창의설계프로젝트 최종보고서

<table>
<tr><td>기술개발
과 제</td><td colspan="5">로드셀을 활용한 무게 감지에 따른 자동주문 시스템
(Automatic ordering system by weight-sensing using load cell)</td></tr>
<tr><td>과제팀 이름</td><td colspan="2">세 공돌이</td><td>지도교수</td><td colspan="2">신동헌</td></tr>
<tr><td>개발기간</td><td colspan="5">2022 년 3 월 ~ 2022 년 6 월 (총 4 개월)</td></tr>
<tr><td rowspan="2">개발소요비용</td><td rowspan="2">총 액</td><td rowspan="2">315(천원)</td><td>학교부담금</td><td colspan="2">315 천원</td></tr>
<tr><td>과제팀부담금</td><td colspan="2">천원</td></tr>
<tr><td rowspan="4">과제팀 구성원</td><td>이름</td><td>강재언</td><td>김 진</td><td colspan="2">이휘호</td></tr>
<tr><td>사진</td><td>  </td><td>  </td><td colspan="2">  </td></tr>
<tr><td>학번</td><td>2016430001</td><td>2018430019</td><td colspan="2">2016430033</td></tr>
<tr><td>연락처</td><td>010-8490-8784</td><td>010-8390-2711</td><td colspan="2">010-5334-2718</td></tr>
</table>

창의설계프로젝트 과제를 성실히 수행하고자 최종보고서를 제출합니다.

2022 년 6 월 15 일

과제 수행자 1 : 강 재 언 (인)
과제 수행자 2 : 김 진 (인)
과제 수행자 3 : 이 휘 호 (인)

지도교수 : 신 동 헌 (인)

서울시립대학교 기계정보공학과 귀중

1. 서론

1.1 개발 과제의 개요

가. 개발 과제 요약

본 프로젝트는 중소 규모의 사업장에서 사용자가 지정한 품목들의 효율적인 관리를 위하여 품목들의 사용량 정보를 데이터화 하여 사용자에게 보여주고 제품의 무게를 감지하는 시스템을 통하여 재고 소진율을 파악하고 주문이 필요한 품목들을 자동으로 주문할 수 있도록 구현하기 위해 아두이노-로드셀 모듈을 설계하고 이를 연동할 수 있는 PC 프로그램을 함께 개발하고자 한다.

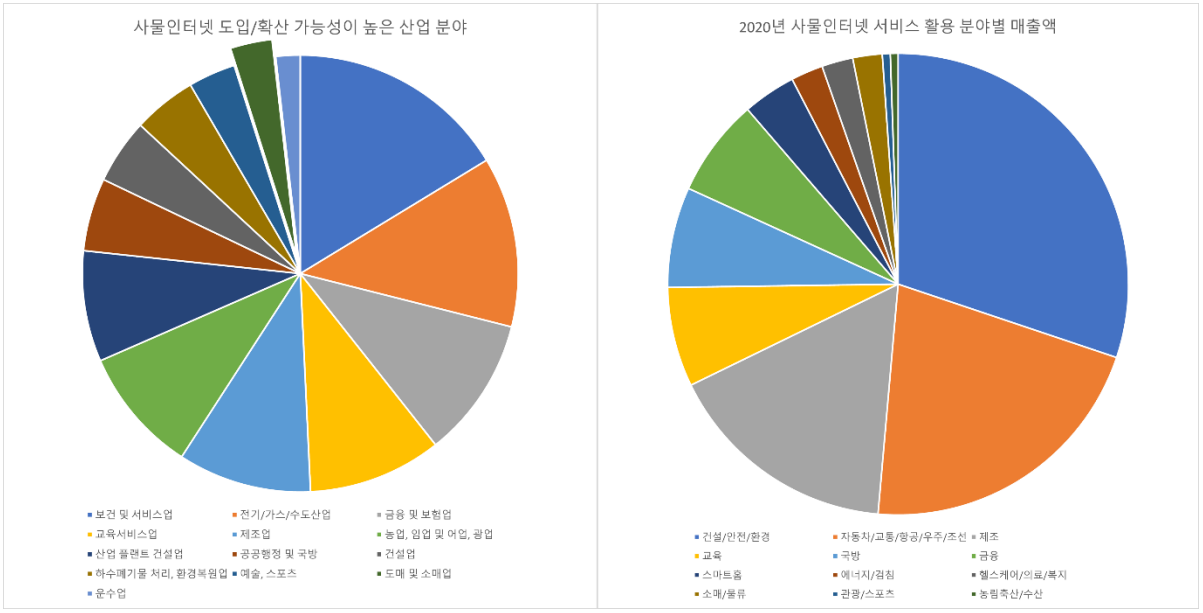
나. 개발 과제의 배경 및 효과

1. 제안배경

주변의 사물이 네트워크로 연결되어 다른 사물들과 상호작용하는 사물인터넷(IoT; Internet of Things)은 매우 빠르게 발전하고 있다. 리서치 전문회사 '마켓앤마켓'의 IoT 기술 시장 전망 보고서에 따르면 Iot 기술 시장 규모는 연간 6.7%씩 성장해 2027년에는 5,664억 달러에 이를 것으로 전망하고 있다. 하지만 현재 사물인터넷 서비스의 활용 분야는 특정 분야에 치중되어 있다. 2020년 사물인터넷 산업 실태조사에 따르면 사물인터넷 서비스 활용 매출액 중 건설/안전/환경이 30.1%, 교통이 21.2%, 제조 분야가 16.4%를 차지한다고 한다. 즉, 현재 대부분의 사물인터넷이 대규모 사업장을 대상으로 개발, 활용되고 있다고 할 수 있다.

또한, 2020년 사물인터넷 산업 실태조사의 결과에 따르면, 앞으로 사물인터넷이 도입, 확산될 가능성이 높은 산업 분야는 보건/서비스업이 16.3%, 전기/가스/수도산업이 12.6%, 금융 및 보험업이 10.4%를 차지하였다. 이에 비해, 이번 프로젝트의 대상으로 설정한 소매업의 경우에는 3.1%를 차지해 전체 13개의 분야 중 12위를 차지하였다.

이러한 조사 결과를 통해, 현재 상용화되고 있는 사물인터넷 서비스는 대부분 대규모 사업장을 대상으로 하고 있으며 앞으로 소매업에 도입하기 위한 연구 및 개발이 부족한 실정임을 알 수 있다.



(2020년 사물인터넷 산업 실태조사 보고서)

하지만, 이러한 대규모 사업장이 아닌 중소규모의 사업장에서도 사물인터넷을 활용한다면 업무를 더욱 효율적으로 처리할 수 있다. 특히, 지속적으로 소모가 있는 제품을 관리해야 하는 경우 사물인터넷을 활용한다면 사람이 일일이 확인하지 않아도 수량을 파악할 수 있다.

현재 사용되고 있는 로드셀을 활용하여 재고의 수량을 파악하는 장치는 대부분 앞서 언급했듯이 대규모 산업을 위한 장치였다. 하지만, 같은 원리로 장치의 크기를 축소한다면 중소규모 사업장에서 충분히 사용할 수 있다고 판단했다. 또한, 작은 크기의 장치도 존재했지만 전용 쇼핑몰을 통해서만 상품의 구매가 가능해 사업주가 가격비교를 할 수 없다는 단점이 존재했다. 사업주의 입장에서 재고를 주문할 때, 가격비교를 통해 최저가로 구매하는 것이 이익을 창출할 수 있는 방안이기 때문에 최저가 상품을 선택하는 기능이 필요하다고 생각했다.

이러한 필요성에 주목하여 중소규모 사업장을 대상으로 제품의 재고를 파악하고 자동으로 부족한 수량에 대해 주문을 해주는 장치 및 PC 프로그램을 설계하고자 한다.

2. 기대효과

본 프로젝트를 통해 지정된 제품의 무게를 측정하여 제품의 소모를 사용자에게 알려주는 장치 및 PC 프로그램을 제안한다. 해당 장치는 중소규모 사업장에서 사용할 수 있는 규모로 제작되어 기존 사업장을 변화시킬 필요없이 바로 사용할 수 있도록 제작한다. 또한, 함께 제공되는 PC 프로그램을 통해 제품의 수량이 특정 기준 이하로 측정될 경우, 별도의 주문과정 없이 자동으로 주문할 수 있어 사업 운영의 효율을 증가시킨다.

이를 통해, 중소규모 사업장에서 재고 관리 및 주문에 소모되는 시간과 인력 소모를 줄이고, 효율적으로 사업장 운영을 할 수 있도록 돕는다.

다. 개발 과제의 목표와 내용

1. 개발목표

사용자가 지정한 품목에 대해 최대 150kg까지 0.2kg 내의 오차로 무게를 측정하여 PC로 전송할 수 있는 무게측정 모듈을 개발한다. 측정한 무게를 이용하여 중소규모 사업장에서 소모품, 부품, 원자재 등에 대한 효율적인 관리를 할 수 있도록 하는 PC 프로그램을 개발한다. 프로그램은 모듈로부터 전송받은 무게를 통해 사용량을 추적하여 사용량에 대한 통계를 제공하는 기능과 물품이 얼마 남지 않았을 때 모두 소진되기 전 자동주문을 하는 기능을 포함한다. 본 과제의 개발단계에서는 대상 품목을 생수로 정하고, 생수의 무게가 10kg 이하가 되었을 때 자동주문을 하는 것으로 한다.

2. 개발내용

가) 무게 측정 및 데이터 전송

아두이노와 로드셀을 이용하여 무게를 측정하고 측정한 데이터를 PC에 전송하는 장치를 개발한다. 최대 150kg의 무게를 0.2kg내의 오차로 측정할 수 있도록 한다. 이 때 통신방법은 Bluetooth와 Wi-Fi중 적합한 것을 선택한다. 로드셀, 로드셀 증폭기, 아두이노, 배터리를 포함한 하나의 회로를 구성한다.

나) 케이스 제작

앞서 설계한 회로를 부착하고 무게를 측정할 때 생수를 올려놓을 케이스를 설계하고 3D 프린터를 이용하여 제작한다. 케이스에 올려놓은 생수의 무게가 모두 로드셀에 전달되도록 설계하고, 구조해석을 통해 최대 150kg의 하중을 버틸 수 있도록 설계한다

다) PC 프로그램 개발

1) 사용량 통계 제공

무게측정 모듈를 통해 측정된 무게를 기록하여 일간, 주간, 월간 등 생수 사용량에 대한 통계를 제공한다.

2) 자동주문

생수가 10kg 이하가 되었을 때 사용자에게 알림을 주고 사용자가 미리 등록해놓은 제품명을 통해 자동으로 상품을 검색하고 주문하도록 하여 재고가 떨어지지 않도록 한다.

1.2 관련 기술의 현황

가. State of art

1) 사물인터넷 (Internet of Things)

사물인터넷(Internet of Things, IoT)은 사물들의 주변의 다른 사물과 네트워크를 통해 연결되어 서로 상호작용하는 기술 및 서비스를 말한다. 사물인터넷의 기반은 각종 센서라고 할 수 있다. 사물에 설치된 센서를 통해 수집한 데이터를 취합, 분류, 분석하여 의미있는 정보를 생성하고 공유한다.

현재 사물인터넷 기술은 유무선 네트워크 기술과 인공지능, 데이터 처리 기술이 발전함에 따라 함께 발전해 국내외에서 많은 연구 및 개발이 이루어지고 있다. 가정의 가전제품, 보안장치 등을 연결하는 '스마트 홈'에서부터 에너지 공급망을 관리하는 '스마트 그리드', 도시의 각종 데이터를 통해 도시를 효율적으로 관리하는 '스마트 시티'까지 다양한 분야에 걸쳐 사물인터넷이 활용되고 있다.

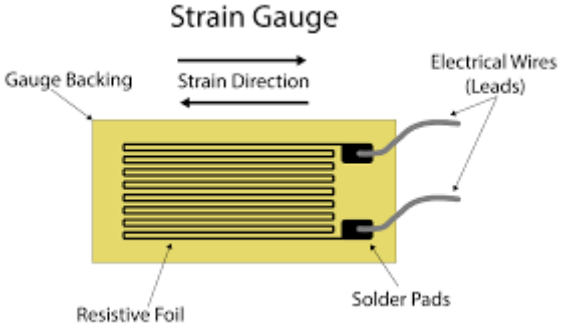
또한, 사물인터넷의 기술 발달과 함께 사물인터넷의 기술적인 기반을 제공하는 IoT 플랫폼 역시 계속해서 개발되고 있다. IoT 플랫폼을 개발하는 업체는 기업과 사용자에게 자신들의 플랫폼을 오픈해 다양한 장치를 연결해 테스트해볼 수 있는 환경을 제공하고 있다.

2) 로드셀(Loadcell)

로드셀(Loadcell)은 힘 또는 하중과 같은 물리량을 측정할 수 있는 센서다. 로드셀에 힘이 가해지면 그 힘만큼 전기 신호가 발생하고 이를 통해 가해진 힘, 하중을 측정하는 것이다. 로드셀의 표면에는 스트레인 게이지(strain gage)가 붙어있는데 로드셀에 힘이 가해지면 스트레인 게이지가 함께 변형된다. 이에 따라 스트레인 게이지의 저항값이 변함에 따라 전기신호가 변화하는 것이다. 로드셀은 다른 센서에 비해 비교적 가격이 저렴하고 수명이 길다는 장점이 있어 보편적으로 사용되게 되었다.

(1) 스트레인 게이지(strain gage)

스트레인 게이지는 로드셀의 핵심이 되는 장치로 물체의 스트레인, 즉 변형을 측정하는 데 사용되는 저항형 센서다. 스트레인 게이지는 일반적으로 아래와 같은 구조를 보인다.

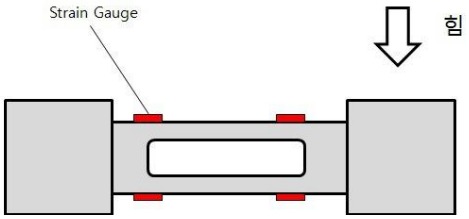


(▲스트레인 게이지의 구조)

가운데 위치한 금속저항체가 스트레인 게이지의 핵심이라고 할 수 있다. 이 금속저항체에 힘이 가해지면 단면적에 변화하면서 저항값이 변화하게 된다. 이 저항값의 변화로 인해 측정되는 전기신호가 달라지는 것이다.

(2) 로드셀의 구조와 원리

로드셀은 앞서 설명한 스트레인 게이지를 활용하여 힘을 측정할 수 있는 센서다. 부착된 스트레인 게이지에 힘이 가해지면 로드셀과 함께 스트레인 게이지가 늘어나거나 수축된다. 이러한 변화에 의해 스트레인 게이지의 저항값이 변화하고 측정되는 전기신호가 변하는 것을 이용하여 하중을 측정하는 것이다. 로드셀의 구조는 아래와 같다.

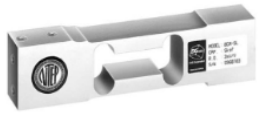


(▲로드셀의 구조)

위의 이미지에서 볼 수 있듯이 하나의 로드셀에는 4개의 스트레인 게이지가 부착되어 있다. 하나의 스트레인 게이지만 있어도 힘을 측정할 수는 있다. 하지만 하나의 스트레인 게이지만 사용하면 출력되는 전기신호가 매우 작고 오차가 발생한다. 그렇기 때문에 스트레인 게이지를 로드셀의 인장 및 압축 방향으로 각각 2개씩 부착하여 로드셀을 제작한다.

(3) 로드셀의 종류

로드셀의 종류는 형태에 따라 크게 빔형, 원주형, S자형, 다이아그램형의 4가지 종류로 구분된다. 각 형태별 로드셀은 아래의 모습과 같다. 아래의 로드셀이 가장 기본적인 형태이며, 이 외에도 다양한 형태의 로드셀을 제작할 수 있다. 일반적으로 하중을 측정할 때 가장 많이 사용되는 형태는 '빔형 로드셀'이다.

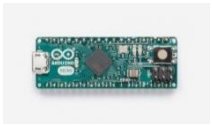
빔형 로드셀	원주형 로드셀	S자형 로드셀	다이아그램형 로드셀
			

3) 아두이노(Arduino)

아두이노는 오픈 소스를 기반으로 한 단일 보드 마이크로 컨트롤러로 완성된 보드와 관련된 개발 도구 및 환경을 말한다. 아두이노는 다수의 스위치나 센서로부터 데이터를 받아, LED나 모터와 같은 외부 장치를 통제해 상호작용할 수 있도록 한다. 아두이노 통합 개발환경(IDE)를 제공하며, 오픈 소스이기 때문에 여러 가지 프로젝트를 수행할 수 있다.

아두이노의 보드는 8비트 AVR 마이크로 컨트롤러와 프로그래밍 및 다른 서킷과의 결합을 용이하게 해주는 부속품으로 구성되어 있다. 현재 Uno와 같은 주요 모델은 14개의 디지털 I/O핀을 제공하고 있다. 이 중 6개의 핀은 PWN(pulse-width modulated) 신호를, 다른 6개의 핀은 디지털 I/O핀으로 혼용이 가능한 아날로그 입력 단자다.

아두이노는 크기, 입출력 단자의 개수, 부가적인 기능 등에 따라 다양한 모델이 있다. 그 중 대표적으로 활용되고 있는 종류는 아래의 표와 같다.

모델명	사진	기능 소개
아두이노 우노 (Arduino Uno)		일반적으로 가장 많이 활용되는 보드 6개의 아날로그 입력, 14개의 디디지털 입출력을 제공
아두이노 나노 (Arduino Nano)		우노와 유사하지만 크기가 작음 후속모델인 Nano 33 IoT는 와이파이 기능을, Nano 33 BLE는 블루투스 기능을 지원함
아두이노 메가 2560 (Arduino Mega 2560)		우노보다 크기가 큰 보드 다른 모델보다 메모리 용량이 크고 제공하는 입출력 핀의 개수가 많음
아두이노 마이크로 (Arduino Micro)		작은 사이즈의 보드 우노나 메가에서 표준 USB 케이블을 사용하는 것과 달리 Micro-USB 케이블이 필요

4) 블루투스(Bluetooth)

블루투스는 디지털 통신 기기를 위한 개인 근거리 무선 통신 산업 표준을 말한다. 2.4~2.485GHz의 단파 UHF 전파를 이용해 전자 장비 간 통신을 통해 정보를 주고받는다. 현재 개인용 컴퓨터에서 사용하는 마우스, 키보드를 비롯해, 휴대전화 및 스마트폰, 태블릿, 스피커 등에서 문자 정보 및 음성 정보를 비교적 낮은 속도로 무선 통신하는 용도로 사용되고 있다.

2016년 사물인터넷 기술에 초점을 맞춘 블루투스 5가 소개되었다. 또한, 코로나 19로 인한 팬데믹을 겪으며 감염병 위험에 대한 노출을 최소화하면서 진료 방식 개선을 위한

의료기기에 대한 솔루션으로 블루투스 기술이 각광받고 있다. 또한, 재택 근무가 증가하는 추세에 따라 블루투스 PC 액세서리 시장이 계속해서 성장하고 있다.

5) 와이파이(Wi-Fi)

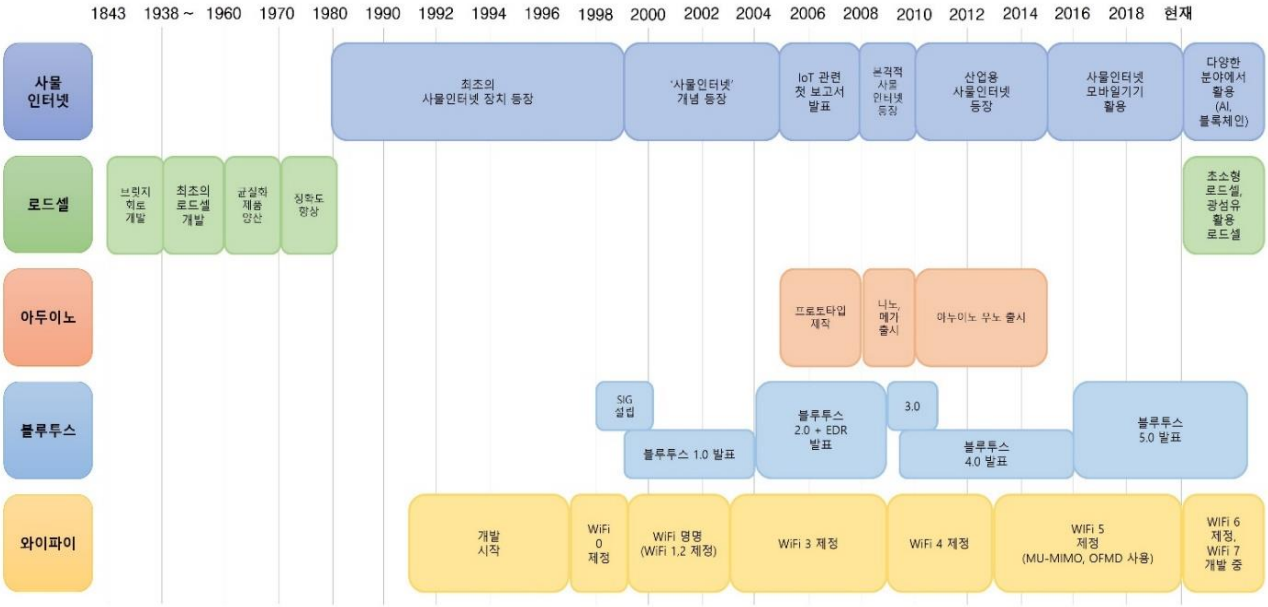
와이파이는 무선 통신 표준 기술 중 하나 인 IEEE 802.11에 기반한 서로 다른 장치들 간의 데이터 전송 규약을 말한다. 일반적으로 부채꼴 모양의 아이콘으로 Wi-Fi를 나타내며, 부채꼴을 통해 신호의 강도를 표현한다.

와이파이를 사용하기 위해서는 무선 인터넷 공유기가 필요하다. 데이터 전송 규약인만큼 표준만 준수한다면 스마트폰, 태블릿 컴퓨터, 컴퓨터, 노트북 등 어떤 장치에서도 사용이 가능하다. 와이파이의 상용화에 따라 현재에는 휴대용 기기 외의 다양한 가전 제품에서도 와이파이를 지원하고 있다.

