개발완료보고서 [스마트 팩토리]

0. 작성 시 주의사항

※아래의 작성 양식(제출분량, 폰트, 크기, 줄 간격 등)을 미준수 시 서류 평가의 감정요인됨

※ 제출 분량 : A4 용지 상세내용 포함 30 page 이내
※ 작성 양식 (폰트 : 맑은 고딕 / 폰트 크기 : 10pt / 자간 : 0% / 장평 : 100% / 줄 간격 : 130%)
※ 제출 포맷 : pdf

1. 팀 정보

팀명	리더스	팀장	윤주영
팀원	이승헌	팀원	김동현
팀원	홍준표	팀원	

2. 개발완료보고서

작품명 : 자가 진단 시스템

1. 개요

1.1. 작품 개요

● **스마트 팩토리**

스마트 팩토리란 제품의 기획, 설계, 생산, 유통, 판매 등 전 생산과정을 ICT로 통합하여 최소의 비용과 시간으로 고객 맞춤형 제품을 생산하는 진보된 공장을 의미한다. 이러한 전 과정에서 IoT, AI, 빅데이터 등으로 통합하여 자동화와 디지털화를 구현하는 것이 기존 공장 자동화와 차별되는 요소이다.

스마트 팩토리는 전공정, 후공정 모두 데이터를 자유롭게 연계할 수 있어 총체적인 관점에서 최적화를 이룰 수 있으며, 모든 설비나 장치가 무선통신으로 연결되어 정보를 주고받고, 모든 공정을 실시간으로 모니터링, 분석, 판단해 최적의 생산환경을 만들 수 있다.

그 원리를 간단하게 설명하자면 공장 곳곳에 IOT 센서와 카메라를 부착하고 이 센서들이 현장의 무수히 많은 크고 작은 모든 데이터(불량 등)를 수집한다. 각 기업들은 구축한 데이터 인프라를 통해 이 데이터들을 저장하고 분석하여 왜 어디에 불량이 발생했는지, 어떤 공정(기계)에 이상징후가 보이는지 분석을 한다. 그를 통해 Error를 제어하고 보수를 한다.

이렇듯 스마트 팩토리에는 모든 상황에서 데이터를 분석하고 그 동안 몰랐던 장애, 불량의 원인을 알아낼 수 있는 등 많은 장점이 있다.

우리 팀에서는 그 중 데이터 분석을 통한 불량의 원인 검출에 초점을 맞추었다. 영상처리기술을 통해 불량률을 실시간으로 수치화하고, 그를 통해 관리자들이 실시간으로 소통하고 공정을 개선하는데 도움을 줄 소프트웨어를 개발하고자 한다.

● **자가 진단 시스템의 필요성**

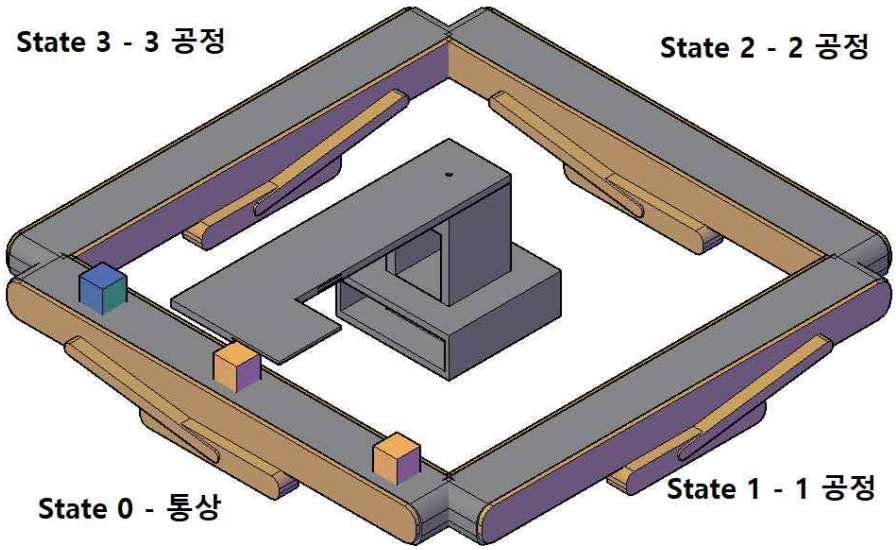
현재 스마트 팩토리가 많은 관심을 받고 있지만 대부분의 구성원들이 스마트 팩토리에 대한 이해도가 떨어지다 보니 시스템을 갖춰 놓고도 운영에 어려움을 겪는 사업장이 많다고 한다. 따라서 무턱대고 스마트 팩토리를 구축했다가는 오히려 손해를 보는 결과를 초래할 수 있다.

우리는 이러한 혼란을 줄이기 위해 전문가가 아닌 누구라도 웹에서 불량률을 쉽게 확인 할 수 있도록 수치화 하여 나타내었다. 영상처리 기술을 기반으로 AI를 통해 데이터를 분석하여 이상이 있는 공정을 쉽게 검출할 수 있도록 하여, 관리자가 한눈에 판단할 수 있고 웹에서 실시간으로 관리자들과 실시간으로 소통하여 간편하고 대처를 신속히 할 수 있도록 시스템을 구현하고자 한다.

이 시스템을 통해 전문인력이 부족하다는 스마트 팩토리의 문제점을 보완할 수 있다.

1.2. 개발 목표

<자가 진단 시스템>



[시스템을 적용하기 위해 설계한 가상 스마트 팩토리 공정]

공장에서 제품 생산 시 제조 과정에서 다양한 원인으로 인해 불량품이 생성된다. 불량 검출을 인간이 할 수 있지만 신체 피로 등 기타 원인으로 인해 정확한 검출이 불확실하다. 그러므로 인공지능을 이용하여 외부 요인에 영향을 받지 않고 정확하게 불량공정을 검출하는 것을 목표로 한다.

현재 대부분의 생산라인에서 컨베이어 벨트를 이용한다. 이를 토대로 가상의 생산라인을 제작하여 실제 현장을 연출한다.

아두이노에 적외선 센서를 이용하여 물체의 위치를 파악한다. 적절한 위치에 도달하면 라즈베리파이에 신호를 주어 촬영을 하고 인공지능이 학습된 데이터와 비교하여 불량을 검출한다.

