제17회 임베디드SW경진대회

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	엑스페디오	**팀장**	오택규
팀원	곽승윤	**팀원**	천우담
팀원	봉재민	**팀원**	지혜주

2. 개발완료보고서

0. 작품명

로트사이즈(Lot size) 기반 재고 관리시스템

<목차>

1. 개요

1.1. 작품 개요

본 시스템은 물품의 생산 또는 고객의 수요변화에 따른 출하량 관리를 위해 보유하고 있는 재료 물품이나 재고(Inventory)를 효율적으로 관리하는 제어시스템이다. 본 시스템은 생산 공정 라인을 구축하는 스마트팩토리(공장자동화) 관리용 임베디드 제어시스템과 영상처리를 활용한 산업차량(Gripper 로봇)용 임베디드 제어시스템으로 구성된다. 생산 공정 라인의 상황에 따라 물품을 운반하고, 생산품에 대한 소비를 수요량으로 수집하는 빅데이터 개념을 도입하여, 수요 예측에 따른 재고 소비량을 분석해 생산 공정 라인의 효율성을 강화하는 것이다. 이 시스템은 재고 물품들은 색상/라인별로 구분하며, 산업차량(Gripper 로봇)은 영상처리기(이하 픽시)를 통한 입력 데이터값에 따라 지정된 생산 물품을 운반 및 데이터 수집하는 시스템이다. 이 시스템은 지그비 통신을 이용하여 구현하였지만 차후 IoT를 접목하여 생산 공정 간의 커뮤니케이션으로 공장자동화 및 생산관리 효율화 및 모니터링시스템에 적용이 가능하다.

1.2. 개발 필요성

생산 공정에서의 자재투입과 관련해서 중요한 과제로 삼고 있는 것은 원가절감 및 생산과 판매 촉진이다. 재고량을 분석하면 미래의 예상되는 수요나 공급의 변화에 따른 불확실성에 대처할 수 있다. 로트사이즈(Lot size; 재고/셋업 비용을 고려한 최적 생산 단위 결정)로 인해 발생하는 손실 재고를 줄임으로써 준비 비용을 절감하고 생산 공정 효율을 높일 수 있다.

1.2.1. 광공업생산 동향 (출하지수와 재고지수)

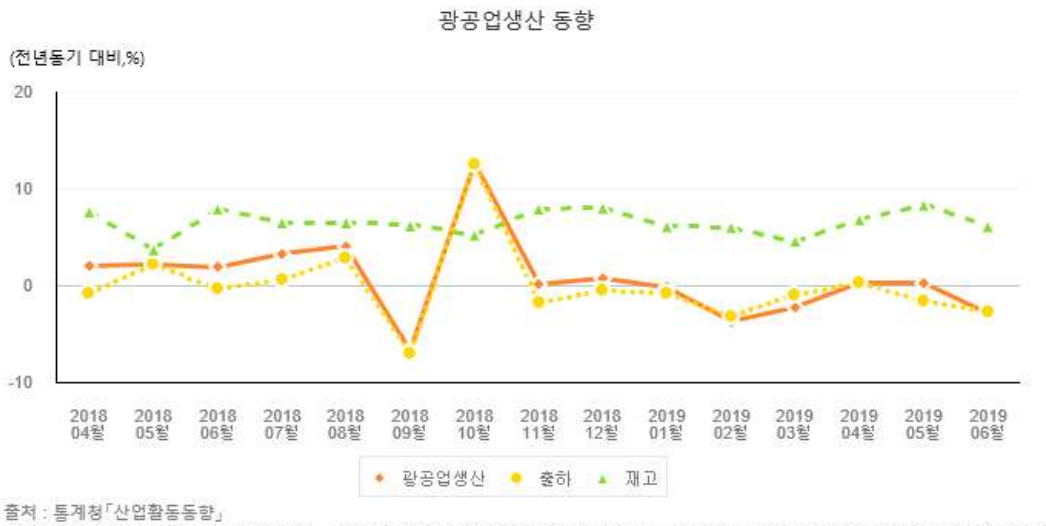


그림 1. 광공업생산 동향

<출처: 통계청 '산업활동동향'>

그림 1.에서 보듯이 광공업생산지수는 광업, 제조업, 전기/가스 사업을 대상으로 어느 기간 동안 이루어진 산업 생산 활동의 수준을 나타내는 지표로써 전체경기의 흐름과 유사하게 움직이는 대표적인 동행지표이다. 이를 통해 물건이 판매되는 비율보다 재고 비율이 높다는 것을 의미하고 재고 비율이 높다는 것은 기업의 비용부담으로 작용한다. 이 재고 비율을 적정 수준으로 유지함으로써 손실을 최소화할 수 있도록 시스템을 구성한다.

1.2.2. 제조업 평균가동률

표 1. 광공업생산 동향 (단위 : 전년동월(기)비,[%])

	2017	2018	2019 02월	2019 03월	2019 04월	2019 05월	2019 06월	2019 07월
광공업생산	2.4	1.3	-3.8	-2.3	0.2	0.5	-2.6	0.6
출하	1.2	0	-3.3	-1.1	0.2	-1.4	-2.8	0.8
재고	8	8	6	4.6	6.8	8.4	6.3	8
제조업평균가동률[%]	73.3	73.5	70.3	71.5	72.7	72.1	72.2	74.8

우리는 경제가 불황일 때 '공장의 생산라인 일부의 가동을 중단했다.'는 소식을 종종 들을 수 있다. 경제가 불황일 때 제품이 판매되지 않으므로 기업주로서는 당연히 생산량을 조정하여 불필요한 생산 공정을 줄여나갈 것이다. 따라서 공장설비의 가동 현황은 시장경제의 단면을 잘 보여주는 지표가 된다. 표 1.에서 보듯이 출하량이 증가할 경우 그에 비례하여 재고량 증가를 보이고 있으나, 출하량이 감소할 경우는 그에 비례하여 재고량이 줄어들지 않는다. 이는 생산 공정의 효율화가 필요하다.

1.3. 개발 목표

본 시스템은 산업차량(Gripper 로봇)을 활용한 생산 공정 라인을 구축하고 수요에 따른 생산량 변화를 수집하는 빅데이터 개념을 도입하여 수요변화에 따른 이상적인 공급량을 분석해 생산 공정 라인을 효율적으로 관리하는 것을 목표로 한다. 이에 따른 개발 목표는 다음과 같다.

- 생산 공정 라인을 위한 산업차량(Gripper 로봇) HW 개발
- 생산 공정 라인을 위한 산업차량(Gripper 로봇) SW 개발
- 생산 공정을 위한 자동 생산라인 HW 구축
- 생산 공정을 위한 자동 생산라인 SW 개발
- 생산 공정 자동화를 위한 통신 시스템 구축
- 생산/소비 효율화를 위한 자료 빅데이터화
- 효율적인 생산 공정 관리를 위한 모니터링시스템 개발

본 개발 목표를 위해 생산라인을 구축하여 물품의 유/무를 판별하고 산업차량(Gripper 로봇)에 송신하여 산업차량(Gripper 로봇)이 물품을 운반할 수 있도록 한다. 산업차량(Gripper 로봇)이 작업을 수행한 후에는 작업 종료에 따른 신호를 지그비 통신을 통해 생산 공정으로 송신한다. 이때, 물품의 생산량을 수집하여 필요한 시기에 주문함으로써, 재고의 부족으로 인한 생산 공정 라인의 가동 중지를 방지하는 등 생산 공정 관리(ex. 생산 물품 재고) 효율을 향상시킨다.

2. 개발 환경 설명

2.1. Hardware 구성

2.1.1. 생산 공정 라인 전체 시스템 구성

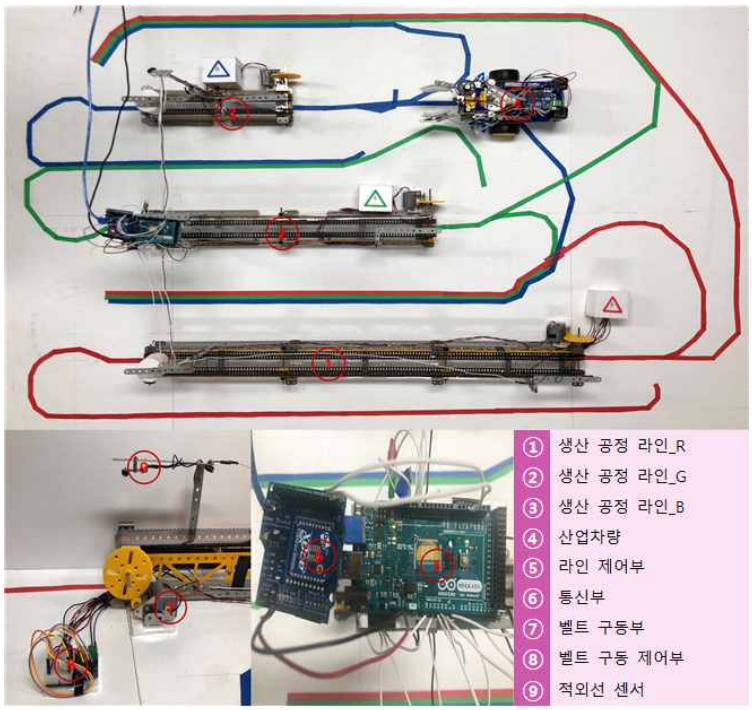


그림 2. 생산 공정 라인 전체 시스템 구성

그림 2.는 생산 공정 라인을 구축하기 위해 자동 생산라인(컨베이어 벨트시스템)과 이를 물품에 따라 R, G, B 색상으로 구현하였다. 각각의 생산라인은 색상에 따른 3개의 물품이 있고 이 물품의 시작점(**물품이 생산 준비 중임을 의미**) 각 3개, 도착점(**물품을 재 생산해야 함을 의미**) 각 3개로 총 6개가 존재하며 물품이 입력됨으로써 산업차량(Gripper 로봇)과 생산 공정 라인이 동작하게 된다. 각각의 생산라인에 적외선 센서를 사용하여 물품의 입/출력을 감지하여 라인의 동작 유/무를 결정한다.

