원	30	100	- 스마트팜 장비업체들을 대상으로 14회 워크숍 및 간담회를 개최 - <농식품ICT우수업체 가이드북>을 제작·배포하여 표준에 대한 관심과 인식을 제고함
2	표준화 컨버터 개발	30	100	- 이기종 장비의 연동을 위한 표준 API를 개발하여 15개 기업에 2회에 걸쳐 기술이전 진행 - 표준 컨버터 제작기술 - 표준 드라이버 제작기술
3	자료 분석 장치 개발	10	100	- 표준컨버터로부터 산출되는 센서 데이터의 시각화 서비스로 모니터링 시스템을 개발
4	스마트팜내 센서/구동기-제어기간 상호 운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용	30	100	- RS485 Modbus 기반 통신 표준 제정 - 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 표준 개발 - 스마트팜 기업들이 표준 제품 개발에 참조 가능한 레퍼런스 모델 개발
합계		100점		

III. 종합의견

1. 연구개발결과에 대한 종합의견

- 연구 초기 스마트팜 표준에 대한 거부감이 상당하였는데 본 과제를 통해 다양한 워크숍과 간담회를 진행하였고, 실제 표준 연동을 위한 표준API를 기술이전하면서 스마트팜 기업들이 표준에 대한 이해도를 제고하였음
- 또한, 농식품부 정책과도 연동하여 스마트팜에 표준 기반 제품개발이 되도록 정책 수립에 기반이 됨
- 무엇보다 본 과제의 결과물인 “RS485 MODBUS 기반의 스마트팜 통신 표준”을 활용하여 ETRI가 추가 노력을 하여 2019년 12월 KS표준(KS X 3267, 3268, 3269)으로 제정되었음

2. 평가시 고려할 사항 또는 요구사항

- 본 사업은 스마트팜 표준의 확산을 위해 스마트팜 기업들의 모임인 “한국농식품ICT융복합산업협회”가 실행한 사업임
- 스마트팜 표준에 대한 인식이 낮은 상황에서 표준 통신에 대한 의견을 모으고, 실제 TTA표준을 만들었고, 이를 활용한 레퍼런스 모델을 개발하는 등 많은 노력을 기울여 왔음

3. 연구결과의 활용방안 및 향후조치에 대한 의견

- 이기종 장비의 연동을 위해 표준화컨버터를 개발하였으나 현재 농식품부 정책은 스마트팜 장비에 표준을 직접 적용하는 것을 목표로 하고 있음
- 따라서, 표준화 컨버터의 개발 결과물을 바로 확산하기 보다는 개발 과정에서 산출된 표준API 가이드와 통신표준(TTA표준을 제정 -> KS표준으로 확대됨)을 기반으로 스마트팜 표준장비 개발로 성과의 활용이 필요함

IV. 보안성 검토

(해당사항 없음)

※ 보안성이 필요하다고 판단되는 경우 작성함.

1. 연구책임자의 의견

2. 연구기관 자체의 검토결과

[별첨 3]

연구성과 활용계획서

1. 연구과제 개요

사업추진형태	<input type="checkbox"/> 자유응모과제 <input checked="" type="checkbox"/> 지정공모과제	분 야	기술사업화지원사업	
연구과제명	스마트 팜 표준화를 위한 기업의 기술개발 지원(CMO)			
주관연구기관	사)한국농식품ICT융복합산업협회	주관연구책임자	정영환	
연구개발비	정부출연 연구개발비	기업부담금	연구기관부담금	총연구개발비
	1,200,000천원			1,200,000천원
연구개발기간	2016. 09. 05 - 2019. 09. 04(36개월)			
주요활용유형	<input type="checkbox"/> 산업체이전 <input checked="" type="checkbox"/> 교육 및 지도 <input checked="" type="checkbox"/> 정책자료 <input type="checkbox"/> 기타() <input type="checkbox"/> 미활용 (사유:)			

