객체 탐지 기술을 활용한
컨베이어 벨트 자동화 시스템 개발

2023년

안양대학교 스마트창의융합대학

컴퓨터공학전공

김 연 수

지 도 교 수 하 은 용

이 논문을 공학 학사 학위논문으로 제출함.

2023년 12월

안양대학교 스마트창의융합대학

컴퓨터공학전공

김 연 수

논 문 인 준 서

이 논문을 김연수의 공학 학사 학위논문으로 인준함.

2023년 12월

주 심 이 상 홍 인

부 심 박 성 순 인

부 심 하 은 용 인

차 례

표 및 그림 차례

표 차례

그림 차례

요 약

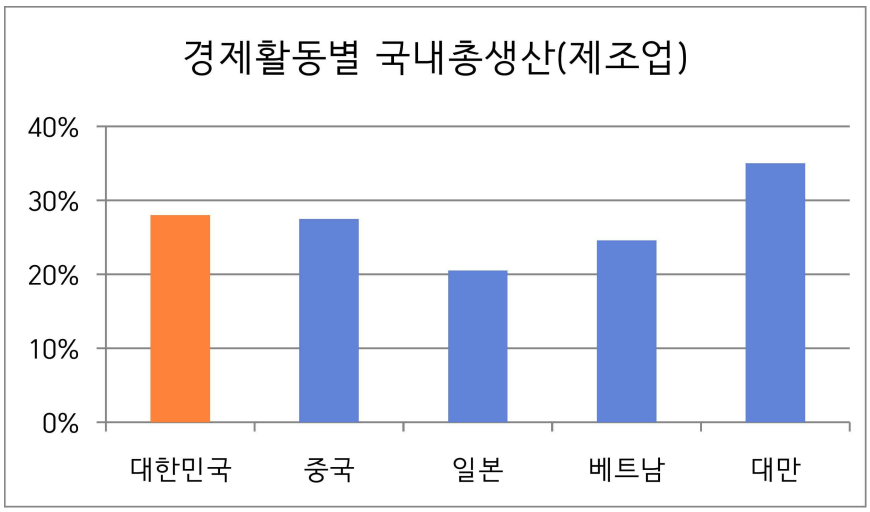
큰 편에 속하는 한국의 제조업 비중과 많은 공장 및 산업 생산 현장에서 4차 산업 혁명에 따른 스마트팩토리 관련 솔루션에 주목하고 있다. 이러한 제조업 패러다임의 변화에 따라가기 위해 사람의 간섭을 최소화 시킬 수 있는 스마트팩토리 솔루션에 대한 기본적인 요소들부터 개선할 점까지 순차적으로 연구를 진행하였다.

스마트팩토리의 핵심 요소인 인공지능 컨베이어벨트를 개발하였다. 객체 탐지 기술을 기반으로, 스스로 물건을 분류하는 인공지능 컨베이어벨트를 개발해 보았다. 컨베이어벨트가 카메라를 통해 물류를 인식하고, 인식한 결과에 따라 물류를 로봇팔을 통해 물리적으로 분류작업을 진행하게 된다. 분류 작업이 완료된 물류는 물류로봇을 호출하여 분류된 물류를 해당 물류 창고에 운반하고, 이러한 물류 분류 정보들은 모두 웹 서버에 기록된다. 이러한 과정을 통해 생산성과 유연성을 극대화 시킬 수 있고, 제조, 물류 관리의 효율성을 극대화 시킬 수 있는 스마트팩토리의 일부를 구현하는 것을 목표로 한다.

제 1장 서 론

제 1절 배경

스마트 팩토리는 기존의 생산과정에 디지털 자동화 솔루션이 결합된 지능형 생산 공장을 말한다. 따라서 현재까지 많은 공장 및 산업 생산 현장에서 스마트 팩토리 관련 솔루션에 주목하고 있다.

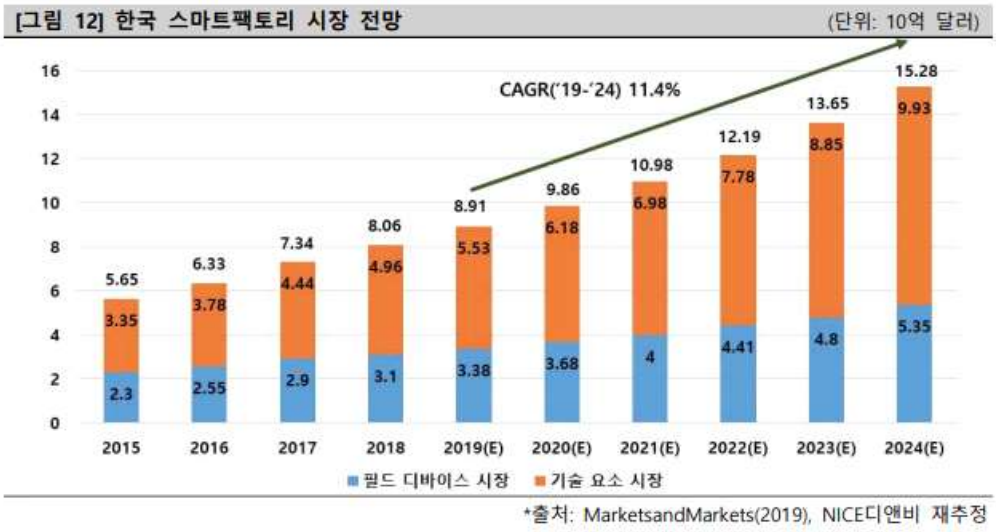


[그림 1] KOSIS 제공 - 경제활동별 국내총생산 2021년 자료

컨베이어 벨트 자동화 시스템은 제조업의 혁신을 이끄는 스마트 팩토리의 아주 기본적이고 꼭 필요한 기술이라고 생각된다. 스마트 팩토리는 기획·설계, 생산, 유통·판매 과정에서 인공지능의 도움을 받을 수 있다면 시간적, 비용적 이득을 볼 수 있다. 기획·설계 단계에서는 가상공간에서의 제품 제작 시뮬레이션을 통한 기획 기간 단축, 생산 단계에서는 설비, 자재, 시스템 간 실시간 정보교환을 통한 대량생산 및 설비 효율 증가, 유통·판매 단계에서는 생산 현황에 맞춘 실시간 자동 발주를 통한 재고비용을 감소시킬 수 있다. 한국 경제의 제조업 비중이 2021년 기준 27.9%로 상당히 큰 비중을 차지하고 있다는 것을 생각하면 제조업 패러다임의 빠른 변화가 필요할 것이라 생각한다.

컨베이어 시스템의 경우 자동화를 도입한 곳도 있지만, 대다수의 공장에서는 아직까지 사람의 손길을 필요로 하고 있다. 컨베이어 벨트는 대량 생산을 목적으로 개발한

것인데, 사람의 손길을 필요로 한다면 효율이 급격하게 떨어질 수밖에 없다.



[그림 2] 한국 스마트팩토리 시장 전망 (출처: MarketsandMarkets)

위 [그림 2]에서 보듯, 제조업 비중이 큰 한국의 스마트팩토리 시장 규모는 계속해서 커질 것으로 예측된다. 따라서 데이터 패턴 분석과 같은 인공지능 기술을 기반으로 한 효율적인 컨베이어 시스템과 인공지능 물류 로봇 개발을 통해 빠르게 성장하는 스마트팩토리 시장을 따라갈 필요성이 있다.

따라서 사람의 간섭을 최소화시킬 수 있는 스마트팩토리 솔루션중 컨베이어 벨트 자동화 시스템에 대해 기본적인 요소들부터 개선할 점까지 순차적으로 연구를 진행한다. 이 논문의 2장에서는 컨베이어 벨트의 종류에 따른 장단점을 살펴보고 이번 연구에 적합한 컨베이어 벨트 유형을 고른다. 또한 컨베이어 벨트 자동화 시스템에 사용될 객체 탐지 기술에 대한 관련 연구들을 소개하고 라즈베리파이를 활용한 관련 연구들도 살펴본다. 3장에서는 컨베이어 벨트를 구현하기 위한 설계 단계를 다룬다. 먼저 컨베이어 벨트 구조물을 만들기 위한 하드웨어 설계 과정에 대해 이야기하고 만들어진 컨베이어 벨트 구조물을 어떤 방식으로 구동시킬지에 대한 소프트웨어 설계 과정에 대해 이야기한다. 두 설계 과정을 소개한 후 구현하고자 하는 스마트 팩토리 전체 구성도를 보여준 뒤 컨베이어 벨트의 역할에 대해 세부적으로 설명한다. 4장에서는 구현한 컨베이어 벨트를 보여주며 객체 탐지 기술을 활용하여 물류를 분류하고 분류 결과를 서버

로 전송하는 내용에 대해 이야기한다. 마지막으로 5장에서는 이번 연구를 통해 얻은 결론과 기대효과에 대해 이야기하고 향후 해결할 부분에 대해 이야기하며 마친다.

제 2장 관련 연구

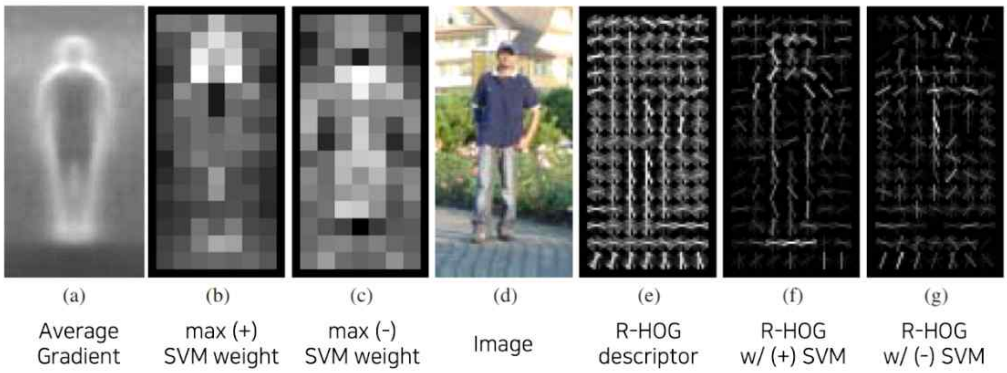
제 1절 컨베이어 벨트

컨베이어 벨트 즉 컨베이어 시스템은 한 지점에서 다른 지점으로 물건 또는 소재를 운반하는 기계적 장치를 말한다. 컨베이어 시스템은 흐름작업 조직으로서의 두 가지 형태가 있다. 하나는 전진하는 컨베이어 위에서 일정한 간격으로 배치되어 있는 물류를 이동시키는 이동작업형 컨베이어 시스템이고, 다른 하나는 컨베이어에 의해 운반되는 물류가 추가적으로 설치되어 있는 작업대 또는 기계 위로 옮겨지는 형태인 정지작업형 컨베이어 시스템이다. 일반적으로 사용되는 형태는 이동작업형 컨베이어 시스템이고, 이 경우에는 주로 수공작업 공정에 사용된다. 기계작업 공정이 들어가는 경우 정지작업형을 혼합하여 사용되기도 한다. 이번 연구에서 구현할 컨베이어 시스템은 이동작업형 컨베이어 시스템이고, 여기에 수공작업이 아닌 기계작업 공정을 사용하는 방식을 사용하고자 한다.

컨베이어 벨트는 형태에 따라서도 다양하게 나뉜다. 흔히 볼 수 있는 수평이송 벨트 컨베이어, 경사를 가진 경사 벨트 컨베이어, 높이 조절 및 이동이 가능한 포터블 벨트 컨베이어, 이송품의 방향을 전환시켜줄 수 있는 커브드 벨트 컨베이어 등 다양하다. 이 중 대부분의 산업 현장에서 흔히 사용되는 수평이송 벨트 컨베이어를 사용하여 구현할 것이다. 수평이송 벨트 컨베이어는 두 개의 롤러에 벨트를 걸어 구동시키는 방식을 가지고 있다. 또한, 간단한 구조를 가지기 때문에 설치가 간단하다는 장점을 가지고 있다.