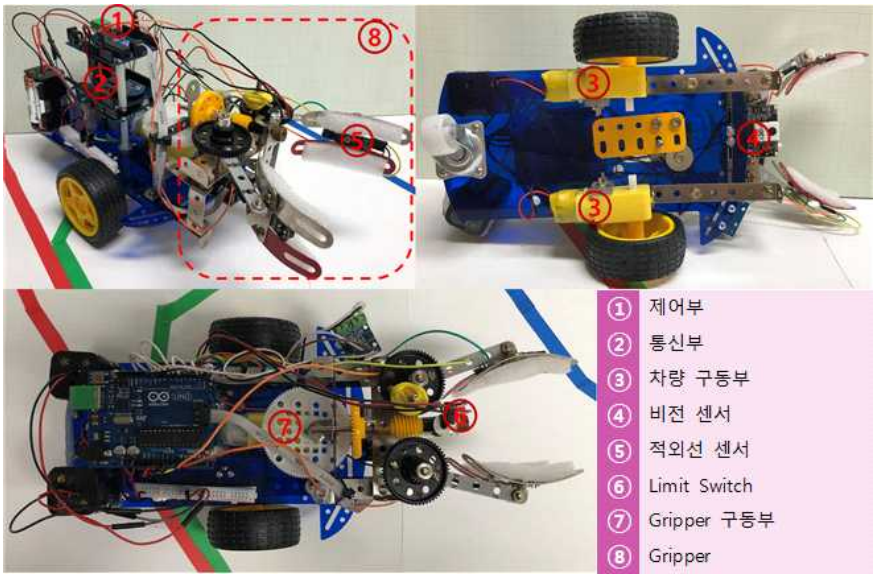


그림 3. 생산 공정 라인용 산업차량(Gripper 로봇)

그림 3.은 생산 공정 라인용 산업차량(Gripper 로봇)이다. 이 산업차량은 Pixy2 CMUcam5 Smart Vision Sensor(이하 픽시)를 사용하여 생산라인을 영상처리로 학습하고, 학습된 영상처리 데이터를 통해 생산 물품에 따른 색상/라인을 따라 움직이도록 설계하였다. 각각의 생산 물품에 따라 색상/라인을 R, G, B로 표현하여 산업차량(Gripper 로봇)이 이동할 수 있도록 하였다.

2.1.2. 생산 공정을 위한 자동 생산 라인 구성

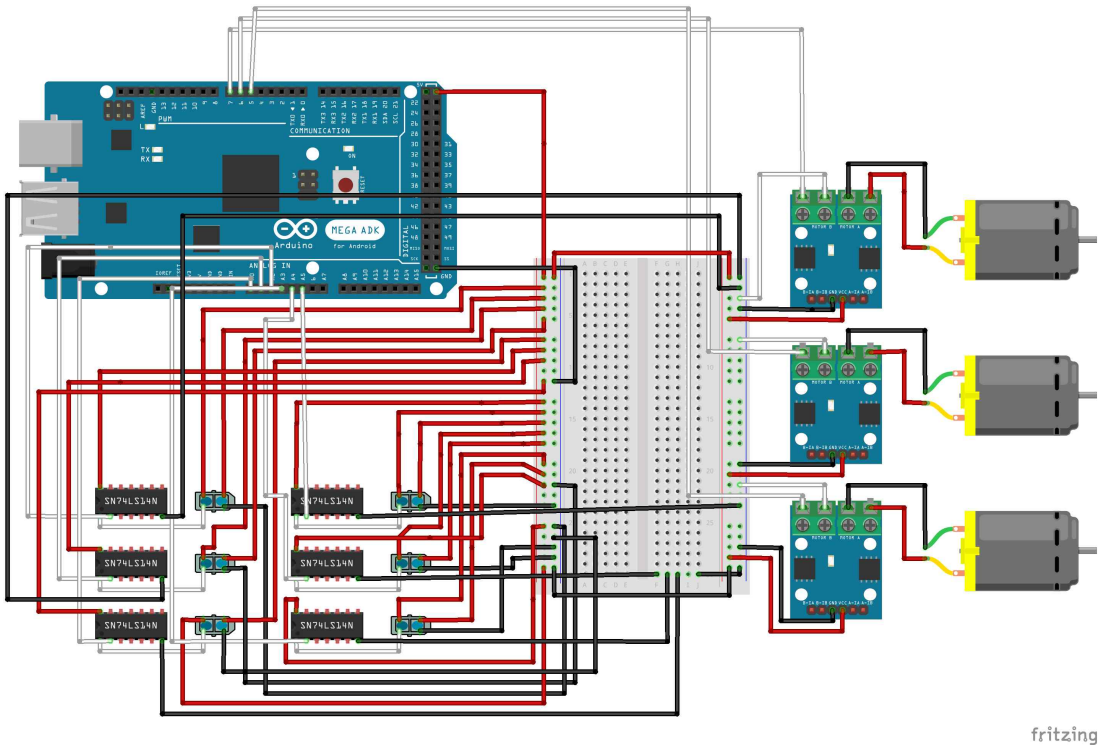


그림 4-(a). 생산 공정 라인 제어 결선도

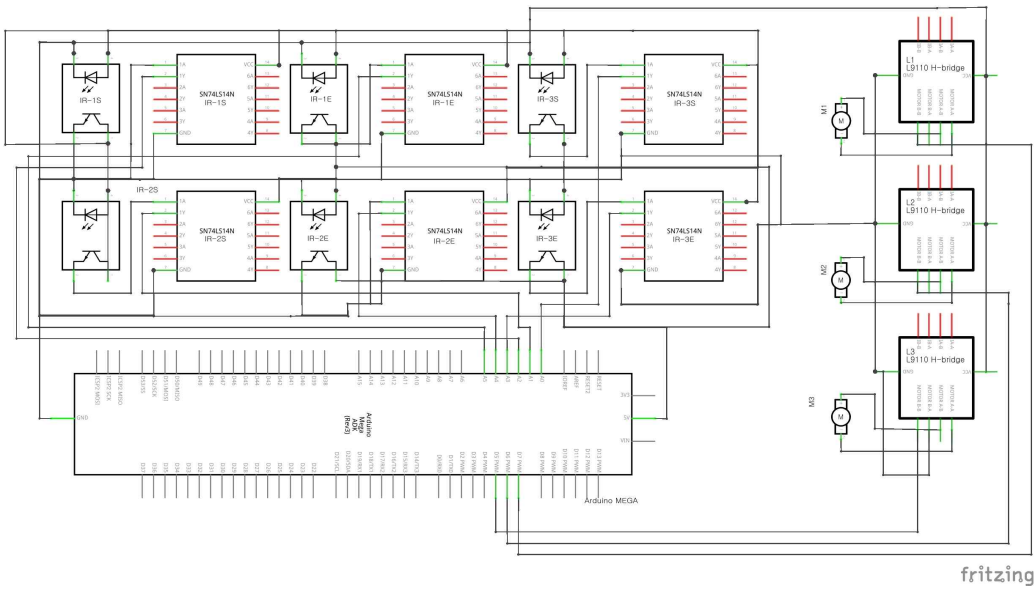


그림 4-(b). 생산 공정 라인 제어 회로도

그림 4.는 생산 공정 라인의 Hardware 제어 결선도 및 회로도이다. 생산 공정 라인은 R, G, B 총 3개의 생산 공정 라인으로 구성된다. 생산 공정 라인의 시작점(**물품이 생산 준비 중임을 의미**)와 도착점(**물품을 재 생산해야 함을 의미**)에는 적외선 센서를 부착하여 물품의 입/출력을 확인할 수 있다. 이에 따라 물품의 위치와 상태에 따라 직류전동기(DC Motor)의 동작을 제어할 수 있다.

생산 공정 라인의 시작점의 적외선 센서에서 물품을 확인하면 임베디드 제어 보드1(Arduino Mega; 이하 Mega)에서 직류전동기 드라이브에 모터 구동 신호인 PWM(Pulse Width Modulation) 신호를 인가한다. 모터 구동 드라이브와 연결된 직류전동기가 동작하여 물품을 운반하게 된다. 물품이 생산 공정 라인의 도착점에 도달하면 도착점의 적외선 센서 신호를 입력받아 Mega에서 직류전동기의 동작을 멈추게 한다. 이때, 생산 공정 라인의 시작점(**물품이 생산 준비 중임을 의미**)에서 도착점(**물품을 재 생산해야 함을 의미**)까지 물품이 이동하는데 걸린 시간과 도착점에 물품이 도착하였을 때 산업차량(Gripper 로봇)이 물품을 운반하기까지의 소요시간 등의 데이터를 모니터링시스템과 연동하며, 생산 공정 및 물품 재고, 생산에 필요한 물품, 수요 등의 정보를 데이터를 통해 축적시켜 빅데이터화 할 수 있다.