이를 위해 먼저 공정에서 생산되는 정상품의 데이터가 필요하다. 사전에 각각 200~300장의 정삼품과 불량품을 다양한 위치에서 촬영한 데이터를 이용하여 머신러닝을 실시한다.

사전 작업을 통해 판별이 가능한 인공지능을 구축한 후, 시스템을 적용한다.

통상시엔 제품 출하 직전의 공정을 검출하고 있다. 이 상황을 'state 0'으로 설정한다. 'state 0'에서 불량이 3회 초과 검출되면 웹에 '자가진단 시스템 시작' 알림과 함께 공정이 중단되고 자동으로 시스템이 시작된다.

먼저 'state 1' 상태가 된다.
먼저 타워형 프레임이 첫 번째 공정을 검출하기 위해 90° 회전한다. 해당 공정만 가동되고 공정을 촬영하여 인공지능으로 불량을 판별한다. 그 후 수집한 데이터를 DB에 전송하고 웹에 수치화, 시각화하여 나타낸다. 해당 과정을 'state 2', 'state 3'에도 반복하여 모든 공정을 공정별로 분석하여 상태를 알리고 불량 원인 공정을 찾아내어 관리자가 효율적으로 관리할 수 돕는 것을 목표로 한다.

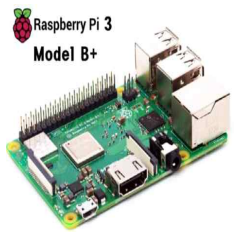
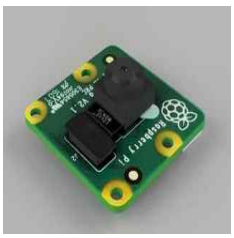
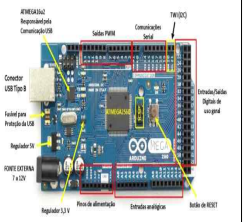
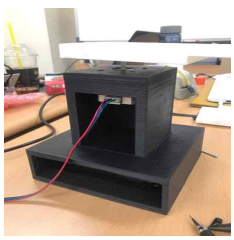
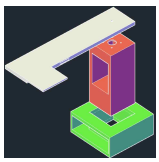
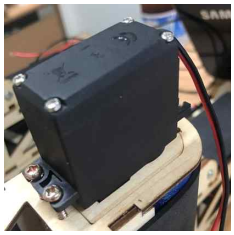
<전제 조건>
1) 표본이 많을수록 정확도가 상승하지만 제작시간과 제품사양을 고려하여 200~300장 정도의 표본으로 제작
2) 실제 공정에서는 불량률이 1%이하가 되어야 실용성이 있는 라인이라는 피드백을 받았으나 시간과 제품사양을 고려하여 구현을 하기 위해 5%로 설정한다.
3) 실제 공정에서는 처음 공정에서 마지막 공정까지 한 번에 진행되는데, 현실적으로 구현하기 어려

워 각 공정에서 도색, 절단 등의 개별 작업하는 것으로 가정. 제품의 공정 별 이동은 생략하고 시연한다.
4) 불량률(%) 이용하는 것이 바람직 하지만 시연을 위해서 횟수(n)로 수정, 필요시 불량률로 대체 가능하도록 plan B 준비

2. 개발 환경 설명

2.1. Hardware 구성

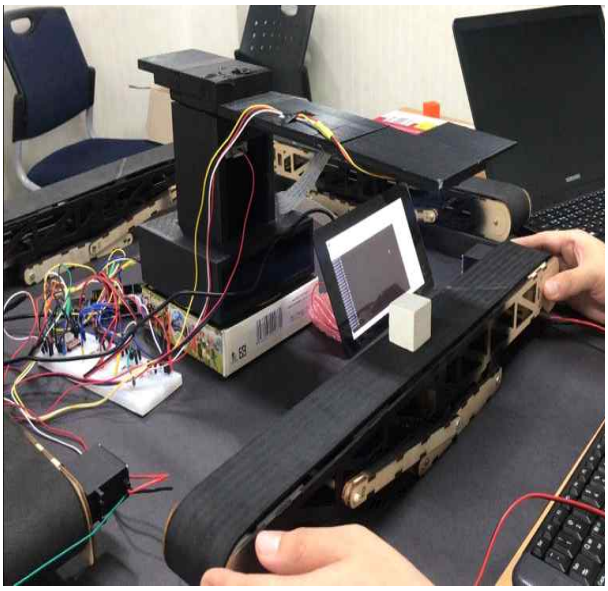
2.1.1 제작 부품

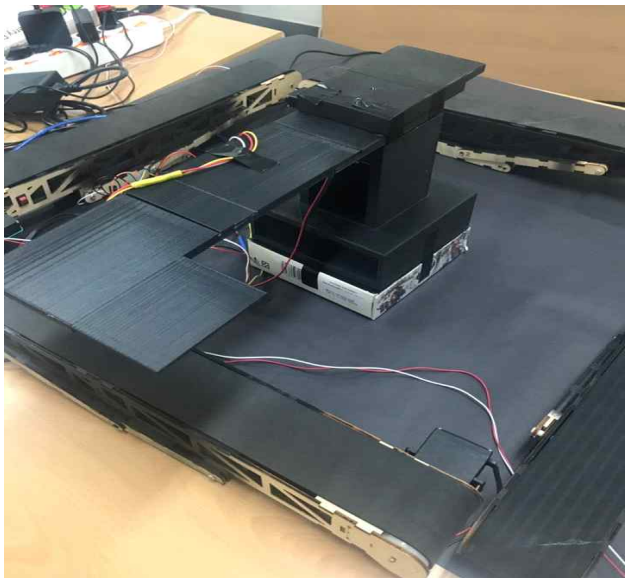
	**raspberry pi 3 b+**		**raspberry pi Camera Module V2.1**
	**ARDUINO 적외선 거리측정센서(2Y0A21)**		**Arduino Mega 2560 (R3)**
	**스테핑 모터 42HB34F08AB**		**스테핑 모터 드라이버 모듈 A4988-Module +전열용 방열판**
	**3D 프린터로 제작한 중심 지지부와 회전부**  <CAD 설계 모습>		
	**컨베이어벨트-중형 x 4**		**컨베이어 벨트 회전용 서보모터 x 4**