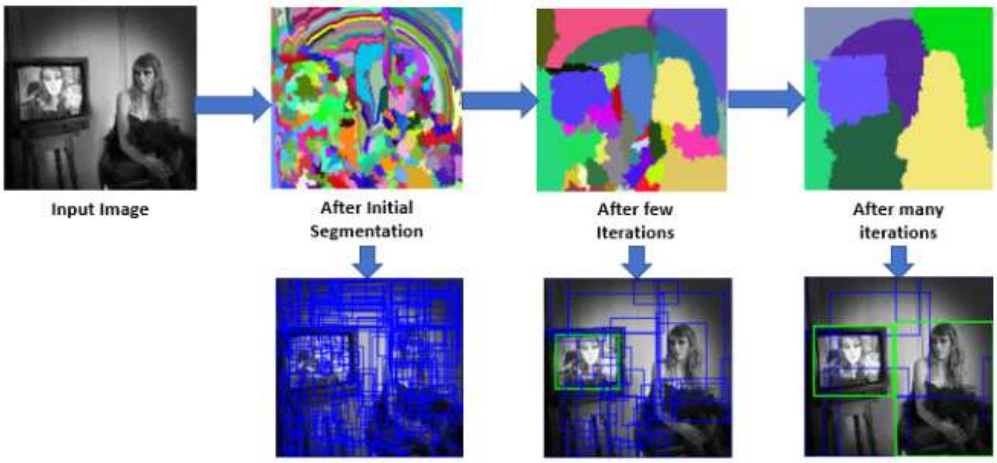
제 2절 객체 탐지 기술

인공지능 컨베이어 시스템을 구현하기 위해 필요한 객체 탐지(object detection) 알고리즘은 크게 두 가지(Two-stage Detector(R-CNN), Single-stage detector(YOLO))가 있다. 우선, 크게 사용되고 있는 두 가지 방법을 살펴보기 전에 과거의 객체 탐지 방법에 대해 알아보았다.



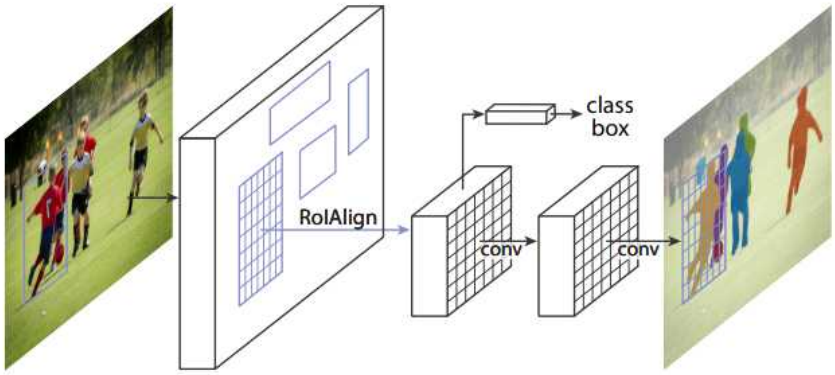
[그림 3] Gradient-based detector (출처: ganta-Object detection)

과거에는 객체 탐지를 위해 Gradient-based detector(경사기반 검출)방법과 Selective search(선택적 검사)방법을 사용하였다. Gradient-based detector는 단어 의미 그대로 경계선을 특징으로 객체를 탐지하는 방법이다. 보행자 검출 알고리즘으로도 많이 알려져 있다. 위 [그림 3]에서 볼 수 있듯 인간이 만든 알고리즘을 통해 (a)와 같이 평균 경계 이미지를 만들어준 모습이고, (b)와 (c)는 픽셀 형태로 각 픽셀의 최대 양, 음의 SVM(Support vector machine) 가중치를 보여준 모습이다. 선형분류기를 통해 관심 문제인지 아닌지를 판별하게 되는 것이다. (e)는 (a)를 이용해 히스토그램을 구하여 나타낸 모습이고, SVM에 의해 분리되어 (f)와 (g)의 모습으로 나타내게 된다.



[그림 4] Selective search (출처: geeksforgeeks.org)

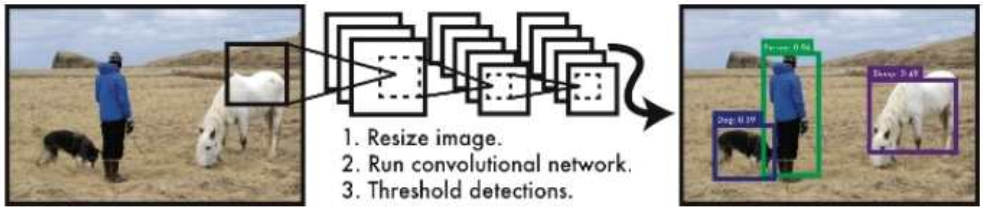
Selective search는 object 인식이나 검출을 위한 가능한 후보 영역을 알아낼 수 있는 방법을 제공하는 것을 목표로 하고 있다. Selective search는 입력 영상을 분할하게 되는데, 초기에는 많은 양의 후보들이 만들어지고 반복적으로 작은 영역을 큰 영역으로 결합하는 과정을 거치면서 영역을 찾아낸다. 하지만 실시간 적용에는 부적합하다.



[그림 5] Mask R-CNN (출처: Mask R-CNN-Kaiming He...)

Two-stage Detector에는 R-CNN(Region-based Convolutional Neural Networks)계열이 속하며, Mask R-CNN이 여기에 해당된다. R-CNN은 딥러닝 기반의 객체 탐지 알고리즘으로 과거 객체 탐지 방법들보다 높은 정확도와 실용적인 속도를 보여준다. 이러한 R-CNN은 대표적으로 Fast R-CNN에서 Mask R-CNN

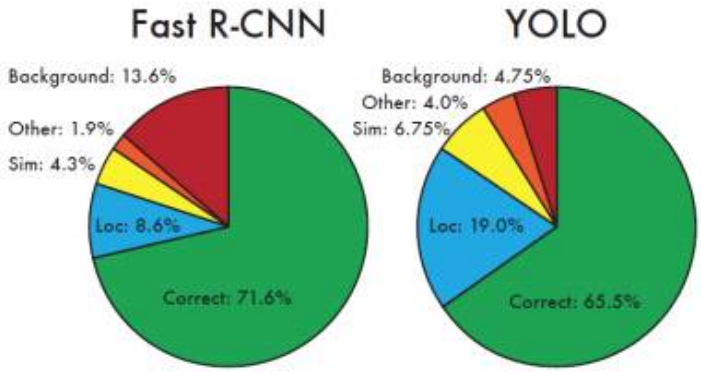
으로 개선되어왔다. Mask R-CNN은 객체 탐지에 객체 분할(segmentation)까지 수행하는 모델로 객체 검출, 객체 분할, 객체 분류를 한 번에 수행할 수 있다. Mask R-CNN은 기존 Faster R-CNN에서 발생하던 RolPool 영역의 위치에서 생기는 소수점 오차를 2D 선형보간법(bilinear interpolation)을 통해 감소시킨 RolAlign을 사용하여 더욱 정확한 분할을 수행할 수 있다.



[그림 6] YOLO Detection System (출처: 반도체디스플레이기술학회지 제19권 제 1호)

Single-stage Detector에는 대표적으로 YOLO 알고리즘이 있다. YOLO는 이미지 내에 존재하는 객체와 해당 객체의 위치를 하나의 입력 이미지를 통해 예측할 수 있는 알고리즘으로, R-CNN 대비 빠른 객체 탐지와 분류를 지원한다. 빠른 처리시간 덕분에 실시간 객체 검출 또한 가능하여 대중성 있게 사용될 수 있는 알고리즘이다.

두 알고리즘 중 Mask R-CNN을 사용하여 인공지능 컨베이어 시스템을 구축할 것이다. Mask R-CNN은 YOLO에 비해 처리 시간이 느리다는 단점이 있지만, 처리시간이 느린 만큼 많은 학습량을 통해 객체 분할에 대한 높은 정확도를 보여준다.



[그림 7] Fast R-CNN vs YOLO
(출처: 반도체디스플레이기술학회지 제19권 제 1호)

위 [그림 7]은 CNN과 YOLO의 오류 비율을 보여준 것이다. 검출 객체에 대한 정확한 지역화(Loc)에서 YOLO는 Fast R-CNN에 비해 2배 낮은 정확도를 보여주고 있다. 단순히 객체 종류를 구분하는 수준이라면 YOLO가 유리하겠지만 객체 간의 세부적인 부분까지도 정확하게 구분이 가능한 Mask R-CNN이 진행할 연구에 더 적합하다고 판단하였다.

제 3절 라즈베리파이에서의 객체 탐지 기술 적용

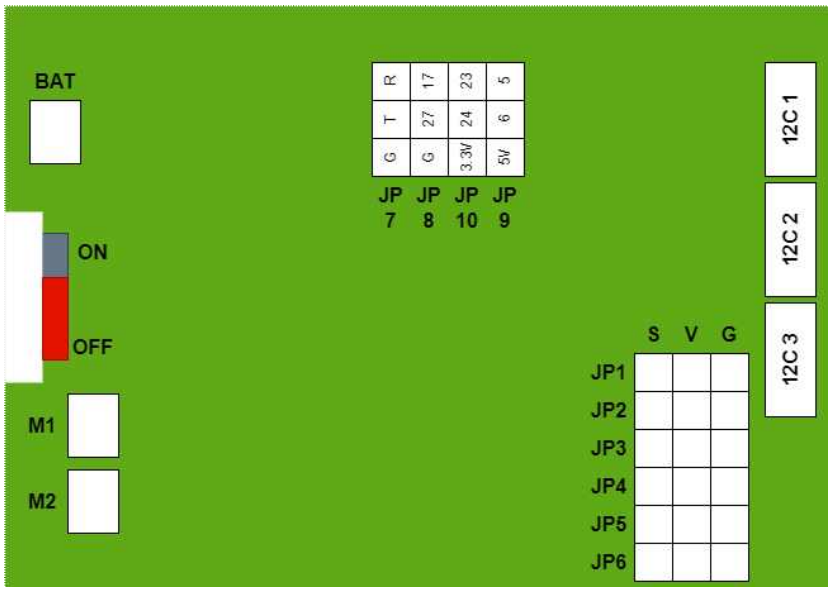
라즈베리파이에서는 TensorFlow Lite와 Open CV 모듈을 사용하여 객체 탐지 알고리즘을 사용할 수 있다. TensorFlow(텐서플로)는 구글에서 개발한 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 오픈소스 소프트웨어 라이브러리이다. 이 텐서플로는 임베디드 장치를 위한 Lite 버전이 존재하며 이는 기존 텐서플로를 경량화시킨 버전이다. 윈도우 환경에서 테스트할 경우 일반 TensorFlow를 사용하고, 구현 단계에서는 라즈베리파이에서 사용할 것이므로 TensorFlow Lite를 사용하도록 한다. Open CV는 인텔에서 개발한 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리이다. 실시간 이미지 프로세싱에 중점을 둔 라이브러리로 라즈베리파이에 카메라를 연결하여 실시간으로 영상처리를 할 수 있도록 하기 위해 사용된다.

제 3장 시스템 설계

제 1절 하드웨어 설계

컨베이어 벨트의 벨트를 구동시킬 수 있는 모터와 물류를 물리적으로 분류시킬 수 있게 해주는 모터, 물류를 인식할 수 있는 카메라가 필요하다. 추가적으로 초음파 센서를 사용하여 컨베이어 벨트 위에서의 물류 위치를 파악할 수 있도록 해준다.