2.1.3. 생산 공정 라인을 위한 산업차량(Gripper 로봇) 구성

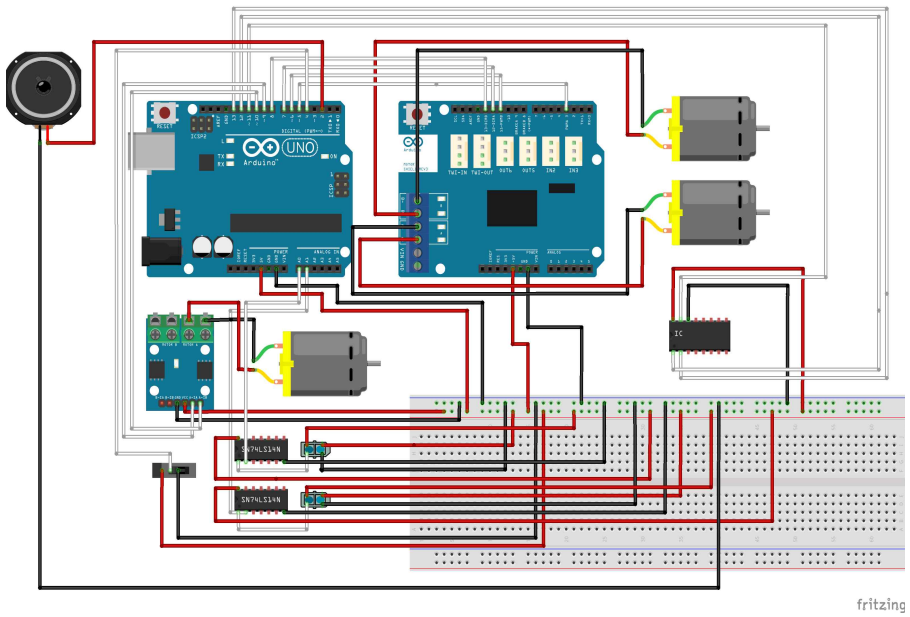


그림 5-(a) 산업차량(Gripper 로봇) 제어 회로도

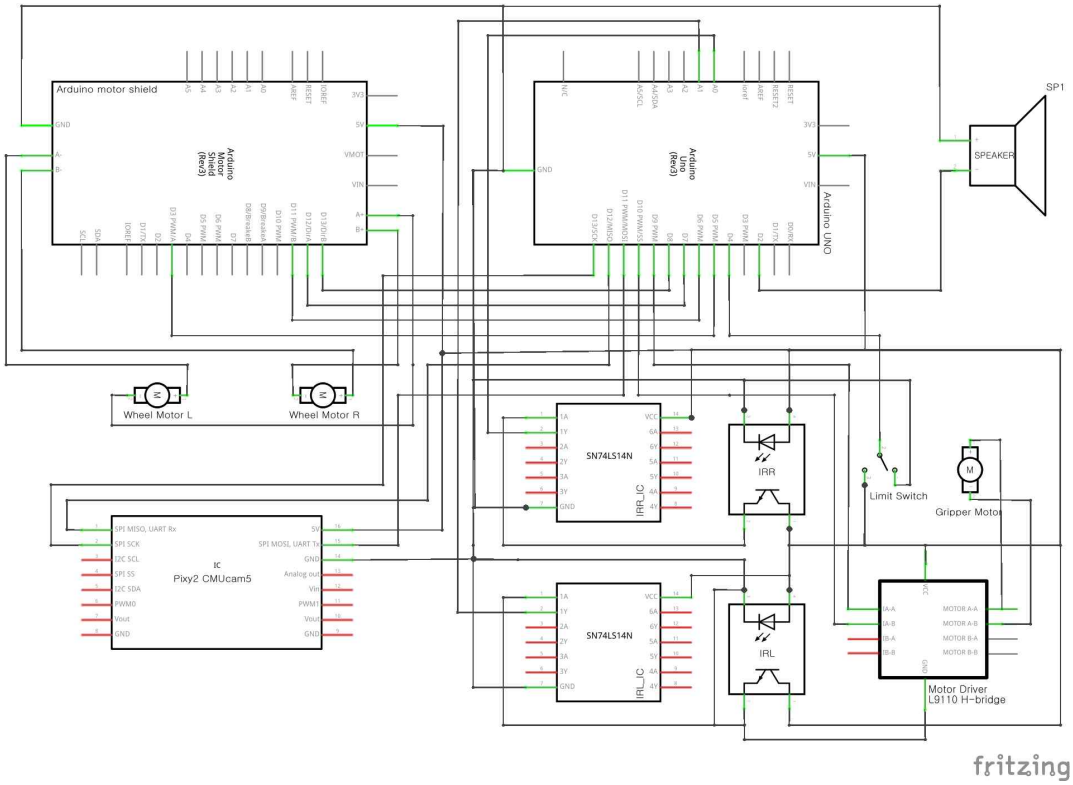


그림 5-(b). 산업차량(Gripper 로봇) 제어 회로도

그림 5.는 산업차량(Gripper 로봇)의 Hardware 제어 결선도 및 회로도이다. 산업차량(Gripper 로봇)은 물품을 집는 Gripper와 색상/라인을 픽시로 영상 처리하여 직류전동기를 구동하는 구동부로 나뉜다. 그림 5-(a)의 좌측 아랫부분은 Gripper 로봇 부분을 나타내고 우측은 직류전동기 구동부이다. 구동부의 IC 칩이 의미하는 것은 픽시이다(회로 결선 시뮬레이터에서 픽시를 지원하지 않아 부득이하게 임의로 만들게 되었다). 그림 5-(b) 회로도에서는 우측 아랫부분이 Gripper이고 좌측 아랫부분이 구동부이다.

먼저 구동부의 동작을 살펴보면 픽시가 바닥에 있는 색상 라인과 색상을 영상 처리하여 각 라인의 경계선을 판단, 물품이 있는 생산 공정 라인을 따라 이동한다. 픽시가 생산라인을 인식하여 경계선을 정하면 그 선을 따라 구동하기 위해 데이터를 임베디드 제어 보드2(Arduino Uno; 이하 Uno)로 보내고 Uno에서 모터 쉴드로 모터 구동 신호인 PWM 신호를 인가해 직류전동기를 동작시켜 산업차량(Gripper 로봇)이 동작한다.

Gripper의 동작은 모터드라이브를 이용하여 직류전동기를 제어함으로써 Gripper를 동작시킨다. Gripper를 동작시킬 때, Gripper가 물품을 놓을 경우는 limit 스위치를 활용하고, 물품을 집을 때는 물품을 잡기 위해 적외선 센서를 Gripper의 양쪽에 부착하여 가변적인 크기의 물품에 효과적으로 대응할 수 있다.

2.1.4. 생산 공정 자동화를 위한 통신 시스템 구성(Zigbee 통신)

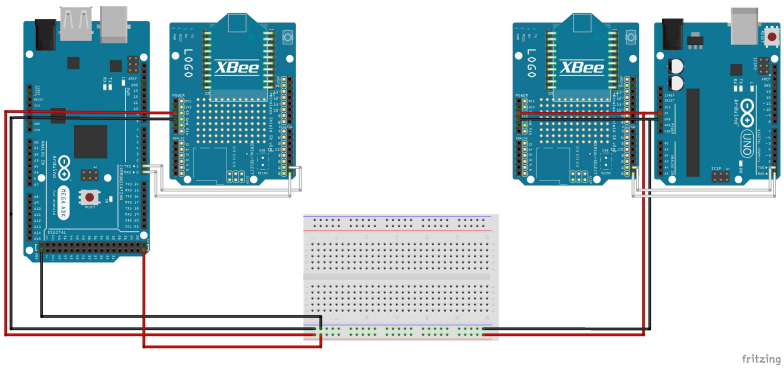


그림 6-(a). 통신 시스템(Zigbee) 제어 결선도

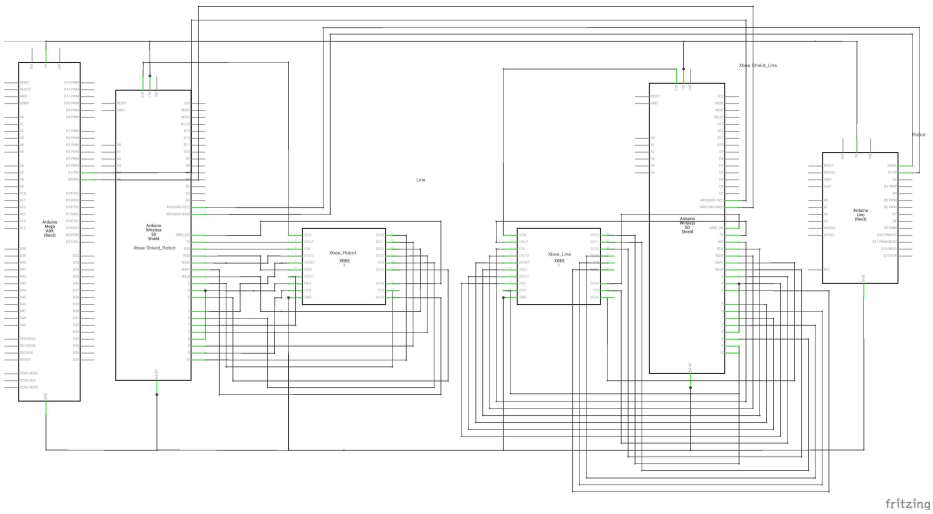


그림 6-(b). 통신 시스템(Zigbee) 제어 회로도

그림 6.은 통신 시스템의 Hardware 제어 결선도 및 회로도이다. 좌측은 Mega에 Zigbee 통신 모듈을 부착한 생산 공정 라인 부분의 통신부이고 우측은 Uno에 Zigbee 통신 모듈을 부착한 산업차량(Gripper 로봇)의 통신부이다.

Zigbee 통신 모듈을 각각의 임베디드 보드에 연결할 때 Xbee shield를 이용하여 연결하여 결선한다. Zigbee 통신은 시리얼 통신을 사용하므로 Zigbee 통신 모듈이 연결된 Xbee shield의 Tx/Rx 포트와 각각의 임베디드 보드의 Tx/Rx를 연결하여 Zigbee 통신 모듈 간의 회로를 결선한다.

2.2. Hardware 기능

2.2.1. 생산 공정을 위한 자동 생산라인 기능

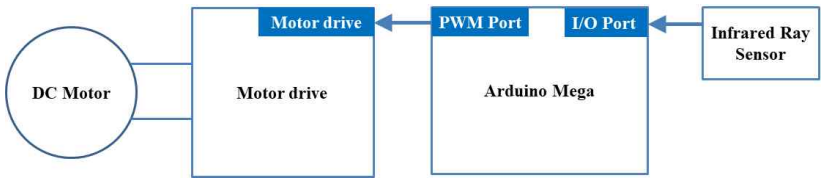


그림 7. 자동 생산라인 Hardware 블록도

그림 7.은 자동 생산라인 Hardware 블록도를 나타내며 기능은 다음과 같다.

① 물품 운반 기능

생산 공정 라인에 전원이 공급되면 생산 공정 라인 각각의 입력 부분에 있는 적외선 센서가 물품을 인식하고 직류전동기가 동작한다. 도착점에 있는 적외선 센서에 물품이 인식되면 전동기가 정지한다. 그리고 물품 도착 신호를 지그비 통신 모듈을 통해 산업차량(Gripper 로봇)으로 송신한다. 도착 신호를 수신한 산업차량(Gripper 로봇)이 라인의 도착점에서 물품을 운반하여 다시 각 라인의 시작점에 놓아두도록 한다.

② 산업차량(Gripper 로봇)과의 지그비 통신

물품이 생산 공정 라인의 도착점에 도착하면 지그비 통신을 통해 산업차량(Gripper 로봇)에 지정된 신호를 보낸다. 우선순위를 결정하는 함수를 통해 순차적으로 산업차량(Gripper 로봇)은 물품을 운반한다. 물품을 가지고 시작점으로 이동하여 산업차량(Gripper 로봇)이 물품을 생산 공정 라인에 두면 다시 지그비 통신 신호를 통해 물품을 놓고 다음 물품을 운반하러 이동한다.

③ 생산 공정 라인에서의 재고 측정

물품이 생산 공정 라인의 도착점에 도착한 후 다시 시작점으로 올 때까지의 시간을 계산해 데이터화한다. 데이터는 생산 공정에서의 물품 소모 시간을 의미하고 이를 토대로 창고에 있는 물품 재고량을 계산할 수 있다. 이 데이터는 모니터링시스템으로 확인 가능하다. 또한 PLX_DAQ를 통해 실시간으로 데이터를 축적하고 그래프를 통해 가시적으로 확인할 수 있다. 게다가 여러 연산을 통한 다양한 데이터 분석이 가능하다.

2.2.2. Gripper 기능

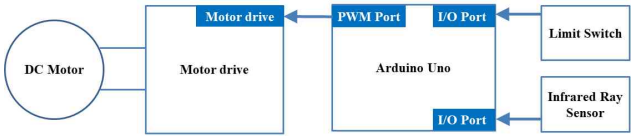


그림 8. Gripper Hardware 블록도

그림 8.은 Gripper Hardware 블록도를 나타내며 기능은 다음과 같다.

① 물품을 집는 기능
산업차량(Gripper 로봇)이 각 라인으로 이동한 후 지그비 통신 모듈을 통해 신호를 받으면 모터드라이브에 직류전동기 동작 신호인 PWM 신호가 주어지고 모터드라이브에 연결되어있는 직류전동기가 Gripper를 작동시켜 물품을 집는다.

② Gripper 제어
적외선 센서 및 리미트스위치를 활용하여 Gripper가 물체의 크기에 관계없이 동작할 수 있게 제어한다.

③ 생산 공정 라인과의 지그비 통신
산업차량(Gripper 로봇)이 라인의 도착점으로 이동하고, 물품을 라인에 내려놓아야 할 때 라인에 있는 적외선 센서가 물품을 인식하여 지그비 통신을 통해 Gripper가 물품을 내려놓게 한다.