	**그 외** **열 수축 튜브(선 보호용)** **0.65mm 연선 (색상별로)** **브레드보드** **12V 어댑터, 9V 어댑터** **각종 저항, NPN 트랜지스터**

제품명	사양	
raspberry pi 3 b+	Chip	Broadcom BCM2837 SoC
	Core architecture	Quad-core ARM Cortex-A7
	Memory	1GB LPDDR2
	Operating System	Boots from Micro SD card, running a version of the Linux operating system
Arduino Mega 2560 (R3)	microcontroller	ATmega2560
	Input voltage	7-12V
	Digital I/O Pins	54 (14 PWM outputs)
	Analog Inputs	16
	Flash Memory	256k
	Clock Speed	16Mhz
raspberry pi Camera Module V2.1	Dimensions	8.5 x 8.5 x 5mm
	Height	5mm
	Length	8.5mm
	Maximum Frame Rate Capture	30fps
	Maximum Operating Temperature	70°C
	Maximum Supported Resolution	2592 x 1944
	Minimum Operating Temperature	-30°C
	Number of Channels	1
	Supported Bus Interfaces	I2C
	Supported Video Ports	DVI
	Width	8.5mm
ARDUINO 적외선 거리측정센서(2Y0A21)	Voltage	DC4.5~5.5
	Current	40mA
	Package size	29.5 × 13.0 × 13.5mm
	Short measuring cycle	16.5ms
	Distance measuring range	10~80cm
스테핑 모터 42HB34F08AB	Step angle	1.8 +- 5%
	Rated voltage	4.96 V
	Current	0.8A/Phase
	Resistance	6.2 +- 10%/Phase
	Inductance	10 +- 20%/Phase
	Holding torque	2.4kg.cm
Servo Motor	Speed	0.19 sec/60° at 6V
	Torque	4.1 Kg-cm at 6V
	Size	40.4 x 19.9 x 39.5 mm
	Weight	40.4 g