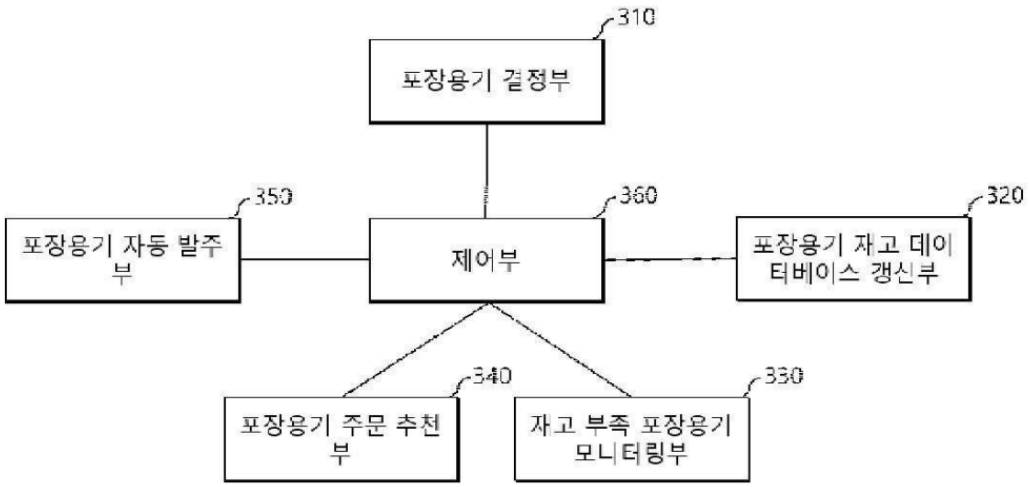
와이파이는 일대다 통신 방식을 사용하기 때문에 연결된 장치가 증가하면 전송속도는 반비례하여 감소한다. 또한, 근거리 통신을 전제로 제정된 규약이기 때문에 커버리지는 개활지를 기준으로 200m 정도이다.

와이파이의 버전은 802.11 뒤에 붙은 알파벳으로 구분된다. 모바일 기기의 보급이 증가함에 따라 와이파이가 함께 발달되었다. 또한, 최근에는 사물인터넷의 발달로 속도와 효율성을 모두 만족해야 하는 와이파이의 필요성이 대두되고 있다. 2019년 이러한 필요성에 의해 등장한 '와이파이 6(IEEE 802.11ax)'는 다중 사용자, 다중 입출력 기술을 활용하여 여러 장치가 동시에 연결되어도 속도 저하를 막을 수 있다.

나. 기술 로드맵

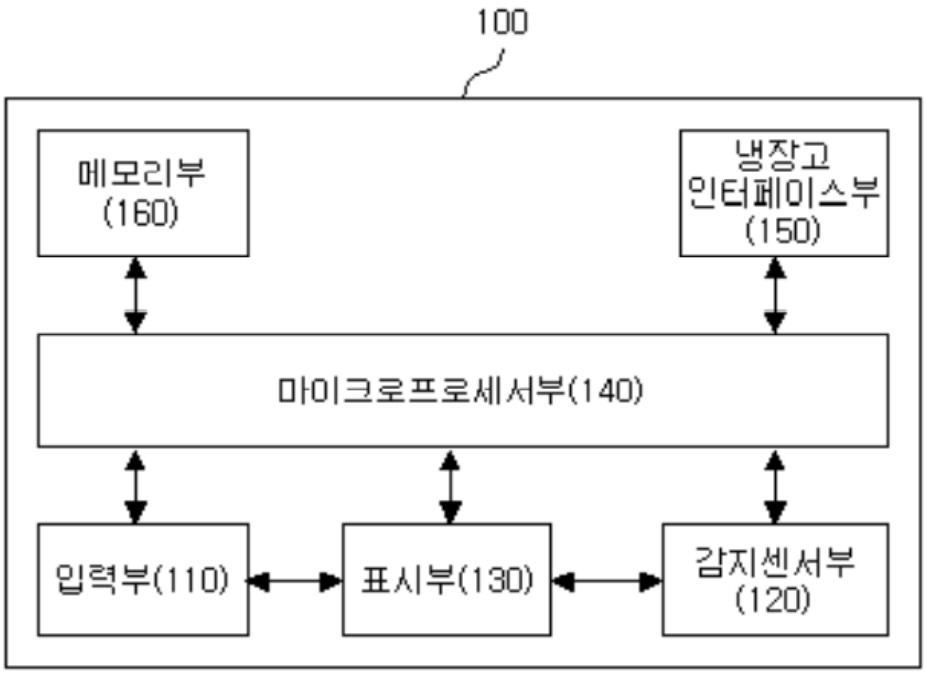


다. 특허조사

1) 포장용기 자동 주문발주 장치
출원번호 : 10-2019-0112512

포장용기 재고 데이터베이스를 주기적으로 모니터링하여 자동으로 재고 부족 포장용기 주문을 추천할 수 있는 포장용기 자동 주문발주 장치를 제공하고자 한다. 재고 부족 포장용기의 종류를 기초로 포장용기 주문 입찰 문서를 생성하고 적어도 하나의 포장용기 발주처에 의한 가격 입찰을 기초로 특정 포장용기 발주처 정보를 추천에 포함시킬 수 있는 사용자 맞춤형 자동 발주 시스템 및 방법을 제공하고자 한다.

2) 인터넷 냉장고의 식품 자동 주문 시스템

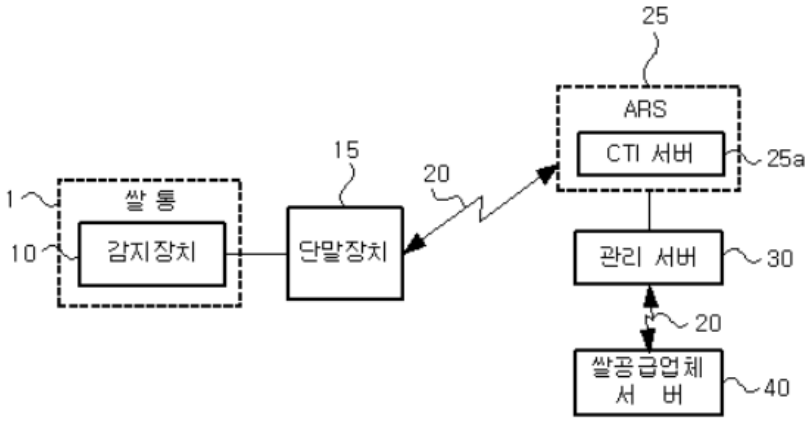
출원번호 : 10-2002-0061861



냉장고에 저장되는 식품의 상태정보의 입력이 가능하고 감지한 현재 보관량 및 상기 식품정보가 표시되는 표시부를 포함하는 동시에 상기 보관량이 최소 요구량 미만인 경우 인터넷 냉장고에 주문 요청 신호 또는 주문 실행 신호를 출력하는 마이크로 프로세서를 갖추는 식품 용기가 상기 인터넷 냉장고와 연동하여 동작됨에 따라 상기 인터넷 냉장고의 외관에 부착된 디스플레이 수단을 통해 상기 식품의 주문을 요청하거나 자동 접속하여 식품 주문을 실행하는 인터넷 냉장고의 식품 자동 주문 시스템을 제공하는 데 있다

3) [거절] 쌀 수납장치 및 이 장치를 이용하여 쌀의 재고량을 관리하는 시스템

출원번호 : 10-2001-0048890

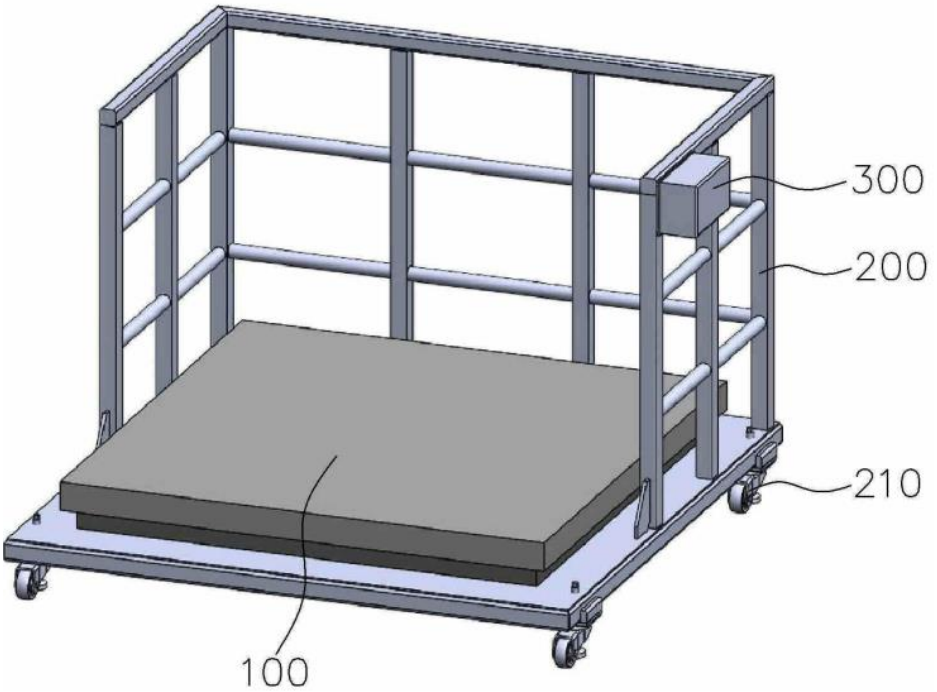


쌀통 내의 쌀 재고량을 측정하여 기준량 이하로 쌀이 남아있는 것으로 감지되었을 때, 통신망을 통해 쌀 공급업체에 배달주문이 자동으로 이루어지도록 하는 쌀통 재고량 관리를 위한 데이터통신 시스템을 제공하는 것을 그 목적으로 한다

거절 사유	인용한 발명과 기술적으로 큰 차이가 없고 이를 이용해서 해당 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 발명할 수 있는 것

4) [거절] 소모품 자동발주 시스템

출원번호 : 10-2019-0026093



소모품의 무게 측정을 통하여 소모품 잔량을 실시간으로 확인하고 발주하여 소모품 재고를 관리할 수 있도록 하는 소모품 자동발주 시스템을 제안하고자 한다.

거절 사유	인용발명 쌀 수납장치 및 이 장치를 이용하여 쌀의 재고량을 관리하는 시스템을 통상의 기술자가 단순 설계 변경하여 쉽게 도출 가능한 시스템임

라. 특허전략

특정 물품에 대해 자동주문을 하는 시스템에 관한 특허는 이미 과거부터 출원된 것들이 존재했다. 그 특허들을 살펴보면, 먼저 1) 포장용기 자동 주문발주 장치는 무게가 아닌 전산을 통해 남은 수량을 파악하고, 별도의 프로그램을 통해 공급사와 사용자에게 각각 알림이 가게 된다. 하지만 본 과제에서는 재고 파악에 무게를 이용하며, 자동 주문에는 오픈마켓을 이용하기 때문에 근본적으로 작동원리가 달라 문제가 되지 않을 것으로 보인다.

재고 파악에 무게를 이용한 특허들도 존재했는데, 2) 인터넷 냉장고의 식품 자동주문 시스템은 2002년에 출원된 특허로 이미 그 특허가 소멸되었다. 3) 쌀 수납장치 및 이 장치를 이용하여 쌀의 재고량을 관리하는 시스템의 경우에는 특허가 거절되었는데, 인용한 일본의 특허와 기술적으로 큰 차이가 없고, 해당 기술분야에서 통상의 기술자가 일본의 특허를 이용하여 쉽게 개발할 수 있다는 이유 때문에 거절되었다. 4) 소모품 자동발주 시스템은 3) 쌀 수납장치를 인용하였는데, 이 역시 인용한 특허를 통해 쉽게 도출 가능한 시스템이라는 이유로 특허 출원이 거절되었다.

따라서 국내에서는 본 과제를 통해 개발된 제품을 판매하는 것에 대해 특허상 문제가 없을 것으로 사료되며, 해외에서의 판매에 대해서는 조금 더 면밀한 조사가 필요하다.

1.3 관련 시장에 대한 분석

가. 경쟁제품 조사 비교

회사	Amazon	트라이포드랩	올트
제품명	Amazon dash smart shelf	weightrip	IoT Smart Scale
사진			
목표고객	개인 및 기업	개인 및 기업	기업
특징	- 아마존 내의 상품만 주문 가능 - 3개의 크기 옵션(7*7″, 12*10″, 18*13″)	- 디바이스는 무료, 정기구독 - AI를 활용하여 배송시 특이사항을 감지하고 주문 시점 추천	- 최대 25개 종류의 부품 적재 - 소형 부품의 수량도 정확하게 파악(최소 10g ~ 최대 100kg) - 각 저울별 품목 개수, 위치 확인 가능 - 블츨 지시 기능

분석내용

국내와 해외에 모두 무게를 측정하여 재고를 파악하고 자동주문을 하는 비슷한 제품이 있었다. 아마존의 smart dash smart shelf는 개인 및 기업을 목표 고객으로 하고 세계 최대의 유통 플랫폼인 아마존을 이용하는 것이 특징이며, 3개의 크기 옵션을 제공한다. 아마존 제품의 가격 19.99달러는 가격 책정에 있어서 기준점이 될 수 있을 것이다. 아마존 내의 상품만 주문 가능하다는 점이 이 제품의 최대 장점이자 단점이다. 국내 등 아마존이 진출하지 않은 곳에서는 아마존을 이용하기 힘들기 때문에 비교우위를 가질 수 있을 것이다

국내 회사인 트라이포드랩의 weightrip은 개인 및 기업을 목표고객으로 하고, 많은 수익을 낼 수 있는 정기구독을 지향한다. 하지만 실제 제품 판매 페이지나 회사 홈페이지 등이 없어 자세한 정보를 찾아볼 수 없었다.

올트 역시 국내 회사로 IoT Smart Scale은 경쟁제품 중에서 가장 강력한 기능을 지원한다. 경쟁 제품들과 달리 최대 25개 종류의 부품을 적재하고 무게 측정의 정밀도가 높아 소형 부품의 수량도 정확하게 파악할 수 있다. 각 저울별로 품목 개수와 위치가 확인이 가능하며, 저울마다 LED가 있어 불출지시가 가능하다. 기업들을 대상으로 하며, 일반적으로 사용할 수 있는 제품은 아니고 올트의 자체 솔루션으로 자사에서 판매하는 소모품(볼트, 너트 등)만을 대상으로 사용할 수 있는 제품이라는 한계가 있다. 일반적으로 사용할 수 있는 자유도 면에서 비교우위를 가지지만 올트사가 사업을 확장한다면 강력한 경쟁자가 될것이다.

나. 마케팅 전략

1.4 개발과제의 기대효과

가. 기술적 기대효과

사물인터넷 시장의 규모는 꾸준히 성장세를 보이고 있지만 대부분 대규모 상업 시설에 그 비중이 편중되어 있어 본 프로젝트에서는 중소 규모의 사업장에서도 효율적으로 재고를 관리할 수 있도록 지정된 제품의 무게 측정을 통하여 해당 정보를 등록할 수 있는 하드웨어 기반 모듈을 개발한다. 또한 사용량 정보를 데이터화 하여 보여주고 제품 잔여량이 10% 이하로 표시되면 사용자가 등록해 놓은 구매 링크를 바탕으로 자동 재주문을 하는 기능을 구현할 수 있도록 PC 프로그램을 개발하기 때문에 기존의 개발사의 전용 쇼핑몰을 통해서만 구입하는 구조가 아닌 사용자가 가격비교를 하고 직접 결정할 수 있도록 설정되었기 때문에 가격 경쟁성이 우수하다는 장점이 있을 것으로 보이며 생산 규모의 맞춤형 제작으로 산업내 자동화를 촉진시키고 불필요한 시간적 인력 소모를 줄이고 원활한 생산조달이 이루어져 전반적인 업무 효율을 향상시키는 효과를 가져올 수 있을 것으로 기대된다.

또한 본 프로젝트에서 적용하고자 하는 무게 감지를 통한 제품 소모량 파악 기술의 안정화는 기존의 기업들의 전용 혹은 독자적인 서비스 운영방식에서 벗어나 대중적 사용 가능성을 제고시켜 사용자에게 선택권을 부여하는 방향으로 추진될 것으로 전망되며 이로 인해 산업 시설, 무인 마켓, 개인 주거 공간 등 다양한 분야에 시스템적 결합을 통하여 전자 기기간의 호환성을 높인 제품들의 수요가 늘어날 것으로 보인다.

나. 경제적 및 사회적 파급효과

본 프로젝트에서 목표로 하는 로드셀을 활용한 무게 감지 자동 주문 시스템은 중소 규모의 사업장에서 공간의 제약을 적게 받고 생산 규모에 맞춤형으로 제작을 목표로 하고 있기에 공간 점유율이 낮고 사용 편의성이 높을 것으로 예상되며 설정된 무게 기준내에서는 모든 품목들을 자유롭게 등록하여 사용할 수 있게 설계되었다. 또한 개인 주거 공간에서도 필요에 따라 활용이 가능하다. 예를 들어 쌀, 세탁 세제, 화장품 등 생활용품의 소모도를 측정하여 PC로 전송해주어 사용자가 편리하게 주거 시설의 전반적인 소요량을 체크할 수 있어 시간 효율을 높여주는 등 사용자 만족도가 높게 형성될 수 있을 것으로 파악된다. 제작 비용 또한 시중에 판매되고 있는 스마트 저울 모델과 가격대가 유사하게 책정되어 소비자 가격 부담을 줄여주고 선택의 범위를 넓혀주어 더 실용성 있는 제품을 선호할 수 있도록 기획하여 판매의 폭을 넓힐 수 있을 것으로 예상한다.

<사용 정보를 공유하여 내수 시장의 안정화>

사업장을 운영하는 사용자가 등록한 제품의 연, 월, 일 단계로 사용량에 대한 통계를 판매처에 공유하여 판매사가 주수요 품목에 대한 공급계획을 미리 수립하고 원재료를 조달할 수 있는 시간을 확보할 수 있을 것으로 예상된다. 따라서 불용재고와 같은 과잉공급 현상을 줄이고 불특정한 수요의 가격 변동폭을 안정화시킬 수 있을 것으로 기대된다.

<사회 계층 간의 실용성을 높이고 기술적 접목으로 분야의 확장가능성>

사회 계층간의 디지털 기술 능력에 차이가 있어 서비스 이용과 정보 획득에 불평등이 발생하는 등 문제점이 존재한다. 이러한 정보취약계층의 디지털 소외 현상을 줄이고자 본 프로젝트에서 개발하는 방향성은 간결한 조작으로 실시간 정보를 시각적으로 보여주고 버튼 클릭 한번 혹은 등록이후의 변동성이 없을 경우 자동 주문이 가능하기에 디지털에 대한 이해도가 떨어져도 충분히 실용성 있게 사용할 수 있도록 초점을 맞췄기에 산업 시설 이외에도 고령층, 장애인 등 사회약자에 특화되어 사용범위가 확대되고 가정용 AI로봇 등과 기술적 결합을 통해 돌봄 서비스 범위를 확대하여 사회적 고부가가치를 만들어 갈 수 있을 것으로 보인다.

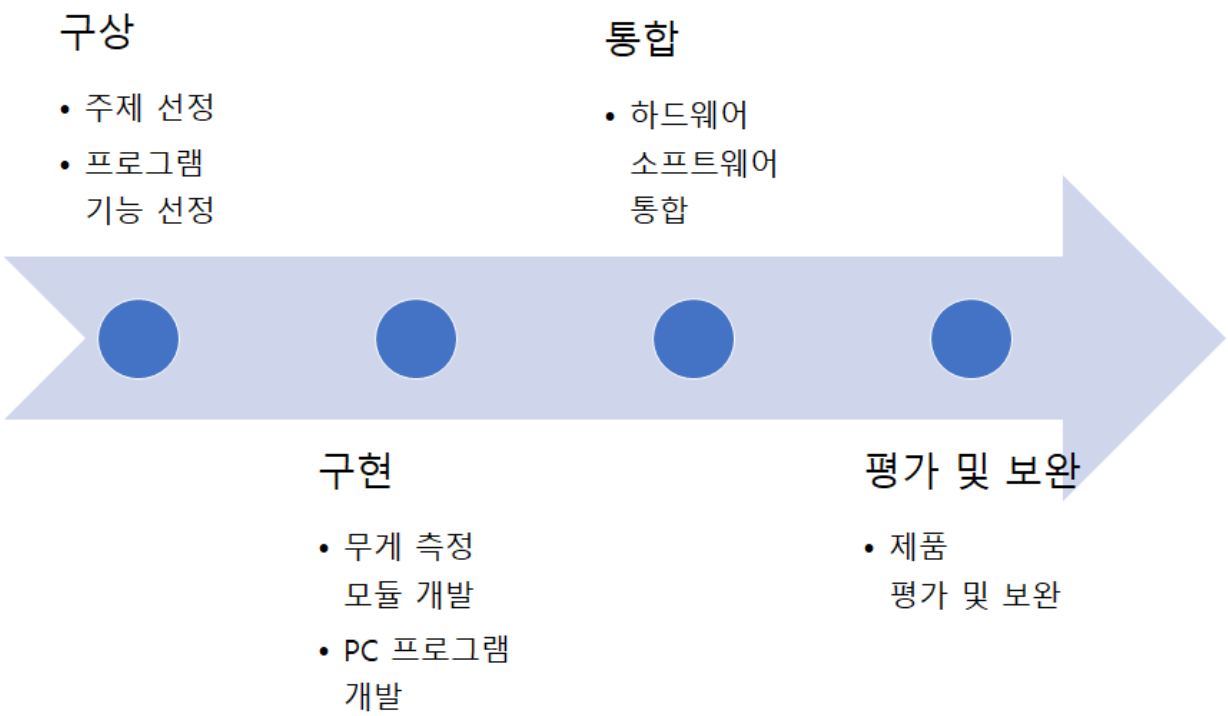
1.5 구성원 및 추진체계

가. 개발 일정

단계별 세부개발 내용	담당자	개발기간 (월단위)				비 고
		3	4	5	6	
프로젝트 주제 선정	전원					
PC 프로그램 제공 기능 선정	전원					
아두이노 회로 구성	강재언 이휘호					
아두이노-PC 연결	강재언 이휘호					
무게 측정 모듈 제작	강재언 이휘호					
무게 감지, 재고 수량 파악 알고리즘 구현	강재언					
재고 부족 시 알림 기능 구현	이휘호					
자동 주문 알고리즘 구현	김진					
PC 프로그램 UI 구축	김진					
하드웨어-소프트웨어 연동 확인	전원					
1차 테스트 및 보완	전원					
특허출원서 작성	전원					
2차 테스트 및 보완	전원					
최종 테스트 및 보완	전원					
최종 발표 및 시연	전원					

나. 구성원 및 추진체계

팀원명	역할
강재언	무게 측정 모듈 제작
이휘호	무게 측정 모듈 제작
김진	PC 프로그램 개발



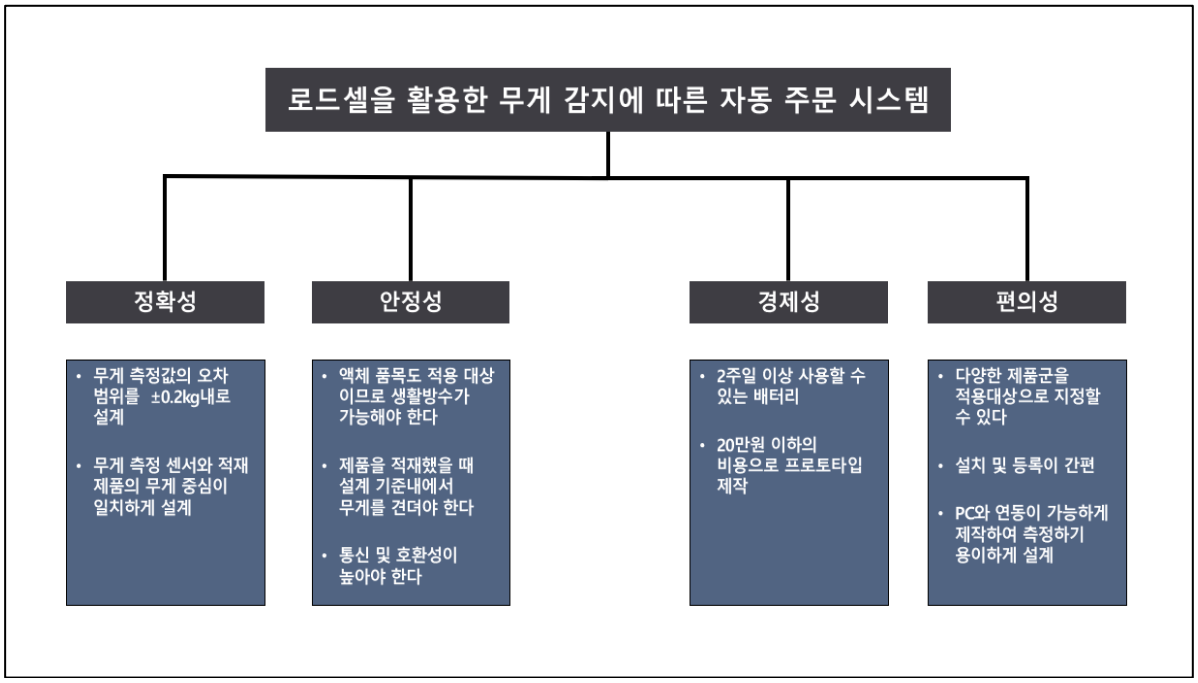
2. 설계

2 .1 설계사양

가. 제품 요구사항

번호	요 구 사 항	D or W	중요도
1	무게 측정값이 정확해야 한다	D	상
2	무게 중심이 잘 맞아야 한다	D	상
3	생활방수가 가능해야 한다	D	하
4	일정 거리까지 모듈과 PC가 연결이 가능해야 한다	W	중
5	모듈과 PC간 연결 안전성이 높아야 한다	D	상
6	프로토타입 제작 비용이 저렴해야 한다	D	상
7	배터리 지속 시간이 길어야 한다	D	중
8	제품의 설치 및 등록이 간편해야 한다	D	상
9	PC 프로그램의 GUI와 사용은 직관적이어야 한다	W	중
10	연결이 끊겼을 때 모듈 자체에 데이터가 저장되어야 한다	D	상

목적계통도



나. 설계사양

하드웨어

1) 무게 측정 오차 ±0.2kg 이내

본 프로젝트에서의 무게 감지 모듈은 중소 규모의 사업장에 특화하여 제작을 목표로 하고 있기 때문에 산업 현장에서 재고관리의 효율성을 높이는 것이 중요하게 작용하기 때문에 무게를 측정하는 센서의 정확도가 높아야 한다. 따라서 무게 측정에 정밀도가 높은 센서를 사용하고 미끄럼 방지 고무 패킹을 사용하여 바닥에 밀착시켜 무게 중심이 흔들리는 것을 줄여 주어 상대오차가 ±0.2kg 이내로 선정될 수 있도록 설계하고자 한다.