2.2.3. 산업차량 구동부 기능

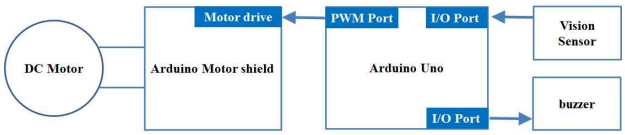


그림 9. 산업차량(Gripper 로봇) 구동부 블록도

그림 9.는 산업차량(Gripper 로봇) 구동부 Hardware 블록도를 나타내며 기능은 다음과 같다.

① 생산라인 판별
픽시를 이용하여 색상/라인을 구별하여 각 라인의 경계선을 찾는 것을 학습시켜 더 효율적인 방향으로 생산라인을 판별한다.

② 생산 공정 라인과의 지그비 통신 모듈
물품이 생산 공정 라인의 도착점에 도착하면 물품을 운반해야 하기 때문에, 생산 공정 라인의 지그비 통신 모듈에서 산업차량(Gripper 로봇)에 데이터를 전송한다. 데이터를 전송받은 산업차량(Gripper 로봇)은 라인을 따라 이동하여 물품을 운반한다.

2.2.4. 생산 공정 자동화를 위한 통신 시스템 기능

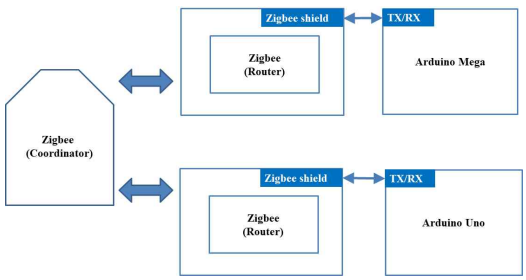


그림 10. 통신 시스템(Zigbee 통신) 블록도

그림 10.은 통신 시스템(Zigbee 통신) Hardware 블록도를 나타내며 기능은 다음과 같다.

① Mesh Topology 이용한 1 : n 양방향 통신

Zigbee S2C를 이용함으로써 Mesh Topology를 이용할 수 있다. 그림 10.에서 위쪽의 Arduino Mega와 연결된 Zigbee 통신 모듈은 생산 공정 라인의 통신부이고 아래의 Arduino Uno와 연결된 Zigbee 모듈은 산업차량(Gripper 로봇)의 통신부이다. 각각의 ZIgbee 통신 모듈은 Routor로써 서로 양방향 통신이 가능하며 좌측의 Coodinator는 전체 통신의 Master 개념으로 각각의 통신 모듈을 하나의 Mesh Topology로 묶어주는 역할을 한다.

2.3. Software 구성

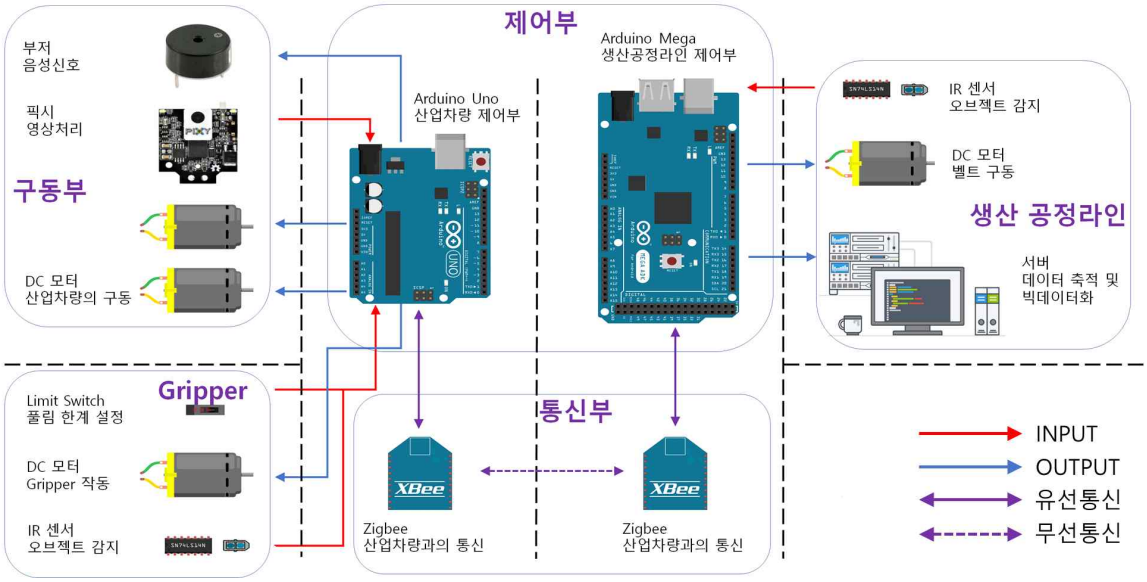


그림 11. Software 구성도

2.4. Software 기능

2.4.1. 전체 시스템

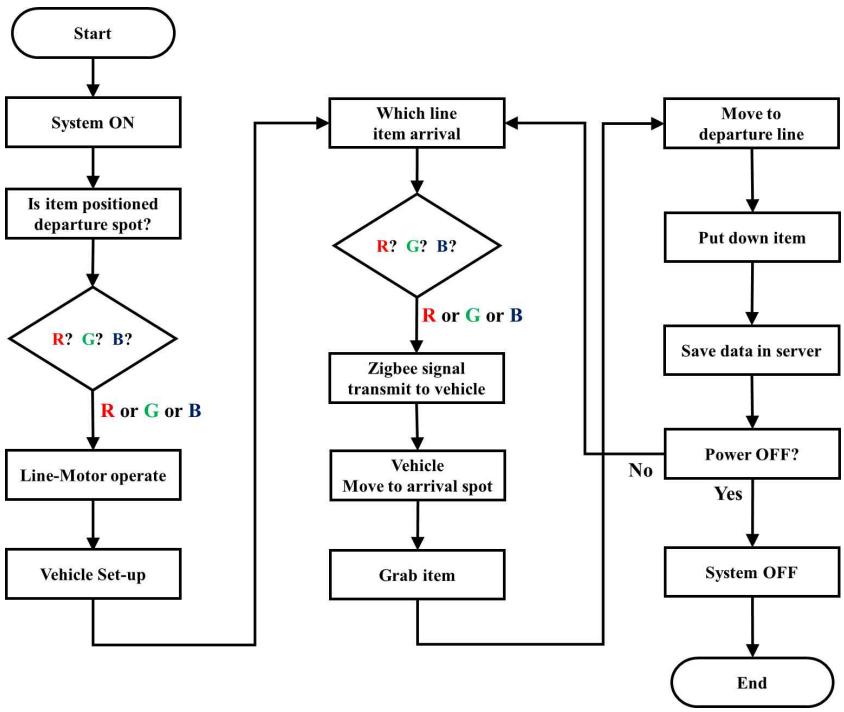


그림 12. 전체 시스템 Flow Chart

그림 12.는 영상처리와 산업차량(Gripper 로봇)을 활용한 생산 공정 자동화 시스템에 대한 일련의 동작들에 대한 프로그램 순서도를 보여준다.

시스템 전원을 인가해 시스템을 동작시키면 생산 공정 라인의 시작점에 있는 물품을 적외선 센서가 인식하고 직류전동기를 구동시킨다. 물품이 도착점에서 도착하면 도착점 적외선 센서가 물품을 인식하여 라인의 구동을 멈춘다. 그리고 지그비 통신을 통해 데이터를 산업차량(Gripper 로봇)에 보내면 산업차량(Gripper 로봇)은 생산 공정 라인의 도착점으로 이동하여 물품을 시작점으로 옮긴다. 이러한 생산 공정들이 무한루프로 구성되어있다. 시스템의 전원이 차단되면 무한루프에서 빠져나와 시스템이 종료된다.

2.4.2. 생산 공정 라인

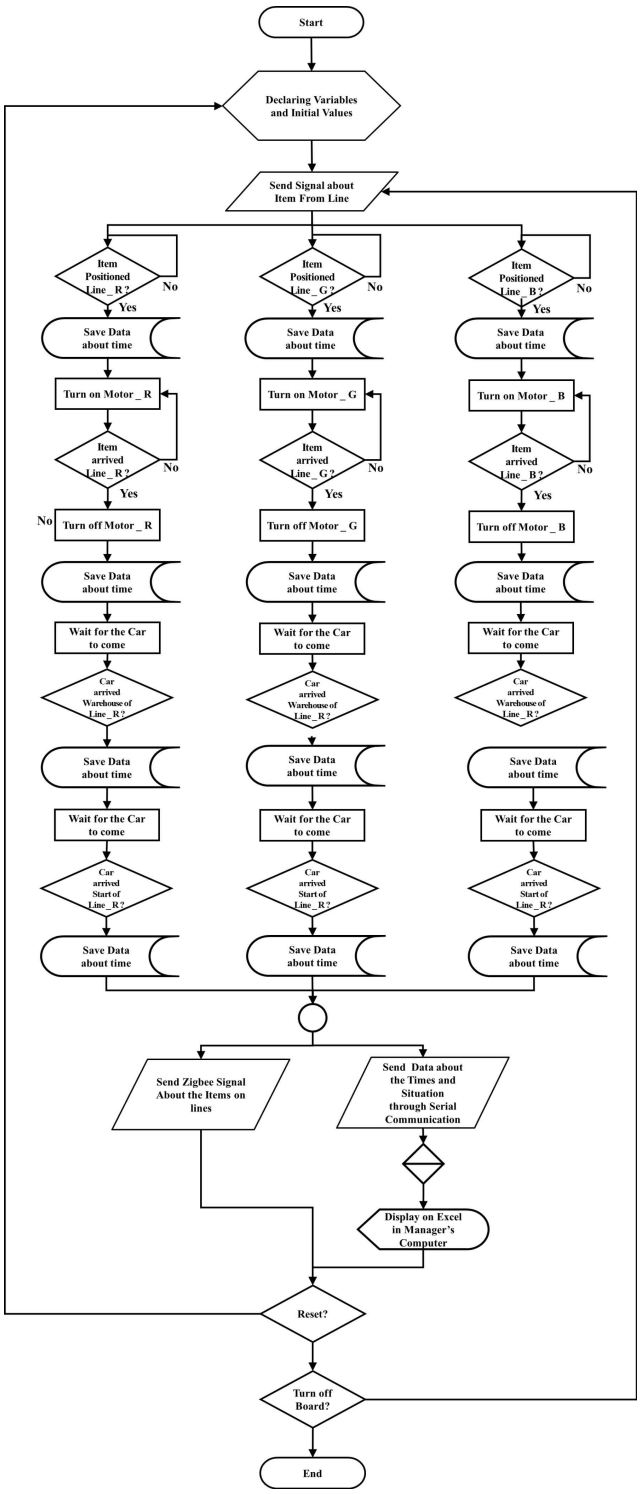


그림 13. 생산 공정 라인 Flow Chart

그림 13.은 생산 공정 라인의 각 라인에 물품이 존재함으로써 시작 신호가 주어졌을 때 라인을 어떠한 방식으로 제어할 것인가에 대한 일련의 공정들에 대한 순서도를 보여준다.