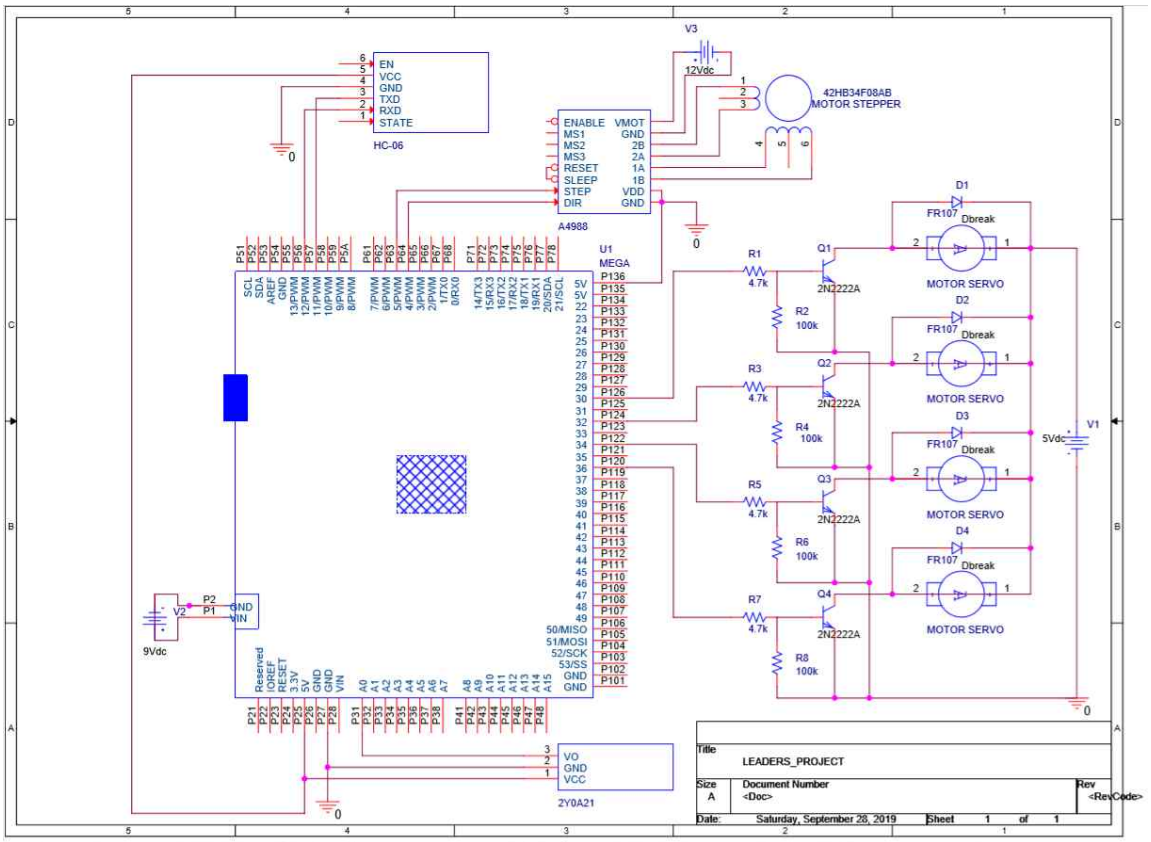
2.1.2 Hardware 전체 구성



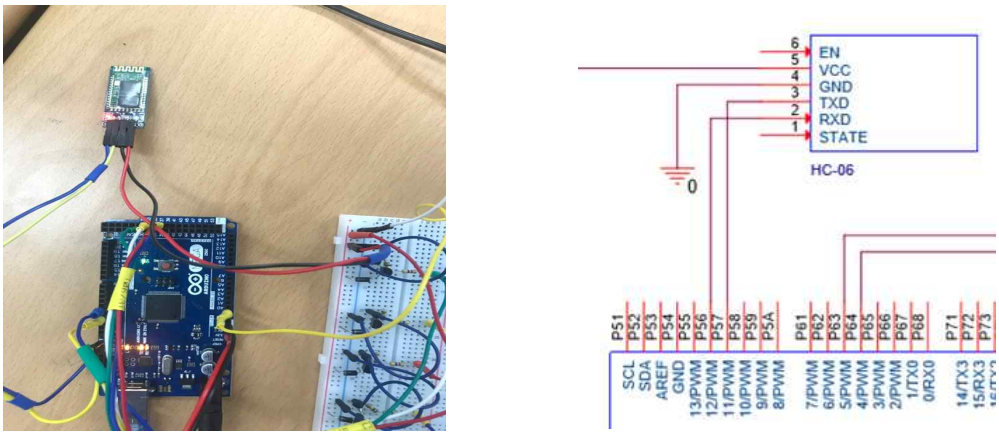


2.1.3 Hardware 구성 세부 설명

<아두이노 전체 구성 회로도(OrCAD)>



1) 블루투스 모듈(HC-06) 구성 회로도

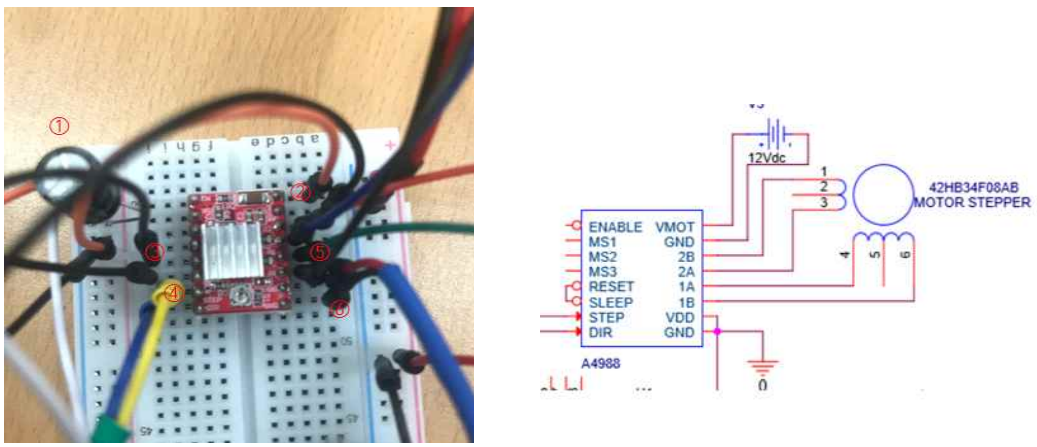


<raspberry pi 3 b+ 보드와 통신하기 위한 블루투스 모듈 HC-06 회로도>

블루투스 모듈 HC-06의 핀을 Arduino MEGA 2560 보드의 핀과 연결
(VCC핀 - 5V , GND - GND , TXD - 11 , RXD - 12)
위와 같이 연결을 하면 raspberry pi 3 b+와 시리얼 통신이 가능하게 된다.

<용도>
시스템 구동 시 적외선 센서가 작동을 할 경우 신호를 raspberry pi에 전달하여 카메라로 촬영
불량률 진단 공정 변경 시 스텝모터를 회전시키는 신호를 Arduino로 전달

2) 스테핑 모터 구동 회로



<스테핑 모터 드라이버(A4988)을 사용하여 스테핑 모터(42HB34F08AB)를 구동>

① 스테핑 모터를 구동하기 위하여 외부 12V 어댑터를 사용하여 ②에 위치한 VMOT핀과 GND핀에 연결한 뒤 100㎌의 콘덴서와 함께 결선하였다.
(콘덴서는 초기 스테핑 모터 구동시 동력을 향상시키고 소음을 줄여주기 위해서 사용하였다.)

③ RESET과 SLEEP은 사용하지 않는다

④ STEP과 DIR핀은 아두이노에서 코딩을 통해 스테핑 모터의 회전 방향과 각도를 결정할 수 있다.

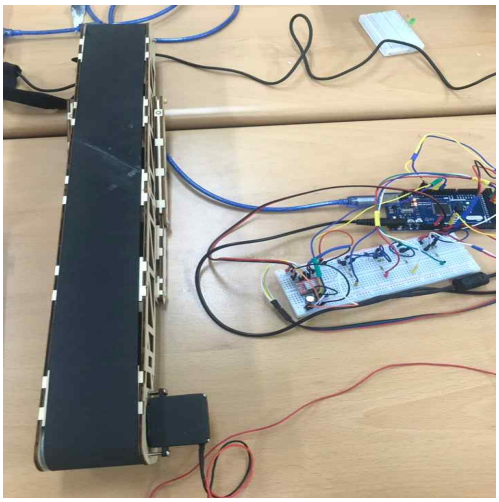
⑤ 스테핑 모터에서 빨간 선을 2B, 파란 선을 2A, 초록 선을 1A, 검정 선을 1B에 연결 하여 구동하고

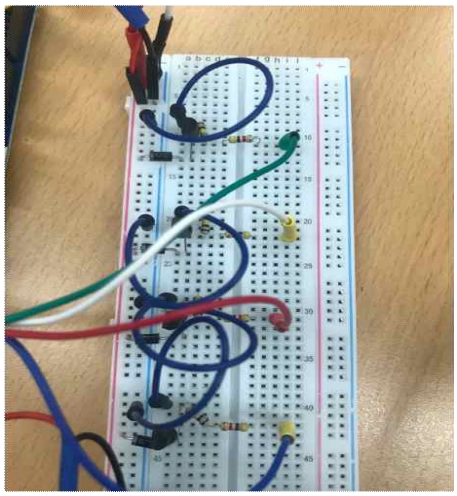
자 한다.

⑥ 스테핑 모터 드라이버의 구동을 위해 5V의 전압을 인가해 주어야 한다. 그렇기 때문에 VDD핀을 아두이노 보드의 5V핀, GND핀을 아두이노 보드의 GND핀과 결선 하였다.

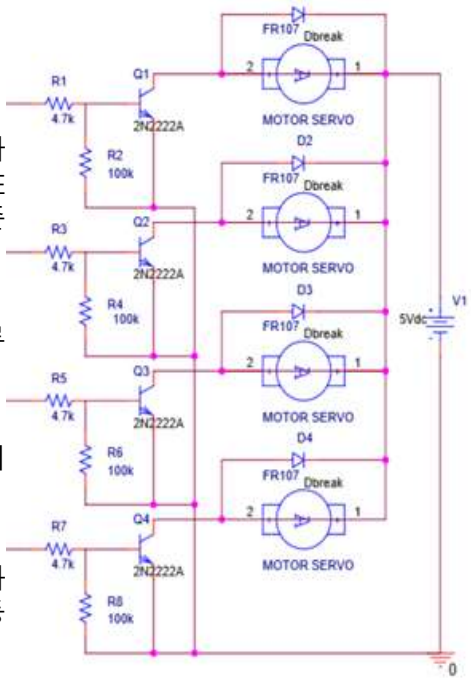
마지막으로 가변저항을 조절하여 구동전류를 적절하게 조절하였다.