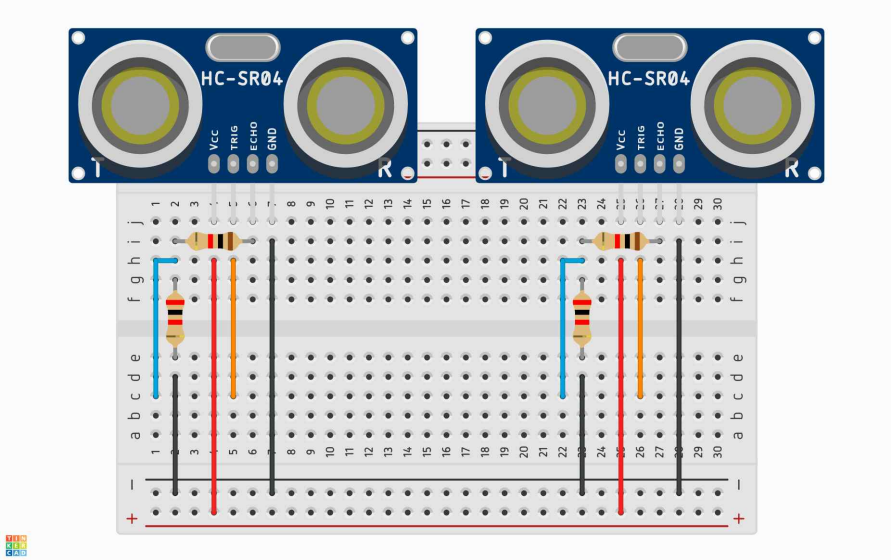
모터의 경우 두 가지 모터가 필요하다. 벨트를 움직일 수 있게 해주는 모터의 경우, 모터의 힘이 강하고 속도 조절이 가능한 모터를 사용하는 것이 좋다. 또한, 이번 연구에서 사용하려는 라즈베리파이4와의 호환성을 고려하여 DC 모터를 결정하였다. 물류를 물리적으로 분류시켜주기 위해 사용하는 모터는 서보 모터를 사용한다. 서보 모터는 다른 모터들과 달리 각도에 따라 정확하고 견고하게 동작하여 물류를 분류시키는 로봇팔에 사용하기에 적합하다고 판단하였다. DC 모터와 서보 모터의 스펙은 컨베이어 벨트의 크기, 분류할 물류의 크기, 무게 등을 고려하여 선택하게 된다.



[그림 8] HAT 보드 모습

모터들을 원활히 동작시키기 위해서 JD Edu사의 'DEEPTHINKCAR _V2 _V3 _230215' HAT 보드를 사용한다. 이 HAT 보드에는 최대 2개의 DC 모터와 10개의 서

보 모터를 제어할 수 있도록 하네스 및 GPIO핀이 구성되어있다. 기존 라즈베리파이 GPIO 핀을 사용하는 것보다 선 정리와 전원 공급 면에서 안정적이므로 HAT 보드를 사용한다. 위 [그림 8]이 사용할 HAT 보드의 모습이다. 해당 보드의 BAT 커넥터를 통해 라즈베리파이와 모터보드에 전원을 공급해주게 되고, M1, M2 커넥터를 통해 DC 모터 2개를 제어할 수 있게 된다. 서보 모터는 JP라고 쓰여 있는 GPIO 타입의 커넥터를 사용하면 되는데 4개의 서보 모터만 사용할 예정이므로 JP1~4까지만 사용하게 된다. JP7~10 에서는 라즈베리파이의 GPIO 핀을 사용할 수 있게 되어있다. JP7의 G, JP9의 5, 6, 5V, JP10의 23, 24 GPIO 핀을 사용하여 초음파 센서를 아래 [그림 8]과 같이 연결해준다.



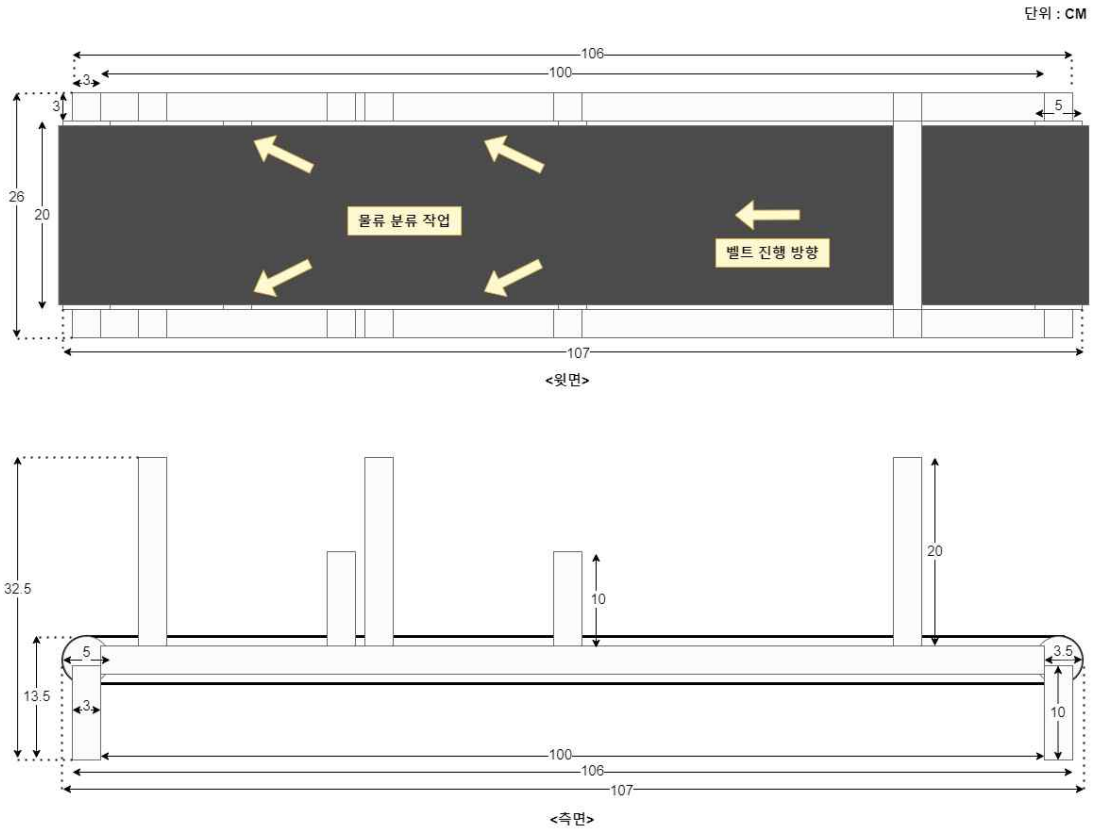
[그림 9] 초음파 센서 연결 회로도

2개의 초음파를 사용하고, [그림 9]과 같이 Echo에는 6번과 24번, Trig에는 5번과 23번을 연결해주고 전원은 5V로 연결해준다. 이때 주의할 점은 Echo쪽의 저항 연결이다. 라즈베리파이의 GPIO는 3.3V 레벨로 동작하는데 위 [그림]과 같이 저항을 연결하지 않을 경우 Echo 핀에서 High level 신호 발생 시 5V를 전송하게 되어 라즈베리파이에 손상을 줄 수 있다. 따라서 1K옴과 2K옴을 사용하여 전압을 낮춰야 한다.

카메라는 라즈베리파이용 카메라를 사용한다. 카메라는 웹캠 등의 USB 타입의 카메라를 사용하거나 라즈베리파이 보드 상의 CSI 인터페이스에 연결할 수 있는 15pin 케이블을 이용한 라즈베리파이 카메라를 사용할 수 있다. 15pin 케이블을 사용한 라즈베리파이 카메라를 선택한 이유는 USB 타입의 웹캠들에 비해 빠른 속도를 자랑하기 때문이다. 라즈베리파이에 사용되는 CSI 15pin 커넥터는 2개의 데이터 라인을 갖는 MIPI CSI 인터페이스로, 2라인 기준 640MB/s 정도의 속도를 갖은 데 비해 USB 2.0은 60MB/s 정도로 매우 큰 속도 차이를 보이고 있다. 물론 USB 3.0 의 경우엔 625MB/s 로 2.0 대비 약 10배 정도 빠른 속도를 보이지만 라즈베리파이에서는 하나의 USB 버스에서 각 USB 포트로 분산시키는 방식을 사용하기 때문에 속도 저하가 일어날 수밖에 없다. 따라서 더 안정적이고 빠른 CSI 카메라를 사용한다.

제 2절 컨베이어 벨트 구조물 설계

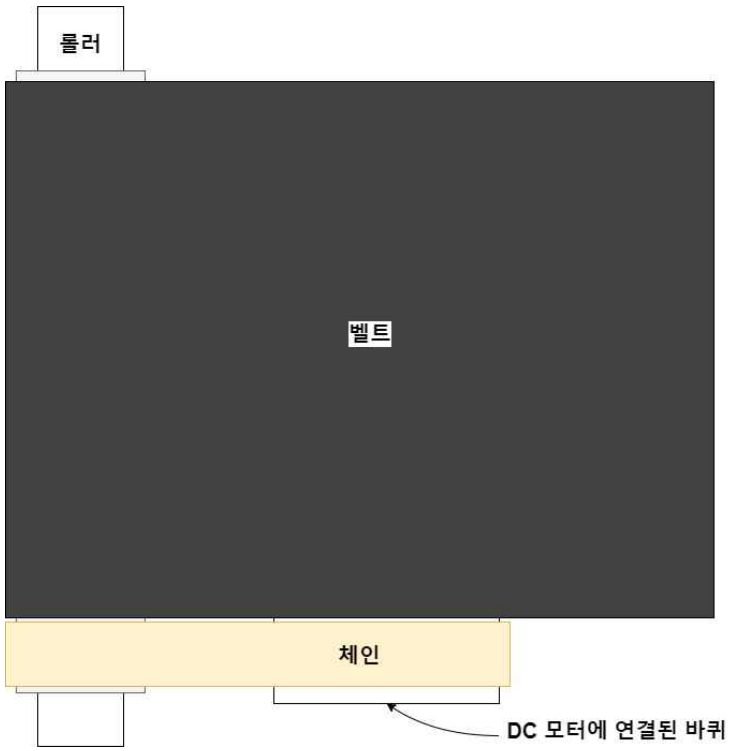
위 1절 하드웨어 설계 부분에서 다룬 각종 부품을 설치할 수 있는 구조물을 제작한다. 이 부분에서 고려해야 할 부분은 구조물의 크기와 소재이다. 구조물의 크기의 경우 분류할 물류의 크기에 맞게 설정해준다. 이번 연구에서는 작은 크기의 박스를 탐지하여 분류시키는 연구를 할 것이므로 컨베이어 벨트 구조물의 크기가 실제 공장에서 사용하는 것처럼 매우 클 필요는 없다. 소재의 경우엔 구조물의 크기에 맞게 튼튼하고 쉽게 조립할 수 있고 마음대로 구조물 위치를 변경 가능한 알루미늄 프로파일을 선택한다.



[그림 10] 컨베이어 벨트 설계도

위 [그림 10]은 구조물의 크기와 소재가 정해진 후 위 작성한 컨베이어 벨트 구조물의 도면이다. 소재는 30x30(mm)의 알루미늄 프로파일과 벨트를 움직이게 도움을 줄 수 있는 지름 50mm 롤러로 구성되어있다. 컨베이어 벨트의 벨트는 고무를 사용하려

하였으나 고무를 가공할 수 있는 환경이 아니었기 때문에 두께 2mm의 부직포 원단으로 대체하여 사용한다.