생산 공정 라인에 전원이 공급되면 각각 라인의 시작점의 적외선 센서가 물품이 있는지 인식하고 "HIGH" 신호를 보낸다. 각 라인에 물품이 존재한다면 벨트를 구동하는 직류전동기를 구동시켜 물품을 도착점으로 이동시킨다. 도착점에 물품이 도착하면 도착점의 적외선 센서가 물품을 인식하여 "HIGH" 신호를 보낸다. 이 신호에 의해 벨트를 구동하는 직류전동기를 정지하고 지그비는 산업차량(Gripper 로봇)에게 도착점으로 오라는 신호를 전달한다. 각 라인들은 이 작업들을 무한루프로 수행하게 된다. 시스템의 전원이 차단되면 무한루프에서 빠져나와 시스템이 종료된다.

2.4.3. Gripper

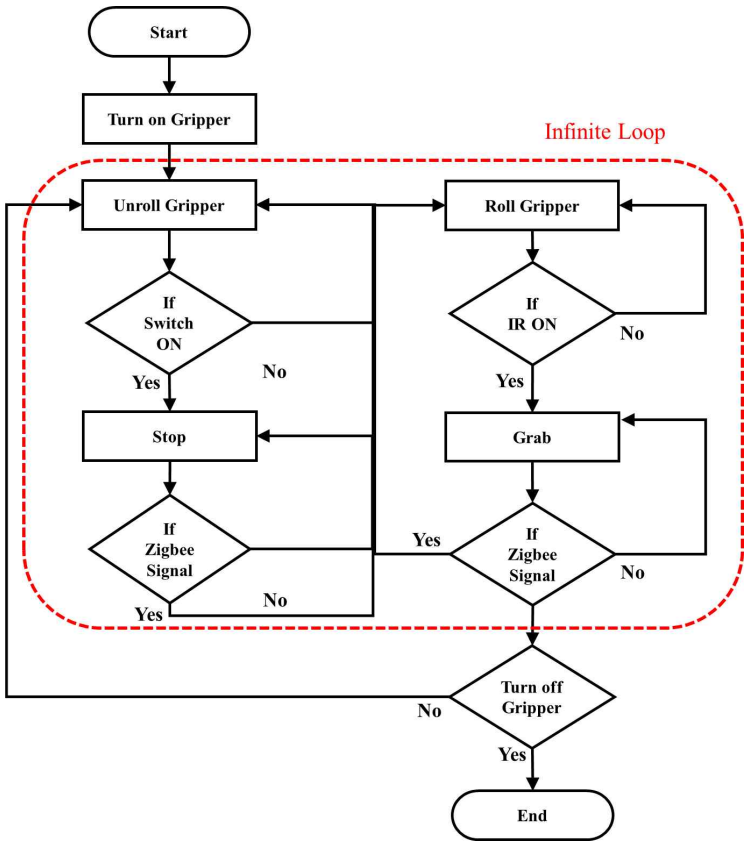


그림 14. Gripper의 Flow Chart

그림 14.는 산업차량(Gripper 로봇)에서 Gripper에 시작 신호가 주어졌을 때 어떠한 방식으로 물품을 집고 놓을 것인가에 대한 일련의 공정들에 대한 순서도를 보여준다.

Gripper에 전원이 공급되면 초기값으로 Gripper가 물품을 집을 수 있게 펼쳐진다. 펼쳐지는 도중 리미트스위치가 on 상태가 되면 "HIGH" 값을 입력하게 되고 직류전동기는 멈추게 된다. 생산 공정 라인에서 지그비 통신을 통해 들어오는 데이터를 기다린다. 생산 공정 라인에서 지그비 통신을 통해 물품을 집으라는 신호를 보낸다. 이후 산업차량(Gripper 로봇)이 이동하여 물품을 운반하여 내려놓는 지점에 도착하면 다시 생산 공정 라인에서 Gripper를 펼치라는 신호를 보낸다. 이 작업들을 무한루프로 수행된다. 시스템의 종료 명령이 수행되면 무한루프에서 빠져나와 시스템이 종료된다.

2.4.4. 산업차량(Gripper)의 구동부

그림 15.는 산업차량(Gripper 로봇)이 어떠한 순서로 생산 공정 라인 도착점의 물품을 운반할 것인가에 대한 일련의 공정들에 대한 순서도를 보여준다.

산업차량(Gripper 로봇)에 전원이 공급되면 생산 공정 라인에서 지그비 통신 신호를 기다리며 대기한다. 생산 공정 라인의 도착점에 물품이 도착하면 산업차량(Gripper 로봇)에 물품이 도착했다는 명령을 지그비 통신 신호로 송신한다. 생산 공정 라인으로의 이동 명령 신호를 받은 산업차량(Gripper 로봇)은 물품를 집기 위해서 생산 공정 라인의 도착점으로 이동한다.

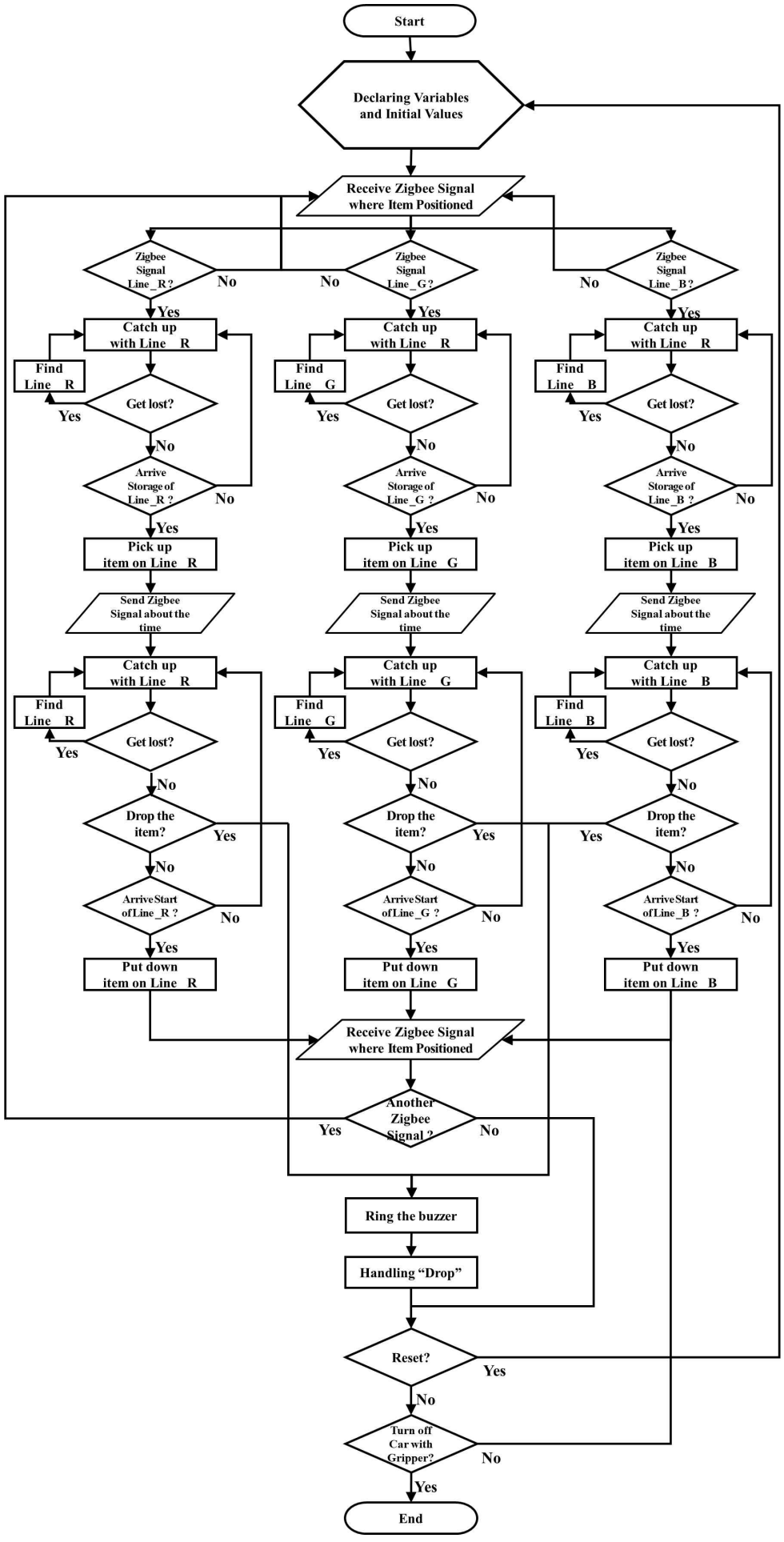


그림 15. 산업차량(Gripper 로봇)의 구동에 관한 Flow Chart

이때의 시간을 측정하여 지그비 통신 신호로 생산라인으로 보내 생산/소비 효율화를 위한 데이터를 축적한다. 도착점의 물품을 집으면 생산 공정 라인의 시작점으로 이동하여 물품을 내려놓는다. 시작점으로 이동 도중 물품을 떨어뜨리게 된다면 작동을 멈추고 신호를 보내 후속 조치를 기다린다. 물품을 내려놓은 산업차량(Gripper 로봇)은 다른 생산 공정 라인의 도착점에 물품이 도착했는지 지그비 통신 신호를 받는다. 새로운 신호가 주어진다면 새로운 신호가 주어진 생산 공정 라인의 물품을 옮기기 위해 이동하고 새로운 신호가 없다면 다시 임무를 수행할 것인지 작동을 중지할 것인지를 결정한다.

2.5. 생산 공정 관리를 위한 모니터링 시스템

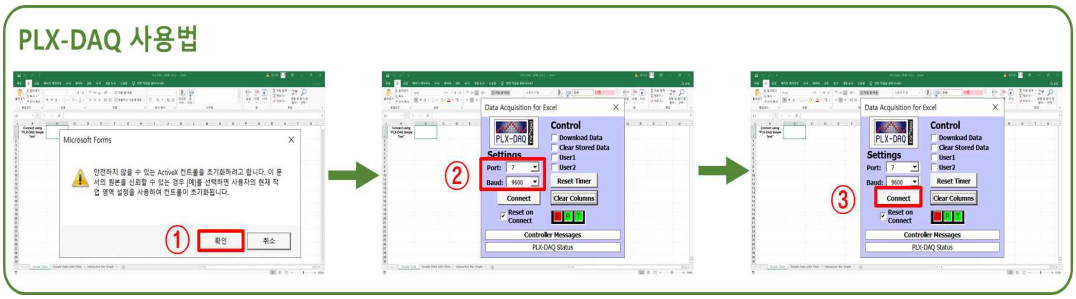


그림 16. PLX-DAQ 사용법

그림 16.의 PLX-DAQ는 Microsoft Excel 용 Parallax 마이크로 컨트롤러 데이터 수집 애드온 툴이다. 아두이노 시리얼 모니터에서 확인할 수 있는 데이터를 엑셀로 가져와 엑셀의 기능을 사용할 수 있게 해준다. PLX-DAQ 프로그램은 아래와 같은 순서로 사용한다.

① 엑셀의 매크로를 통해 만든 프로그램임으로 ActivezX 컨트롤을 초기화 해주기 위해 '확인' 버튼을 누른다.