3) 컨베이어 벨트 서보모터 구동회로





<컨베이어 벨트 4개의 서보모터를 각각 구동하기 위한 회로>



트렌지스터는 전류, 전압 증폭의 기능을 가지는 스위칭 소자이다. 회로에 사용한 2N2222A NPN트렌지스터의 경우 BASE 단자에 전류가 들어오면 증폭하여 Emitter 단자로 흘려 보내준다.

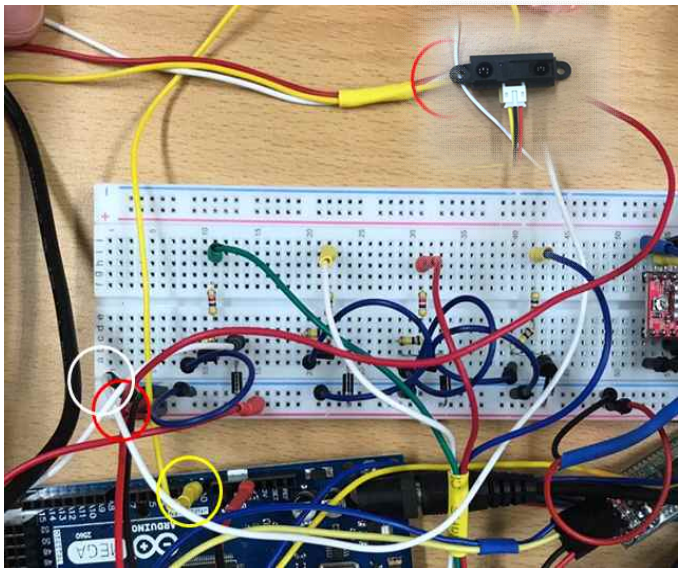
R1~R8의 저항은 트렌지스터 BASE 단자에 흘러 들어가는 전류를 조정하고 소자를 보호하는 역할을 한다.

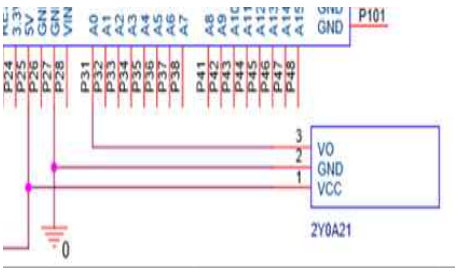
다이오드(FR107)는 역방향 전류로 인한 서보 모터 소손을 방지하기 위하여 설치한다.

서보 모터 한 개를 제어하기 위해서는 6V의 전압으로 가능하지만 4개를 동시에 제어하고 스테핑 모터 드라이버에 전압 등을 고려하여 9V의 어댑터를 사용하였다.

Arduino MEGA 2560 보드의 30, 33, 34, 37번 디지털 핀을 사용하여 0번~3번 공정에 해당하는 서보 모터를 각각 연결하여 신호를 주면 트랜지스터가 작동하며 모터를 제어하였다.

4) 적외선 거리측정 센서 구동 회로





적외선 센서의 각각의 선을 Arduino MEGA 2560 보드의 핀에 연결
(노란색 선 - A0 , 빨간색 선 - 5V , 검은색(+흰색) - GND)

발사된 적외선이 되돌아오며 거리 측정(시리얼 모니터로 cm 단위로 확인 가능)

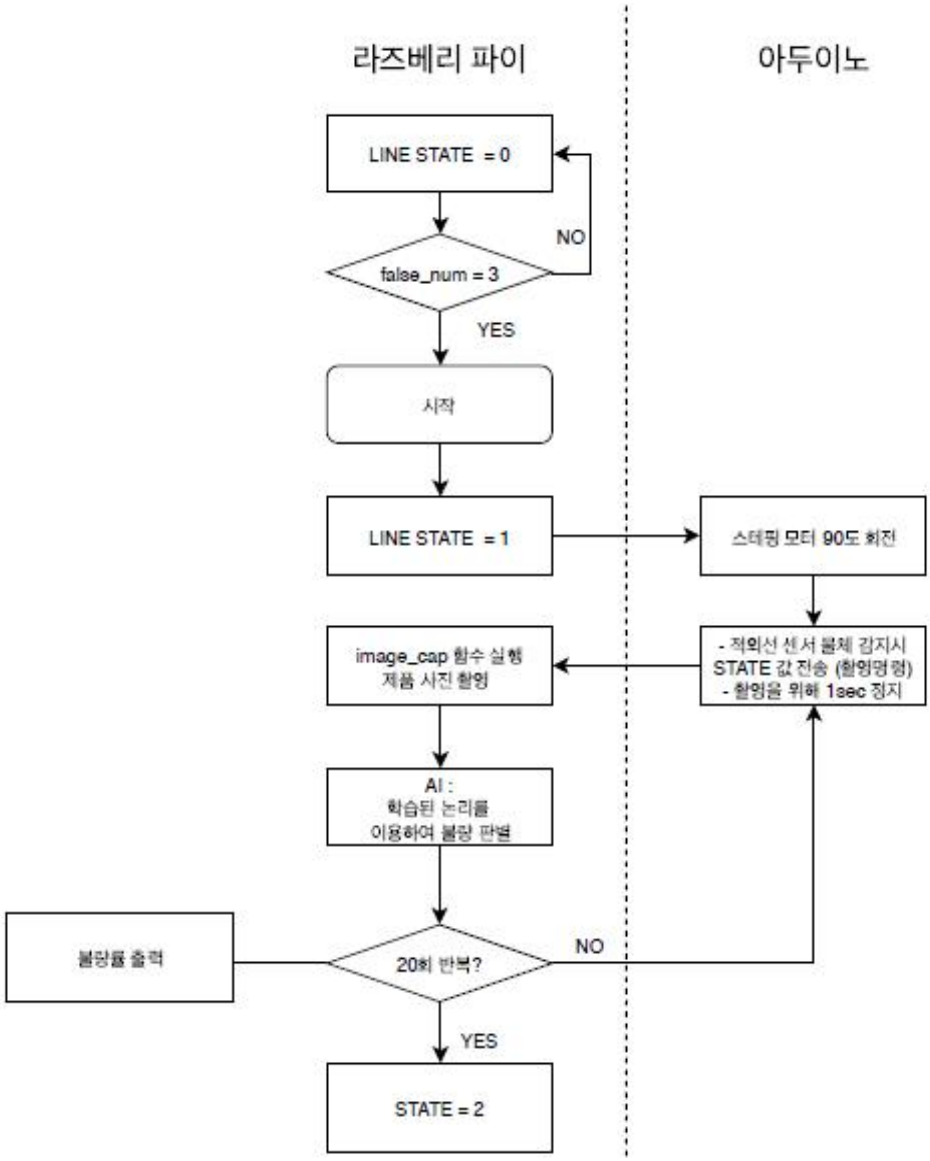
공정에 제품이 지나갈 때 측정값이 낮아지게 됨
-> 거리 측정값이 일정 수준이 되었을 때 컨베이어 벨트를 멈추고 촬영을 하는 용도로 사용