2) 최대 하중 150kg 이내

시중에서 판매되고 있는 체중계 모델과 본 프로젝트에서의 무게 감지 모듈 응용 방향성이 유사한 관계로 기존의 일반적인 체중계 측정 범위인 150kg~200kg 를 적용 기준으로 잡았다. 또한 중소 규모의 사업장의 크기와 보편적으로 사용되고 있는 제품의 무게가 100kg 미만인 특성을 고려해봤을 때 150kg 이내의 무게 적용 기준을 잡고 설계하고자 한다.

3) 프로토타입 크기 30cm * 30cm * 5cm 이내 (가로 * 세로 * 높이)

기존의 체중계 모델과 디자인을 유사하게 제작하여 소비자에게 친숙함을 주어 가격대비 성능면에서 더욱 차별성이 있고 효율성이 높음을 강조하고자 무게 감지 모듈 크기를 30cm * 30cm * 5cm 이내로 제작하고자 한다.

4) 배터리 지속시간 336 시간 이상

일반적인 사업장의 운영 특성을 분석하였을 때 일 평균 12 시간에서 24 시간으로 운영시간대가 불일치하므로 일 24 시간을 기준으로 가정을 했을 때 2 주간 배터리 교체 없이 사용을 하기 위해선 배터리 지속시간이 336 시간 이상이어야 한다. 따라서 이러한 설계 사양을 만족할 수 있는 회로를 구성하고 소프트웨어를 설계하고자 한다.

5) 최소 연결거리 10m 이상

무게 감지 모듈과 PC 간의 연결을 필요로 하기 때문에 무선 통신 기술을 적용하여야 한다. 단 각각의 사업장 구조 및 공간 밀도가 상이하므로 신호 감쇄 등 원인으로 목표 연결 거리까지 원활한 통신이 어려울 수 있는 관계로 신호 증폭기 등 부가적인 기술적 융합성도 고려하여 설계 사양을 설정한다.

소프트웨어

1) 자동 주문 알고리즘 작동 성공률 90% 이상

본 설계에 있어서 자동 주문 알고리즘의 구현은 핵심 기능 부분 중에 하나이다. 시제품을 제작하여 지정된 구매 링크를 등록하고 자동 주문 명령을 실행하면 그 이후로 추가적인 실행 명령이 필요없이 자동으로 주문이 완료되는 편리성을 제공하기 위함이다. 소프트웨어 설계에서 다양한 경우의 수를 파악하여 작동 오류를 최소화하고 하드웨어에서도 결함의 존재 여부를 파악하여 최종적으로 프로토타입에서 작동 성공률 90% 이상이어야 한다.

2) 프로그램 설치가 간편하고, 사용방법이 직관적

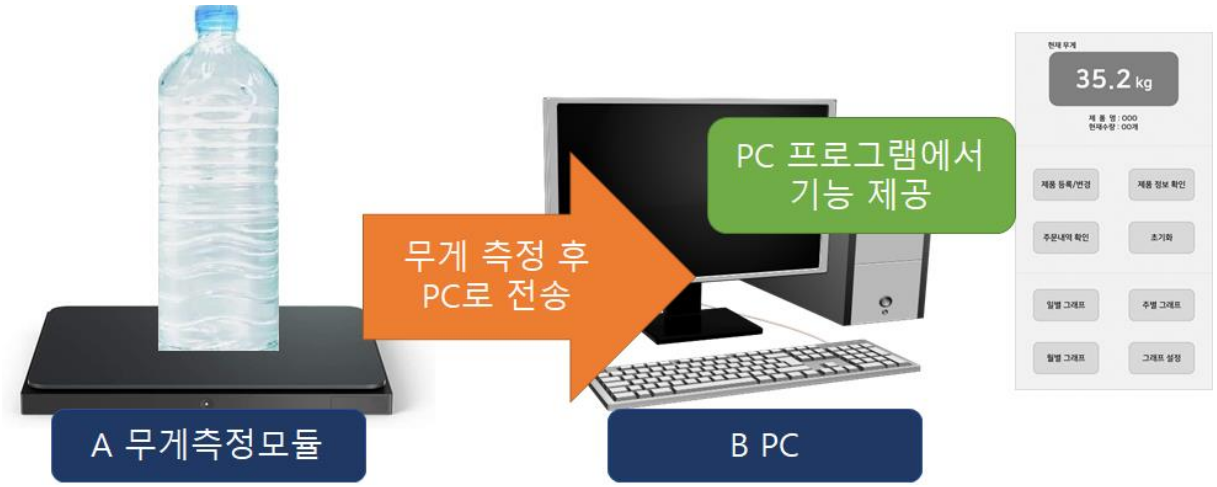
무게 감지 자동 주문 모듈은 사용 편리성에 최적화가 되어 있어야 시장에서의 경쟁력을 갖출 수 있다. 컴퓨터 조작을 어려워하는 계층에게도 손 쉽게 설치를 하고 사용할 수 있도록 매뉴얼을 제공하고 프로그램 다운로드 링크를 통해 설치만 하면 프로그램상 간단하게 구성된 데이터 표를 확인이 가능하고 조작 버튼 또한 간결하게 설계를 하여 소비자 맞춤형으로 제작 목표를 설정한다.

3) 연결이 끊겼을 때 모듈 자체에 데이터 저장

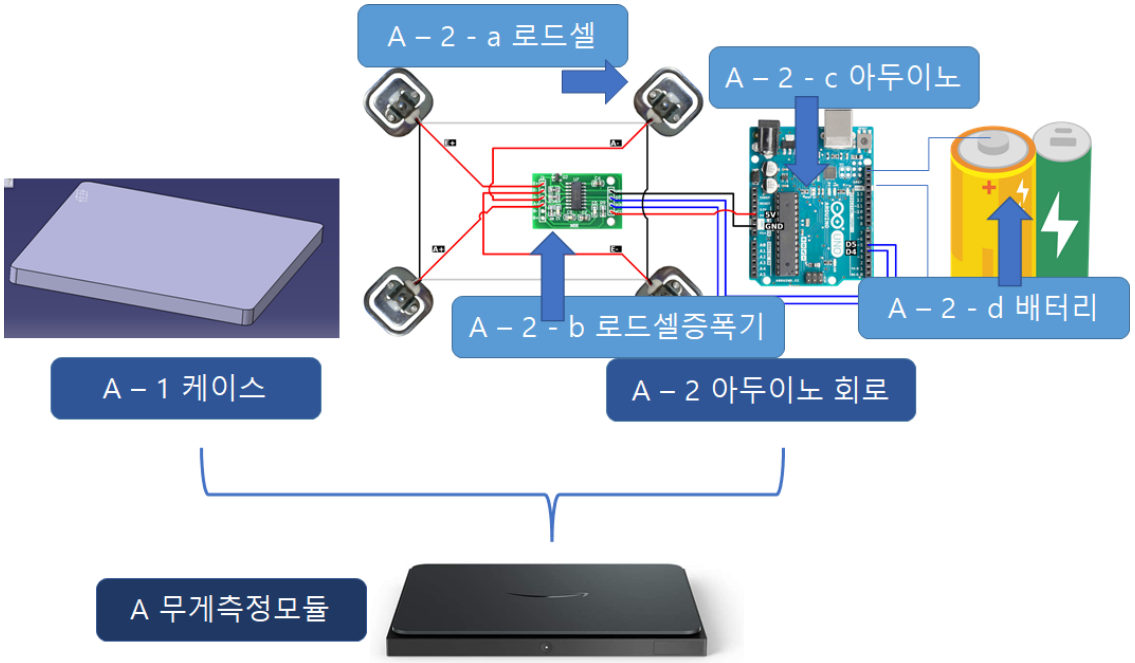
시스템 구동 중 예상치 못한 오작동 혹은 서버 과열로 발생한 연결 끊김 현상을 미연에 방지하고자 연결이 끊겼을 때 모듈 자체에 데이터가 저장되도록 설계 방향을 정하였다. 데이터의 손실을 최소화하여 사용자가 이용에 있어서 불편함이나 서비스 품질의 만족도가 떨어지는 상황을 미리 개발 단계에서 고려하여 향상시키는 방향으로 제작을 한다.

2 .2 개념설계안

작동원리 및 구조



무게측정모듈(A)에 그림의 생수와 같이 물품을 올려놓으면 무게가 측정되고, 측정된 무게를 PC(B)로 전송하면 PC 프로그램을 통해 일별, 주별, 월별 등의 사용량 통계를 제공해주는 사용량 통계 기능과 물품이 거의 다 떨어졌을 때 자동주문을 해주는 자동주문 기능 등을 제공한다.



무게측정모듈의 구성

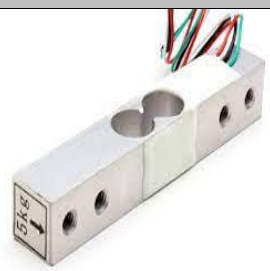
무게측정모듈(A)은 케이스(A – 1) 내부에 아두이노 회로(A – 2)를 부착한 형태로 구성된다. 케이스(A - 1) 부분은 아두이노 회로를 보호하고 사용 대상 물품을 올려놓고 그 무게를 아두이노 회로에 전달하는 역할을 하고, 아두이노 회로(A – 2)는 전달받은 무게를 측정하고 이를 PC에 무선통신을 통해 전달하는 역할을 한다.

아두이노 회로(A – 2)는 로드셀(A –2 – a), 아두이노(A – 2 – b), 로드셀 증폭기(A – 2 – c), 배터리(A – 2 – d)로 구성되어 있다. 로드셀(A –2 –a)은 로드셀에 가해진 무게를 전기신호로 바꾸어 무게를 측정할 수 있게 해주는 센서이다. 로드셀 증폭기(A –2 –c)는 로드셀에서 출력된 작은 전기신호를 아두이노가 인식할 수 있는 큰 신호로 증폭시켜주는 역할을 한다. 아두이노(A – 3 – c)는 로드셀과 로드셀 증폭기를 통해 출력된 전기신호를 무게로 환산하여 무게를 측정하고 측정된 무게를 무선통신을 통해 PC전송해주는 역할을 한다. 배터리(A – 2- d)는 아두이노가 동작하는데 필요한 전력을 공급해주는 역할을 한다.

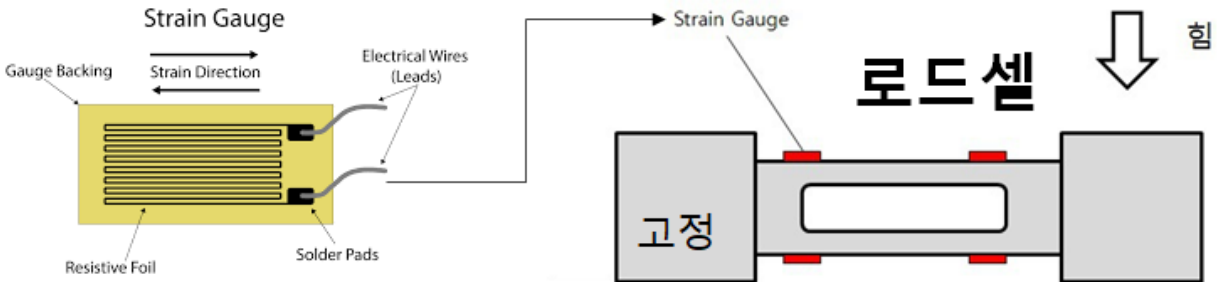
가. 무게측정모듈

1) 로드셀 종류

빔형, S자형, 원주형 등 다양한 종류의 로드셀이 있지만 부피가 작고 가격이 싸며 일반적으로 많이 사용되는 빔형 로드셀과 체중계용 로드셀 2가지를 비교해본다.

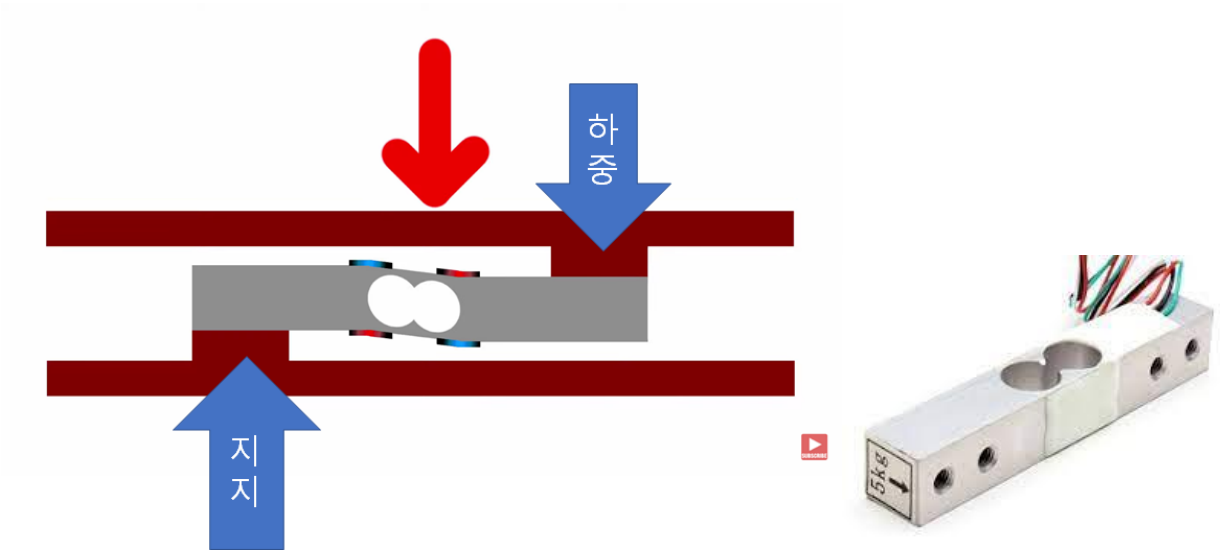
종류	빔형	체중계용
사진		

가) 로드셀 동작원리



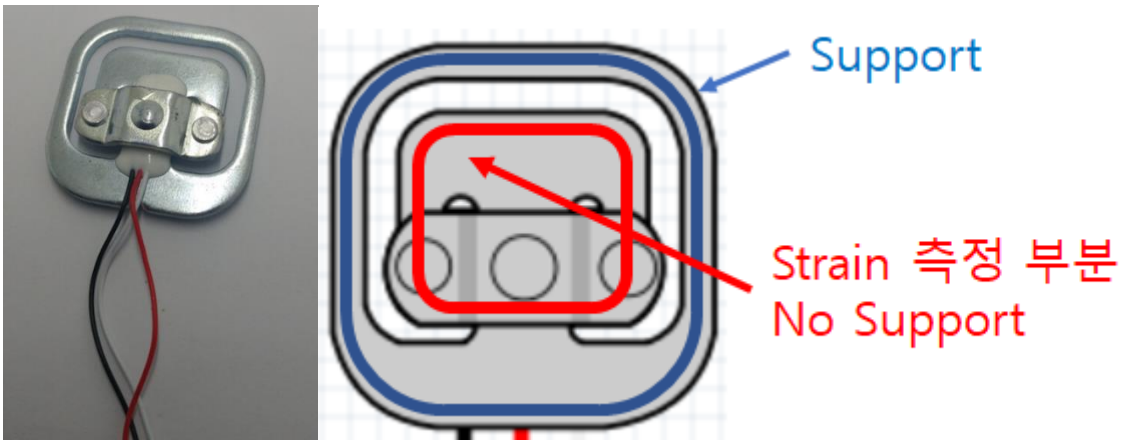
로드셀의 한쪽 부분을 고정시키고 다른 부분에 힘을 가하게 되면 로드셀에 변형이 생기게 된다. 로드셀에는 스트레인 게이지가 부착되어 있고 변형이 생김에 따라 스트레인 게이지 내부의 저항값이 달라지게 되고, 스트레인 게이지를 흐르는 전류에도 변화가 생긴다. 로드셀에 가해지는 하중이 클수록 전류의 변화도 커지게 되고 이를 이용하여 하중을 측정한다. 로드셀은 금속의 변형을 이용하기 때문에, 서포트를 두지 않고 의도적으로 변형이 생기게 만들 부분이 필요하다.

나) 빔형 로드셀



빔형 로드셀을 옆쪽에서 바라본 그림이다. 빔형 로드셀은 그림과 같이 한쪽 끝에만 서포트를 두고 반대편 끝에 힘을 가하여 로드셀에 변형이 생기도록 한다.

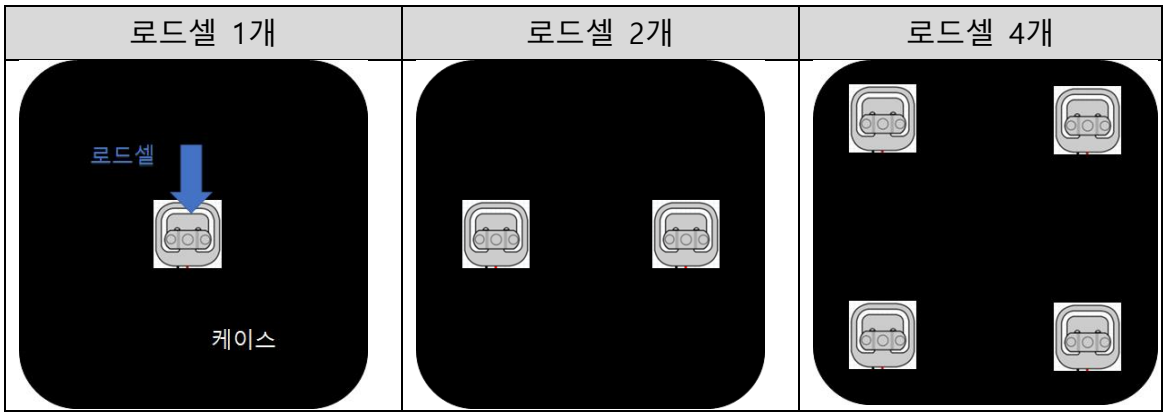
다) 체중계용 로드셀



체중계용 로드셀을 위쪽에서 바라본 그림이다. 안쪽 사각형(빨간색)과 바깥쪽 사각형(파란색)으로 되어있는 구조인데, 바깥쪽 사각형에선 로드셀을 지지하고, 가운데 있는 원 부분에 하중을 가하게 되면 안쪽 사각형(빨간색) 부분에서 변형이 생기도록 하고 안쪽에는 서포트를 두지 않는다.

빔형 로드셀과 체중계용 로드셀을 비교했을 때 가장 큰 차이점은 서포트를 두는 형태라고 할 수 있다. 빔형 로드셀은 한쪽 끝에만 서포트를 둘 수 있고 체중계용 로드셀은 바깥쪽 테두리를 따라서 서포트를 둘 수 있다.

2) 로드셀 개수

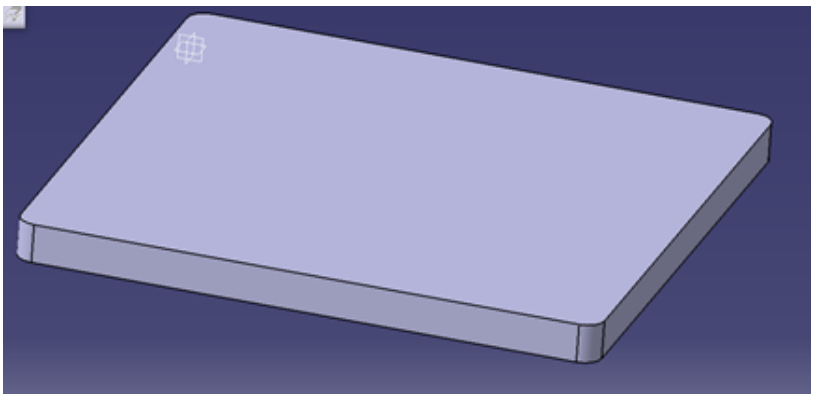


위에서 본 모습

무게측정모듈에 설치할 로드셀의 수이다. 로드셀의 수가 적으면 제작비용이 저렴해진다는 장점이 있지만,무게를 측정하는 물품의 무게중심이 케이스의 중앙에서 벗어나 가장자리로 향할수록 구조적 안정성이 떨어진다는 단점이 있다.