② PORT 옆의 숫자를 아두이노를 연결한 포트 번호와 맞추고, Baud 옆의 숫자를 시리얼 모니터 보드레이트와 맞춘다.

③ 아두이노 보드를 연결하고 'Connect' 버튼을 누르면 아두이노에서 출력된 데이터 값이 엑셀에 출력된다.

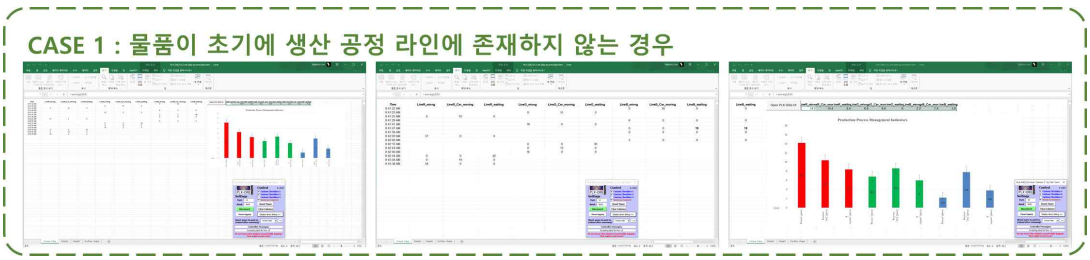


그림 17. 데이터 분석 CASE 1

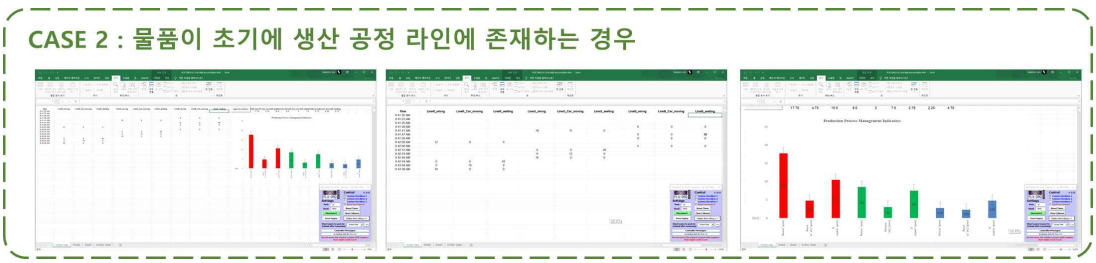


그림 18. 데이터 분석 CASE 2

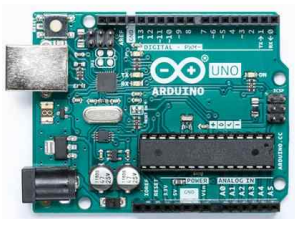
위 그림 17., 18.의 데이터 분석 CASE1, CASE2는 시스템의 프로그램이 출력하는 빅데이터(시간 값)에 대한 해석에서 다룬다.

2.6. 개발환경

2.6.1 HW 개발 사양

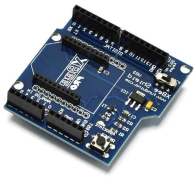
그림 19.는 본 시스템의 하드웨어를 나타내면 이에 따른 자세한 하드웨어 사양은 표 2.에 나타내었다.

Arduino Mega ADK 2560 Rev3

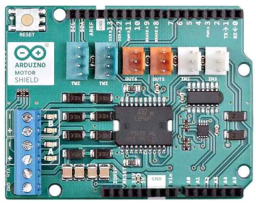


Arduino Uno Rev3

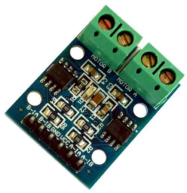
Zigbee S2C



XBee Shield 3.0

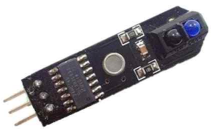


Arduino Motor Shield R3



Motor Driver [L9110]

Pixy2 CMUcam5 Smart Vision Sensor



IR [TCRT5000]

Limit Switch [KW-10R]

Piezo buzzer

그림 19. HW 개발 사양

표 2. 각 모듈별 하드웨어 사양

제품명	주요 정보	
Arduino Mega ADK 2560 REV3	Microcontroller	ATmega2560
	Flash Memory	256 KB of which 8 KB used by bootloader
	Operating Voltage	5V
	Digital I/O Pins	54 (of which 15 provide PWM output)
	Analog Input Pin	16
	External interrupt	UART serial port 4 (0/1, 2/3, 18/19, 20/21) (Rx0/Tx0 ~Rx3/Tx3)
Arduino Uno Rev3	Microcontroller	ATmega328P
	Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
	Operating Voltage	5V
	Digital I/O Pins	14 (of which 6 provide PWM output)
	Analog Input Pin	6
	External interrupt	serial port 1, i2c (A4, A5)
Zigbee S2C	Supply voltage	2.7 to 3.6V
	Transmit current	47 mA (3.3 VDC)
	Receive current	42 mA (3.3 VDC)
	Serial data interface	UART, SPI
	Indoor / urbanrange	Up to 60 m (200 ft)

	Transmit power output (maximum)	6.3 mW (+8 dBm), boost mode 3.1 mW (+5 dBm), normal mode channel 26 max power is +3 dBm
	RF data rate	250,000 b/s
	Receiver sensitivity	-102 dBm, boost mode -100 dBm, normal mode
	Operating current (transmit)	45 mA (+8 dBm, boost mode) 33 mA (+5 dBm, normal mode)
	Operating current (receive)	31 mA (boost mode) 28 mA (normal mode)
XBee Shield 3.0	power output	3.3V and 5V
	IO compatible	3.3V and 5V
	XBee-setting support software X-CTU	
Arduino Motor Shield R3	Operating Voltage	5V to 12V
	Motor controller	L298P, Drives 2 DC motors or 1 stepper motor
	Max current	2A per channel or 4A max (with external power supply)
	Current sensing	1.65V/A
Motor Driver [L9110]	Input voltage	2.5-12V DC
	Output current	800 mA
Pixy2 CMUcam5 Smart Vision Sensor	Processor	NXP LPC4330, 204 MHz, dual core
	Image sensor [Aptina MT9M114]	1296×976 resolution with integrated image flow processor
	Lens field-of-view	60 degrees horizontal, 40 degrees vertical
	Power consumption	140 mA typical
	Power input	USB input (5V)
	Available data outputs	UART serial, SPI, I2C, USB, digital, analog
IR [TCRT5000]	Operating Voltage	5V
	Working voltage	digital signal (0 and 1)
Limit Switch [KW-10R]	Ratings	3(0.5)A 250VAC
		3A 125VAC
	Insulation resisance	$\geq 100M\Omega$
	Contact resistance	$\leq 50m\Omega$
	Operating force	$\leq 0.46N$
Piezo Buzzer	Operating Voltage	3 to 30 V dc
	Rated Current	30 mA

2.6.2. SW 개발환경

표 3.은 본 시스템의 개발에 필요한 SW 개발환경, 개발언어, 개발도구에 대하여 나타내었다.

표 3. SW 개발 사양

구분	내용
개발환경	Arduino (ATMega2560, ATMega328P)
개발언어	C-language
개발도구	Arduino Sketch, Fritzing, XCTU(Windows 10 x64 기반)

3. 개발 프로그램 설명

3.1. 파일 구성

Router_1_Line.ino	Arduino Mega ADK Rev3에 업로드하는 파일로써 생산 공정 관리의 제어부 역할을 수행한다.
Pixy2.h	Arduino Uno Rev3에서 픽시를 사용하기 위한 헤더 파일이다.
Router_2_Car.ino	Arduino Uno Rev3에 업로드하는 파일로써 산업 차량(Gripper 로봇)의 제어부 역할을 수행한다.
PLX-DAQ-v2-Line data accumulator .xlsm	Arduino Mega ADK Rev3에서 Serial 통신으로 출력되는 시간 값을 엑셀로 문서화하여 빅데이터를 축적한다.

3.2. 함수별 기능

3.2.1 산업 차량

Line	void Forward (int F_W, int TIME_FORWARD)	차량이 전진하기 위한 함수
	void L_Turn(int L_T)	차량이 전진하기 위한 함수
	void R_Turn(int R_T)	차량이 전진하기 위한 함수
	void Same_Position_Turn_for_Find_L (int TIME_L)	차량이 제자리에서 좌회전하기 위한 함수
	void Same_Position_Turn_for_Find_R (int TIME_R)	차량이 제자리에서 우회전하기 위한 함수
	void Brake()	차량이 멈추기 위한 함수
	void Back(int TIME_BACK)	차량이 후진하기 위한 함수
	void Find(int COLOR)	차량이 길을 잃었을 때 돌아오기 위한 함수
	void Red()	차량이 빨간색을 따라가기 위한 함수
	void Green()	차량이 초록색을 따라가기 위한 함수
	void Blue()	차량이 파란색을 따라가기 위한 함수
Gripper	void Pick_Up()	차량이 물품을 들어올리기 위한 함수
	void Pick_Down()	차량이 물품을 내리기 위한 함수
	void Drop()	차량이 물품을 떨어뜨림을 알리기 위한 함수

3.2.2 생산 공정 라인

함수를 이용하지 않고 loop() 프로그래밍을 사용하였다. 이에 주요 함수의 흐름도에서 설명한다.

3.3. 주요 함수의 흐름도

3.3.1 산업 차량(Gripper 로봇) (해당 경로 파악)

① char data = Serial.read(); (Zigbee로 수신된 정보를 char 형으로 변환)의 값이 'R', 'G', 그리고 'B' 인지에 따라서 산업 차량(Gripper 로봇)의 경로를 결정한다. ('R'은 빨간색, 'G'는 초록색, 그리고 'B'는 파란색이다.)

② pixy.ccc.getBlocks()로 픽시에서 인식하는 값의 위치가 (pixy.ccc.blocks[i].m_x > 138 && pixy.ccc.blocks[i].m_x < 178) (x축을 기준으로 중심)일 때는 Forward(250, 25) (직진)를 동작한다. (pixy.ccc.blocks[i].m_x >= 0 && pixy.ccc.blocks[i].m_x <138) (x축을 기준으로 좌측)일 때는 L_Turn(250) (좌회전)을 동작한다. (pixy.ccc.blocks[i].m_x >= 178 && pixy.ccc.blocks[i].m_x <= 316)일 때는 (x축을 기준으로 우측)일 때는 R_Turn(250) (우회전)을 동작한다.

③ 만일 산업 차량(Gripper 로봇)이 길을 잃었을 때는 Find(int COLOR)를 이용하여 다시 경로로 복귀한다. int *array = new int[pixy.ccc.numBlocks];를 사용해서 동적행렬을 선언하고 복귀를 경로를 찾는 동안에 발생하는 경로를 저정한다. 변수 j를 이용하여 좌측으로 '30', 우측으로 '60', 좌측으로 '60', 다시 우측으로 '30'을 이동한 뒤, 후진을 '10'이라는 값으로 이동한다. 동작 도중에 해당 경로를 찾게 되면 다시 Red(), 또는 Green(), 또는 Blue()로 복귀한다.