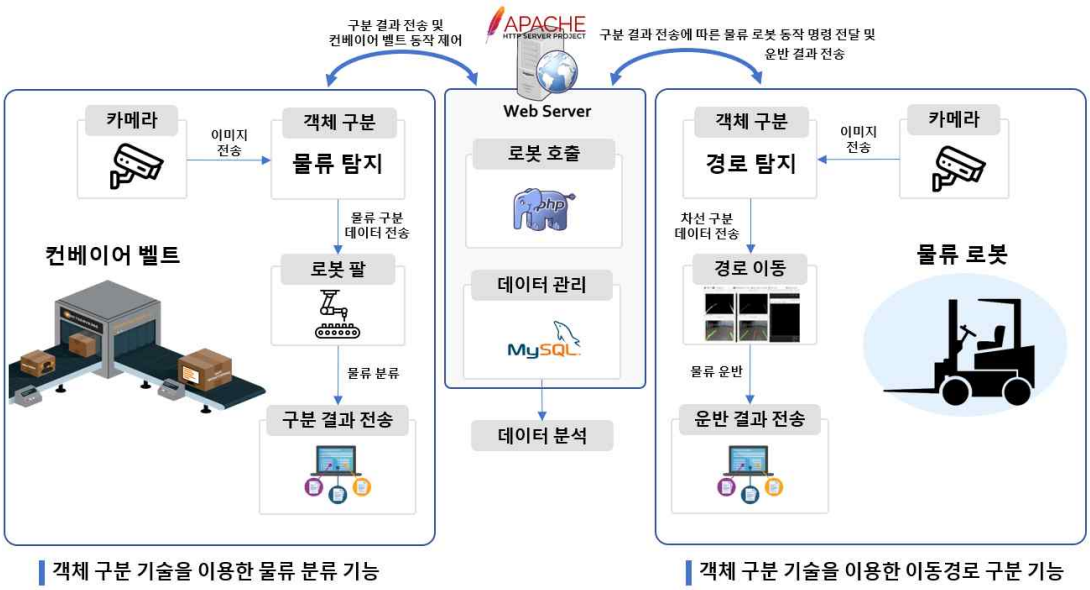
[그림 11] 컨베이어 벨트 동작 원리

컨베이어 벨트의 동작은 위 [그림 11]과 같은 방식으로 동작할 수 있도록 설계하였다. 위 그림은 컨베이어 벨트 구조물의 앞부분의 일부를 가져온 모습으로 롤러에 연결된 벨트와 체인의 모습을 볼 수 있다. 체인은 롤러와 DC 모터에 연결된 바퀴, 총 2곳에 연결되어 DC 모터 회전 시 롤러를 같이 회전시켜 벨트가 움직일 수 있도록 해준다. 체인의 소재는 롤러와 DC 모터에 연결된 바퀴의 회전 시 미끄러지지 않을 수 있는 소재여야 하고, 이번 연구에서는 벨트와 동일한 부직포 원단을 사용하여 톱니 모양의 체인을 만들어 사용한다.

제 3절 이미지 처리 설계

컨베이어 벨트를 동작시키는 프로그램이 실행되면 컨베이어 벨트 상단에 부착된 카메라를 통해 촬영한 실시간 영상을 받아올 수 있다. 이 영상에는 컨베이어 벨트 주변의 모습, 벨트의 움직임과 물류의 움직임을 볼 수 있다. 이번 연구에서는 물류 분류를 중점으로 두고 이미지 처리를 할 것이기 때문에 물류를 탐지할 수 있는 위치 이외의 부분은 객체 탐지가 되지 않도록 처리해주어야 한다. 물류의 경우에는 색상, 모양, 크기별로 분류시킬 것이므로 모니터를 통해 각각의 분류 상황을 모니터링 할 수 있도록 도와주는 이미지맵 창이 필요하다. 사용자가 이 이미지맵 창을 통해 컨베이어 벨트에 올라온 물류가 어떤 색상이고, 어떤 모양이고, 어느 정도의 크기를 가지고 있는지 한눈에 볼 수 있도록 프로그래밍을 진행한다.

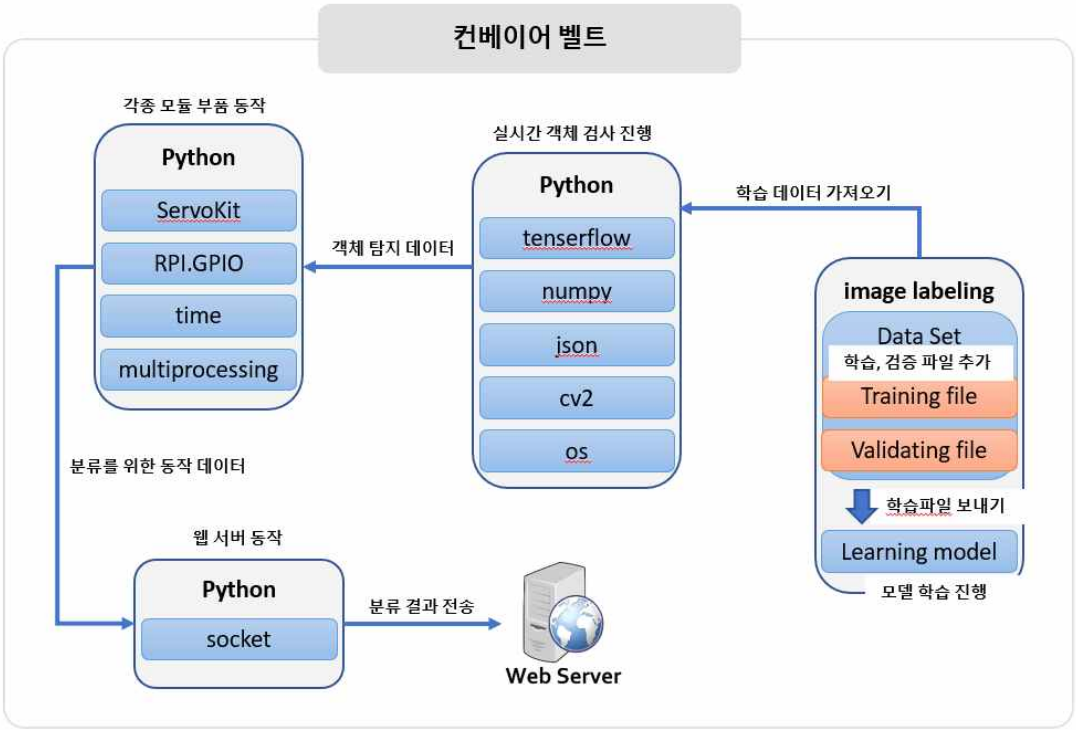
제 4절 전체 구성도



[그림 12] 스마트팩토리 시스템 구조도

위 [그림 12]는 스마트팩토리의 일부를 구현하기 위해 연구를 진행한 컨베이어 벨트, 물류 로봇, 웹 서버에 대한 전체 시스템 구조를 간략하게 보여주고 있다. 이 연구에서는 컨베이어 벨트 자동화 시스템에 대해서만 다루지만 컨베이어 벨트를 개발함과 동

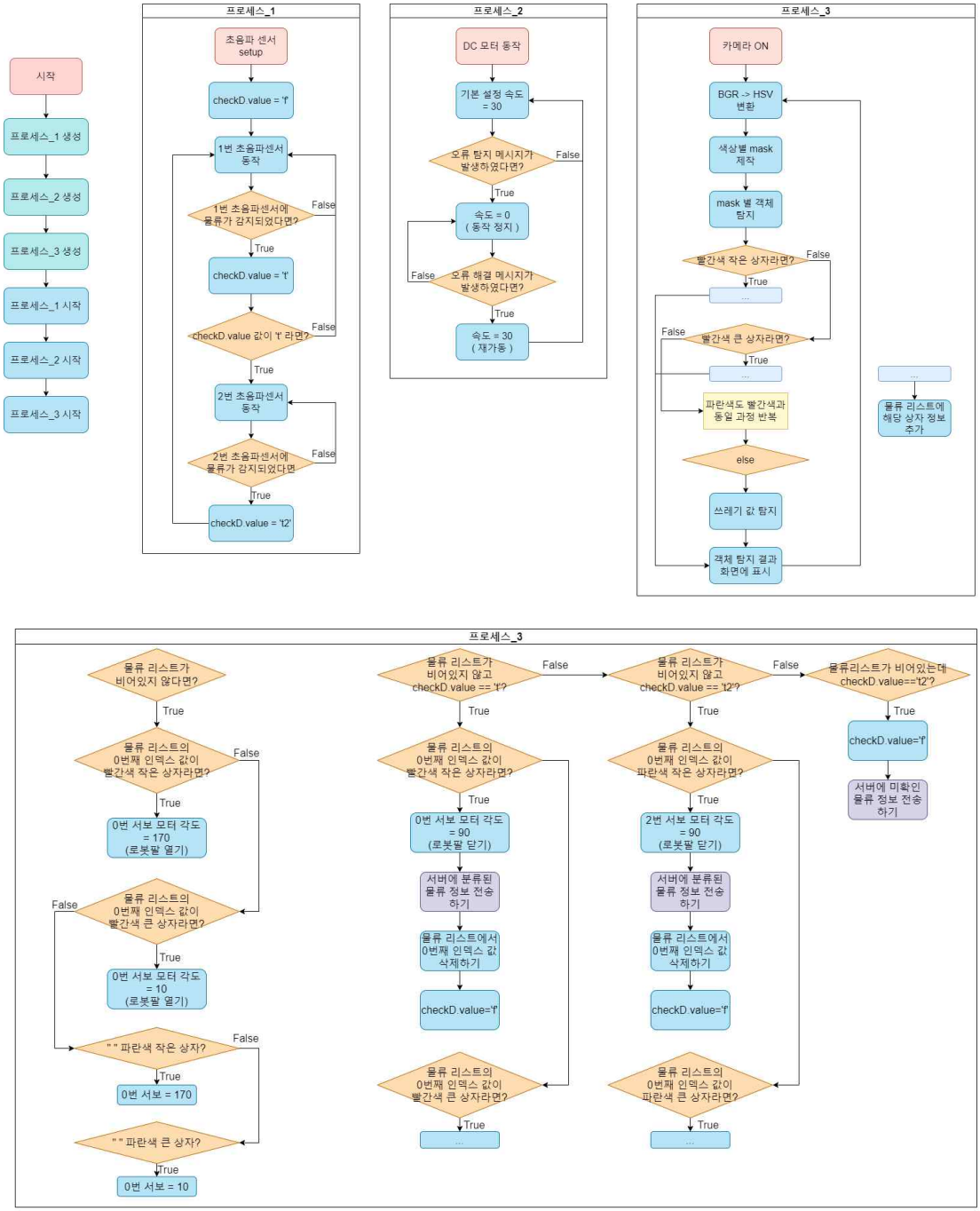
시에 다른 팀원들은 물류 로봇과 이를 연동시킬 웹 서버 개발을 진행한다.



[그림 13] 컨베이어 벨트 모듈 구조

[그림 13]의 전체 구성도에서 컨베이어 벨트의 구조를 자세히 본다면 위 [그림]과 같다. 우선 이미지 라벨링 등의 작업을 통해 얻은 학습 데이터 셋으로 모델 학습을 진행한 후, 학습된 데이터를 기반으로 컨베이어 벨트에서 실시간 객체 검사를 진행하게 된다. 라즈베리파이에 연결된 카메라를 통해 들어오는 객체 모습을 Python 프로그램을 통해 어떤 객체인지 탐지하게 되고, 탐지된 결과를 기반으로 초음파 센서와 서보 모터를 동작시켜 객체를 물리적으로 분류시키는 작업을 진행하게 한다. 이때 초음파 센서는 물류의 위치를 탐지할 수 있게 해주는 모듈이고, 초음파 센서가 탐지한 물류 위치와 카메라를 통해 탐지된 객체 정보를 기반으로 서보 모터를 동작시켜 물류를 분류시키게 한다. 물류 분류가 정상적으로 완료되었다면 socket 통신을 통해 웹 서버로 분류 결과를 전송하게 된다. 이렇게 분류된 결과는 웹 서버의 데이터 분석 결과에 따라 물류 로봇을 호출하여 분류된 물류를 운반시킬 것인지를 판단하는데 사용되게 된다.