3) 케이스 형상

가) 케이스 외부 형상

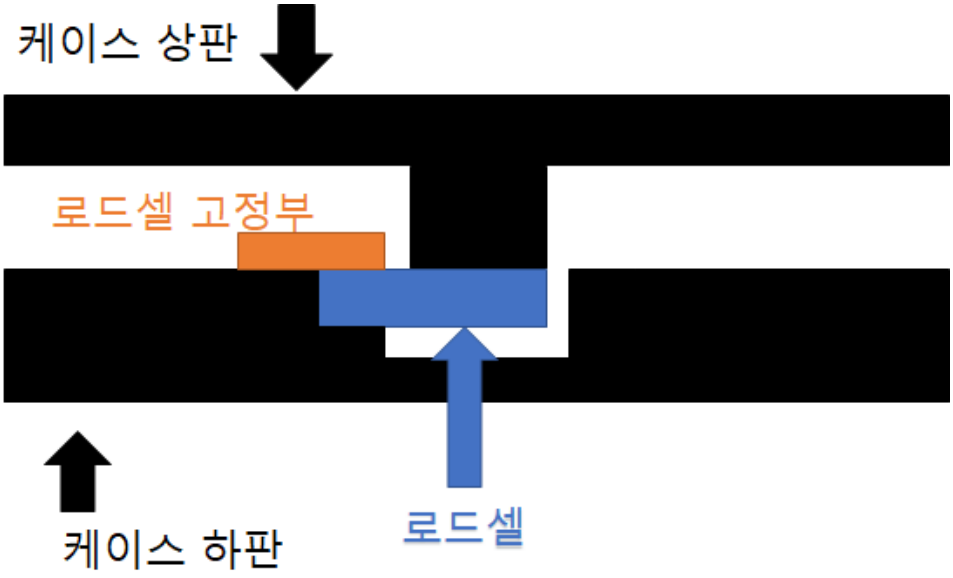


실제 사용시 사용자가 물품을 올려놓고 사용할 케이스의 외부 형상은 단순하며 활용도가 높은 직육면체 모양으로 하고, 중소규모의 사업장을 목표로 하므로 크기는 30cm * 30cm 정도로 한다.

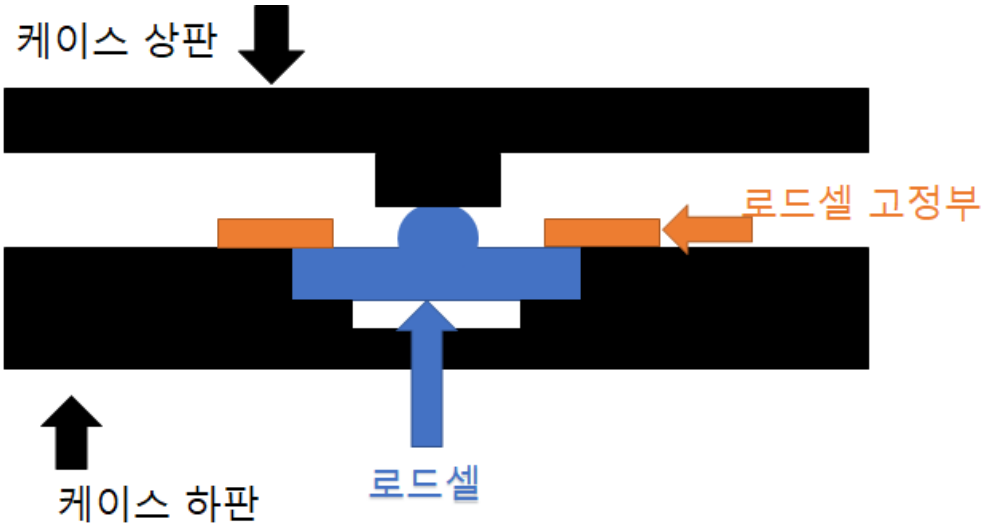
나) 로드셀 부착방식

로드셀을 케이스에 부착할 때 주의할 사항들은 다음과 같다.

(1) 로드셀은 움직이지 않게 단단히 고정되어 있어야 한다.
(2) 하중이 로드셀의 하중을 가하는 부분에 정확히 전달되어야 한다.
(3) 로드셀에는 수직 방향의 하중 외에 비틀림 등이 전달되어서는 안된다.
(4) 로드셀의 변형이 일어나는 부분에는 서포트가 없어야한다.



옆에서 본 모습(빔형 로드셀)

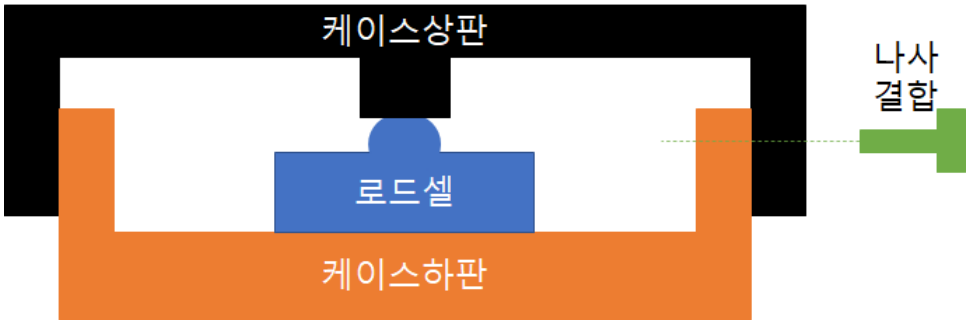


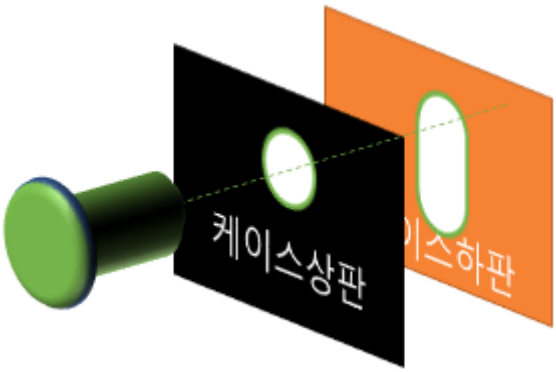
옆에서 본 모습(체중계용 로드셀)

로드셀을 설치할 때는 로드셀의 변형이 일어나는 부분에 변형이 생길 수 있도록 서포트를 없애기 위하여 케이스 하판의 높이를 낮춘다. 그 후 로드셀을 단단히 고정시키기 위하여 로드셀 고정부를 두고, 나사로 결합한다. 케이스 상판과 로드셀 중앙의 하중을 가하는 부분이 정확히 만나도록 하여 케이스 상판 위에 올려놓는 물품의 무게를 정확하게 측정할 수 있도록 한다.

다) 케이스 상판과 하판의 결합방식

케이스 상판과 하판이 분리되지 않도록 상판과 하판을 결합해주어야 하는데, 이 때 케이스 상판에 올려놓는 물품의 하중이 모두 로드셀에 전달되어야 하며, 케이스의 결합부위에 전달되지 않도록 주의하여야 한다.





케이스 상판의 나사구멍은 나사와 딱 맞는 원형으로 하고, 케이스 하판의 나사구멍은 둥근 직사각형 모양으로 한다. 케이스 상판과 하판이 멀어지는 방향으로 힘이 작용할 때는 나사에 힘이 전달되어 케이스가 분리되지 않고, 케이스 상판과 하판이 가까워지는 방향으로 힘이 작용할 때는 나사에 힘이 전달되지 않고 로드셀에 모두 전달된다.

4) 무선연결방식

	블루투스	와이파이
사용시전력	3mA	300mA
대기전력	0.05mA	1mA
통신거리	최대 40m (ver 5.0) 최대 10m (ver 4.2)	인터넷 연결이 가능한 모든곳
전송속도	최대 2Mbps	최대 100 ~ 1000Mbps

블루투스는 와이파이에 비해 낮은 속도, 낮은 전력 소모, 통신 거리의 제약 등을 장단점으로 갖는다. 본 프로젝트에서 전송하는 데이터는 블루투스의 전송속도로도 충분할 것으로 예상되며, 전송속도는 큰 단점이 되지 않을 것이다. 블루투스는 와이파이에 비해 사용시전력, 대기전력 모두에서 큰 우위를 가지고 있고, 와이파이는 블루투스에 비해 상대적으로 통신 가능 거리에서 제약이 없다는 장점이 있지만, 연결을 위해 외부 인터넷망과 라우터가 필요하다는 단점이 있다. 배터리 사용시간과 통신가능거리는 모두 사용자에게 중요한 요소이므로 적절한 타협이 필요할 것이다.

5) 아두이노 종류

종류	Micro	Nano 33 IoT	Nano 33 BLE
사진			
크기	18*48	18 * 45	18*45
동작전압	5V	3.3V	3,3V
가격	24,200	28,930	48,400
Bluetooth	X	4.2	5.0
Wi-Fi	X	O	X
비고	Bluetooth 모듈 필요		

3종류의 아두이노를 비교해보았다. 우선 크기는 아두이노 마이크로가 나노보다 가로로 3mm가 더 길지만 프로토타입 제작시 공간에 충분히 여유가 있을 것으로 추정되기 때문에 유의미한 차이는 아닐 것이다. 아두이노 마이크로는 상대적으로 저렴하지만, Bluetooth를 지원하지 않아 Bluetooth를 이용하기 위한 추가적인 모듈이 필요하고, 모듈을 위한 추가적인 비용과 공간이 필요하게 된다.

Nano 33 IoT와 Nano 33 BLE 2개 모델의 가장 큰 차이는 블루투스 버전과 와이파이 지원 여부이다. Nano 33 IoT는 Bluetooth 4.2를 지원하며 이론상 최대 통신 가능거리는 10m이고, Nano 33 BLE는 Bluetooth 5.0을 지원허묘 이론상 최대 통신 가능거리는 40m이다. 다만 Bluetooth 5.0을 이용하기 위해서는 PC 측에도 5Bluetooth 5.0 이상의 버전이 사용 가능해야 허며, 그 미만의 버전이라면 하위호환으로 PC측의 버전에 맞게 연결된다. 또한 Nano 33 IoT 모델은 Wi-Fi를 지원하지만, Nano 33 BLE 모델은 Wi-Fi를 지원하지 않는다.

현재시점에서 Nano 33 BLE의 가격이 크게 올라 Nano 33 IoT보다 약 20,000원 정도가 더 비싸 가격차가 크게 나는 점 또한 고려해야 할것이다.

6) 전원 공급 방식

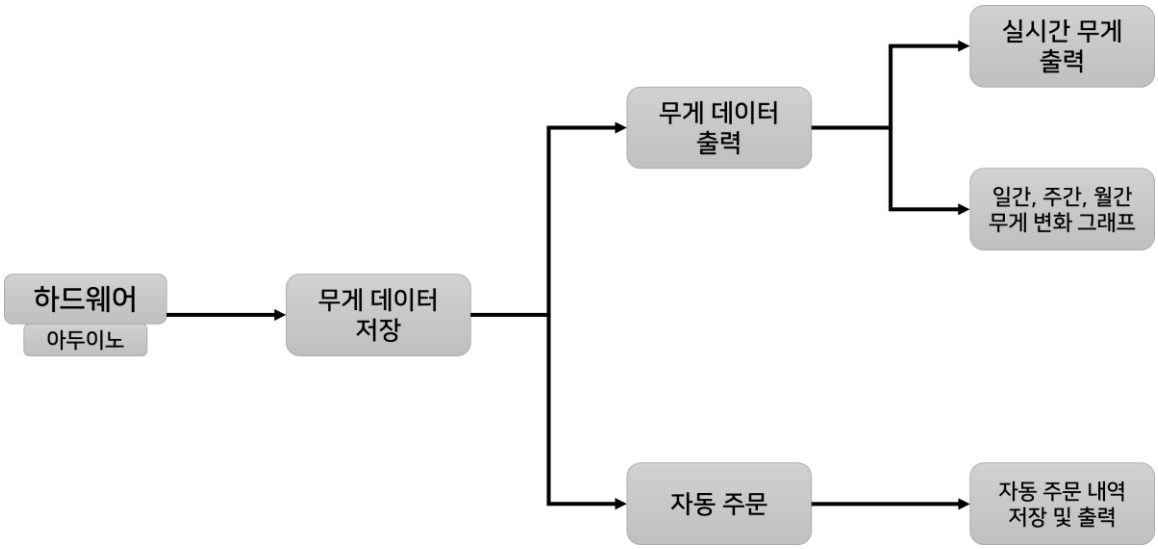
종류	동전형(CR2450)	리튬폴리머 배터리	어댑터
사진			
전압	3V	3.7V	-
크기	24mm * 5.4mm	20mm * 50mm	-
용량	620mAh	1000mAh	-
가격	1,000	7,380	5,000

3가지 종류의 전원공급방식을 비교해보았다. 우선 전원공급방식은 크게 어댑터 방식과 배터리 방식으로 나누어볼 수 있다. 어댑터방식은 전원공급을 위해 항상 유선 연결이 필요하며, 배터리 방식은 배터리를 통해 외부 전원 연결 없이도 사용이 가능하다. 어댑터 방식은 유선연결로 인해 설치공간에 제약이 생긴다는 단점이 있지만 배터리 교체나 충전 등을 신경 쓸 필요가 없어 편리하다는 장점이 있다.

배터리 방식은 배터리가 다 떨어졌을 때 교체를 해주는 교체식 방식과 충전을 해주는 충전식 방식이 있다. 리튬폴리머 배터리를 사용했을 때는 제품 가격이 높아져 소비자의 초기 투자 비용이 증가하지만, 추가적인 배터리 구입이 필요 없기 때문에 유지비가 들지 않는다. 배터리 용량을 자유롭게 선택할 수 있다는 장점도 있다. 하지만 배터리 충전을 위한 추가적인 모듈이 필요하고, 장기간 사용하였을 때 배터리 성능 자체에 열화가 발생해서 충전 주기가 점점 짧아질 수 있다는 단점이 있다.

나. 소프트웨어

본 프로젝트에서 구현할 PC 프로그램은 하드웨어에서 측정한 무게 데이터를 받아 사용자에게 필요한 정보를 제공한다. PC 프로그램은 크게 아래와 같은 알고리즘으로 동작한다.



가장 기본적으로 사용되는 데이터는 하드웨어를 통해 측정된 제품의 실시간 무게 데이터다. 이 데이터는 아두이노와 연결된 블루투스 또는 와이파이 장치를 통해 PC로 전송되어 저장된다. PC프로그램은 이렇게 저장된 데이터를 활용하여 동작한다. PC 프로그램의 기능은 크게 무게 데이터의 출력과 자동 주문으로 구분할 수 있다.

첫째, 무게 데이터의 출력은 전송된 데이터를 화면에 출력해 사용자에게 무게와 관련된 정보를 제공해주는 기능이다. 시간 간격을 설정해 프로그램의 메인 화면에 출력하는 기능과 저장된 데이터를 기반으로 일간, 주간, 월간 재고의 무게 변화를 확인할 수 있는 그래프 출력 기능으로 구분할 수 있다.

둘째, 자동 주문 기능은 본 프로젝트의 핵심 기능 중 하나로 재고의 무게가 일정 기준 이하가 되면 저장된 구매 링크로 접속해 자동으로 필요한 양의 재고를 주문해주는 기능이다. 본 프로젝트에서는 생수를 기준으로 10kg(20%) 이하가 되면 자동으로 주문하는 것으로 설정했다. 자동 주문을 위해 웹페이지를 제어할 수 있어야 한다. 또한, 자동 주문이 이루어진 날짜, 시간, 주문 수량 등을 별도로 저장해 사용자가 주문 내역을 확인할 수 있는 기능이 제공된다.

이러한 기능을 가진 PC 프로그램을 일반 사업장을 운영하는 사용자가 쉽게 프로그램을 이해하고 사용할 수 있도록 직관적인 GUI를 구현할 예정이다.

1) 파이썬 웹페이지 제어 모듈

	Selenium	BeautifulSoup	Requests
모듈			
특징	사용자가 실제로 웹페이지를 동작하는 것처럼 웹페이지를 제어함.	웹페이지의 주요 정보를 다루기 쉬운 형태로 변환하는 것에 적합함.	주로 http 문서를 가져오는 기능에 활용됨.
장점	코드의 구현이 쉬움. 버튼 클릭, 스크롤 조작 등 자동화 테스트가 가능함.	코드의 구현이 쉽고 간결함. 다른 모듈보다 가볍고 빠름.	http 문서를 가져오는 속도가 가장 빠름.
단점	웹페이지를 직접 실행시키기 때문에 상대적으로 속도가 느림.	스스로 웹페이지를 불러오는 기능이 없어 다른 모듈과 함께 사용됨.	동적 페이지 요소를 처리하지 못함.

온라인 쇼핑몰에서 물품을 구매하는 과정을 자동화하기 위해서는 프로그램 자체에서 해당 물품을 구매할 수 있는 웹페이지를 열고, 제어할 수 있어야 한다. 이러한 기능을 구현하기 위해 파이썬에서 활용이 가능한 웹페이지를 제어할 수 있는 모듈 3가지를 비교해보았다.

첫째, '셀레니움'은 파이썬을 통해 실제로 웹페이지를 열고 사용자가 동작하는 것처럼 웹페이지를 제어할 수 있는 모듈이다. 웹페이지를 직접 열어서 실행하기 때문에 버튼 클릭, 스크롤 조작 등을 활용해 자동화 테스트가 가능하다. 단, 웹페이지를 직접 실행하기 때문에 상대적으로 속도가 느려진다는 단점이 있다.

둘째, '뷰티풀수프'는 웹페이지의 주요 정보를 파이썬에서 다루기 쉬운 형태로 변환하는 것에 적합한 모듈이다. 다른 모듈에 비해 구현한 코드가 가볍고 빠르지만, 스스로 웹페이지를 여는 기능을 제공하지 않아 다른 모듈과 함께 사용해야 한다.

셋째, '리퀘스트'는 주로 http 문서를 가져오기 위해 사용되는 모듈이다. 다른 모듈에 비해 http 문서를 가져오는 속도가 빨라 많이 사용되지만 동적 페이지 요소를 처리하지 못한다는 단점이 있다.

이번 프로젝트에서 필요한 기능은 구매 링크가 있는 웹페이지를 열고, 웹페이지를 조작하여 필요한 물품을 주문하는 것이다. 이를 위해서는 웹페이지를 열어 정보를 읽고, 웹페이지 내의 버튼, 입력창, 스크롤 등을 조작할 수 있는 기능이 필요하다.

2) 파이썬 GUI 구성 모듈

	Tkinter	PyQT	PyGTK
모듈			
특징	파이썬에 기본적으로 내장된 GUI 모듈임.	별도로 제공되는 디자인툴인 Qt Designer를 활용해 GUI의 구현이 가능함.	별도로 제공되는 디자인툴인 Glade를 활용해 간단하게 GUI의 구현이 가능함.
장점	내장된 모듈이라 별도의 설치가 필요 없음. 코드의 구현이 쉽고 간결함.	미적 요소를 갖춘 GUI 구현 가능. 대부분의 GUI에서 요구하는 기능을 제공.	Tkinter에 비해 제공하는 기능이 다양함.
단점	타 모듈에 비해 기능이 한정적임. 미적 요소가 부족함.	별도의 설치가 필요함. 상업용으로 활용할 경우, 유료 라이센스의 구입이 필요함.	별도의 설치가 필요함. 윈도우 환경보다는 리눅스 환경에 적합함.

일반 사용자가 프로그램을 쉽게 사용하기 위해서는 단순하고 직관적인 형태의 GUI가 제공되어야 한다. GUI의 구현을 위해 파이썬에서 활용할 수 있는 GUI 구현 모듈을 비교해보았다.

첫째, Tkinter는 파이썬에 기본적으로 내장되어 있는 GUI 모듈이다. 이미 내장된 모듈이기 때문에 별도의 설치가 필요없고, 코드의 구현이 쉽고 간결하다. 코드의 구현이 쉬운 만큼, 타 모듈에 비해 기능이 한정적이고 미적인 요소가 부족하다는 단점이 있다.

둘째, PyQT는 별도로 제공되는 디자인툴인 Qt Designer를 활용하는 GUI 모듈이다. 별도의 디자인툴을 제공하는만큼 미적 요소를 갖춘 GUI의 구현이 가능하고, 대부분의 기능을 지원한다. 하지만 별도의 설치가 필요하고, 프로그램을 상업용으로 개발할 시에는 유료 라이센스의 구입이 필요하다.

셋째, PyGTK 역시 PyQT처럼 Glade라는 별도의 디자인툴을 활용해 GUI를 구현할 수 있는 모듈이다. PyQT보다는 부족하지만 Tkinter보다는 다양한 기능을 제공한다. 다만, 윈도우 개발환경보다는 리눅스 개발환경에 적합한 모듈이다.

현재 진행 중인 프로젝트에서는 버튼을 클릭했을 때, 새로운 창이 열려 저장된 정보를 보여주거나, 특정한 함수가 동작되도록 하는 기능이 필요하다. GUI에서 미적인 요소 역시 중요한 부분이지만, 현 프로젝트에서는 미적인 요소보다는 기능적인 요소에 집중하는 것이 맞다는 판단을 내렸다.

3) 프로그램 인터페이스 구성

종류	단순형 인터페이스	종합형 인터페이스
		
특징	측정되고 있는 현재 무게에 초점을 둔 인터페이스. 버튼을 누르면 새로운 창이 열리면서 사용자에게 필요한 기능을 제공함.	측정되고 있는 현재 무게를 포함해 등록된 제품 정보, 주문내역, 사용량 그래프 등을 하나의 화면에서 모두 보여줌.
장점	차지하는 화면이 작아 다른 프로그램과 함께 하나의 모니터에서 동시에 사용하는 것이 편리함. 메인으로 실행되는 기능이 무게 표시이기 때문에 상대적으로 프로그램 동작이 가벼움.	별도의 조작없이 프로그램이 제공하는 모든 정보를 하나의 창에서 확인할 수 있음.
단점	사용자가 기능 확인을 위해서는 버튼을 직접 눌러봐야 함. 필요한 정보를 제공 받기 위해서는 직접 프로그램을 조작해야 함.	필요하지 않은 기능에 대해서도 프로그램이 동작하기 때문에 동작이 느려짐. 차지하면 화면이 넓음.