2. 연구목표 대비 결과

당초목표	당초연구목표 대비 연구결과
① 스마트팜 표준화 확산 지원	<ul style="list-style-type: none"> - 스마트팜 장비업체들을 대상으로 14회 워크숍 및 간담회를 개최하였고, - <농식품ICT융복합우수업체> 가이드북을 제작·배포하여 표준에 대한 관심과 인식을 제고함
② 표준화 컨버터 개발	<ul style="list-style-type: none"> - 이기종 장비의 연동을 위한 표준API를 개발하여 15개 기업에 2회에 걸쳐 기술이전 진행 - 표준 컨버터 제작기술 - 표준 드라이버 제작기술
③ 자료 분석 장치 개발	<ul style="list-style-type: none"> - 표준 컨버터로부터 산출되는 센서 데이터의 시각화 서비스로 모니터링 시스템을 개발
④ 스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용	<ul style="list-style-type: none"> - RS485 Modbus 기반 통신 표준 제정 - 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 표준 개발 - 스마트팜 기업들이 표준 제품 개발에 참조가능한 레퍼런스 모델 개발

* 결과에 대한 의견 첨부 가능

3. 연구목표 대비 성과

성과 목표	사업화지표										연구기반지표									
	지식 재산권			기술 실시 (이전)		사업화					기술 인증	학술성과				교육 지도	인력 양성	정책 활용-홍보		기 타 (타 연 구 활 용 등)
	특 허 출 원	특 허 등 록	품 종 등 록	건 수	기 술 료	제 품 화	매 출 액	수 출 액	고 용 창 출	투 자 유 치		논문		학 술 발 표	정 책 활 용			홍 보 전 시		
												SC I	비 SC I							
단위	건	건	건	건	백 만 원	백 만 원	백 만 원	백 만 원	명	백 만 원	건	건	건	건	명	건	건			
가중치																				
최종목표	4	2		30		3			3			5			3	2	1			
기간내 달성실적	5	2		30		3			3			3	2		0	8	2	1		
달성율(%)	125	100		100		100			100			60			0	400	200			

당초목표	당초연구목표 대비 연구결과
① 스마트팜 표준화 확산 지원	<ul style="list-style-type: none"> - 스마트팜 장비업체들을 대상으로 14회 워크숍 및 간담회 개최, - 농식품ICT우수업체 가이드북을 제작 배포하여 표준에 대한 관심과 인식을 제고함
② 표준화 컨버터 개발	<ul style="list-style-type: none"> - 이기종 장비의 연동을 위한 표준API를 개발하여 15개 기업에 2회에 걸쳐 기술이전 진행 - 표준 컨버터 제작기술 - 표준 드라이버 제작기술
③ 자료 분석 장치 개발	<ul style="list-style-type: none"> - 표준 컨버터로부터 산출되는 센서데이터의 시각화 서비스로 모니터링 시스템을 개발
④ 스마트팜내 센서/구동기-제어기간 상호운용 기능구조 및 통신 프로토콜 국내 표준개발 및 적용	<ul style="list-style-type: none"> - RS485 Modbus 기반 통신 표준 제정 - 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 표준 개발 - 스마트팜 기업들이 표준 제품 개발에 참조 가능한 레퍼런스 모델 개발

4. 핵심기술

구분	핵심기술명
②	- 표준 컨버터 제작 기술 및 표준 드라이버 제작 기술
③	- 클라우드 기반 컨버터 관리 및 데이터 수집시스템 개발
④	- RS485 Modbus 기반 통신 표준 제정 - 스마트팜 장비 연동을 위한 디바이스 드라이버 어플리케이션 프로그래밍 인터페이스 표준 개발

5. 연구결과별 기술적 수준

구분	핵심기술 수준					기술의 활용유형(복수표기 가능)				
	세계 최초	국내 최초	외국기술 복제	외국기술 소화흡수	외국기술 개선개량	특허 출원	산업체이전 (상품화)	현장애로 해결	정책 자료	기타
②의 기술					V	V	V	V		
③의 기술					V					
④의 기술		V				V			V	