제 5절 전체 흐름도



[그림 14] 컨베이어 벨트 자동화 시스템 순서도

컨베이어 벨트 자동화 시스템의 흐름은 위 [그림 14]과 같다. 멀티 프로세싱을 사용하여 모듈 동작에 따라 3개의 프로세스로 나누어주고, 3개의 프로세스가 동시에 실

행될 수 있도록 프로그램을 설계한다. 멀티 프로세싱을 사용하는 이유는 초음파 센서 사용 시 time sleep을 활용하여 거리를 측정하게 되는데 멀티 프로세싱으로 분리하여 실행해주지 않으면 time sleep 시간 동안 초음파 센서를 제외한 모든 동작이 정지되기 때문이다. 멀티 프로세싱을 사용하면 초음파 센서 프로그램에서 time sleep이 발생하더라도 나머지 모듈들도 원활하게 동작할 수 있다. 3개의 프로세스는 각각 거리 측정, 컨베이어 벨트 동작 모터 제어, 카메라 및 서보 모터 제어 기능을 가지고 있다. 각 프로세스는 시스템이 종료될 때까지 각 과정을 무한 반복하게 된다. 우선 거리 측정 프로세스부터 살펴보면, 2개의 초음파 센서를 통해 지속적으로 거리 변화를 측정하는 방식으로 동작한다. 다만 2개의 초음파 센서가 계속 동시에 동작하는 방식이 아닌, 첫 번째 초음파 센서만 동작하다가 어떠한 물체가 지나가게 된다면 측정을 멈추고 두 번째 초음파 센서로 측정을 시작하게 하는 방식이다. 현재 구현하고자 하는 컨베이어 벨트의 로봇팔에 사용될 모터 방식과 컨베이어 벨트의 길이를 생각하였을 때 물체가 두 초음파 센서의 거리보다 좁게 이동한다면 물리적으로 분류시키기 어렵다는 문제점이 있다. 때문에 두 물체를 동시에 분류시키고자 한다면 일정한 거리를 두고 물체를 컨베이어 벨트에 올려야 하는데, 이렇게 되면 굳이 2개의 초음파 센서를 동시에 동작시켜 에너지 및 자원 낭비를 할 필요가 없기 때문이다. 상황에 맞게 두 초음파 센서를 번갈아 가며 동작하게 프로그래밍을 해줌으로써 에너지 및 자원 효율을 증가시킬 수 있었다.

모터 동작 프로세스는 모터의 속도를 제어해줄 수 있다. 에러 상황이나 서버를 통해 시작, 정지 명령을 받았을 때 컨베이어 벨트의 벨트 동작을 모터의 속도 변화를 통해 제어해줄 수 있게 만들어준다.

카메라 및 서보 모터 동작 프로세스는 객체 탐지 및 물리적인 물류 분류 기능을 가지고 있다. 컨베이어 벨트 위에 물류가 올라오면 제일 먼저 카메라를 통해 객체 탐지 알고리즘을 적용하여 물류를 구분하게 된다. 이렇게 구분된 정보를 바탕으로 서보 모터를 동작시켜 물류를 분류할 준비를 하고, 거리 측정 프로세스에서 받아오는 물류 위치 정보를 바탕으로 서보 모터를 닫아 물류를 분류할 타이밍을 정한다. 모든 과정이 오류 없이 끝난다면 서버에 해당 물류 데이터를 전송시키는 작업을 하게 한다.

제 4장 구현

제 1절 시스템 구현 환경

구분		항목	적용내역
S/W 개발 환경	OS	라즈비안	라즈베리파이 OS
	개발환경 (IDE)	Thonny Python IDE	Python 프로그래밍을 위해 사용
	개발언어	Python	객체 탐지 알고리즘 사용에 적합한 언어 선택
H/W 구성 장비	디바이스	RaspberryPi 4B	객체 탐지 알고리즘 동작 가능 및 각종 센서 모듈들을 쉽게 다룰 수 있는 디바이스 선택
	센서	HC-SR04	초음파 센서를 통해 측정한 거리 변화로 물류 위치 측정
	통신	iptime N604R	서버와의 소켓 통신을 위한 공유기 사용
	모터	CD12-6, SG-90	벨트 동작을 위한 DC 모터와 로봇팔 제어를 위한 서보 모터
	카메라	라즈베리파이 카메라 모듈 V2	실시간 물류 정보 확인을 위한 영상 촬영용 카메라

[표 1] 소프트웨어 및 하드웨어 구현 환경

컨베이어 벨트 자동화 시스템은 [표 1]과 같은 시스템 구현 환경을 가지고 있다. 라즈베리파이를 기반으로 구현하였기 때문에 운영체제는 '라즈비안'을 사용한다. 라즈비안은 '라즈베리파이 OS' 라고도 하며 라즈베리 파이 재단이 개발한 라즈베리파이 전용 운영체제이다. 리눅스를 기반으로 만들어졌으며 라즈베리파이 계열의 저성능 ARM CPU에 최적화가 되어있는 운영체제이다. 이 운영체제에서 쉽게 사용할 수 있는 개발 툴이 바로 'Thonny Python IDE'이다. 이 개발 툴을 사용하여 라즈베리파이에서 파이썬 프로그래밍을 할 수 있다. 하드웨어는 각종 모터와 센서 등으로 구성되어 있다. 영상 분석을 위한 카메라, 벨트 동작과 로봇팔 동작을 위한 DC, 서보 모터, 서버와의 연동을 위한 공유기로 구성되어 있다.

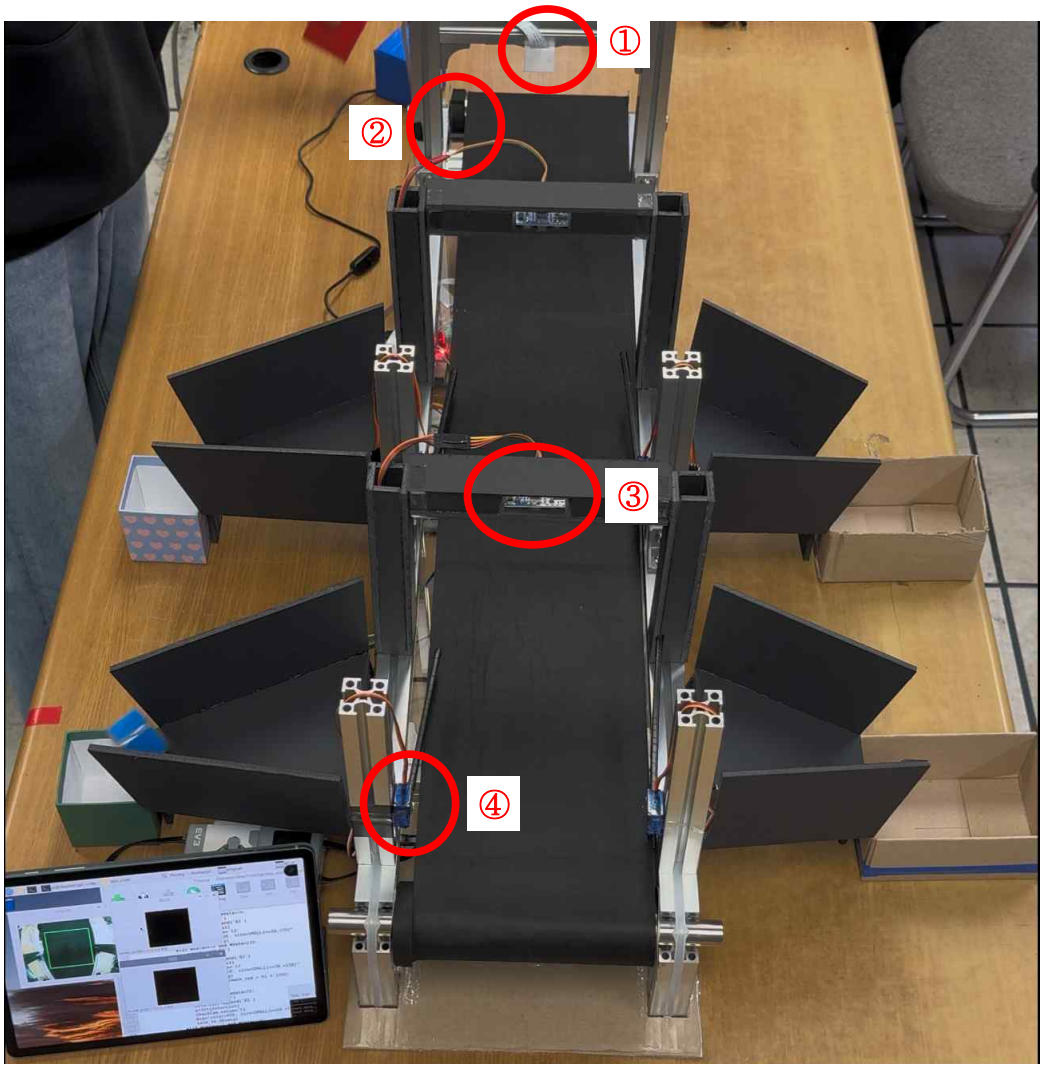
제 2절 구현된 주요 기능

구분	기능	설명
S/W	실시간 객체 탐지 영상	실시간으로 분류되는 객체 탐지 모습을 영상을 통해 화면에 표시하여 사용자에게 시각적으로 보여줌
	오류 제어	컨베이어 벨트에서 발생할 수 있는 다양한 오류 발생 시, 해당 오류에 대응할 수 있도록 각종 모터들을 제어함
	예외 처리	분류 항목에 포함되어 있지 않은 물류가 컨베이어 벨트에 올라올 경우에도 컨베이어 벨트가 정상 작동할 수 있도록 처리
H/W	영상 촬영	CSI 카메라를 통한 실시간 영상 촬영
	벨트 동작	물류가 자동으로 이동하여 분류될 수 있도록 DC 모터를 사용한 벨트 동작
	로봇팔 동작	객체 탐지가 완료된 물류를 서보 모터를 통해 물리적으로 분류시킴
	물류 위치 탐지	초음파 센서를 활용한 물류 위치 탐지

[표 2] 소프트웨어와 하드웨어 구현 내용

컨베이어 벨트 자동화 시스템에 구현된 주요 기능들은 [표 2]와 같다. 먼저 소프트웨어적인 부분에서는 자동화 시스템의 필수 요소인 실시간 객체 탐지 기능이 구현되었다. 카메라를 통해 촬영된 실시간 영상을 통해 물류의 탐지가 이루어지고, 탐지된 내용을 바탕으로 물류 분류가 이루어지게 된다. 물류 분류는 구현한 하드웨어 장비들을 이용하여 물리적으로 이루어진다. 이 외에도 오류, 예외 처리 기능을 추가하여 컨베이어 벨트 자동화 시스템에서 발생할 수 있는 다양한 상황에 대처할 수 있도록 시스템을 구현하였다.