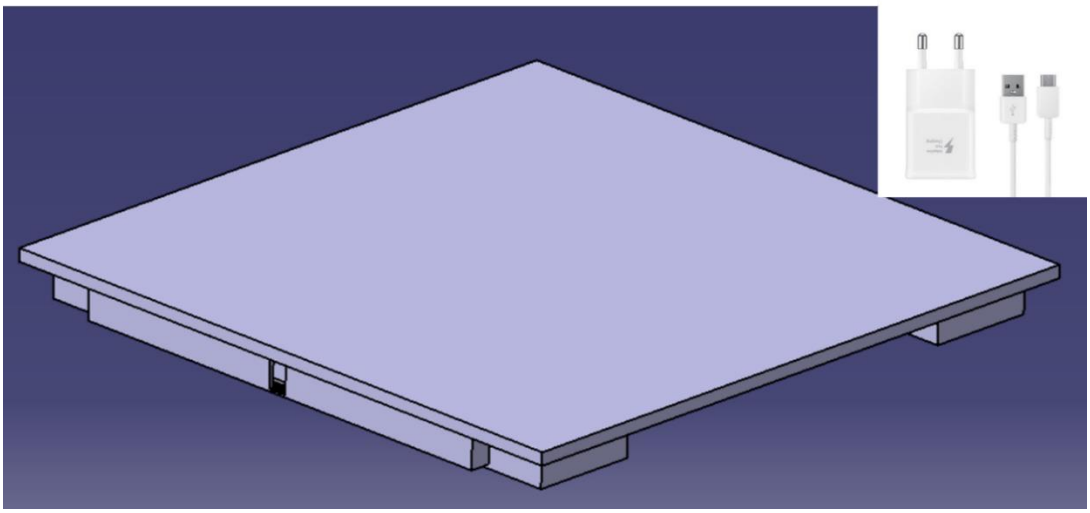
프로그램의 인터페이스는 프로그램의 코드를 구현함에 있어 어떻게 구현하게 될 것인지를 설정하는 요소이자, 사용자에게 어떤 정보에 초점을 두고 제공할 것인가를 결정하는 요소이다. 이를 결정하기 위해 두 가지 방안을 고려해보았다.

첫째, 단순형 인터페이스는 무게 측정 모듈로부터 받은 현재 무게 정보를 보여주는 것에 초점을 맞춘 인터페이스이다. 본 프로젝트의 기본적인 목표가 측정한 무게를 사용자에게 알려주고 자동 주문을 하는 것이기 때문에 무게 정보가 가장 중요한 요소라고 판단했다. 세로형 화면에 무게 정보를 출력하고, 그 외의 부가적인 기능은 버튼을 통해 사용자가 이용할 수 있도록 구현한 형태이다. 단순형 인터페이스는 필요한 화면의 크기가 작아 하나의 모니터에서 다른 프로그램과 함께 효율적으로 사용이 가능하다는 장점이 있지만, 필요한 정보를 확인하기 위해서는 별도의 버튼을 눌러야한다는 단점이 있다.

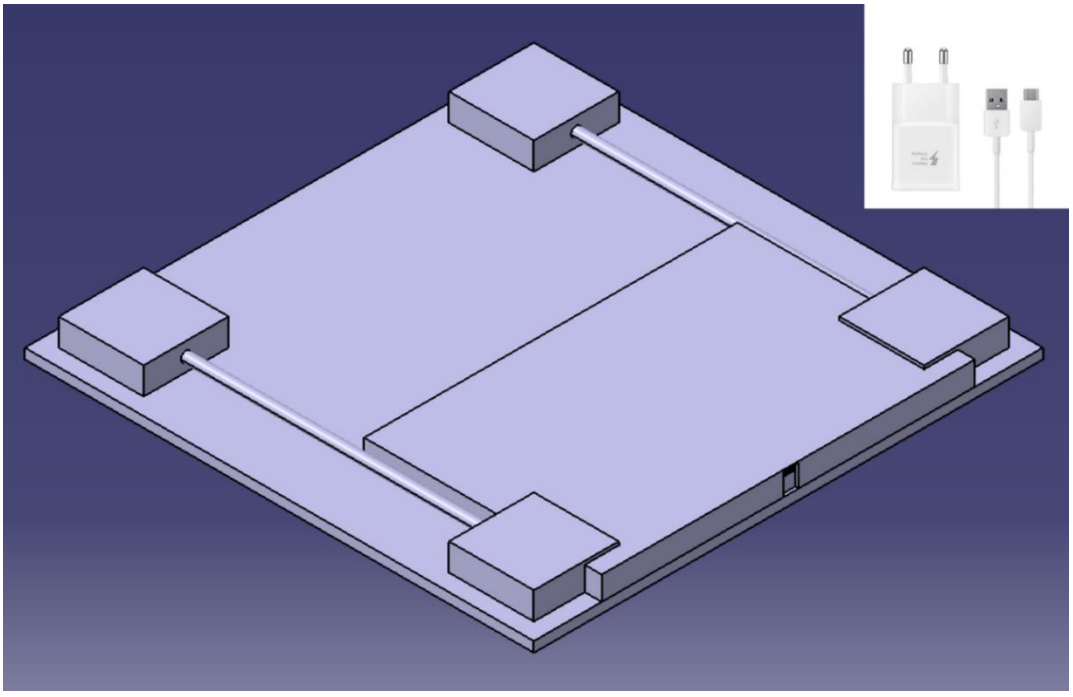
둘째, 종합형 인터페이스는 하나의 화면에 사용자에게 제공되는 모든 정보가 출력되는 것이다. 측정된 현재 무게 정보 외에도 제품 정보, 주문내역, 사용량 그래프 등을 하나의 화면에 출력하는 방식이다. 별도의 조작 없이 프로그램이 제공하는 모든 정보를 확인할 수 있다는 장점이 있지만, 차지하는 화면이 넓고 현재 필요하지 않은 기능에 대해서도 프로그램이 동작하기 때문에 프로그램 자체의 속도가 느려질 수 있다는 단점이 있다.

2 .3 조립도

가. 조립도

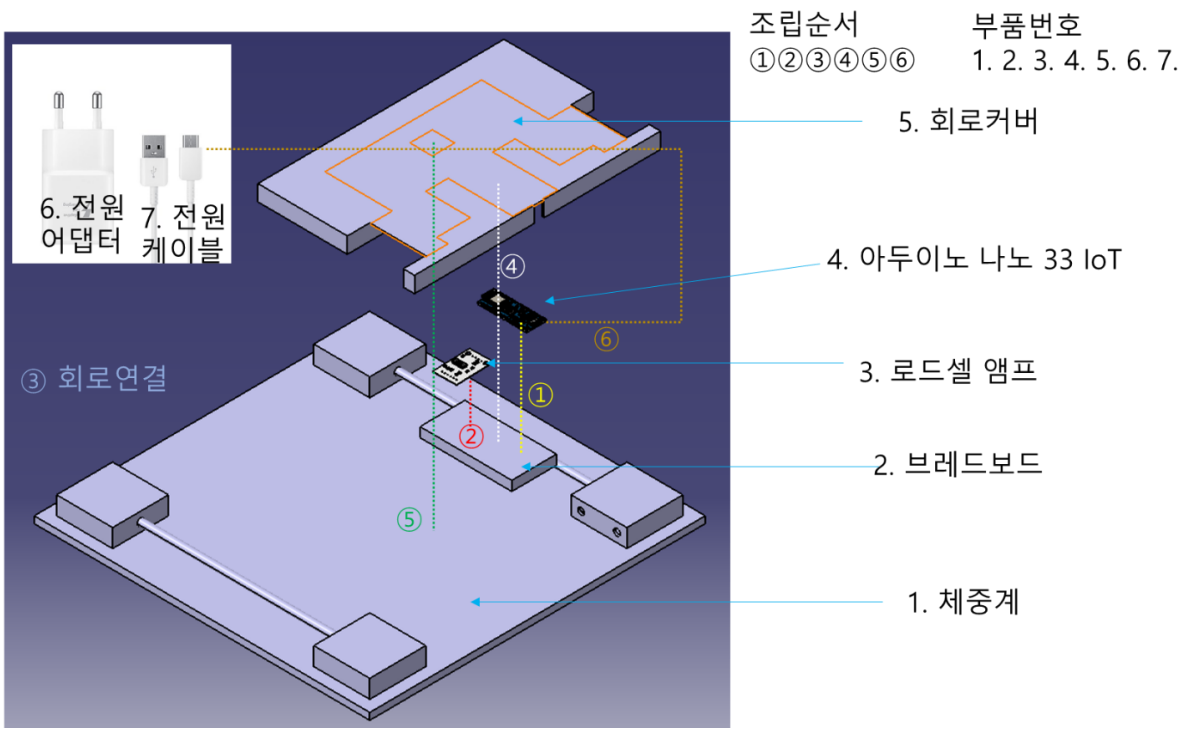


위쪽에서 본 모습



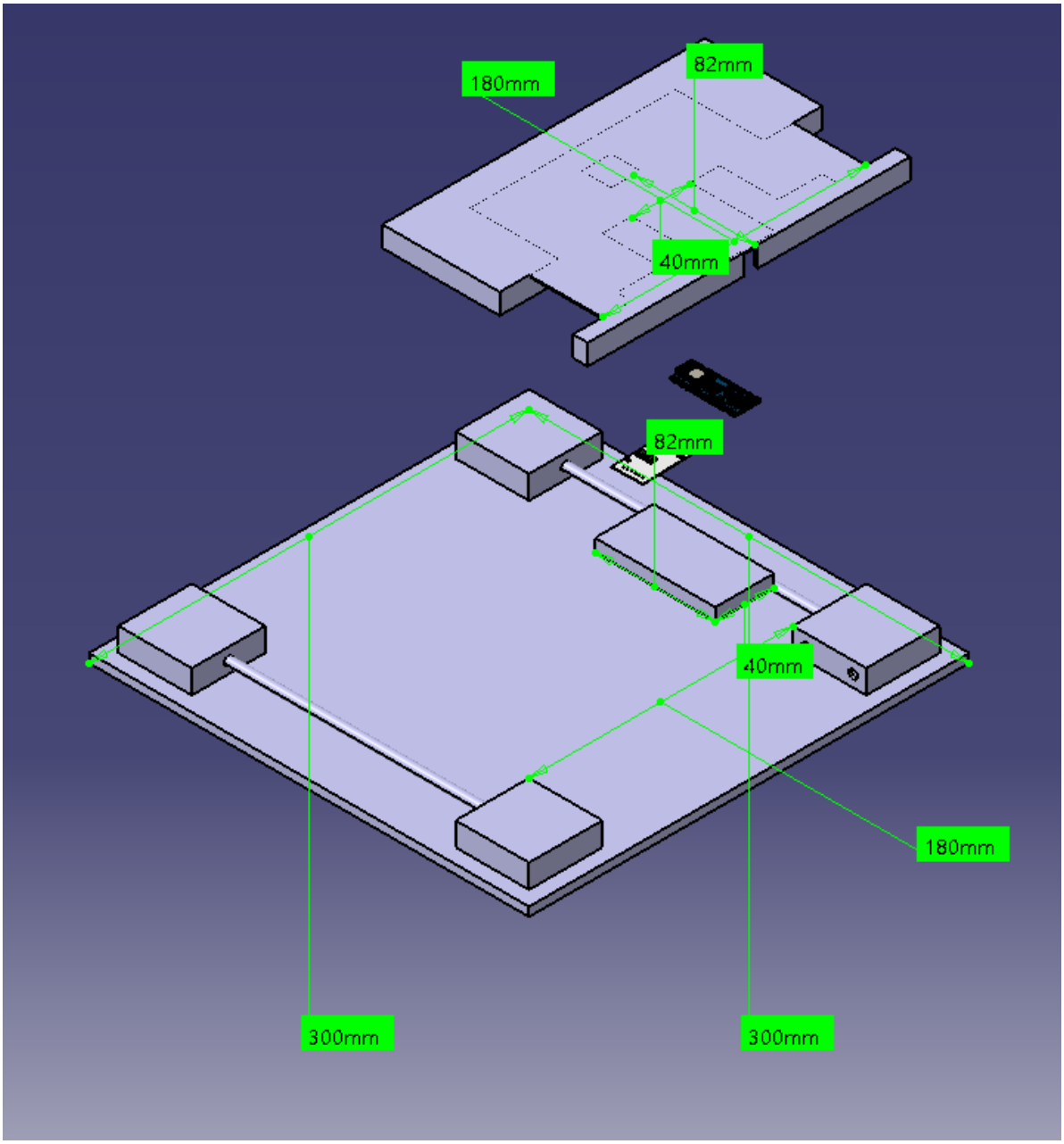
아래쪽에서 본 모습

위쪽의 평평한 판 부분에 물건을 올려놓고 무게를 측정하고, PC로 무게를 전송하게 된다. 별도의 어댑터와 케이블을 사용해 외부로부터 전원을 공급받는다.



로드셀이 들어있는 체중계와 회로를 구성하는 브레드보드, 로드셀 엠프, 아두이노 나노 33 IoT, 회로를 보호할 회로커버, 전원 공급을 담당하는 전원 어댑터와 케이블로 구성되어 있다.

우선 핀을 이용해 브레드보드에 로드셀 앰프와 아두이노를 차례로 장착하고 이를 체중계에 있는 로드셀과 연결시켜 회로를 구성한다. 그 후 브레드보드를 회로커버에 접착제를 이용하여 고정한 후, 접착제를 이용하여 회로커버를 체중계 하단에 부착한다. 그 후 외부로 드러난 아두이노의 USB 포트를 통해 어댑터를 이용하여 외부 전원을 공급해준다.



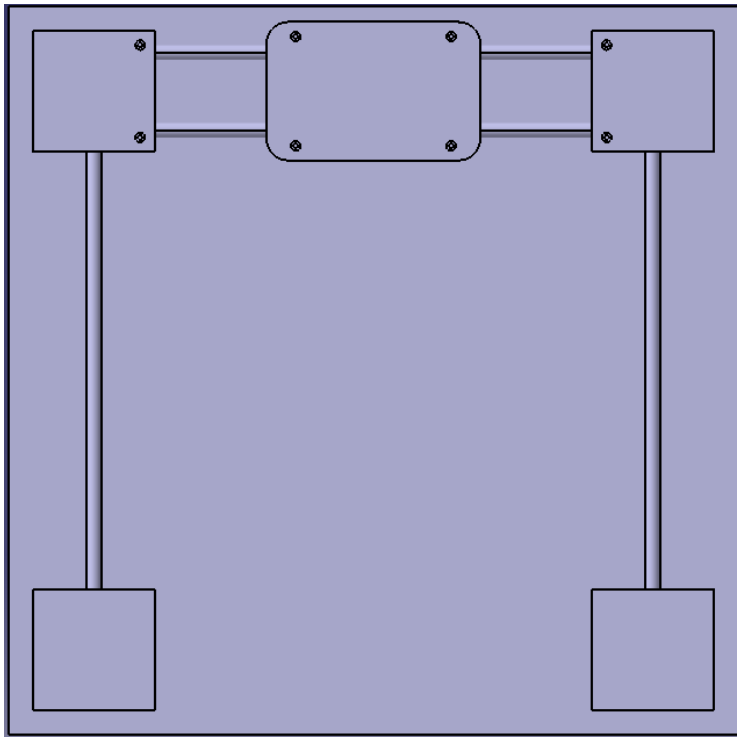
제품의 주요 치수이다. 제품의 전체 크기는 300mm * 300mm 이며, 브레드보드와 회로커버, 회로커버와 체중계의 서로 맞닿으면서 결합하는 부분의 치수를 같게 하여 단단하게 결합될 수 있도록 하였다.

나. 조립순서

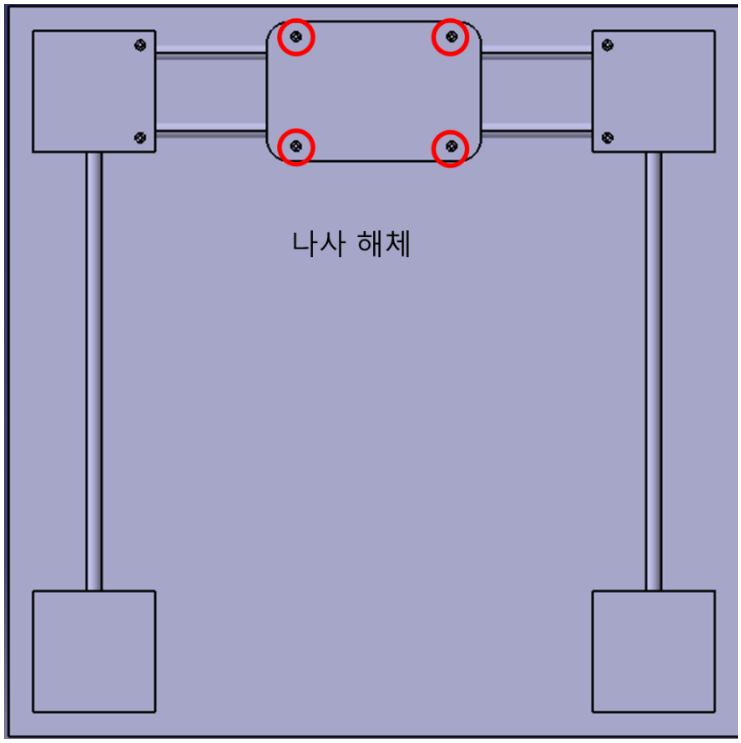
0) 체중계 분해

부품 조립에 앞서, 본 프로젝트에서는 로드셀을 이용하여 무게를 측정하는 장치가 필요하며, 체중계에는 이미 로드셀이 포함되어 있고 로드셀을 이용하기 위한 기하적 형상이 이미 구현되어 있으므로 시중의 체중계를 구매하여 사용하기로 한다. 체중계 모델 선정 기준은 개조가 용이한 형상으로 카스 HE-70 모델을 선정하였다.

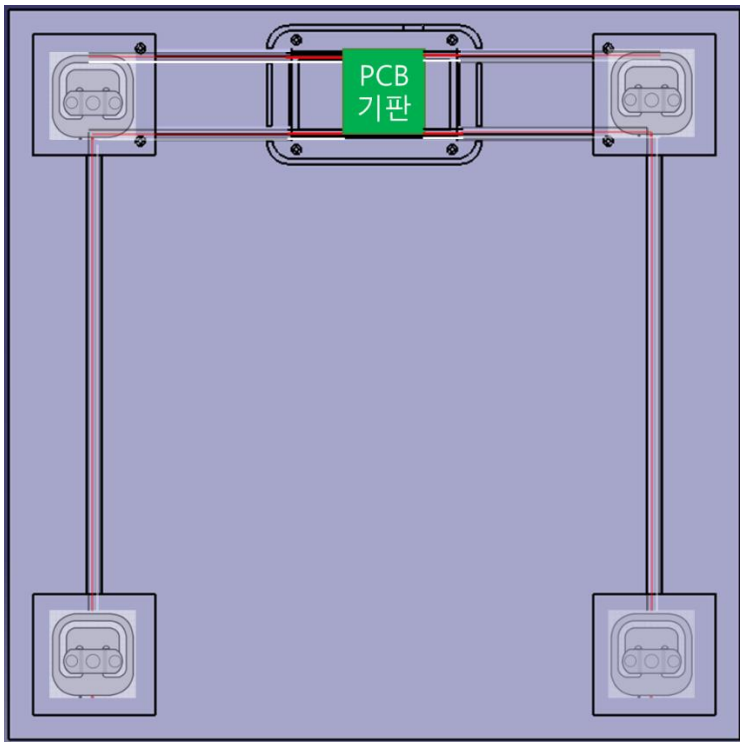
로드셀을 이용하여 측정한 무게를 PC로 전송해주는 역할을 하는 아두이노 회로가 필요하므로, 체중계의 로드셀을 아두이노에 연결하기 위한 사전 작업이 필요하다.



체중계 원본 형상 (아래쪽)

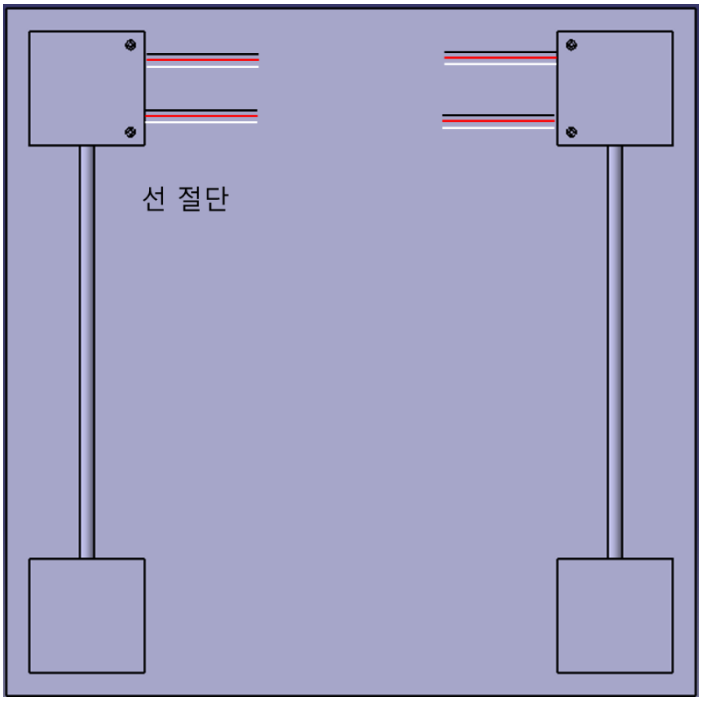


빨간색 원 부분에 있는 나사 4개를 해체한다.



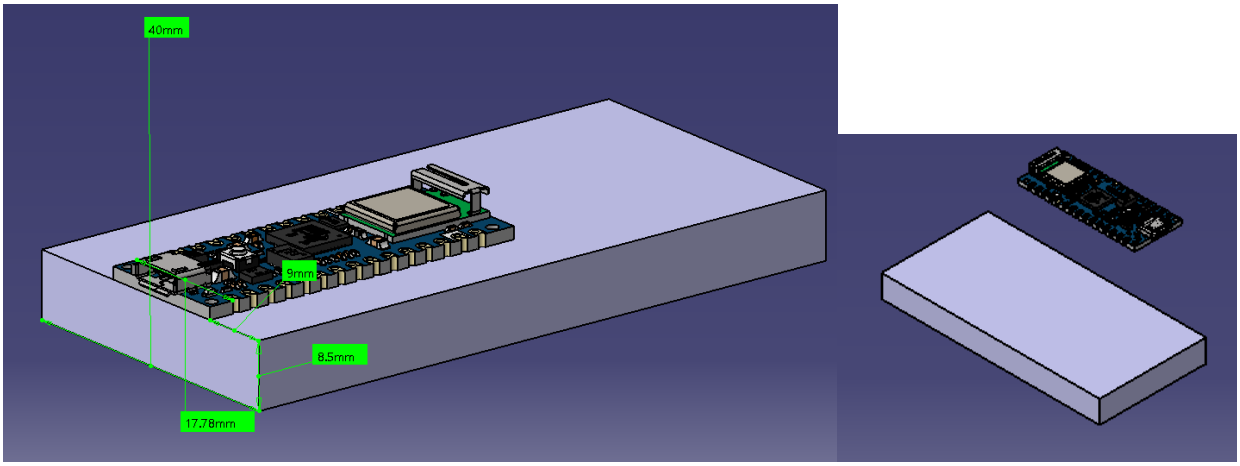
(투명도가 있는 부분은 외부에서 보이지 않는 부분)

체중계의 각 꼭짓점 부분에 로드셀이 부착되어 있고, 봉을 통해 로드셀의 선이 연결되어 있으며 중앙 상단에 있는 PCB 기반에 부착되어 있음을 확인할 수 있다.