6. 각 연구결과별 구체적 활용계획

핵심기술명	핵심기술별 연구결과 활용계획 및 기대효과
②의 기술	<ul style="list-style-type: none"> ○ 오픈소스로 표준컨버터와 드라이버 API 소스를 공개 <ul style="list-style-type: none"> - https://github.com/ebio-snu/stdcvt, https://github.com/ebio-snu/cvtdriver - 협회에서는 본 사이트를 지속적으로 운영하여 소속 회사들의 표준 전환 시 기술 참조가 가능하도록 지원 ○ 농식품부의 표준화 정책 기여 <ul style="list-style-type: none"> - 표준화 컨버터 방식에서 나아가 현재 농식품부는 스마트팜 장비에 직접 표준을 적용하는 방식으로 정책과제를 진행중임 - 본 과제에서 개발된 표준화 API 가이드를 활용하여 업체들의 스마트팜 표준 적용 시 활용 가능(일부 수정이 필요함)
③의 기술	<ul style="list-style-type: none"> ○ 협회 회원사들이 표준화 컨버터로 개발한 제품의 원활한 통신 테스트용으로 활용
④의 기술	<ul style="list-style-type: none"> ○ 농식품부의 스마트팜 표준화에 기반이 되는 KS표준으로 제정됨에 따라 이를 활용한 제품 개발 가능 <ul style="list-style-type: none"> - 본 과제에서 개발한 TTA 표준(TTAK.KO-10.1044)이 2019년 12월 KS표준으로 제정됨(ETRI에서 별도 연구로 진행)

7. 연구종료 후 성과창출 계획

성과목표	사업화지표										연구기반지표								
	지식 재산권			기술실시 (이전)		사업화					기술인증	학술성과			교육지도	인력양성	정책 활용-홍보		기타 (타 연구활동등)
	특허출원	특허등록	품종등록	건수	기술료	제품화	매출액	수출액	고용창출	투자유치		논문		학술발표			정책활용	홍보전시	
												SCI	비SCI						
단위	건	건	건	건	백만원	건	백만원	백만원	명	백만원	건	건	건	건	명				
가중치																			
최종목표	4	2		30		3			3			5			3	2	1		
연구기간내 달성실적	5	2		30		3			3			3		2	0	8	2	1	
연구종료후 성과창출 계획	2	4		40		4			7					7	4		1		

8. 연구결과의 기술이전조건(산업체이전 및 상품화연구결과에 한함)

핵심기술명 ¹⁾	스마트팜 기기 연동을 위한 표준 컨버터 개발 기술		
이전형태	<input checked="" type="checkbox"/> 무상 <input type="checkbox"/> 유상	기술료 예정액	천원
이전방식 ²⁾	<input type="checkbox"/> 소유권이전 <input type="checkbox"/> 전용실시권 <input checked="" type="checkbox"/> 통상실시권 <input type="checkbox"/> 협의결정 <input type="checkbox"/> 기타()		
이전소요기간	연구개발기간 중 완료	실용화예상시기 ³⁾	2019.10
기술이전시 선행조건 ⁴⁾	업체에 기술이전 계약 체결		

핵심기술명 ¹⁾	표준 컨버터용 디바이스 드라이버 개발기술		
이전형태	<input checked="" type="checkbox"/> 무상 <input type="checkbox"/> 유상	기술료 예정액	천원
이전방식 ²⁾	<input type="checkbox"/> 소유권이전 <input type="checkbox"/> 전용실시권 <input checked="" type="checkbox"/> 통상실시권 <input type="checkbox"/> 협의결정 <input type="checkbox"/> 기타()		
이전소요기간	연구개발기간 중 완료	실용화예상시기 ³⁾	2019.10
기술이전시 선행조건 ⁴⁾	업체에 기술이전 계약 체결		

핵심기술명 ¹⁾	스마트팜용 비표준센서장치의 데이터측정장치		
이전형태	<input type="checkbox"/> 무상 <input checked="" type="checkbox"/> 유상	기술료 예정액	1,000천원
이전방식 ²⁾	<input type="checkbox"/> 소유권이전 <input type="checkbox"/> 전용실시권 <input checked="" type="checkbox"/> 통상실시권 <input type="checkbox"/> 협의결정 <input type="checkbox"/> 기타()		
이전소요기간	3개월	실용화예상시기 ³⁾	2020.3
기술이전시 선행조건 ⁴⁾	스마트팜 제품 개발 업체 중 센서노드 제작 가능 업체		

주 의

1. 이 보고서는 농림축산식품부에서 시행한 기술사업화지원의 연구보고서입니다.
2. 이 보고서 내용을 발표하는 때에는 반드시 농림축산식품부에서 시행한 기술사업화지원사업의 연구 결과임을 밝혀야 합니다.
3. 국가과학기술 기밀유지에 필요한 내용은 대외적으로 발표 또는 공개하여서는 안됩니다.