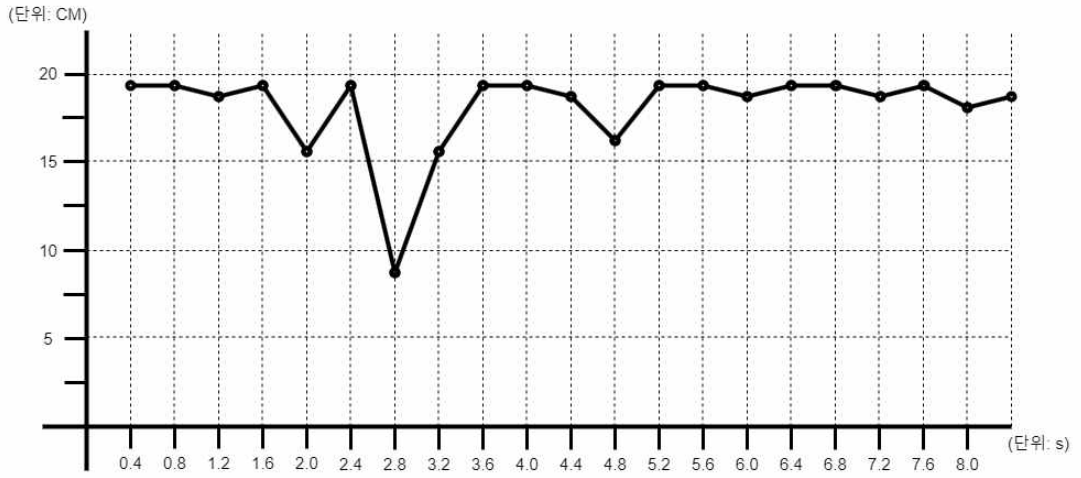
제 3절 구현 결과



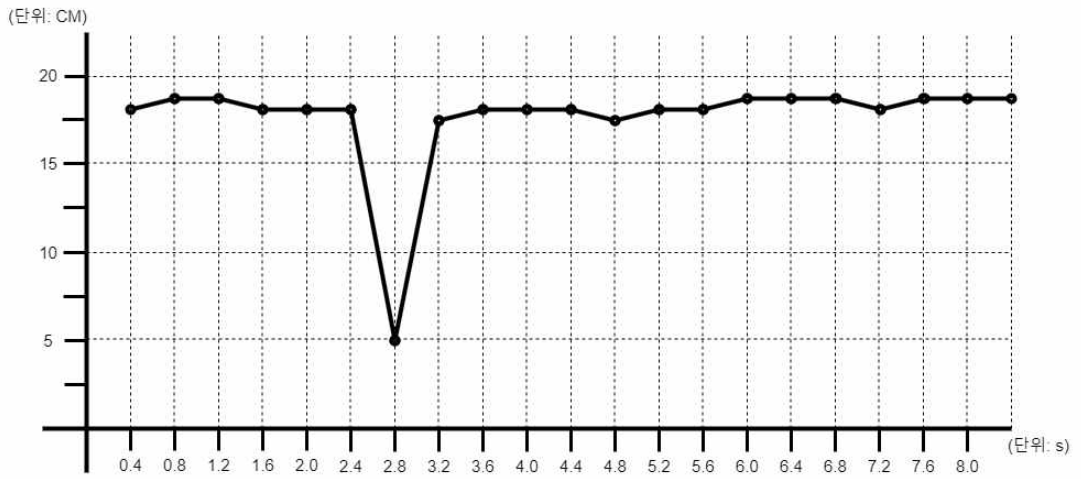
[그림 15] 컨베이어 벨트 자동화 시스템 최종 구현 모습

컨베이어 벨트는 위 [그림 15]과 같이 최종적으로 구현이 되었다. 설계 단계에서 계획한 대로 프로파일을 재단하여 구조물의 뼈대를 제작하였고, 컨베이어 벨트 양 끝에는 롤러가 위치하고, 구조물에 각종 카메라, 모터, 초음파 센서를 부착하는 방식을 사용하였다. 컨베이어 벨트의 앞부분은 ①번 부분이다. ①번은 CSI 카메라가 설치된 지점이다. 물류가 처음으로 올려지는 곳에 카메라가 위치해 있으며 이 지점에서 객체 탐지 과정이 이루어지게 된다. ②번은 DC 모터와 체인으로 연결되어 벨트를 움직일 수 있게 해주는 연결 지점이다. 해당 지점 하단에 DC 모터가 위치해 있고, DC 모터가 동작함에 따라 체인으로 인해 이 지점도 같이 굴러가게 되고 같은 롤러에 위치해 있는 벨트 전체를 움직일 수 있게 해준다. 또한, 벨트를 부

직포로 제작하였으므로 벨트 처짐 현상을 예방하기 위해 벨트 아래쪽에 투명 아크릴판을 설치하여 무거운 물류가 올려지더라도 벨트가 처지지 않고 손상 없이 팽팽한 모습을 유지할 수 있도록 하였다. ③번은 초음파 센서가 설치된 위치이다. 초음파 센서의 경우 측면과 상단 두 곳에 설치하여 연구를 진행해 보았다. 기존에 설계했던 방식은 측면 설치 방식인데, 아래 [그림 16]과 같이 불안정한 거리 측정값을 보여주었다. 사용한 초음파 센서 모듈은 HC-SR04인데 두 개의 원기둥이 각각의 신호를 송수신하는 역할을 한다. 그런데 측면에 설치해두게 된다면 물류의 최저 높이에 맞게 설치해야 할 뿐만 아니라 낮게 설치하다 보면 벨트의 굴곡진 부분으로 인해 초음파 센서의 두 개의 원기둥에 신호가 정상적으로 들어오지 못하게 되는 일도 발생하였다.



[그림 16] 측면에 설치한 초음파 센서 거리 측정 그래프



[그림 17] 상단에 설치한 초음파 센서 거리 측정 그래프

따라서 초음파의 위치를 상단으로 변경하였더니 측면에 설치하였을 때보다 안정적인 거리 측정 결과를 보여주었다. 위 [그림 17]을 보면 이전 [그림 16]보다 확실히 안정된 그래프 모습을 볼 수 있다. 이전 그래프보다 굴곡이 심하지 않고, 물류가 탐지되었을 때만 확실하게 그래프가 하강하고 바로 이전 상태로 복귀하는 모습을 볼 수 있다. 따라서 상단에 설치하기 위해 폼보드를 사용하여 임시로 구조물을 추가로 제작한 후 [그림 15]과 같은 모습으로 초음파 센서를 설치하게 되었다. ④번은 서보 모터가 설치된 위치이다. 서보 모터에는 긴 막대가 연결되어 서보 모터 동작에 따라 벨트의 경로를 막았다 열었다 해주며 물류를 분류시켜주는 역할을 한다. 이번 연구에서는 총 4가지의 물류를 분류하도록 설정하였기 때문에 4개의 서보 모터를 설치하였다. 이제 프로그램을 실행하여 동작 원리에 대해 자세히 살펴본다.

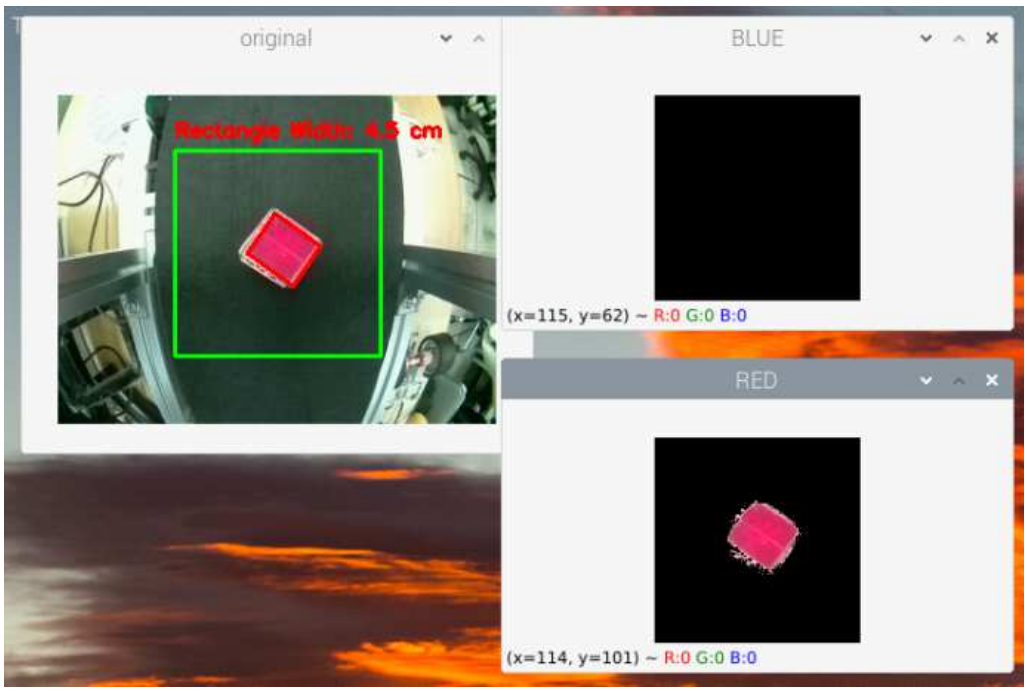
main.py (마스크 처리된 영상 출력 방법 예시)

```
...

# 프레임에서 ROI(region of interest: 관심영역) 추출
roi =frame[roi_y:roi_y+roi_h, roi_x:roi_x+roi_w]

# BGR -> HSV
hsv =cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

# hsv 범위 지정
lower_blue =np.array([90, 50, 100])
upper_blue =np.array([130, 255, 255])
lower_red1 =np.array([0, 50, 100])
upper_red1 =np.array([10, 255, 255])
lower_red2 =np.array([160, 50, 100])
upper_red2 =np.array([180, 255, 255])

# HSV img -> blue, red mask (색상 영역 추출)
mask_blue =cv2.inRange(hsv, lower_blue, upper_blue)
mask_red1 =cv2.inRange(hsv, lower_red1, upper_red1)
mask_red2 =cv2.inRange(hsv, lower_red2, upper_red2)
mask_red=mask_red1+mask_red2

# mask 영역에서 서로 공통으로 겹치는 부분 출력
res1 =cv2.bitwise_and(roi, roi, mask=mask_blue)
res3 =cv2.bitwise_and(roi, roi, mask=mask_red)
...
# 영상 출력
cv2.imshow('original', frame)
cv2.imshow('BLUE', res1)
cv2.imshow('RED', res3)
# 윈도우 위치 설정
cv2.moveWindow('original', 50, 100)
cv2.moveWindow('BLUE', 400, 100)
cv2.moveWindow('RED', 400, 350)
```

컨베이어 벨트 자동화 시스템 프로그램을 실행하면 사용자는 컨베이어 벨트의 앞부분 상황을 실시간으로 관찰할 수 있는 화면 3가지가 나온다. 왼쪽 큰 화면에는 컬러 모드 화면에 객체 탐지 가능 범위가 그려진 모습을 볼 수 있고, 오른쪽에 표시되는 두 개의 화면에서는 객체 탐지에 사용될 물류의 색상 (이번 연구에서는 빨간색과 파란색)이 제대로 탐지되었는지 확인할 수 있는 색상별 마스크 처리된 화면을 볼 수 있다. 또한, 컨베이어 벨트의 실행과 동시에 DC 모터를 속력 30으로 동작시켜 벨트를 설정해둔 진행 방향으로 이동하게 만들어주고, 맨 앞에 위치한 초음파 센서는 거리 측정을 시작하게 된다.

main.py (물류 리스트 작성법 예시)

```
        if checkR<10 and np.sum(mask_red >0) >1300: # 물류 색상 확인
            checkR+=1 # 인식 오류 방지를 위한 확인용 데이터
            if checkR>9: # 정상적인 인식이라면
                if Wdata>=20 and Wdata<70: # 작은 크기 물류라면
                    colorList.append('R1') # 물류 리스트에 추가
                    ...
                elif Wdata>=70 and Wdata<130: # 큰 크기 물류라면
                    colorList.append('R2')
                    ...
```



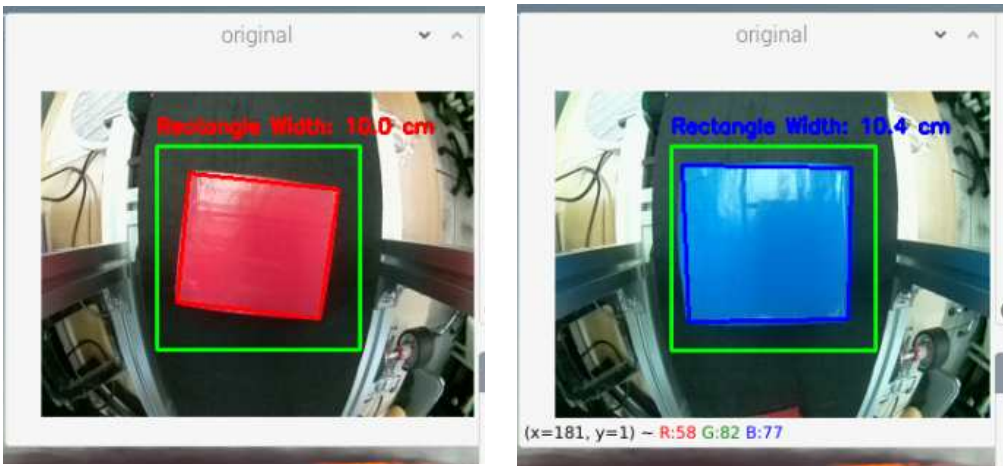
[그림 18] 빨간색 작은 물류 탐지 모습

위 [그림 18]는 컨베이어 벨트 위에 분류시킬 물류가 올라온 모습이다. 한 변의 길이가 약 4.5cm인 정육면체 물류가 컨베이어 벨트 위에 올려졌고 카메라의 객체 탐지 범위(초록색 상자)에 물류가 들어왔다. 객체 탐지 알고리즘이 분류할 물류인 것을 탐지하고 물류의 색상(글씨, 경계선 색상)과 너비를 측정하여 왼쪽 화면(original)에 시각적으로 보여주게 된다. 이번 연구에서는 정육면체를 활용하였기 때문에 너비의 정보만을 제공하는 것을 예시로 보였지만, 소스 코드에는 객체의 너비, 높이, x, y좌표를 측정할 수 있도록 작성하였다. 또한, 크기의 경우 오차 범위를 지정할 수 있어서 완전히 크기가 일치해야만 같은 물류로 분류할 수 있게도

사용할 수 있다. 추가적으로 얼마나 정확하게 객체의 색상을 탐지하였는지 오른쪽 화면(RED)을 통해 볼 수 있다. 이 정보는 컨베이어 벨트 주변 환경의 변화(ex. 조명 변화)로 인해 객체 탐지가 제대로 되지 않을 경우 어떠한 문제인지, 얼마만큼의 값을 변경해주어야 할 것인지를 쉽게 판단하기 위해 제공해주고 있다.