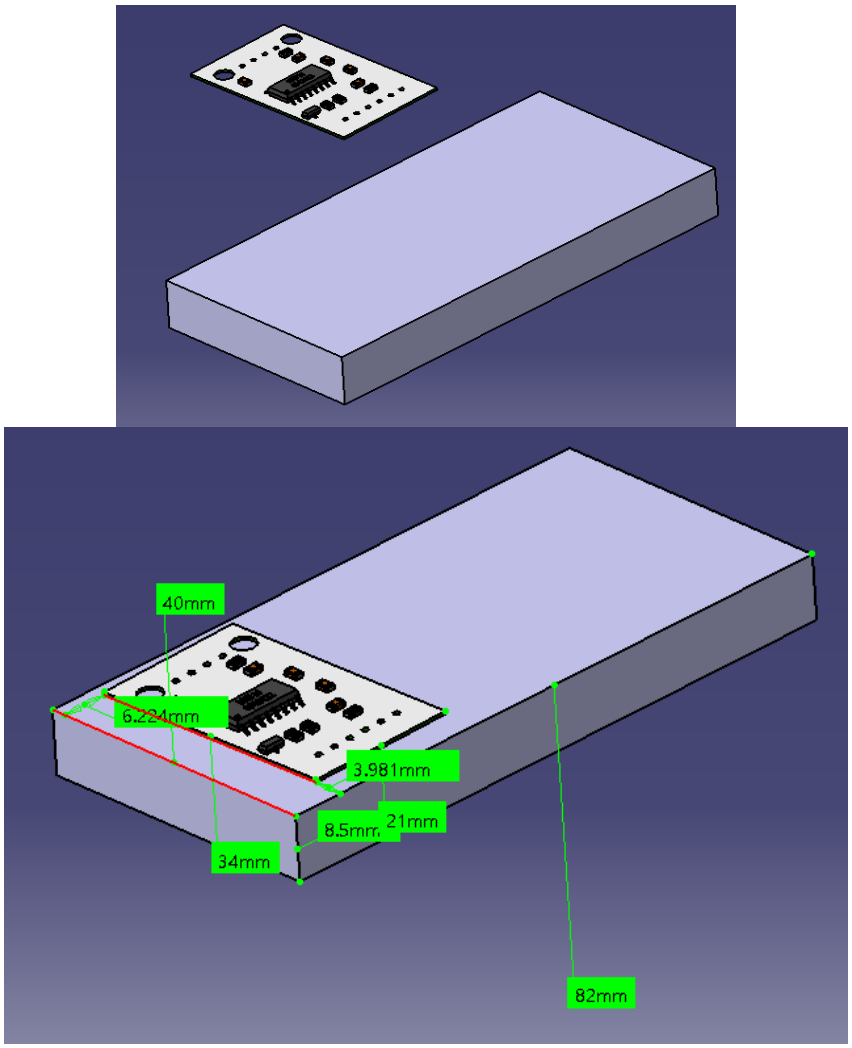
로드셀을 이용하기 위해 PCB와 로드셀 사이의 선을 절단하고, 중앙 상단에 있던 연결봉, 회로커버, PCB 등은 모두 제거한다. 남아있는 로드셀 선은 아두이노와 로드셀 앰프를 포함한 회로에 연결된다.

1) 아두이노 – 브레드보드 결합



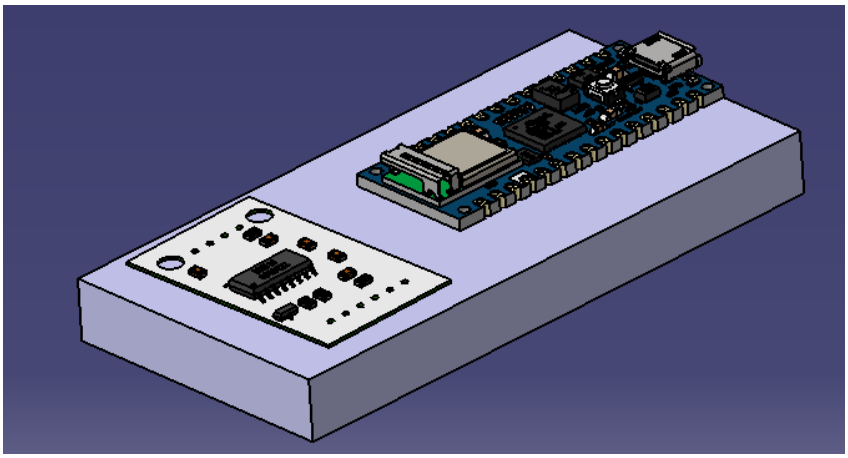
헤더가 포함된 아두이노와 브레드보드를 핀을 이용해 결합한다

2) HX711 – 브레드보드 결합



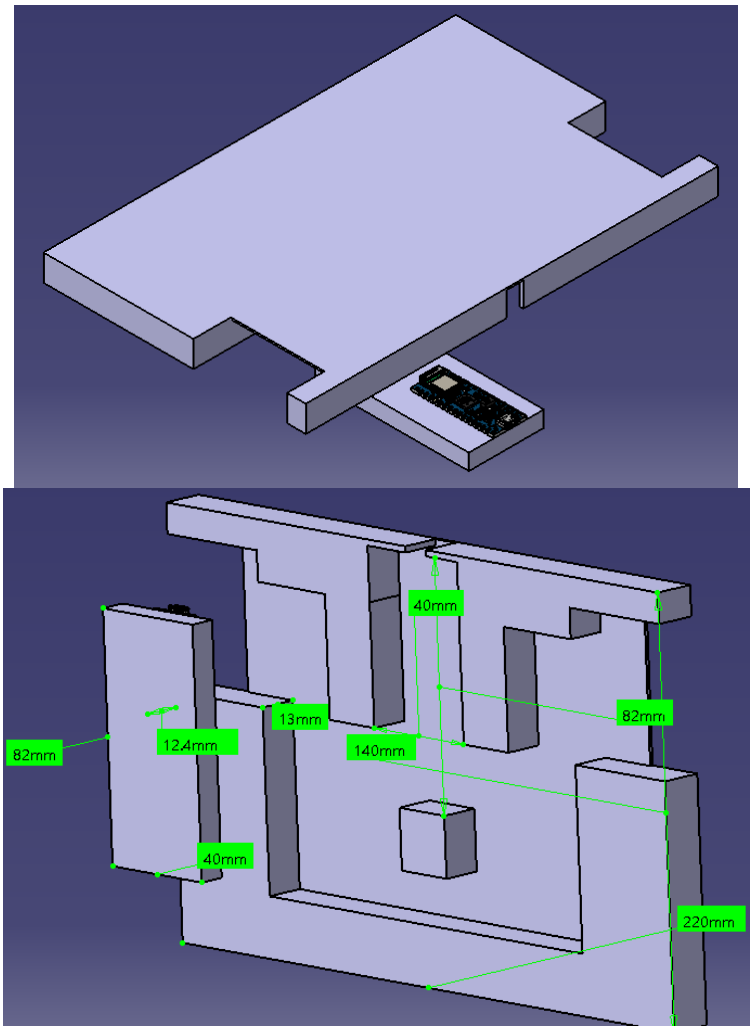
헤더가 포함된 HX711과 브레드보드를 핀을 이용해 결합한다.

3) 회로 연결

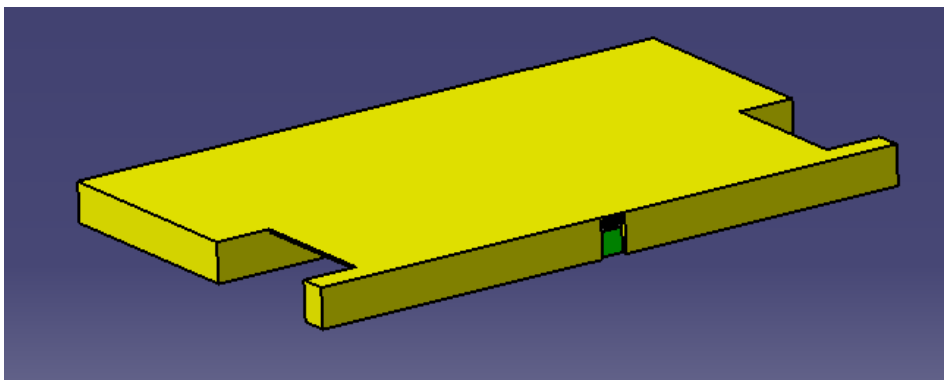


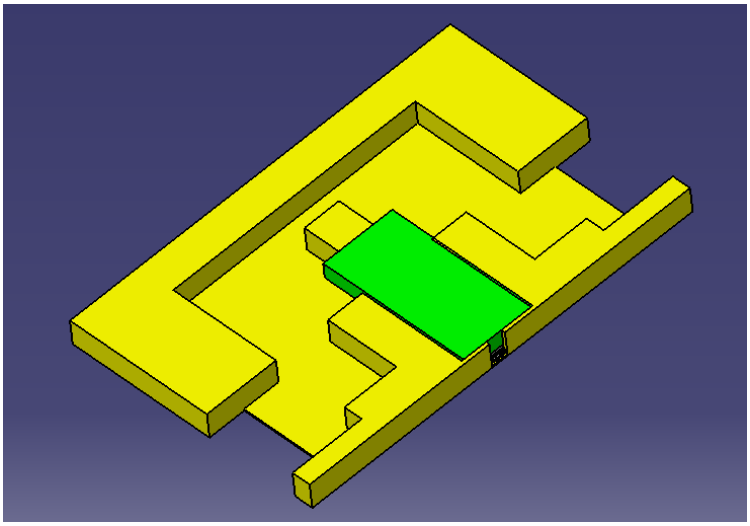
조립된 브레드보드에 5. 제어부 및 회로연결에 있는 회로도대로 체중계의 로드셀을 연결하여 회로를 구성한다.

4) 회로, 회로커버 결합



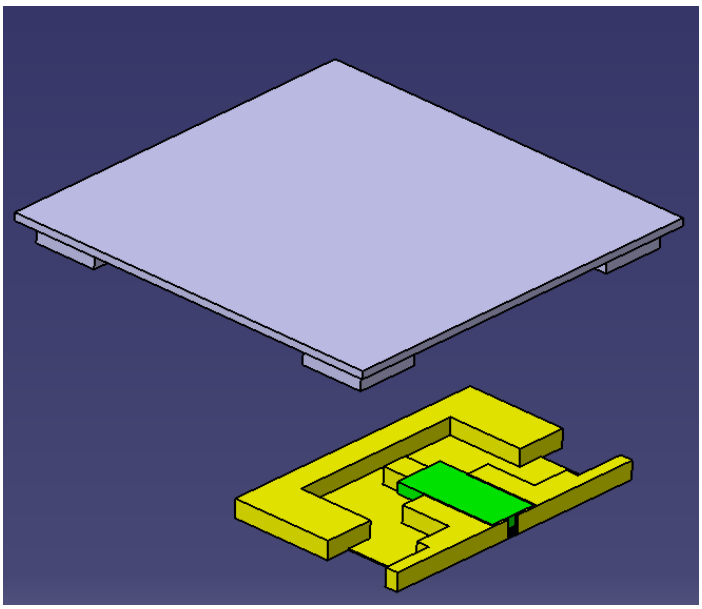
사진과 같은 방향으로 브레드보드와 회로커버를 결합한다. 브레드보드의 가로, 세로와 브레드보드가 들어갈 구멍의 가로, 세로 길이가 일치하기 때문에 가로, 세로 방향으로는 자연스럽게 고정이되며, 높이 방향 고정은 외부 힘을 크게 받지 않는다는 점을 감안하여 브레드보드의 옆면에 접착제를 발라 회로커버에 고정시킨다.

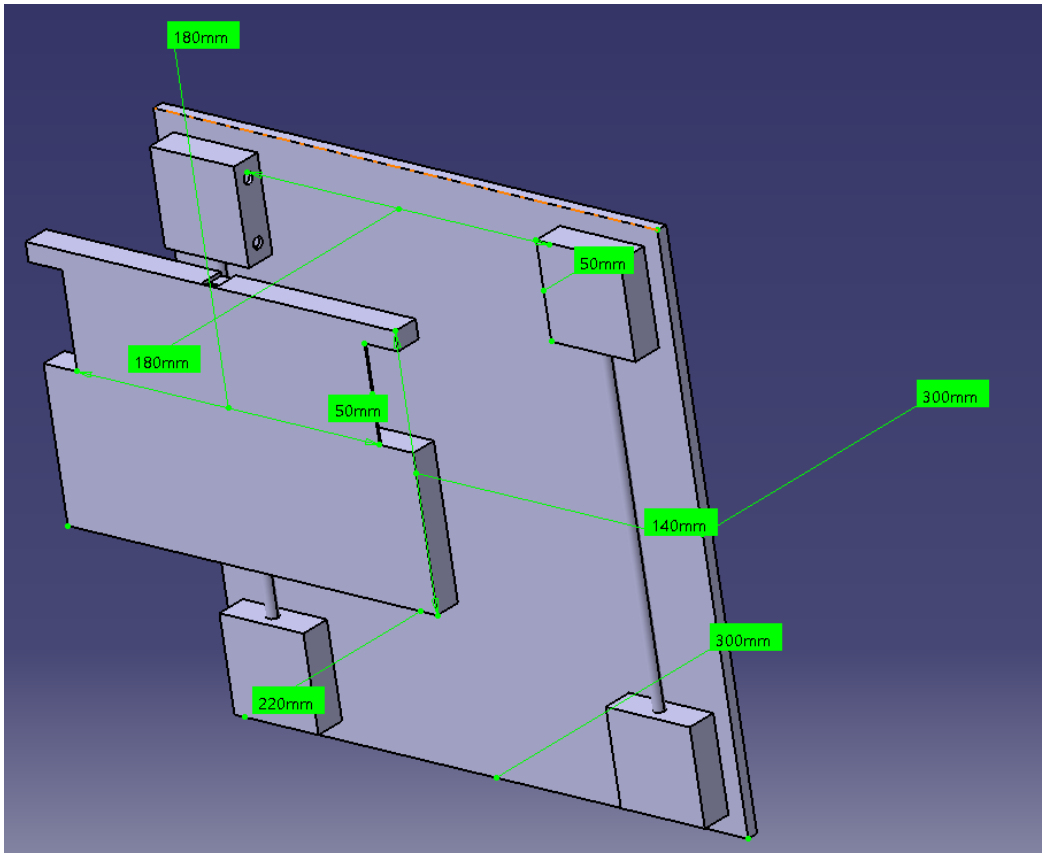




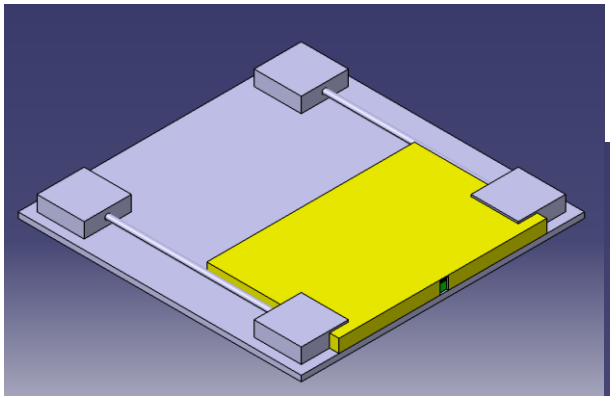
결합이 완료된 모습이다. 회로의 선이 빠져나올 공간을 확보해야 하므로, 브레드보드를 끝까지 밀어넣지 않고 브레드보드의 바닥면과 회로커버의 바닥면이 같은 높이가 되도록 한다.

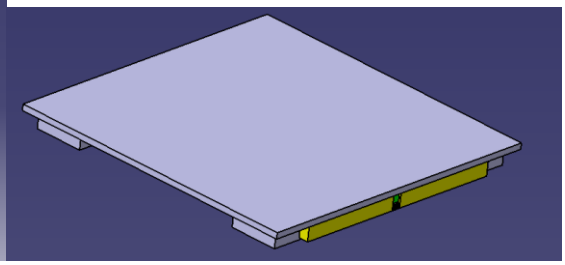
5) 회로커버, 체중계 결합





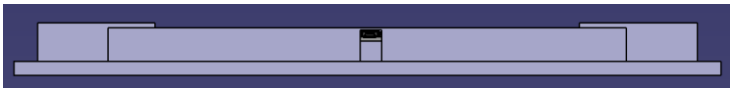
사진과 같은 방향으로 결합을 진행하며, 회로커버는 로드셀 커버에 의해 가로, 세로 방향으로 고정이 된다. 높이 방향 고정은 외부 힘을 크게 받지 않는다는 점을 감안하여 회로커버의 바다면과 브레드보드의 바닥면에 접착제를 발라 결합을 한다.





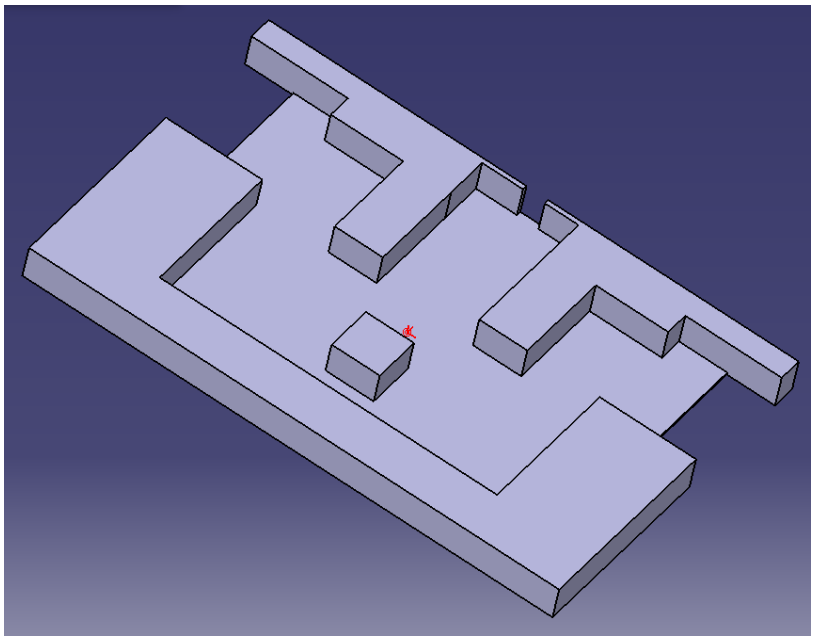
6) 전원 공급

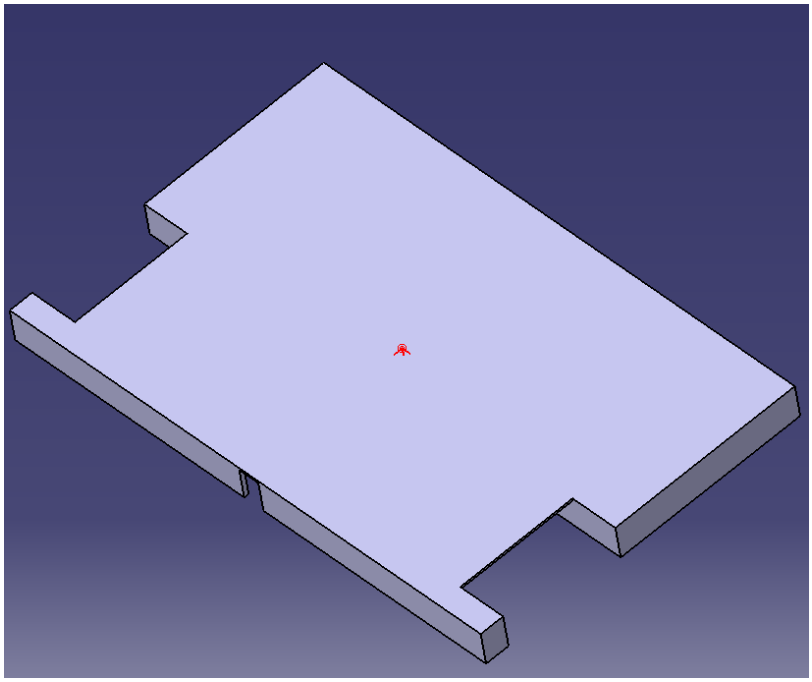
외부로 드러난 아두이노의 USB 포트를 통해 전원을 공급한다. 별도의 어댑터와 케이블을 이용한다.