2.2. Software 구성

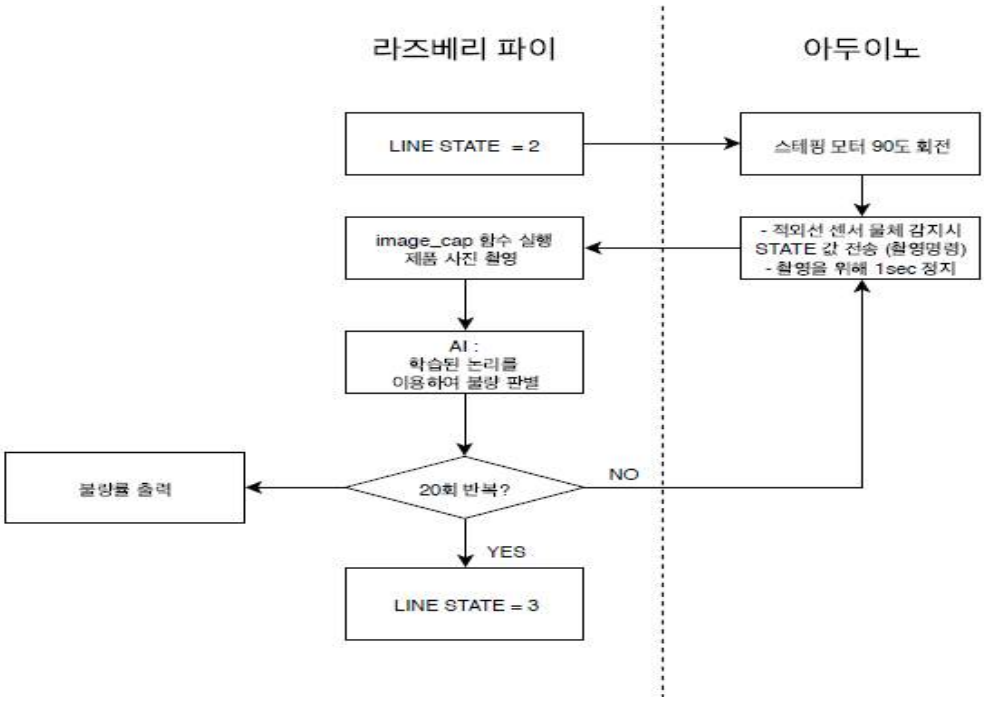
2.3. Software 설계도 (흐름도 및 클래스 다이어그램 등 (개발언어에 따라 선택))

: 시스템은 case : 0 (통상)에서 1, 2, 3 순차적으로 실행 후 종료

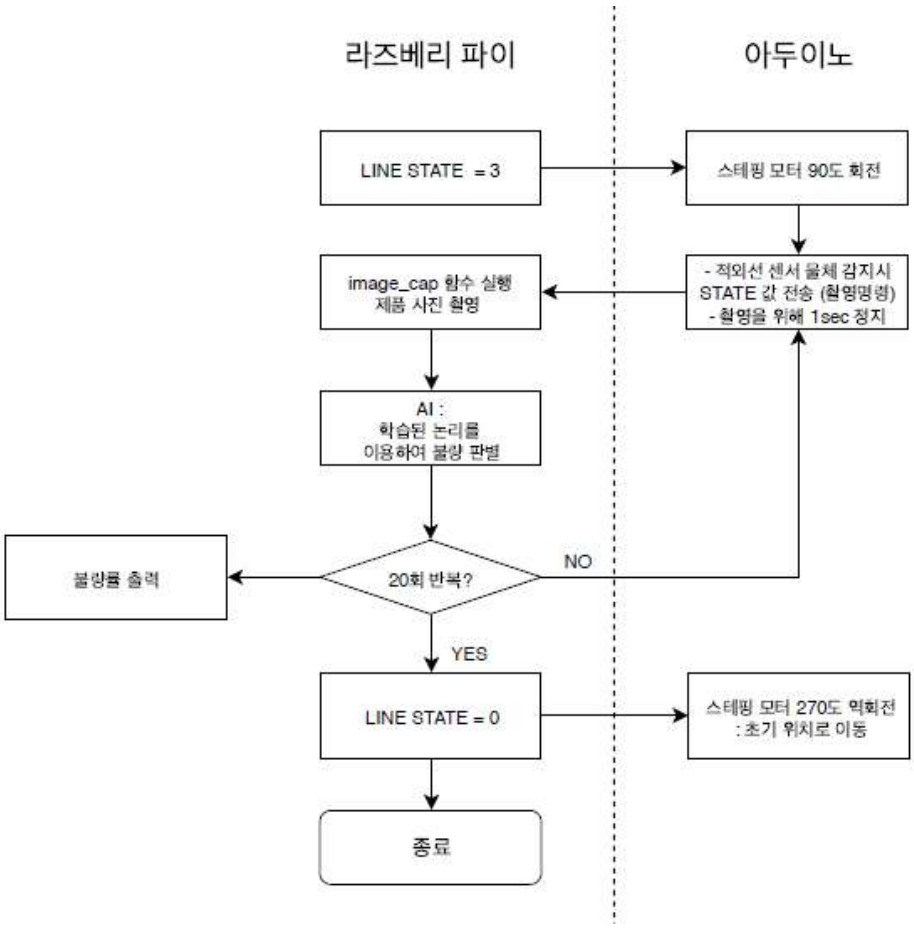
2.3.1. Line_state = 0 to 1, 2 / 자가진단 시스템 시작