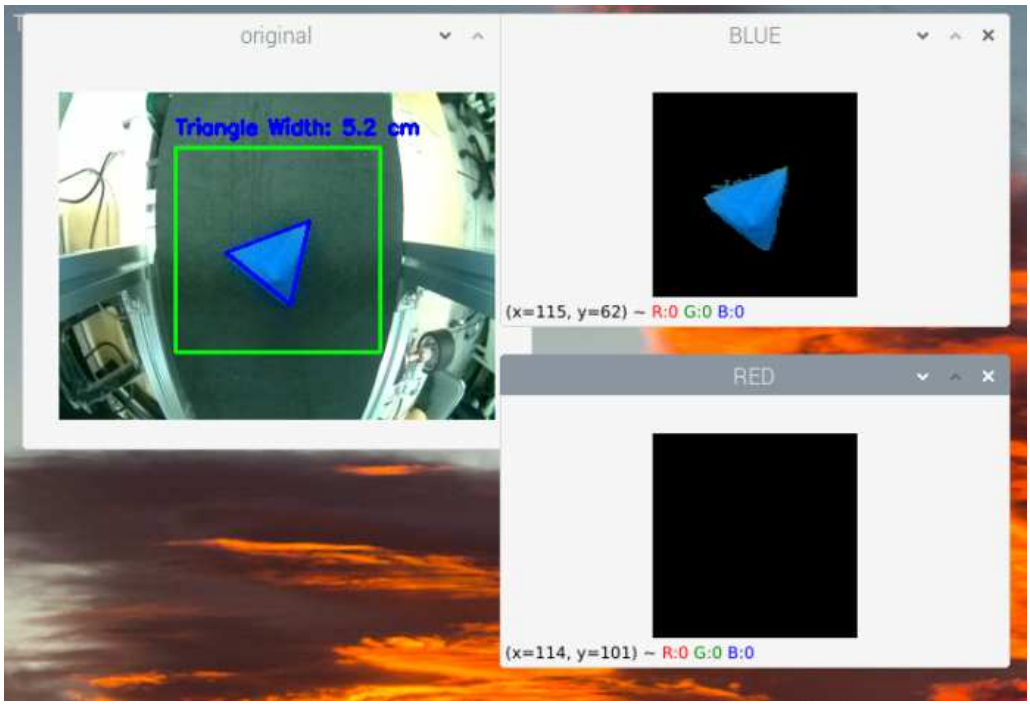
[그림 19] 파란색 작은 물류 탐지 모습

이번에는 비슷한 크기의 다른 색상의 사각형 물류가 올라온 모습이다. 한 변의 길이가 약 3.4cm인 정육면체 물류가 컨베이어 벨트 위에 올려졌고 카메라의 객체 탐지 범위(초록색 상자)에 물류가 들어왔다. 위에서 살펴본 빨간색 상자와 동일한 방식으로 객체 탐지 알고리즘을 사용하여 위 [그림 19]과 같은 탐지 화면을 볼 수 있다.



[그림 20] 두 가지의 큰 물류 탐지 모습

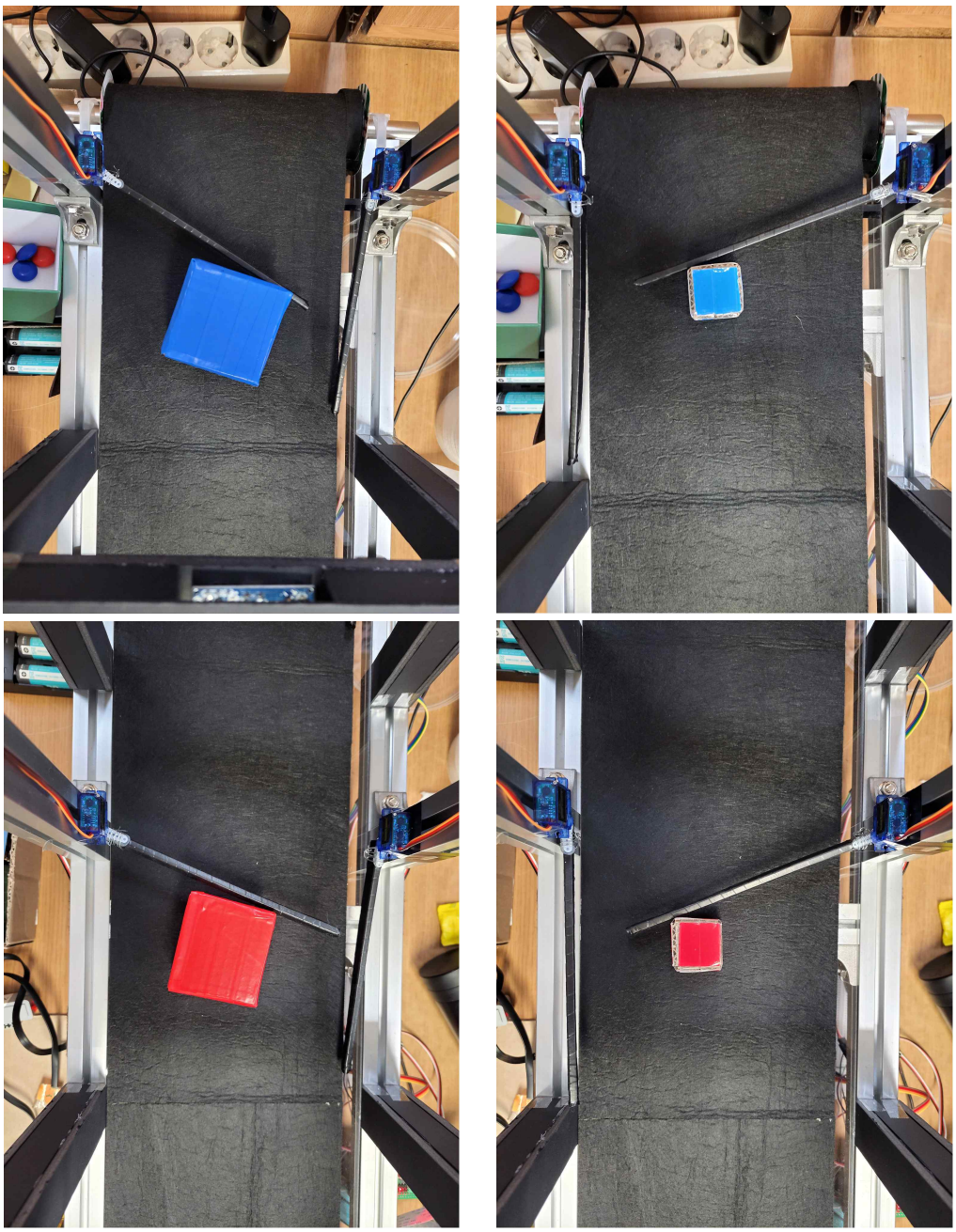
이번에는 이전 작은 크기의 물류와 색상은 같으면서 크기가 달라진 부분에 대해 살펴본다. 빨간색과 파란색 물류 모두 이전에 탐지했던 물류들보다 크기가 커진 모습을 볼 수 있다. 위 [그림 20]에서도 이전보다 폭이 커진 10.0cm, 10.4cm로 크기를 탐지한 것을 볼 수 있다.



[그림 21] 삼각형 물류 탐지 모습

추가로 크기와 색상 이외에도 모양에 따른 분류가 가능하다는 것을 보여주기 위해 삼각형 물류를 탐지하는 모습을 추가하였다. 이번 연구에서는 물류 상자를

분류하는 모습을 담기 위해 육각형의 상자를 대상으로 분류 작업을 진행하였지만, 객체 탐지 기술의 장점을 보이기 위해 모양에 따라서도 분류가 가능한 부분도 구현하였다. 위 [그림 21]는 정사면체를 컨베이어 벨트 위에 올린 모습이다. 이전 정육면체의 화면과는 다르게 초록색 객체 탐지 범위 상자 위에 Triangle이라는 문구의 변화가 생겼다. 이처럼 원하는 물류에 대한 정보만 있다면 모양만 다르더라도 서로 다른 물류로 구분하여 분류할 수 있다.



[그림 22] 로봇팔 동작 모습 (빨간색 물류들이 컨베이어 벨트 앞부분)

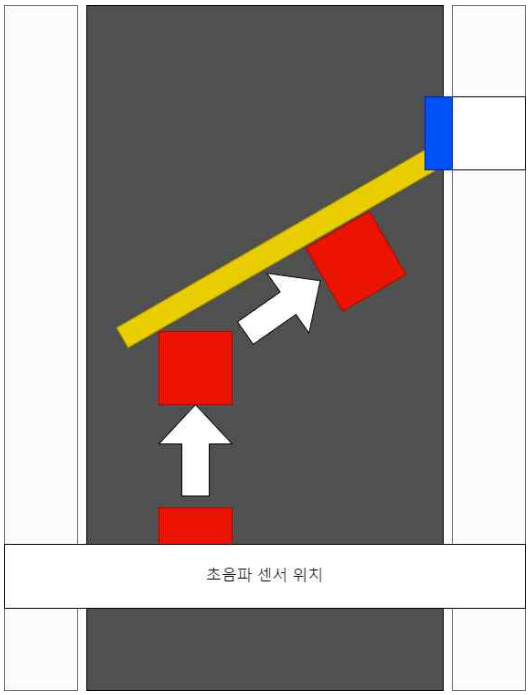
위 [그림 22]은 지금까지 살펴본 객체 탐지 결과와 초음파 센서를 통한 물류 위치 탐지를 바탕으로 동작시킨 서보 모터(로봇팔)의 모습이다. 빨간색 계열의 물류는 컨베이어 벨트의 앞부분에서 좌, 우로 분류되고, 파란색 계열의 물류는 컨베이어 벨트 뒷부분에서 좌, 우로 분류된다. 크기로 본다면 작은 물류 상자는 컨베이어 벨트의 우측 부분으로, 큰 물류 상자는 컨베이어 벨트의 좌측 부분으로 분류된다. 물류를 분류 시킬 때 중요한 점은 로봇팔이 물류를 분류시킬 때 물류가 경로를 이탈하지 않고 정확히 원하는 위치로 들어가 분류될 수 있도록 해야 한다.

distance_check.py (초음파 센서를 활용한 서보 모터 동작 원리 예시)

```
...
if dis<15.5:
    if checkCam.value=='t1': # 빨간색 물류라면
        time.sleep(2) # 2초 후
        checkD.value='t'# 로봇팔 닫을수 있게 명령 내림
    else: # 빨간색 물류가 아니라면
        checkD.value='t3'# 2번째 초음파 센서로 측정 시작
...
```

main.py (초음파 센서를 활용한 서보 모터 동작 원리 예시)

```
...
if len(colorList)>0 and checkD.value=='t': #닫으라는 명령 받음
    if colorList[0]=='R1': #물류 리스트의 첫 번째 물류 확인
        colorList.pop(0) #해당 물류 꺼내옴
        kit.servo[0].angle = 90 #로봇팔 닫음
        checkD.value='f' #닫으라는 명령 기본값으로 초기화
...
```



[그림 23] 물류 분류 원리

이 부분을 해결하기 위해 time sleep을 사용하여 물류가 초음파 센서 아래를 지나는 순간 일정 시간(여기서는 2초)이 지난 후 로봇팔을 닫아 물류를 분류할 수 있도록 하였다. 이 방법을 사용하면 위 [그림 23]에서 보듯 벨트는 계속 진행 방향으로 이동 중이기 때문에 물류가 로봇팔의 경사를 따라 분류될 장소로 미끄러져 내려가기 때문에 벨트의 어느 위치에 있더라도 로봇팔을 닫았을 때 이탈하지 않고 정확하게 분류될 수 있었다.