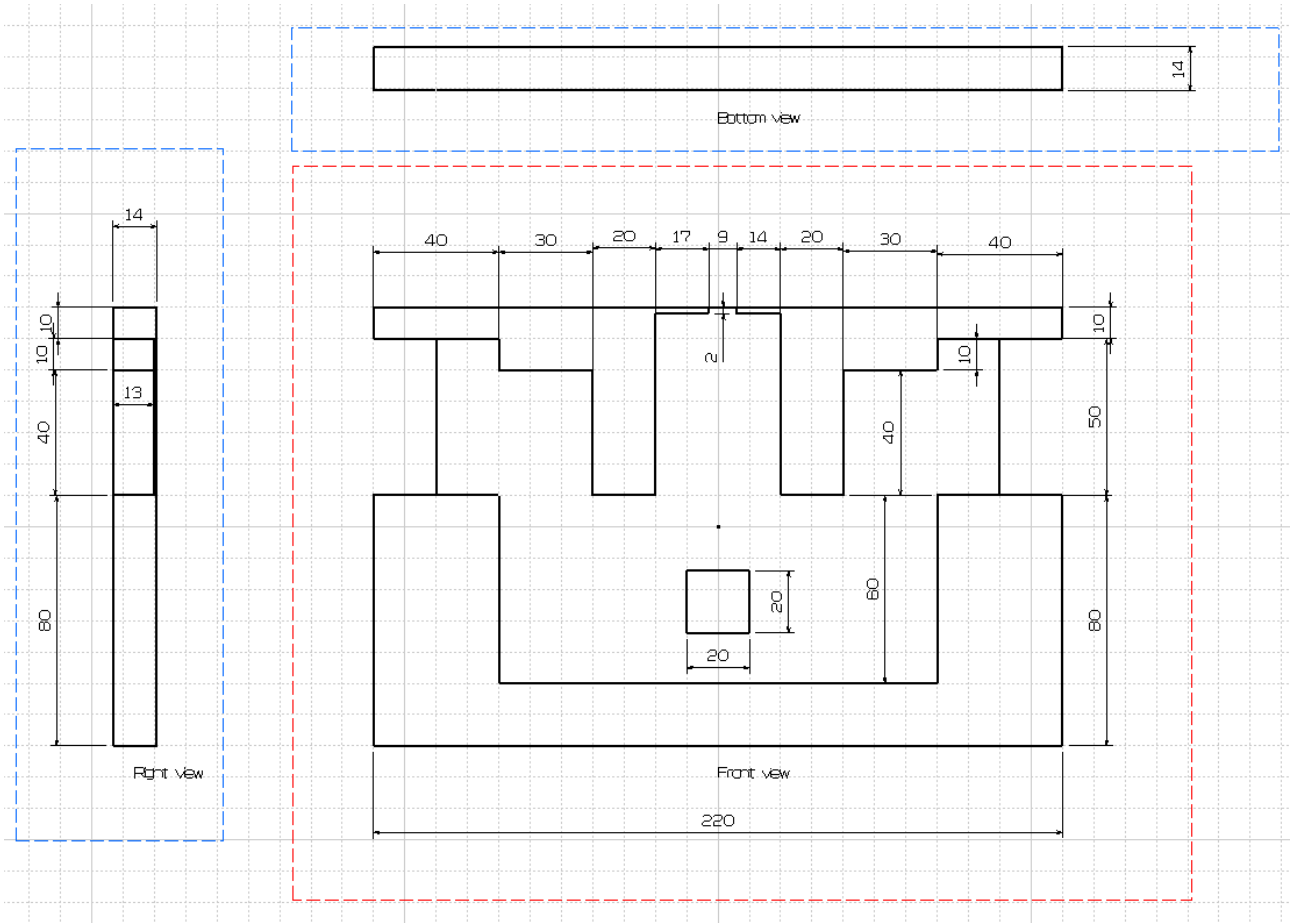


2.4 부품도

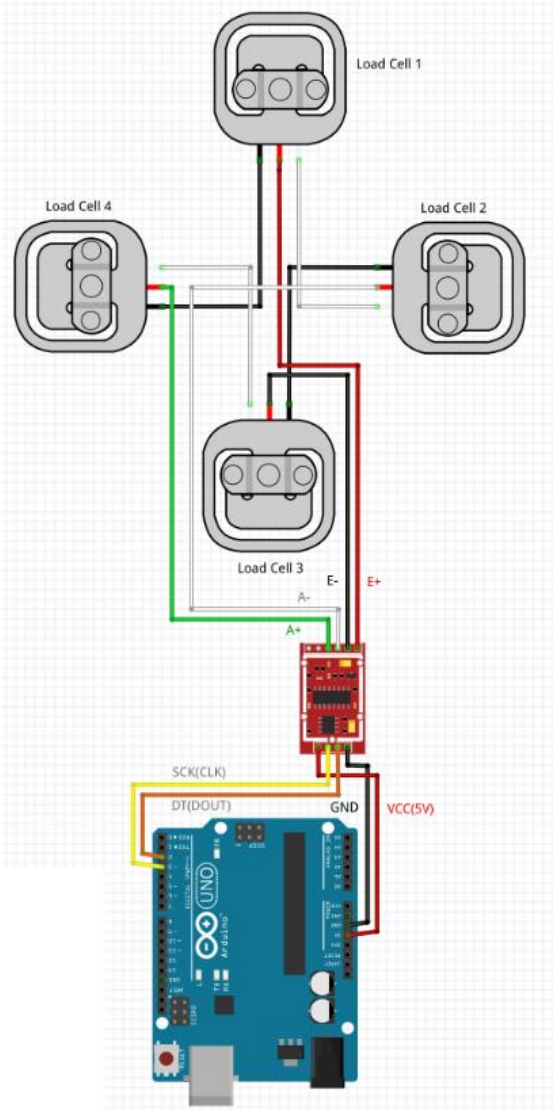
회로커버






Bottom view
14
14
13
10
10
40
80
Right view
40
30
20
17
9
14
20
30
40
2
10
10
40
50
80
80
20
20
Front view
220

2 .5 제어부 및 회로설계



아두이노 나노 33iot, 로드셀 앰프 HX711과 4개의 로드셀을 포함하는 회로다. HX711의 A+, A-, E+, E-에 로드셀이 하나씩 연결되어 있다. 로드셀 간에도 연결이 되어있으며 대칭성을 갖는다. 아두이노와 HX711은 전원에 과녈ㄴ된 핀 2개와 데이터 전송에 관련된 핀 2개로 총 4개 핀의 연결을 갖는다.

로드셀 엠프 HX711은 일반적인 아두이노 나노 33 iot의 출력 전압인 3.3V에서 하중측정의 정확도에 대한 문제가 있다. 따라서 5V 출력을 사용해야 하고 아두이노 나노 33 iot에서 5V 출력을 사용하기 위해서는 2가지 조건이 충족되어야 한다. 아두이노의 usb micro b 단자를 통해 전원이 공급되어야 하며, 사진상의 VUSB에 있는 2개의 핀을 납땜을 통해 연결해주어야 한다. 5V 전압을 사용하기 위해 추가로 전원모듈을 장착하면 추가적인 부품이 필요하고 회로가 복잡해지게 된다. 따라서 외부 어댑터와 usb 단자를 통한 전원공급과 납땜을 통해 5V 출력을 이용하기로 한다.

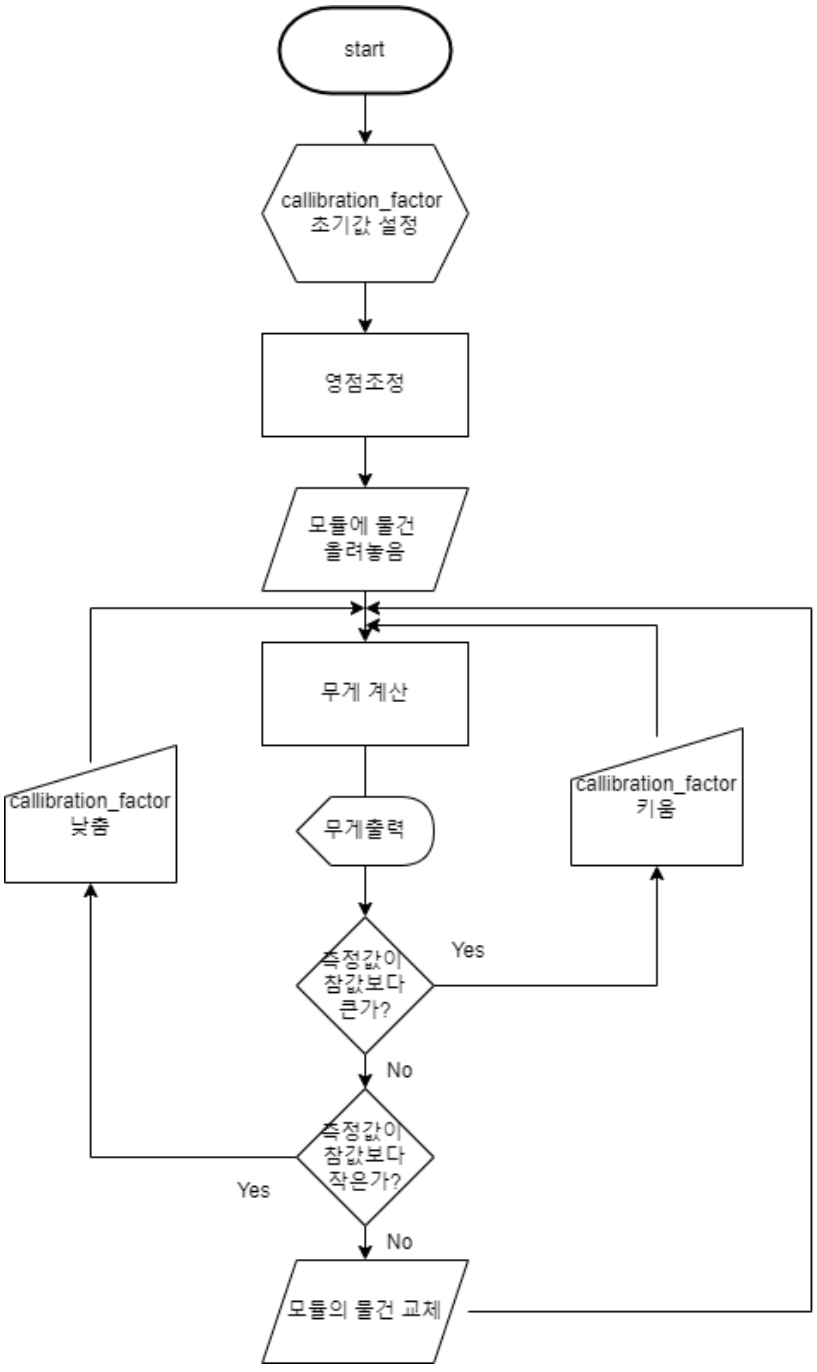
2 .6 소프트웨어 설계

전체적인 소프트웨어 구조를 나타내는 그림이다. 아두이노에서 무게를 측정하는 코드가 실행되고, 아두이노에서 측정한 무게를 PC로 전송하기 위해 BLE 통신을 이용하며, BLE 통신에 관련된 코드가 아두이노와 PC 양쪽에 있다. 전송받은 무게 데이터를 바탕으로 PC에서 무게 데이터 출력, 자동주문 등의 기능을 제공하는 코드를 실행하게 된다.

가. 무게측정

1) 캘리브레이션

로드셀을 통해 측정된 전류와 그에 해당하는 무게 사이의 관계를 조정해주는 것을 캘리브레이션이라고 한다. 무게를 정확하게 측정하기 위해서는 캘리브레이션 작업이 꼭 필요하다. 물, 체중 등 참값을 알고 있는 물건들을 활용해 무게측정모듈로 측정했을 때 참값에 가까운 무게가 나오도록 캘리브레이션을 진행한다. 아두이노에서 제공하는 HX711.h 라이브러리의 내장 함수와 예제를 이용한다.



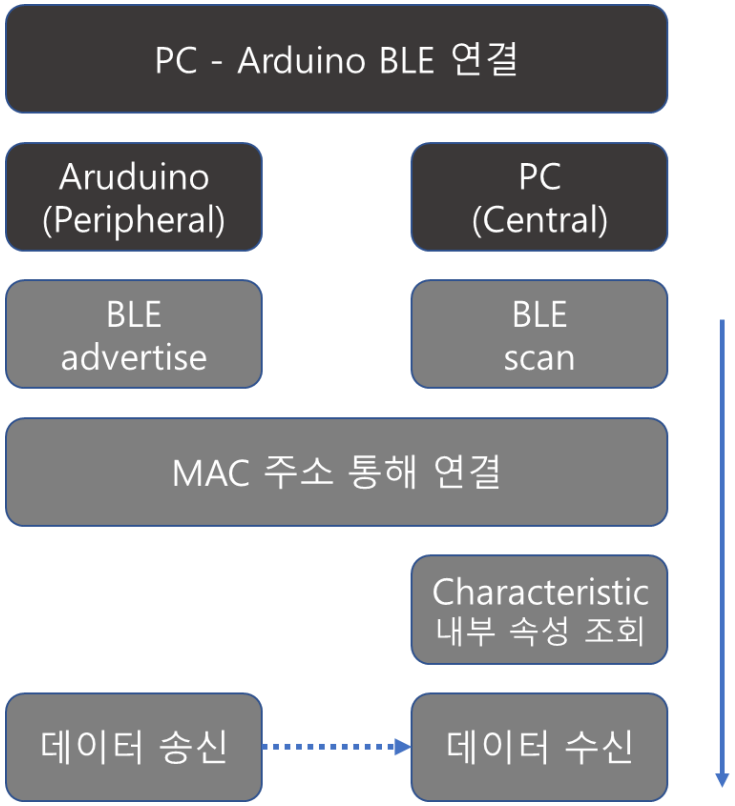
적절한 calibration_factor 값을 찾는 것을 목표로 캘리브레이션을 진행한다.

초기 calibration_factor 값을 설정하면, 라이브러리 내장함수(set_scale())를 통해 영점이 맞춰지게 된다. 그 후로 무게의 참값을 아는 물건을 올려놓고, 측정값이 참값보다 크다면 calibration_factor를 키우고 측정값이 참값보다 작다면 calibration_factor를 낮추면서 참값과 일치하는 무게를 얻도록 한다. 참값과 일치하는 무게를 얻었다면 다른 무게를 가진 물건을 올려놓고 위 과정을 반복한다. 0~150kg 사이의 물건에 대해 시행한다.

캘리브레이션 표

		Calibration factor			
		-200000	**-20000**	**-20500**	**-20300**
참값	**3.0**	**0.3**	**3.0**	**2.9**	**2.9**
	12.1		**12.2**	**11.9**	**12.0**
	24.2		**24.3**	**24.0**	**24.1**
	62.3		**66.6**	**60.3**	**61.2**
	133.3				**125.2**

나. PC-아두이노 BLE 연결



PC와 아두이노를 저전력 블루투스인 BLE를 통해 연결한다. 아두이노 나노 33 iot에 내장된 BLE 4.2를 활용하며, PC에 내장된 블루투스 모듈이 없다면 usb형 블루투스 동글을 활용할 수 있다.

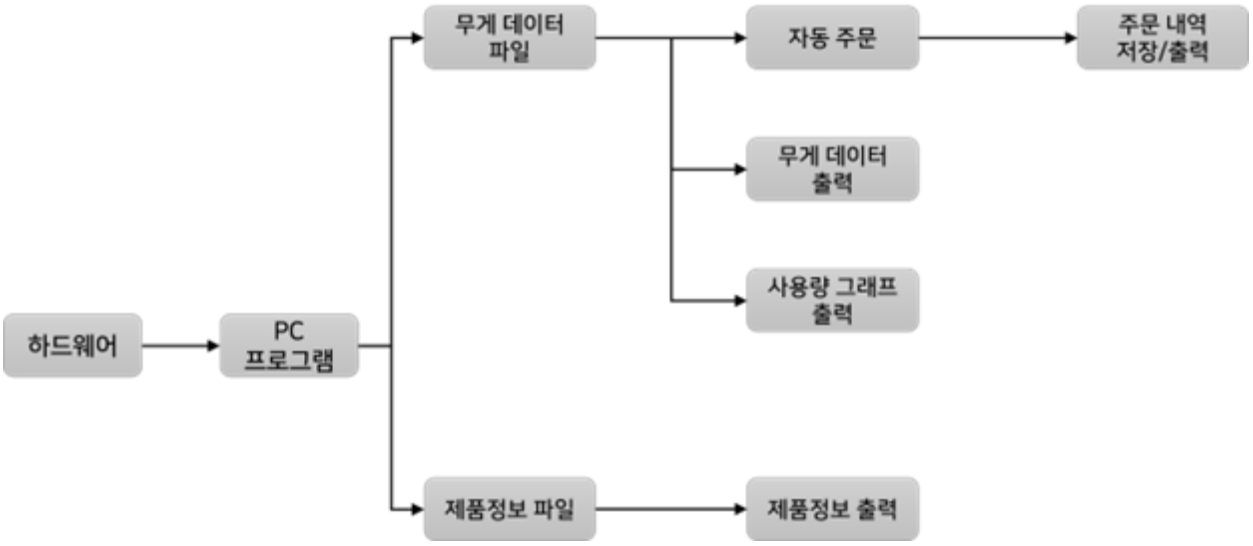
PC는 Central로 동작하는데, 연결 advertising 신호를 주기적으로 스캔하다가 아두이노에 연결을 요청하게 된다. 아두이노는 Peripheral로 작동하며 다른 기기에서 BLE 신호를 스캔할 수 있도록 주기적으로 연결 advertising 신호를 보내고(advertise) 연결 요청이 들어오면 수락하게 된다. 연결에는 디바이스의 고유 주소인 MAC address 값이 키값으로 사용된다.

연결이 되면 아두이노는 데이터를 주기적으로 송신하고, PC는 서비스 내의 고유번호인 Characteristic를 확인하여 원하는 정보를 수신할 수 있다.

데이터 구조 및 형식

Service	Characteristic	Data type	Description
WeightModule	Weight	float	측정한 무게값

다. PC프로그램 전체 동작 시나리오



▶ 무게 데이터 저장

아두이노를 통해 하드웨어로부터 측정된 무게 데이터를 텍스트 파일 또는 CSV 파일의 형태로 저장한다. 이렇게 저장된 데이터는 무게 데이터와 함께 측정한 날짜, 시각 정보가 함께 기록되어 무게 데이터 출력과 사용량 그래프 출력에 활용된다.

▶ 자동주문

측정된 무게 데이터를 바탕으로 설정된 무게 이하가 되면 저장된 구매 링크를 통해 자동으로 제품을 주문한다. 본 프로젝트에서는 생수를 기준으로 10kg(20%)이하가 되면 자동으로 주문하는 것을 목표로 한다. 자동주문이 실행되면 실행된 날짜, 시각, 주문한 제품 수량을 별도의 파일을 통해 주문내역을 저장한다.

▶ 무게 데이터 출력

특정 시간 간격으로 현재 재고의 무게 데이터를 출력한다. 출력은 기존에 저장된 무게 데이터 파일을 활용하여 데이터를 불러와 출력하는 방식으로 구현한다. 현재 무게의 아래에는 등록된 제품명과 기준 무게를 이용한 현재 재고 수량을 함께 표시한다.

▶ 제품 정보 등록 및 변경

사용자가 제품의 정보를 입력하면 입력 내용을 텍스트 파일로 저장한다. 사용자가 입력할 수 있는 정보는 제품명, 기준 무게(1개), 구매 링크이다. 텍스트 파일에는 이 내용이 순서대로 리스트의 형식으로 저장된다. 제품 정보를 변경할 때는 입력창에 수정 내용을 입력하면, 기존의 텍스트 파일 위에 새로운 내용을 덮어 씌우는 방식으로 등록 정보를 수정한다.

▶ 제품 정보 확인

제품 정보의 확인은 제품 정보를 등록하면서 생성된 텍스트 파일의 내용을 불러와 출력하는 방식으로 동작한다. 저장된 정보인 제품명, 기준 무게, 구매 링크를 출력하며, 구매 링크의 경우 버튼을 클릭하면 해당 웹페이지로 연결될 수 있도록 구현한다.

▶ 주문 내역 확인

자동주문이 실행될 때마다 저장된 주문내역 파일을 불러와 출력한다. 순번, 날짜, 시각, 주문 수량이 순서대로 출력된다.

▶ 초기화

기존에 저장된 무게 데이터 파일, 제품 정보 파일, 주문내역 파일 위에 새로운 파일을 덮어씌워 저장된 모든 정보를 삭제한다. 삭제 전 사용자에게 확인 메시지를 출력한 후, 실행되도록 구현한다.

▶ 일별, 주별, 월별 그래프

저장된 무게 측정 데이터를 바탕으로 사용량 그래프를 제공한다. 각 버튼을 클릭할 때 해당하는 그래프가 별도의 화면을 통해 출력되도록 한다.

나. 세부 설계

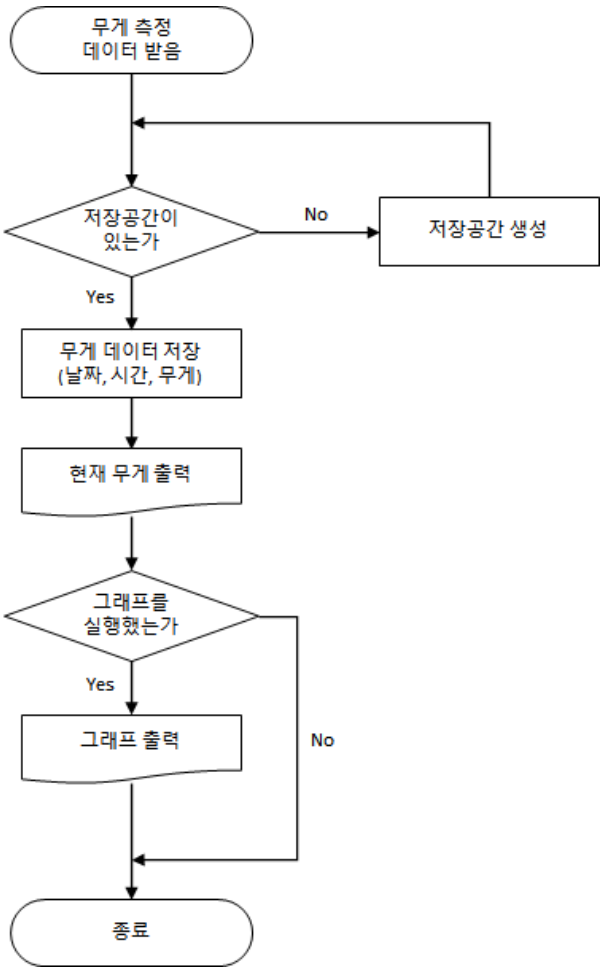
1) 프로그램 인터페이스 구성

프로그램의 인터페이스는 사용자에게 현재 무게 정보를 보여주는 것에 초점을 맞춘다. 이를 위해 단순형 인터페이스를 활용하여 하드웨어로부터 입력받은 현재 무게를 메인 화면에 출력하고, 그 외의 부가적인 기능은 별도의 버튼을 통해 동작하도록 설계한다. 이를 통해, 사용자가 한 눈에 현재 무게를 확인할 수 있도록 한다. 또한, 작은 화면만 차지하기 때문에 하나의 모니터에서 별도의 프로그램과 함께 사용이 편리하다.

2) 프로그램 GUI 구성을 위한 파이썬 모듈

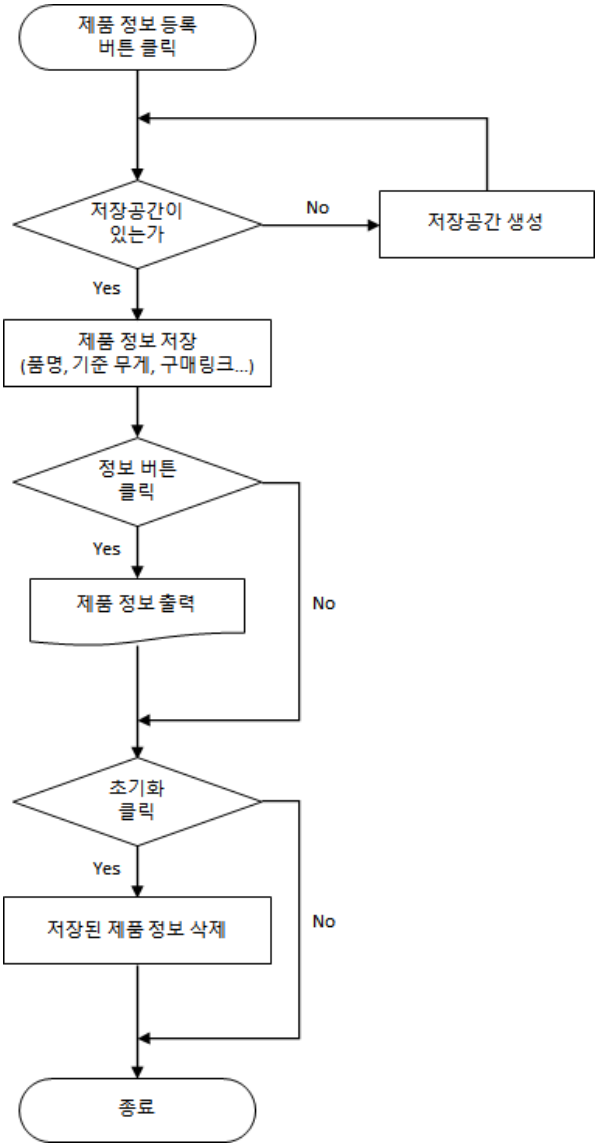
사용자가 프로그램을 쉽게 사용하기 위해서는 단순하고 직관적 형태의 GUI가 제공되어야 한다. 또한, 본 프로젝트를 통해 제공하고자 하는 기능을 모두 구현할 수 있어야 한다. 무료로 사용이 가능한 다양한 GUI 모듈 중 가장 단순하면서도 필요한 기능이 전부 제공되는 'Tkinter'를 활용하여 프로그램의 GUI를 구성한다. 타 모듈에 비해 디자인적인 요소는 부족하지만, 코드의 구현이 간단하고 직관적인 출력이 가능해 'Tkinter'를 활용하여 설계한다.