2.3.2. Line_state = 2 to 3



2.3.3. Line_state 3 = 0 / 자가진단 시스템 종료



2.4. Software 기능 (필요 시 알고리즘 설명 포함)

 # 핵심 기술 -> 영상처리, 딥-러닝(CNN)

1) 영상처리 (OpenCV)

- 모델 학습과 예측의 정확도를 향상시키기 위해 학습시킬 이미지 데이터들을 제품만 나오도록 전처리과정으로 활용
- 기존의 RGB 색모델에서 HSV 색 모델로 변환한 후, 전경과 배경을 분리시켜 제품만 촬영
- 이미지의 크기를 리사이즈하여 데이터 저장

```
image = cv2.imread(imagePath)                                  # 이미지 데이터 읽어오기
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)                   # BGR -> HSV 색변환
img_mask = cv2.inRange(hsv, (0,10,170), (255,255,255))         # 마스크
img_result = cv2.bitwise_and(image, image, mask=img_mask) # 마스크 적용 -> 제품만 촬영
image = cv2.resize(img_result, (32, 32))                       # 이미지 (32,32)로 리사이즈
```

-> 예측모델에서 동일하게 적용됨

2) Keras를 활용한 CNN 모델 학습

- 전처리 과정을 거친 후, 학습용 데이터와 검증용 데이터로 분리

- **모델 생성**
  models.Sequential()
- **합성곱층 생성 , 출력채널 : 8 , kernel_size = (3,3)**
  model.add(Conv2D(8, (3, 3), padding="same", input_shape=(32, 32, 3)))
- **활성화함수 지정**
  model.add(Activation("relu"))
- **풀링층**
  model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
- **합성곱층 생성 , 출력채널 : 16 , kernel_size = (3,3)**
  model.add(Conv2D(16, (3, 3), padding="same"))
- **활성화함수 지정**
  model.add(Activation("relu"))
- **풀링층**
  model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
- **합성곱층 생성 , 출력채널 : 32 , kernel_size = (3,3)**
  model.add(Conv2D(32, (3, 3), padding="same"))
- **활성화함수 지정**
  model.add(Activation("relu"))
- **풀링층**
  model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
- **데이터를 펴기**
  model.add(Flatten())
- **전결합층, 출력 채널 9 설정**
  model.add(Dense(9))

- **출력층**

  model.add(Activation("softmax"))

- **최적화 모델 Adam 적용**

  opt = Adam(lr=1e-3, decay=1e-3 / 50)

- **손실함수 : 크로스엔트로피 / 최적화 : categorical_crossentropy / 정확도**

  model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])

2.5. 개발환경 (언어, Tool, 사용시스템 등)

• 개발 언어 : Python , C언어

• 개발 tool : Arduino, VNC Viewer, OrCAD, AutoCAD

3. 개발 프로그램 설명 (최대한 자세하게 기술)

3.1. 파일 구성

(1) 아두이노

- leaders.c

(2) 라즈베리파이 (python)

```
├─── ml_project
|      ├─── data
|      |      ├─── Default [483 entries]
|      |      ├─── line0_True [492 entries]
|      |      ├─── line0_False [476 entries]
|      |      ├─── line1_True [513 entries]
|      |      ├─── line1_False [493 entries]
|      |      ├─── line2_True [502 entries]
|      |      ├─── line2_False [499 entries]
|      |      ├─── line3_True [467 entries]
|      |      ├─── line3_False [493 entries]
├─── model
|      ├─── activity.model          : 훈련 Keras 모델 저장 파일
|      └─── lb.pickle               : 고유한 클래스 레이블을 포함하는 직렬화된 레이블 이진화기
├─── output                         : 비디오 분류 결과를 저장할 위치
├─── predict.py                     : 모델 예측 (실시간 영상 예측)
├─── train.py                       : 모델 훈련
└─── auto.py                        : 통합 시스템 (통신, 데이터 수집, 예측)
```

3.2. 함수별 기능

3.2.1. 아두이노

1)적외선 센서 거리 : A0에 입력되는 적외선 센서의 아날로그 값을 volt 값으로 변환하고 사용자가 알아보기 쉽도록 거리(distance)로 나타내는 함수.

```
volt = map(analogRead(A0), 0, 1023, 0, 5000);
distance = (27.61 / (volt - 0.1696)) * 1000
```

2)C_time(int x) : 라즈베리파이로부터 받은 값(motor_num)을 x에 저장하고 case문을 이용, 해당 공정번호(0,1,2,3)와 촬영 정지/ 촬영 신호 송신하여 다른 클래스에 저장할 수 있도록 하는 역할. 다수의 delay를 이용하여 정확한 위치/타이밍 유도

3)Turn : 자가 진단 시스템 시작 시 다음 공정 이동을 위한 스테핑 모터 제어 함수.
본 프로젝트에서는 4공정을 다루므로 90°(1.8° * 50) 회전. 라즈베리 파이에서 명령신호 수신 시 동작

4)switch(RA_signal) : 라즈베리 파이로부터 받은 명령 신호(RA_siganl)을 이용하여 해당 state로 이동. case문에서 다른 공정을 멈추고 카메라를 위치로 이동시킴.
그 후 C_time 함수를 실행하여 촬영 신호를 보냄.

3.2.2. 라즈베리파이

1) def image_cap(num=1)

-> 정확한 위치에서의 이미지 데이터 수집(캡처)하기 위해 아두이노 측의 적외선 센서 범위 내에 제품이 들어 올 때까지 수신을 기다림

2) def spin_to_ard(x)

-> 아두이노와 bluetooth통신을 통해 시스템의 하드웨어를 회전 요청하는 함수

-> 매개변수 x는 촬영하고자 하는 공정 라인을 아두이노 측으로 데이터를 보냄

3) def product_predict()

-> 이미지 수집 후, 미리 학습된 모델을 통해 현재 제품 상태를 예측 함수. 예측의 정확도를 올림과 동시에 환경변수를 줄이기 위해 RGB모델에서 HSV색모델로 변환하고 마스크를 씌워 제품과 배경을 분리시킴. 즉, 제품만을 촬영하여 예측함. (모델 또한 마찬가지)

4) def test_init_var()

-> 테스트에 사용하는 변수들을 초기화시키는 함수

3.3. 기술적 차별성

- 적은 데이터로도 정확도를 최대한 높여, 제품이 들어오고 나가기까지 정확하게 인식가능
- 하나의 카메라로 여러 공정을 관리할 수 있고, 이로 인해 경제성이 우수함
- 딥러닝 기술을 적용하여 어떤 제품이더라도 데이터만 존재한다면 학습하고 판별할 수 있음

4. 개발 중 장애요인과 해결방안

개발 과정에서 나타났던 모든 장애 요인(Risk)들을 나열하고 이러한 장애요인들이 발생했던 경우 어떻게 해결했는지 구체적으로 제시한다.