[그림 24] 분류 대상이 아닌 물류 처리 모습

마지막으로 분류하려는 물류 정보에 없는 물류가 컨베이어 벨트에 올라왔을 경우에 대한 처리 결과의 모습이다. 이번 연구에서는 빨간색과 파란색 상자를 크기별로 분류하는 작업을 진행하였다. 때문에 다른 색상, 크기, 모양인 경우 기존과는 다른 처리 방식이 들어가야 한다. 따라서 분류 대상이 아닌 객체가 탐지된 경우에는 위 [그림 24]처럼 컨베이어 벨트 맨 끝부분으로 분류시키도록 하였다. 분류 대상이 아닌 객체의 탐지 방법은 컨베이어 벨트에 물류가 올라왔음에도 카메라를 통해 탐지가 불가능한 상태로 계속 이동하여 첫 번째 초음파 센서에 감지되었을 때 분류 대상이 아닌 물류가 올라왔음을 의심하는 단계로 넘어간다. 이때까지는 단순히 초음파 센서의 오류일 수도 있기 때문에 확정이 아닌 의심 단계까지만 진행이 되어야 한다. 첫 번째 초음파 센서에 감지되었기 때문에 일정 시간 이내에(이번 연구에서는 3초) 두 번째 초음파 센서에도 물류가 감지된다면 분류 대상이 아닌 물류로 확정하고 컨베이어 벨트 맨 끝부분으로 이동시킨 후, 서버에 분류 대상이 아닌 물류가 탐지되었음을 알린다.

세 가지 오류 발생 상황에 대한 오류 처리 기능도 구현하였다. 첫 번째로 맨 앞에 위치한 첫 번째 초음파 센서 오류 발생에 대한 처리 방법을 살펴본다. 첫 번째 초음파 센서의 경우 모든 물류에 대한 위치 정보를 제일 처음으로 측정하는 센서로, 빨간색, 파란색, 미분류 대상 물류 모두 이 초음파 센서 아래를 지나가게 된다. 때문에 카메라에서 물류가 탐지된다면 적어도 3초 이내에 첫 번째 초음파 센

서 아래를 지나가야 하는데 첫 번째 초음파 센서에서 물류가 지나갔음을 인식하지 못한 경우 벨트 동작을 정지시킨 후 서버에 '물류 위치를 알 수 없음' 메시지를 전송하도록 하였다. 이후 서버에서 물류가 정상적으로 첫 번째 초음파 센서 아래를 지났거나 물류가 이동 중 문제가 생겨 첫 번째 초음파 센서까지 도달하지 못했음을 파악한 후 해결하였다는 메시지를 보낸다면 다시 벨트를 동작시켜 다음 작업을 이어가도록 하였다.

두 번째로 초음파 센서 오류 발생에 대한 처리 방법에 대해 살펴본다. 초음파 센서는 컨베이어 벨트가 동작하는 동안 계속 동작하며 물류가 지나가는지를 확인하는 역할을 한다. 하지만 간혹 초음파 센서 자체의 오류일 때도 있고 물류가 컨베이어 벨트 어딘가에 끼거나 하는 상황이 발생할 경우 첫 번째 초음파 센서에서는 탐지가 되었다고 나오는데 두 번째 초음파 센서에서는 아무리 기다려도 물류가 지나가는 것을 탐지하지 못하는 상황이 발생할 수 있다. 이러한 경우를 대비하여 첫 번째 초음파 센서가 물류를 탐지한 후 3초가 지났음에도 두 번째 초음파 센서에 물류가 탐지되지 않는다면 DC 모터의 속도를 0으로 설정하여 벨트를 정지시키고 서버에 컨베이어 벨트에서 물류가 사라짐이 의심되는 오류가 발생하였다는 것을 알린다. 이후 서버에서 물류가 사라진 것인지 아니면 초음파 센서의 측정 오류인지를 확인하였다는 메시지를 보내면 오류를 확인하였다는 사실이기 때문에 벨트를 다시 동작시키게 구현하였다. 이 오류 처리 방식은 카메라에서 파란색 물류가 탐지되었을 때(두 번째 초음파 센서를 지나야 하는 물류인 경우)와 분류 대상이 아닌 물류가 지나갔을 때, 또는 아무런 물류가 카메라 앞으로 지나가지 않았을 때만 동작하게 된다. 즉, 빨간색 물류(첫 번째 초음파 센서만 지나는 물류)인 경우에는 적용되지 않는 오류 처리 방법이다.

마지막으로는 서버와의 연결이 끊어진 경우에 대한 오류 처리 방법이다. 컨베이어 벨트와 서버는 항상 연결되어 있어야 한다. 컨베이어 벨트에서 물류를 분류하는 즉시 서버에 기록이 되어야 물류가 분류된 이후 물류 로봇을 호출하여 수거할지 말지를 결정할 수 있기 때문이다. 서버에 연결되어있지 않는 동안 물류를 분류한다면 서버에서는 분류된 기록을 가지고 있지 않기 때문에 적절한 분류 타이밍에 물류 로봇을 호출하는데 어려움이 생길 수밖에 없다. 따라서 컨베이어 벨트와 서버 간의 연결이 끊어진다면 컨베이어 벨트의 모든 동작을 정지 및 초기화시키기고 시스템이 종료되도록 하였다. 사용자는 컨베이어 벨트 시스템만을 재시작하여 서버와 재연결을 시도하거나 서버도 같이 재시작하여 재연결 해주어야 한다.

제 5장 결론과 기대효과 및 해결 과제

제 1절 결론

본 논문에서는 스마트팩토리 구축 시 필수적 요소 중 하나인 컨베이어 벨트 자동화 시스템을 객체 탐지 기술을 활용하여 다양한 종류의 물류를 분류할 수 있는 시스템을 제안하였다. 제안하고자 하는 컨베이어 벨트 자동화 시스템에서는 객체 탐지 기술을 사용함으로써 다양한 물류를 추가적인 포장 없이 분류할 수 있도록 효율성을 증가시켰다. 바코드 등과 같은 물류 분류를 위한 추가적인 수단을 요구하지 않고 분류하고자 하는 물류 자체를 카메라로 인식하여 어떤 물류인지 탐지하고, 탐지 결과를 바탕으로 물류를 분류시킨 후 서버에 분류 정보를 기록할 수 있도록 하였다. 또한, 사용자는 서버를 통해 컨베이어 벨트의 동작을 원격으로 관리할 수 있으며 오류에 대한 처리도 원격으로 확인할 수 있게 구현하였다. 서버에서 시각적인 분류 현황을 관찰할 수 있기 때문에 컨베이어 벨트 동작에 대한 신뢰도를 확보하였다. 또한, 함께 개발하였던 물류 로봇과도 서버를 통해 연동되어 서로의 상황에 대한 소통이 이루어져 생산 라인의 연결성을 구축하고 유연성을 향상시킴과 동시에 생산성 향상까지 확보하였다고 볼 수 있다.

제 2절 기대효과

대부분의 일반 공장에서는 자동화 및 지능화가 이루어지지 않은 경우가 많다. 수동 작업이 여전히 많고, 생산 흐름은 상대적으로 간단할 수 있지만 스마트팩토리의 경우 센서, IoT, AI 등의 기술을 통합하여 생산과정을 자동화하고 최적화한다. 이번 연구 주제인 컨베이어 벨트 자동화 시스템이 도입된 스마트팩토리를 통해 생산 데이터를 실시간으로 수집하고 분석하며, 의사 결정을 지원하여 최적의 운영 상태를 유지할 수 있도록 도움을 줄 수 있을 것으로 기대한다. 기존 컨베이어 벨트 자동화 시스템에서 사용하는 바코드 인식, 물리적인 방식의 무게 및 크기 인식과는 다르게 더 다양한 분야, 다양한 환경에서 활용될 수 있다. 컨베이어 벨트 시스템에 분류하고자 하는 물류 정보만 입력한다면 컨베이어 벨트의 교체 없이 그대로 사용할 수 있기 때문이다. 더 나아가 상품의 손상까지도 분류할 수 있을 것으로 기대한다. 이번 연구에서는 물류를 분류하는 기능만 넣었지만, 객체 탐지 기술을 이용하면 물건의 손상도도 측정이 가능하기 때문에 추가적인 품질 관리 없이도 물류 분류가 가능할 것으로 기대된다. 품질 관리 과정만 축소된다면 물류 분류 과정에 있어서 엄청난 효율성 및 생산성 증가가 예상된다. 안정성 측면에서도 많은 변화가 있을 것으로 예상한다. 공간적, 비용적, 시간적 문제로 직접 연구하지는 못했지만, 객체 탐지 기술을 이용하면 컨베이어 벨트 전체의 동작 흐름을 카메라를 통해 촬영하여 컨베이어 벨트에서 발생할 수 있는 사고 상황에 대응할 수 있을 것이다. 예를 들어 컨베이어 벨트 근처에 사람이 접근하는 경우 관리자에게 이를 알리고, 컨베이어 벨트에 일정 거리 이상 가깝게 접근한다면 컨베이어 벨트를 정지시켜 사고를 예방할 수 있을 것으로 기대한다. 기존 컨베이어 벨트 시스템보다 다양한 센서와 카메라를 사용하기 때문에 초기 비용이 조금 더 발생할지 몰라도 안정성 및 생산 효율을 생각한다면 시간이 갈수록 비용적인 이득을 줄 수 있다고 기대한다.

제 3절 해결 과제

현 시스템에서 제일 큰 문제는 초음파 센서였다. 비용적인 문제로 비교적 저렴한 HC-SR40 모듈을 사용하였는데, 거리 측정 도중 모듈 동작이 멈추거나 측정값 오류(실제 거리와는 전혀 다른 값 반환)가 자주 발생하였다. 이 문제는 초음파 센서를 새 상품으로 교체하였음에도 발생하였기 때문에 초음파 센서가 아닌 다른 모듈을 사용하여 물류 위치를 탐지하는 것이 좋을 것 같다. 컨베이어 벨트는 계속 움직이기 때문에 소리의 반사를 이용한 방법보다는 적외선 센서 모듈을 이용하여 물류의 위치를 탐지하는 것이 더 정확할 것으로 예상한다. 적외선 센서에 관한 공부를 진행하고 적외선 센서로 교체하는 것이 컨베이어 벨트 동작을 더 원활하게 해줄 것으로 예상한다.

Mask R-CNN 모델에 관한 연구도 진행하였는데, 개발한 컨베이어 벨트 시스템에 적용하려고 하니 오류도 발생하고, 프레임 드랍 현상이 너무 심해 적용하지 못하고 따로 윈도우 환경에서 테스트해 보는 방법밖에 하지 못하였다. 이는 라즈베리파이 성능상 각종 모터를 동시에 제어하면서 알고리즘을 동작시키기에는 무리가 있을 것으로 판단되어 컨베이어 벨트 시스템 하드웨어를 더 좋은 것으로 교체하게 된다면 적용시켜볼 예정이다.

참 고 문 헌

[1] [그림 1] 참고 자료

https://kosis.kr/statHtml/statHtml.do?orgId=101&tblId=DT_2KAA906

[2] [그림 2] + 스마트팩토리 솔루션 참고 자료

https://ssl.pstatic.net/imgstock/upload/research/industry/1625097016084.pdf

[3] [그림 3] Gradient-based detector 참고 자료

https://velog.io/@ganta/Object-detection

[4] [그림 4] Selective search 참고 자료

https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/

[5] [그림 5] Mask R-CNN 참고 자료

https://arxiv.org/pdf/1703.06870.pdf

[6] [그림 6] YOLO, [그림 7] Fast R-CNN vs YOLO 참고 자료

https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=JAKO202011161035249&dbt=JAKO&koi=KISTI1.1003%2FJNL.JAKO202011161035249

[7] 텐서플로 참고 자료

https://ko.wikipedia.org/wiki/%ED%85%90%EC%84%9C%ED%94%8C%EB%A1%9C

[8] Open CV 참고 자료

https://ko.wikipedia.org/wiki/OpenCV

[9] 라즈베리 파이 OS 참고 자료

https://ko.wikipedia.org/wiki/%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC_%ED%8C%8C%EC%9D%B4_OS