3.3.2 Gripper

① 픽시가 (pixy.ccc.blocks[i].m_signature == 4) (노란색)을 인식할 시에 Gripper의 Pick_Up()을 시작한다.

② analogRead(IRL) (Gripper 좌측 적외선 센서)와 analogRead(IRR) (Gripper 우측 적외선 센서)의 값이 모두 900보다 큰 값을 가질 때까지 Gripper의 Pick_Up()을 구동한다. 900보다 큰 값을 가질 때는 구동을 멈춘다.

③ 픽시가 (pixy.ccc.blocks[i].m_signature == 5) (보라색)을 인식할 시에 Gripper의 Pick_Down()을 시작한다.

④ (digitalRead(SWITCH1) == HIGH) (Limit Switch)의 값이 HIGH일 때까지 Gripper의 Pick_Down()을 구동한다. HIGH일 때는 구동을 멈춘다.

⑤ Pick_Up()이후 부터는 analogRead(IRL) (Gripper 좌측 적외선 센서)와 analogRead(IRR) (Gripper 우측 적외선 센서)의 값이 모두 900보다 작은 값을 가질 때 Brake()를 시작하고 digitalWrite(BUZZ, HIGH) (버저를 동작)를 시작한다. 이 동작은 보드의 Reset버튼을 누르기까지 지속된다.

3.3.3 생산 공정 라인

① unsigned long Current_Time_R = millis();를 사용하여 보드가 시작되는 순간부터 1 [ms]를 카운터한다.

② 다음 조건에 따라서 벨트를 구동시킨다. (예) 생산 공정 라인_R)

1. ((i == 0) && (analogRead(senR_S) < sen_val) && (analogRead(senR_E) < sen_val))
생산 공정 라인에 물건이 존재하지 않을 때 (초기 조건)이므로 벨트 구동부는 정지한다.

2. ((i == 0) && (analogRead(senR_S) >= sen_val))
생산 공정 라인의 시작점에 물건이 도착하는 순간, timeR_start = Current_Time_R;를 이용하여 시간을 대입하고 accumulate_data = (timeR_start – timeR_car)/1000;을 이용하여 시간을 출력한다. 벨트 구동부는 동작한다. Current_Time_R = 0;으로 초기화한다. i = 1;으로 대입한다.

3. ((i == 1) && (analogRead(senR_S) >= sen_val))
생산 공정 라인의 시작점에 물건이 존재할 때, 벨트 구동부는 동작한다.

4. ((i == 1) && (analogRead(senR_S) < sen_val) && (analogRead(senR_E) < sen_val))
생산 공정 라인의 벨트가 구동 중일 때, 벨트 구동부는 동작한다.

5. ((i == 1) && (analogRead(senR_E) >= sen_val))
생산 공정 라인의 도착점에 물건이 도착하는 순간, timeR_end = Current_Time_R;를 이용하여 시간을 대입하고 accumulate_data = (timeR_end - timeR_start)/1000;을 이용하여 시간을 출력한다. 벨트 구동부는 정지한다. Current_Time_R = 0;으로 초기화한다. i = 2;으로 대입한다. Things_For_CHAR[COUNT_NUM] = 'R';을 이용하여 이동할 경로의 값을 송신한다.

6. ((i == 0) && (analogRead(senR_E) >= sen_val))
생산 공정 라인의 도착점에 물건이 존재할 때, 벨트 구동부는 정지한다.

7. ((i == 2) && (analogRead(senR_S) < sen_val) && (analogRead(senR_E) < sen_val))
생산 공정 라인의 도착점에 산업 차량(Gripper 로봇)이 물건을 가져가는 순간, timeR_car = Current_Time_R;를 이용하여 시간을 대입하고 accumulate_data = (timeR_car – timeR_end)/1000; 을 이용하여 시간을 출력한다. 벨트 구동부는 정지한다. Current_Time_R = 0;으로 초기화하고 i = 0;으로 대입한다.

그 외 생산 공정 라인_G, 그리고 생산 공정 라인_B 또한 위와 같이 동일하게 동작한다.

③ char Things_For_CHAR[4] 배열을 이용하여 산업 차량(Gripper 로봇)에 전달하고자 하는 경로에 대한 정보를 정렬한다. 이 배열의 마지막 열에 해당하는 Things_For_CHAR[3]의 값은 배열을 정렬하기 위해 사용될 뿐 값을 대입하거나 출력하는 데는 사용되지 않는다. 생산 공정 라인의 도착점에 우선적으로 물건이 도착한 순서대로 산업 차량(Gripper 로봇)이 이동하기 위해 다음 조건에 따라서 값을 정렬해 나간다. (char Things_For_CHAR[4] = {'0', '0', '0', '0'};으로 초기화되어 있다.)

1. ((i == 1) && (analogRead(senR_E) >= sen_val))
생산 공정 라인의 도착점에 물건이 도착하는 순간, '0'이라는 값을 가지는 배열에 파악이 되는 순간

```
for(CONSTANT=0; CONSTANT < 3; CONSTANT++)
{
if(Things_For_CHAR[CONSTANT] == '0')
{
COUNT_NUM = CONSTANT;
break;
}
}
Things_For_CHAR[COUNT_NUM] = 'R';
```

다음과 같은 과정을 통해 해당 열에 정보를 추가한다. (예) 생산 공정 라인_R)

2. ((i == 0) && (analogRead(senR_S) >= sen_val))
생산 공정 라인의 시작점에 물건이 도착하는 순간,

```
for(CONSTANT=0; CONSTANT < 3; CONSTANT++)
{
Things_For_CHAR[CONSTANT] = Things_For_CHAR[CONSTANT+1];
}
```

다음과 같은 과정을 통해 값들은 왼쪽으로 Shift하고, Things_For_CHAR[0]이었던 값은 소멸한다.

④ Serial2.print(Things_For_CHAR[0]);를 이용하여 현재 산업 차량(Gripper 로봇)이 이동해야 하는 경로에 대한 정보를 지속적으로 전달한다.

⑤ PLX-DAQ (Serial 통신 Excel 매크로)를 이용하여 Serial 통신으로 출력되는 값들을 Excel로 전달하여 문서화한다.

3.4. 시스템의 프로그램이 출력하는 빅데이터(시간 값)에 대한 해석

시간데이터를 해석함에 있어서, 짧은 벨트에서 얻을 값을 수요가 많은 제품에서 나올 데이터 값으로 볼 것이고, 긴 벨트에서 얻을 값을 수요가 작은 제품이 배정된 것에서 나올 데이터 값으로 볼 것이다.(벨트길이와 수요는 반비례한다.) 이를 통해 우리가 구상한 system에서 간접적으로 수요에 따른 데이터 값을 측정할 수 있고 이 데이터를 통해 여러 해석을 할 것이다.

① Time_Start (물건이 생산 공정 라인의 시작점에 들어오는 순간의 시각)
② Time_Warehouse (물건이 생산 공정 라인의 도착점에 들어오는 순간의 시각)
③ Time_Car_Pick_Up (산업 차량(Gripper 로봇)이 물건은 집어가는 순간의 시각)

위 시각들을 조합하면 얻을 수 있는 시간의 값들은 다음과 같다.

ⓐ (Time_Warehouse) - (Time_Start) = Time_Belt_Moving
(벨트가 물건은 시작점 부터 도착점까지 보내기 위해 사용된 시간)

ⓑ (Time_Start) - (Time_Car_Pick_Up) = Time_Car_Moving
(산업 차량(Gripper 로봇)이 물건을 도착점부터 시작점 까지 보내기 위해 사용된 시간)

ⓒ (Time_Car_Pick_Up) - (Time_Warehouse) = Time_Waiting_Car_To_Come
(도착점의 물건이 산업 차량(Gripper 로봇)이 올 때까지 대기하는 시간)

이 시간값들은 2개의 경우의 수로 해석할 수 있다.

㉠ 사용자가 주문하는 물건의 수가 모두 동일할 경우
(소비자의 수요는 라인의 길이에 따라 이미 결정 되어 있다는 전제하에 해석 해야함)
따라서 소비자들이 많은 양의 물건을 원한다고 해석할 수 없다. 그런데 이렇게 해석하면 우리가 처음 주장한 소비자의 수요를 예측한다는 주장이 성립하지 않는다.
Time_Belt_Moving의 시간 값은 물건이 소비되는 시간에 해당한다. 시간 값이 작을수록 물건이 소비되는 속도가 빠르다는 것이므로 (생산량이 많다는 것을 확인할 수 있다.) 소비자들이 많은 양의 물건을 원한다고 해석할 수 있다.

㉡ 사용자가 주문하는 물건의 수가 각기 다를 경우
(R > G > B)순으로 주문한다면 빨간색 생산 공정 라인이 가장 많은 물건이 소비되기에 Time_Belt_Moving이 가장 높은 시간 값을 가지게 된다. 만일 (R < G < B)순으로 주문한다면 파란색 생산 공정 라인의 물건을 소비자들이 많이 소비하고 빠르게 소비된다는 것을 알 수 있다. 즉, R, G, B순으로 재고가 쌓이는 속도가 빠르다는 것을 알 수 있으므로 사용자가 주문해야할 양과 시간을 추론할 수 있다.

해석을 통해 생산 공정 라인에 대한 오류 검출을 할 수 있다.

1. Time_Belt_Moving에 대해 고찰하면 다음과 같은 해석을 도출할 수 있다.
Time_Belt_Moving에서 축적된 빅데이터를 기반으로 특정 라인의 시간 값이 늘어났다는 사실을 확인한다면 해당 생산 공정 라인에 이상이 있다는 해석을 끌어낼 수 있다.

2. Time_Car_Moving에 대해 고찰하면 다음과 같은 해석을 도출할 수 있다.
Time_Car_Moving에서 축적된 빅데이터를 기반으로 특정 라인의 시간 값이 늘어났다는 사실을 확인한다면 산업 차량(Gripper 로봇)의 이동 경로 및 산업 차량(Gripper 로봇)의 구동부에 이상이 있다는 해석을 끌어낼 수 있다. (Time_Waiting_Car_To_Come과 함께 시간 값이 증가할 확률이 높다.)

3. Time_Car_Moving과 Time_Waiting_Car_To_Come에 대해 비교하여 고찰하면 다음과 같은 해석을 도출할 수 있다. 산업 차량이 이동하는 경로를 반으로 나눠서 해석한다면 다음과 같이 Time_Car_Moving의 시간값을 가지는 경로와 Time_Waiting_Car_To_Come의 시간 값을 가지는 경로이므로 각각의 시간 값의 변화를 통해서 이상이 있는 경로를 추론할 수 있다는 해석을 끌어낼 수 있다.

-> 다음과 같이 특정 시간대의 변화를 통해서 시스템의 어디에 이상이 있는지를 추론할 수 있다.