4.1. 아이디어 수정

기존 아이디어(영상처리, 안드로이드를 이용하여 조작)에서 스마트 팩토리에 부적합한 요소를 수정함. 2차 기술지원 세미나에서 자문위원님들의 의견과 한국품질재단 SW교육에 자문을 구한 결과 궁극적으로 스마트 팩토리는 인간의 손길이 최소화 될수록 좋다는 결론을 도출함. 따라서 기존의 '영상처리/안드로이드 조작'을 '인공지능/자동화'로 대체함

4.2. Hardware

1) 컨베이어 벨트에 사용하는 서보모터의 문제로 공정의 라인을 나타내는 벨트가 일정한 속도로 회전하지 않는 문제

→ 서보모터의 전류를 제한하여 속도를 낮추어 오차를 줄임 / 카메라 시야각을 넓혀 제품이 시야각 안으로 들어오게 함 / 수많은 반복실험을 통해 오차를 알아내고 범위 내에 위치하게 delay를 이용하여 유도

2) 스텝모터에 과도한 열이 발생하여 원하는 각도로 회전이 어려운 문제

→ A4988 모터드라이브 가변저항을 조절하여 최적의 회전 각도 구현

3) 회전부 끝부분의 처짐 발생으로 인해 원활하지 않은 회전의 문제점

→ 회전부에 다른 물체를 부착시켜 무게 차이로 인해 평행 유지 / 글루건으로 시스템 회전부와 모터 회전부 고정하여 회전 시에도 평행 유지를 통한 적외선 센서 인식의 오차 방지

4) 3D 프린터를 이용하여 시스템 Frame 제작 시 Tough PLA 소재의 특성 파악 및 치수 오차 발생으로 알맞은 조립 불가능

→ 적층되는 구조의 3D 프린터 특성을 이용하여 적층하기 편한 구조로 재설계하여 제작 진행 / 접합 부위의 치수를 기존 치수보다 낮게 설계하여 안전하게 조립

4.3. Software

1) AI를 통해 데이터를 기반으로 학습시킨 Deep running (CNN) 기술을 재현하는 것이 어려움

표본의 수가 부족하면 불량에 대한 정확도가 떨어지고 과적합이 발생함.

이미지 데이터 세트가 존재한다면, 어느정도 구현하기 쉬울지 모르겠지만, 우리가 직접 데이터를 수집하고 학습시키는 것에는 다소 어려움들이 많음. 직접 정상품과 불량품을 선정하고 구상한 시스템이기 때문에 데이터를 직접 수집해야 함

→ 데이터를 수집하는데에 다소 시간이 많이 걸리지만, 촬영을 했을 때 불량에 대한 판단이 최대한 정확하도록 정상품과 불량품에 대한 데이터를 하나씩 찍어서 쌓음

2) 라즈베리파이 성능이 CNN을 구현하기에 비교적으로 우수한 편이 아니므로 모델을 학습하기 위해 여러 고려사항들이 존재함. (컬러 3채널, 이미지 크기, 최적화 알고리즘 모델 선정)

-> 이미지의 특징이 무너지지 않는 범위 내에서 이미지 크기를 조절하였고, 불필요한 데이터들은 OpenCV 라이브러리를 통해 전처리

5. 개발결과물의 차별성

개발한 결과물에 대해 타 팀과의 차별성(우수성) 및 기대효과를 서술한다.

첫 번째, 차별성은 범용성이다.

드론과 로봇의 경우 숙련된 작업자가 필요하거나 사람의 손길이 필요하고 적용가능한 공장이 적다. 하지만 본 '인공지능 자가 진단 시스템'은 어느 분야의 공장이든 적용할 수 있다. 또, 본 프로젝트에서는 하드웨어로 '360°회전 타워형'을 구현했지만 레일형, 호이스트형 등 다양한 형태에 적용할 수 있다. 이를 이용하여 불량검출이 필요한 공장이라면 어디든 적용할 수 있다.

두 번째, 경제성이 우수하다.

카메라(데이터 수집 장치)와 CPU만 있으면 되기 때문에 초기 투자비용이 적게 든다는 장점이 있다. 하지만 더 높은 신뢰도와 판단능력을 갖추기 위해서는 데이터의 양이 방대해지고 CPU와 데이터 저장에 비용이 증가하게 될 것이다. 이는 고객의 수요에 따라 변동되는 사항이다. 최소의 기능을 이용한다면 적은 투자로도 높은 효율을 이끌어 낼 수 있다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

참여 인원의 역할을 구체적으로 제시하고 개발 일정 계획상에서 각 참여 인력이 어떤 역할을 수행했는지, 어떤 부분을 협업하였고, 어떤 개발 절차로 개발은 진행했는지에 대해 자세히 명시한다.

계발단계	기간(주)																							담당
	5월				6월					7월					8월					9월					
	1	2	3	4	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	
아이디어 구상																									전원
회로 구성 및 아두이노																									준표/동현
프레임 제작 및 CAD																									승헌
모터 교체 및 세팅																									승헌/준표
보드간 신호 및 통신 설정																									주영/동현
영상 처리 알고리즘 개발																									주영
통합 Test 및 개발내용 보완																									전원

No.	구분	성명	소속(학교)	부서(학과)	입학년도	맡은 역할
1	팀장	윤주영	동아대학교	전자공학과	2013년	SW 총괄 CNN-딥러닝 알고리즘 개발
2	팀원	이승헌	동아대학교	기계공학과	2012년	CAD 제작 영상 처리 데이터 관리
3	팀원	홍준표	동아대학교	전기공학과	2013년	HW 및 회로 제작 아두이노 / 보드 결선 제작
4	팀원	김동현	동아대학교	전기공학과	2014년	HW 및 회로 제작 아두이노 / 통신