4. 개발 중 장애요인과 해결방안

① 물품을 지게차로 들어 올려 물품을 받아 운반하려 하였으나 물품이 움직이고 운반 도중 떨어질 위험이 있으며 생산라인에 공급 할 때 안정적으로 내려놓기가 힘들었다. 그러므로 Gripper를 사용하여 물품을 안정적으로 잡게 하고, 이동 중에 확실하게 고정하며, 만약 물품을 운반하는 도중에 물품이 떨어질 경우 그 즉시 바로 알 수 있도록 하였다.

② 산업차량(Gripper 로봇)이 이동하는 경로들에 서로 다른 색깔 테이프를 부착하고 산업차량(Gripper 로봇)이 RGB 센서를 이용하여 이 경로를 인식해 움직여서 생산라인에 물품을 공급하려 하였으나, RGB 센서로는 라인의 경계선 확인이 제대로 되지 않을 수 있다. 그러므로 산업차량(Gripper 로봇)이 이동하는 경로에 있는 서로 다른 색깔 테이프를 RGB 센서가 아닌 픽시를 이용해 라인을 명확하게 감지하고 이 라인을 따라 움직여서 필요한 물품을 단시간에 생산라인에 공급할 수 있도록 하였다.

③ 산업차량(Gripper 로봇)의 Gripper 부분에 물품이 잡혔을 때 이를 확인하기 위하여 리미트스위치를 사용하려고 하였으나, 물리적인 센서이기 때문에 정해진 크기의 물품에만 사용할 수 있다. 따라서 적외선 센서를 이용해 물품을 확인하고 Gripper로 잡을 수 있도록 설계하였다. 이때 적외선 센서의 발광부, 수광부를 교차하여 마주 보는 적외선 센서에 방해가 없도록 주의해야 한다.

④ 지그비 통신 신호를 통해 산업차량(Gripper 로봇)의 Gripper를 제어하려고 했으나 너무 많은 송수신 데이터로 인해 지그비 통신에 오류가 많이 발생하여 원활한 물품 운반이 힘들어졌다. 지그비 통신 신호를 오류 없이 처리할 수 있도록 하기 위해 라인의 시작점과 도착점에 RGB가 아닌 다른 색상을 이용해 움직임을 처리할 수 있도록 프로그램을 수정하였다.

⑤ 지그비 통신 모듈 각각이 송수신을 동시에 진행하는 경우 순간적으로 데이터가 송신되기 때문에 수신에 어려움을 겪었다. 따라서 통신이 아닌 변수를 통해 움직임을 처리할 수 있도록 프로그램을 수정하였다.

⑥ 산업차량(Gripper 로봇)이 지정된 경로가 아닌 다른 경로로 진입 시 정상적인 제어가 불가능하다. 그러므로 산업차량(Gripper 로봇)이 다른 경로로 진입할 때 일단 정지하고 좌우로 움직이며 올바른 경로를 찾아 진입할 수 있도록 한다.

5. 개발결과물과의 차별성

① 통신 시스템을 지그비 통신 모듈을 사용하여 구축함으로써 Mesh Topology 이용한 1 : n 양방향 통신을 이용할 수 있다. 이는 생산 공정에서 공정에 필요한 장비들이 유기적으로 연결되어 서로 상호작용하며 공정 효율을 높일 수 있다. 또한 지그비 통신 모듈은 저전력의 특징을 가지고 있어 생산 공정에서 전력 사용을 줄일 수 있다.

② 각 부품 수급의 주기를 파악함으로써 생산라인에 적정 시기에 맞춰 적합한 부품을 공급할 수 있다. 실제로 구현할 때 생산라인의 길이를 다르게 하여 각 생산라인의 주기에 맞추어 부품을 공급한다. 다품종 소량 생산을 넘어 맞춤형 제품을 생산할 때 무수히 많은 부품들이 제각기 다른 주기로 생산라인에서 필요하므로 부품 공급이 수월하다. 부품마다 생산라인의 길이가 다르므로 각 생산라인에 따른 주기가 달라 생산라인 마다 다른 주기로 각각의 부품을 공급하는데 이러한 데이터를 계속 축적하여 빅데이터화 함으로써 각 부품마다 소진될 시기를 예측한다. 또한 부품 주문이 필요한 시기를 파악하여 주문함으로써 원활한 부품 수급이 가능하다. 그러므로 부품 재고 통제가 원활해지고 부품 공급률이 향상되어 부품 부족에 의한 불가피한 생산라인의 정지가

감소한다.

③ 한 대로 여러 가지 부품을 경로에 따라 개별적으로 운반 가능하다. 현대사회는 다품종 소량 생산을 넘어서 맞춤화된 제품을 생산하는 시대로써 이를 위해서는 다양한 종류의 부품들을 최적화된 경로로 생산라인마다 공급해야 한다. 만약 생산차량이 하나의 부품을 한 생산라인에만 공급할 수 있다면 공장의 면적은 매우 커야 할 것이고 생산라인이 구동되는 시간이 짧아 제품 생산에 시간이 많이 걸릴 것이다. 그러므로 한 대의 산업차량(Gripper 로봇)으로 다양한 부품들을 생산라인에 공급하는 것은 적은 수의 산업차량(Gripper 로봇)을 운용하고, 공장의 크기를 소형화하며, 공장의 활용면적을 증대시킬 수 있는 3중 효과를 발휘한다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

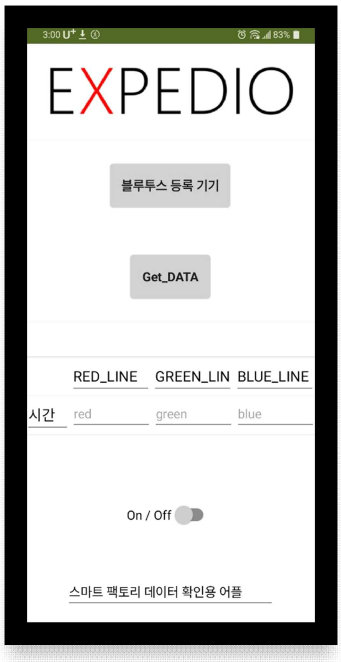
번호	구분	이름	대학	학과	학년	역할
1	**팀장**	**오택규**	국립 경상대학교	제어계측공학과	3	전체 알고리즘 제작, 산업 차량(Gripper 로봇) 하드웨어와 소프트웨어 설계 및 제작, 생산 공정 라인 하드웨어와 소프트웨어 설계 및 제작
2	**팀원**	**곽승윤**	국립 경상대학교	제어계측공학과	3	산업 차량(Gripper 로봇) 하드웨어와 소프트웨어 설계 및 제작, 생산 공정 라인 하드웨어와 소프트웨어 설계 및 제작
3	**팀원**	**천우담**	국립 경상대학교	제어계측공학과	3	산업 차량(Gripper 로봇) 소프트웨어 제작, 생산 공정 라인 하드웨어와 소프트웨어 설계 및 제작, 생산 공정 라인 주변 환경 구축
4	**팀원**	**봉재민**	국립 경상대학교	제어계측공학과	3	요구 사항 및 애로 사항 자료 수집, 생산 공정 라인 주변 환경 구축
5	**팀원**	**지혜주**	국립 경상대학교	제어계측공학과	3	생산 공정 라인 소프트웨어 제작, 블루투스 제어 및 어플리케이션 제작

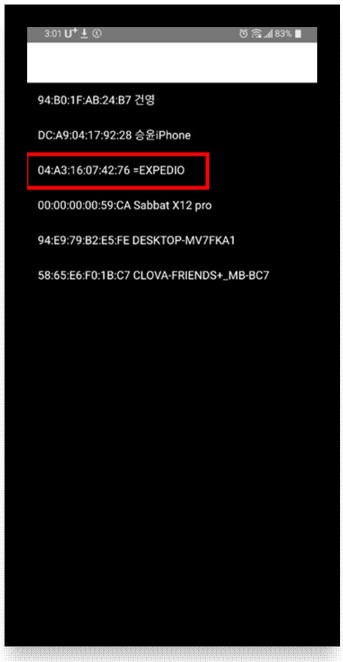
수행 내용		6월				7월				8월				9월			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
심화 개발	요구 사항 및 애로 사항 자료 수집, 고찰 및 해결 방안 탐구																
소프트웨어 구축	Arduino 보드에 필요로 하는 프로그래밍 숙달																
	산업 차량(Gripper 로봇) 프로그래밍 실행																
	생산 관리 라인 프로그래밍 실행																
하드웨어 구축	철제 프레임을 이용한 산업 차량(Gripper 로봇) 하드웨어 제작																
	철제 프레임을 이용한 생산 관리 라인 하드웨어 제작																
기술지원 교육	1차 기술지원 교육 및 LS 산전 R&D센터 견학																
	2차 기술지원 교육 및 세미나																

개발환경 구축	생산 공정 라인 주변 환경 구축																
개발 마무리	수정 및 추가 사항에 대한 토의 및 실행																
	최종 검사 및 버그 수정																

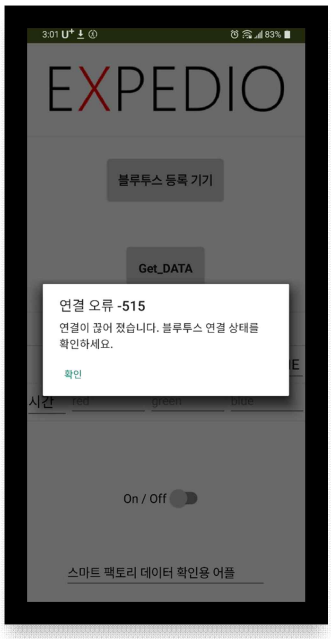
※ 부록: UI 사용법 (본 개발 시스템 미 적용)

안드로이드 앱 모니터링 시스템은 본 개발 시스템에는 미적용 되었지만 추후 본 시스템에 적용시 생산 관리 효율화를 도모할 수 있다.





안드로이드 앱을 실행시키면 제일 먼저 보이는 화면이다. 위에서부터 순서대로 로고, 버튼 2가지, 데이터 목록, 블루투스의 전원으로 구성 되어 있다. 왼쪽 그림에서 블루투스 등록기기 버튼을 누르면 오른쪽과 같은 창이 열린다. 안드로이드 앱에 블루투스 연결 등록이 된 기기들의 목록들이 뜬다. 이 목록 창에서 블루투스 모듈 주소와 이름을 볼 수 있다. 우리가 테스트한 HM-10의 블루투스 모듈의 이름은 EXPEDIO이고, 주소는 04:A3:16:07:42:76이다.



블루투스가 연결되지 않은 상태로 Get_DATA 버튼을 누르면, 연결이 되지 않았다는 오류로 왼쪽의 알림창이 뜬다. 여기서 확인을 누르면 블루투스를 확실히 끊어 새로 연결하도록 한다. 블루투스 연결을 ON 시켜서 Get_DATA 버튼을 누른다면 메인 보드에서 받아온 데이터를 표의 텍스트 창에 띄워 준다. 오른쪽 그림과 같이 통신을 통해 수신된 데이터를 가시적으로 볼 수 있고, 스마트 팩토리를 관리하는 사용자는 현장에 있지 않아도 간단하게 안드로이드 앱을 이용해 공장에서 필요한 데이터를 확인할 수 있다.