3) 무게 데이터 저장 및 출력



무게 데이터 저장 및 출력은 하드웨어에서 측정된 데이터를 전달받으며 시작된다. 데이터가 전달되면 저장을 위한 폴더와 파일의 존재를 확인한다. 저장을 위한 폴더와 파일이 없다면 저장공간을 생성한 후 데이터를 저장한다. 이 때, 저장되는 데이터는 입력된 날짜와 시간, 측정된 무게이다. 저장된 파일의 무게 데이터를 일정 시간 간격으로 읽어 프로그램의 메인 화면에 현재 무게를 출력한다. 이후, 사용자가 그래프 기능을 실행하면 저장된 파일을 바탕으로 그래프를 출력한다.

4) 제품 정보 등록 및 수정

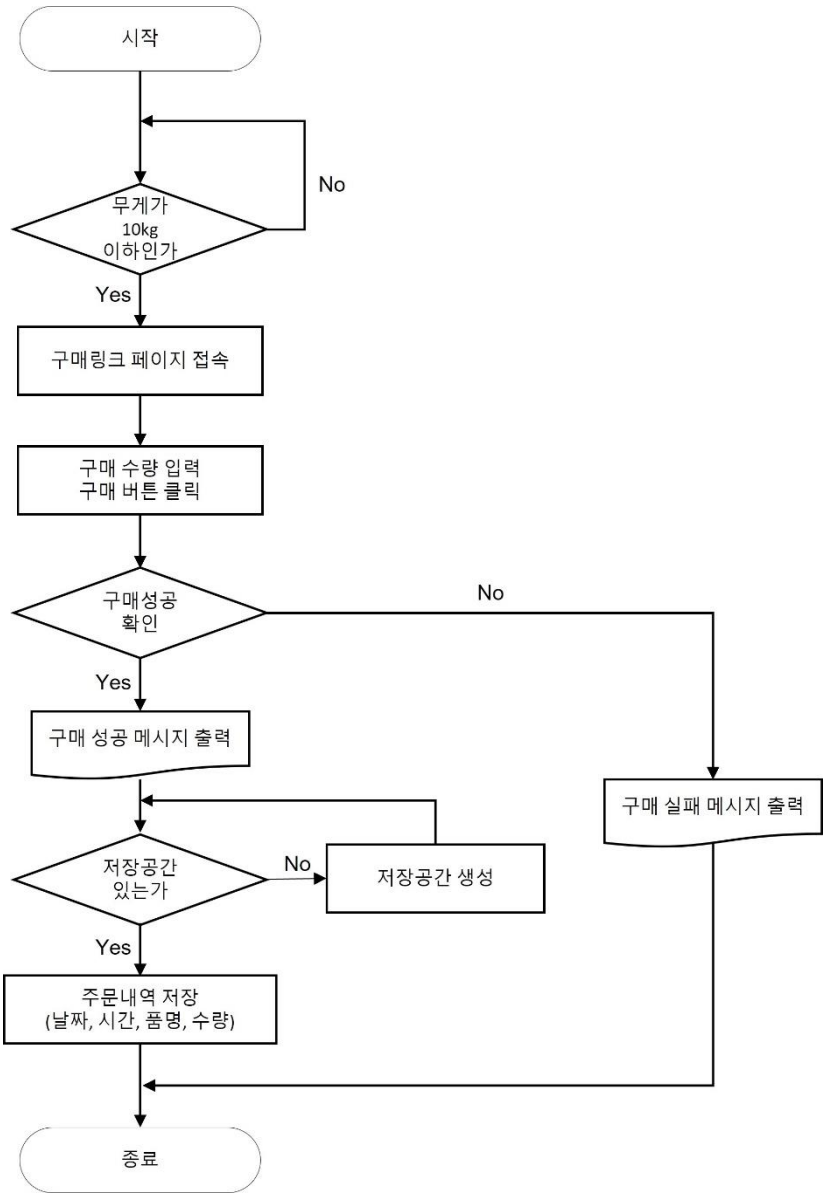


사용자가 무게를 측정할 제품에 대한 정보를 입력할 수 있도록 한다. '제품 정보 등록' 버튼을 클릭하면 정보 저장을 위한 공간을 확인한 후, 없다면 저장 공간을 새롭게 생성해 제품 정보를 저장한다. 이 때 저장되는 정보는 제품명, 기준이 되는 무게, 제품 구매링크, 1회 구매 시 수량 등이 포함된다. 저장 후 메인 화면에서 사용자가 '정보 확인' 버튼을 클릭하면 제품 정보가 저장된 파일을 읽어 저장된 내용을 화면에 출력한다. 이후, 제품 정보의 초기화가 필요할 시 초기화 버튼을 누르면 기존에 정보가 저장된 파일의 내용을 모두 삭제하여 초기화한다.

5) 자동주문을 위한 웹페이지 제어 모듈

자동주문을 위해서는 사용자가 실제로 웹페이지를 동작하는 것처럼 제어할 수 있어야 한다. 이를 위해 웹페이지 제어 모듈인 'Selenium'을 활용하여 웹페이지를 제어한다. 'Selenium'은 타 모듈에 비해 상대적으로 속도가 느리지만 버튼 클릭, 스크롤 조작 등의 기능 구현이 가능해 웹페이지 제어의 자동화를 구현할 수 있다.

6) 자동주문 및 주문내역 저장



저장된 무게 데이터를 기반으로 특정 무게 이하가 되면 저장된 제품 정보에 따라 웹페이지를 제어하여 자동으로 제품을 주문한다. 측정된 무게가 10kg(전체의 20%) 이하가 되면 제품 정보에 저장된 구매링크 페이지로 접속한다. 접속 후, 웹페이지를 제어하여 구매 수량을 입력하고, 구매 버튼을 클릭한다. 구매가 정상적으로 진행되었다면 구매 성공 메시지를 출력하고, 실패했다면 구매 실패 메시지를 출력한다. 구매가 성공한 후에는 주문내역 저장을 위한 공간이 있는지 확인하여 주문내역을 저장한다. 이 때 저장되는 정보는 주문 날짜, 시간, 품명, 주문 수량이다.

2.8 자재소요서

부품번호	부품명	규격	재질	수량	구매, 외주, 제작	비고
1	카스 HE70 (체중계)	300*300*20 (mm)		1	구매	
	무게측정판	300*300*6 (mm)	유리	1	구매	체중계에 포함
	로드셀			4	구매	체중계에 포함
	로드셀 커버		플라스틱	4	구매	체중계에 포함
	연결봉	6*60 (mm) (직경*길이)	금속	2	구매	체중계에 포함
2	브레드보드	40*88*8 (mm)			구매	
3	로드셀 엠프 (HX711)	16*24*1.5 (mm)		1	구매	
4	아두이노 나노 33 IOT	45*18*1.5 (mm)		1	구매	
5	회로커버	220*180*14 (mm)	PLA	1	제작	3D프린터
6	전원어댑터	5V 1A		1	구매	
7	USB 케이블	2m		1	구매	
8	접착제			1	구매	
9	점퍼케이블		구리		구매	
10	테이프				구매	

3. 결과 및 평가

3 .1 완료작품 소개

가. 프로토타입 사진

나. 포스터

서울시립대학교 UNIVERSITY OF SEOUL

로드셀을 활용한 무게 감지에 따른 자동 주문 시스템

(Automatic ordering system by weight-sensing using load cell)

[세 공돌이] 1조: 강재언, 김 진, 이휘호

지도교수: 신동헌

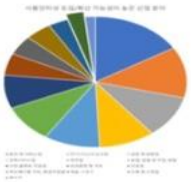
개발 과제 요약

본 프로젝트는 중소 규모의 사업장에서 사용자가 지정한 품목들의 효율적인 관리를 위하여 품목들의 사용량 정보를 데이터화 하여 사용자에게 보여주고 제품의 무게를 감지하는 시스템을 통하여 재고 소진율을 파악하고 주문이 필요한 품목들을 자동으로 주문할 수 있도록 구현하기 위해 아두이노-로드셀 모듈을 설계하고 이를 연동할 수 있는 PC 프로그램을 함께 개발하고자 한다.

개발 배경

사물 인터넷 시장은 전세계적으로 성장하는 추세

사물 인터넷이 도입 될 산업 분야는 보건/서비스업 16.3%, 전기/가스/수도산업 12.6%, 금융/보험 10.4%, 소매업 3.1%



개발 목표 및 내용

모듈에서 전달 받은 무게 데이터를 PC에 전송해주어 잔량이 20%이하이면 자동 주문이 가능하게 설계하여 사용자가 효율적으로 재고를 관리할 수 있도록 제작한다.

① 무게 측정 및 데이터 전송

② 회로물의 케이스 제작

③ PC 프로그램 개발

기대 효과

기술적 기대효과.

산업내의 자동화를 촉진시켜 전반적인 업무 효율을 향상

산업시설 이 외에도 개인의 주거 공간, 무인 마켓 등 기술적 호환 가능성 ↑

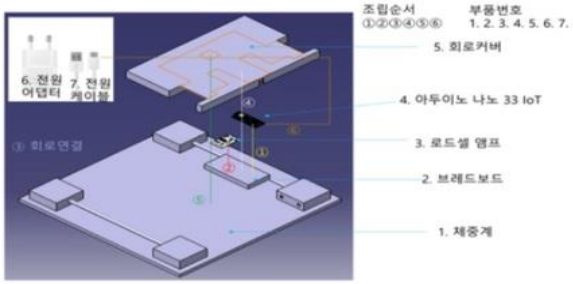
사회적 파급효과.

불용재고와 같은 과잉공급 현상을 줄이고 불특정한 수요의 가격 변동폭을 안정화

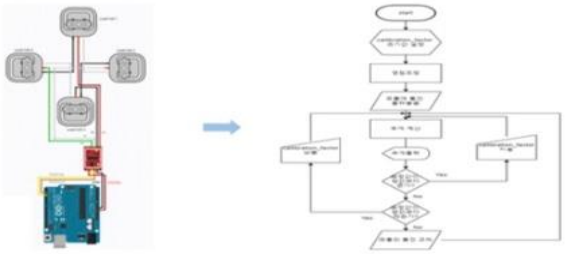
정보취약계층의 사용에 있어서도 실용성이 높게 제작을 하여 조작 편의성을 제공

설계 및 시나리오

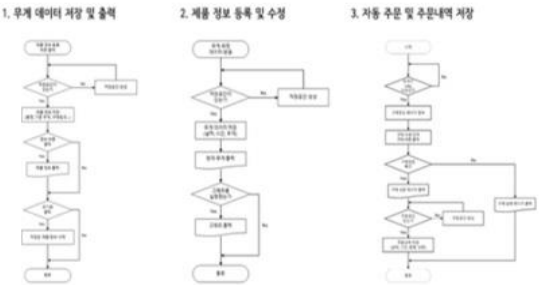
로드셀이 들어있는 체중계와 부품 및 회로 커버 부착 방식



아두이노와 로드셀 회로 연결도 및 알고리즘 작동 방식



PC 프로그램 동작 시나리오



최종 결과물

Department of Mechanical and Information Engineering, University of Seoul

다. 특허출원번호 통지서

22. 6. 21. 오후 11:34 특허로

관 인 생 략

출 원 번 호 통 지 서

출 원 일 자 2022.06.21
특 기 사 항 심사청구(무) 공개신청(무)
출 원 번 호 10-2022-0075880 (접수번호 1-1-2022-0649829-73)
(DAS접근코드3177)
출원인 성명 이휘호(4-2022-031820-8) 외 1명
발명자 성명 이휘호 강재언
발명의 명칭 로드셀을 활용한 무게감지에 따른 자동주문 시스템

특 허 청 장

<< 안내 >>

1. 귀하의 출원은 위와 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 이용하여 특허로 홈페이지(www.patent.go.kr)에서 확인하실 수 있습니다.
2. 출원에 따른 수수료는 접수일로부터 다음날까지 동봉된 납입영수증에 성명, 납부자번호 등을 기재하여 가까운 은행 또는 우체국에 납부하여야 합니다.
※ 납부자번호 : 0131(기관코드) + 접수번호
3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고객번호 정보변경(경정), 정정신고서]를 제출하여야 출원 이후의 각종 통지서를 정상적으로 받을 수 있습니다.
4. 기타 심사 절차(제도)에 관한 사항은 특허청 홈페이지를 참고하시거나 특허고객상담센터(☎ 1544-8080)에 문의하여 주시기 바랍니다.
※ 심사제도 안내 : https://www.kipo.go.kr-지식재산제도

3 .2 개발사업비 내역서

(단위 : 천원)

	항 목 (품명, 규격)	수 량	단 가	금 액			비 고
				계	현금		
직접개발비	Arduino Nano 33 IoT with headers	1	26.7	26.7	26.7		
	컬러 미니 브레드보드 170핀	2	0.6	1.2	1.2		
	브레드보드 400 핀 Half Size Breadboard	1	0.7	0.7	0.7		
	테스트[CH254] 소켓 점퍼 케이블 40P (칼라) (M/F) 20cm	1	0.85	0.85	0.85		
	테스트[CH254] 소켓 점퍼 케이블 40P (칼라) (M/M) 20cm	1	0.85	0.85	0.85		
	1/4W 5% Axial Resistor 152J (1.5KΩ)	1	0.13	0.13	0.13		

	1/4W 5% Axial Resistor 151J (150Ω)	1	0.13	0.13	0.13		
	1/4W 5% Axial Resistor 331J (330Ω)	1	0.13	0.13	0.13		
	1/4W 1% Axial Resistor 102F (1KΩ)	1	0.25	0.25	0.25		
	테프론선 0.6T (검정) 1M	1	0.46	0.46	0.46		
	14종 점퍼와이어 키트 140PCS/SET [OR0014]	1	3.2	3.2	3.2		
	배송비	2	2.7	5.4	5.44		
	카스 체중계 H5	1	21.71	21.71	21.71		
	카스 체중계 HE70	1	14.5	14.5	14.5		
	HX711 납땜	1	9	9	9		
	HAKKO PRESTO 980 인두기 6종세트	1	53.8	53.8	53.8		
	태광 멀티테스터 TK203	1	28.46	28.46	28.46		
	Arduino Nano 33 IoT	1	29	29	29		
	아두이노 로드셀 무게센서 YZC-131A	1	3.19	3.19	3.19		
	HX711	1	1.1	1.1	1.1		
	3D프린터 출력비용 (회로커버 1안)	1	9	9	9		
	3D프린터 출력비용 (회로커버 2안)	1	33	33	33		
	모자이크타일조각 (10 * 10mm), 80g	1	1	1	1		
	논슬립테이프 (50mm)	1	3	3	3		
	가드스티커 32P SET (원형 소)	1	1	1	1		
	가드스티커 27P SET (원형 중)	1	1	1	1		
	가드스티커 48P SET (사각 소)	1	1	1	1		
	가드스티커(A5)	1	2	2	2		
	가위(19cm)	1	2	2	2		
	버니어캘리퍼스	1	30	30	30		
	스카치테이프	1	2	2	2		
	순간접착제(8g)	1	2	2	2		
	색종이	1	2	2	2		
	글루건	1	3	3	3		
	핫멜트알	1	1	1	1		
	EVA 다용도 패드	1	1	1	1		
	Micro usb cable 2m	1	2	2	2		

	충전기 5V 1A	1	3	3	3		
	블루투스 5.0 동글	1	5	5	5		
	니퍼 145mm	1	3	3	3		
	드라이버 세트 20p	1	3	3	3		
	합 계			315	315		

3 .3 완료 작품의 평가

평 가 항 목	평가방법	적용기준	개 발 목표치	비중 (%)	평가결과
1. 무게 측정 최댓값	무게측정모듈 최대인가하중	kg	최대 150Kg	10	최대 130kg
2. 무게 측정 정확도	20회 반복 측정	% (최대 상대 오차)	≤ 1%	25	5%
3. 자동 주문 성공률	20회 반복 시행	%	≥ 90%	30	90%
4. 제품의 크기	프로토타입 장치의 크기	mm (정사각형 한 변)	≤ 300mm	10	300mm
5. 경제성	프로토타입 제작비용	원	< 200,000 원	25	314,220원

3 .4 향후기대

1) 무게 측정 모듈

개발목표를 무게 측정 최댓값 150kg, 측정 무게의 1% 정확도로 설정했지만 측정 무게가 130kg이 넘어가면 모듈이 불안정한 모습을 보였고 측정 무게의 최대 5%까지 오차가 발생했다. 지면에서의 반력이 로드셀에 전달되는 구조를 개선한다면 무게측정의 정확도와 안정성 측면에서 더 발전된 모습을 기대할 수 있을 것이다.

2) 물품 사용량 통계 제공 기능

누적된 무게 데이터를 활용하여 일간, 주간, 월간 물품 사용량 통계를 사용자에게 제공할 수 있다. 현재 하드웨어를 통해 전달 받은 무게 데이터는 별도의 관리 없이 저장되고 있다. 이 데이터를 기반으로 일간, 주간, 월간 데이터의 형태로 나누어 관리한다면 사용자에게 통계 정보를 제공할 수 있다.

통계 정보를 출력하는 GUI의 경우, 현재 제작된 PC 프로그램에 추가적으로 구현할 수 있다. 파이썬에서 'Matplotlib'를 이용하면 그래프를 출력할 수 있는데, 이 라이브러리는 현재 GUI를 구현에 사용된 'Tkinter'와 함께 활용이 가능하다. 이를 통해, 일간, 주간, 월간 그래프를 나타내는 버튼을 클릭하면 각 기능에 맞는 그래프가 출력되는 기능을 구현할 수 있다.

이러한 사용량 통계를 제공하는 기능은 물품 사용의 효율적인 관리를 도울 수 있을 것이다.

3) 다양한 URL을 통한 자동 주문의 구현

현재 자동주문 알고리즘은 '네이버 쇼핑' 페이지에 맞춰 설계되어 있다. 자동주문을 위해 파이썬 라이브러리인 'Selenium'을 사용하는데 웹페이지를 제어하기 위해서는 각 구성요소의 'Xpath'를 읽어야한다. 하지만, 각 구성요소의 'Xpath'는 웹페이지마다 다르기 때문에 모든 웹페이지에서 작동하게 하려면 모든 쇼핑몰 페이지 구성요소들의 'Xpath'를 정리해야 한다.

이러한 작업을 수행한다면, 모든 웹페이지에서 자동주문이 가능해진다. 이는 특정 웹사이트가 아닌 사용자가 원하는 모든 웹사이트에서의 주문을 가능하게 하여 PC 프로그램의 활용도가 높아지도록 할 것이다.

4) 연결 안정성

현재 알 수 없는 이유로 PC와 무게측정모듈간 블루투스 연결이 20분을 넘기지 못하며 불안정한 모습을 보이고 있다. 연결 안정성을 높이고, 연결이 끊기더라도 프로그램을 재부팅하여 다시 연결될 수 있도록 한다면 제품의 안정성을 크게 높일 수 있을 것이다.

부 록

A-1 참고문헌 및 참고사이트

가. 참고 논문

1. 김대현(Dae-Hyun Kim). "구조 건전성 모니터링을 위한 광섬유 브래그 격자 센서와 차동법을 적용한 로드셀 개발." 한국비파괴검사학회지 29.4 (2009): 299-307.
2. 김정훈 (Jeong-hun Kim), 황석준 (Seok-joon Hwang), 장문경 (Moon-kyeong Jang),and 남주석 (Ju-seok Nam). "로드셀을 이용한 농업부산물 수집량 모니터링 시스템 개발." 한국농업기계학회 학술발표논문집 26.2 (2021): 132-132.
3. 한사발 (Sahbal Han), 고영준 (Yeongjun Ko), 김진형 (Jinhyeong Kim), 안형준 (Hyoungjun An), 조은상 (Eunsang Jo),and 김지영 (Jiyeong Kim). "IoT 기기를 활용한 무게 감지 자동 주문 시스템에 관한 연구." 한국정보처리학회 학술대회논문집 24.2 (2017): 1159-1161.

나. 참고 사이트

1. IoT 스마트 저울 개발 '올트' 팁스 선정
(https://www.venturesquare.net/833401)
2. 눈대중 재고관리 그만...'미래배송'까지 해주는 스마트저울
(https://news.mt.co.kr/mtview.php?no=2021102008281920038)
3. 아마존, '대시 스마트 셸프' 출시… "사무용 소모품을 알아서 주문"
(https://www.ciokorea.com/news/137299)
4. [올트 스마트 재고관리 솔루션] 4차 버전 기능 소개
(https://www.youtube.com/watch?v=htobaUzdAUM)
5. 트라이포드
https://www.nextunicorn.kr/company/%ED%8A%B8%EB%9D%BC%EC%9D%B4%ED%8F%AC%EB%93%9C-9d56a6075781a4b5?tab=service
6. 체중계 분해하여 로드셀 사용하기
(https://blog.danggun.net/2141)

A-2 관련특허

1. 포장용기 자동 주문발주 장치(AUTOMATIC ORDERING DEVICE FOR PACKAGING CONTAINER), 1022950180000

2. 인터넷 냉장고의 식품 자동 주문 시스템(Internet Refrigerator's Food Auto Order System), 1006734350000

3. 쌀 수납장치 및 이 장치를 이용하여 쌀의 재고량을 관리하는 시스템 (a rice container and system for controlling stock of rice by using the same), 1020010048890

4. 소모품 자동발주 시스템 (AUTOMATIC ORDERING SYSTEM FOR CONSUMABLE UNIT), 1020190026093

A-3 소프트웨어 프로그램 소스

1) 무게측정모듈 캘리브레이션 코드 (아두이노 c코드)

```
/*
 Example using the SparkFun HX711 breakout board with a scale
 By: Nathan Seidle
 SparkFun Electronics
 Date: November 19th, 2014
 License: This code is public domain but you buy me a beer if you use this and we meet someday (Beerware license).

 This is the calibration sketch. Use it to determine the calibration_factor that the main example uses. It also
 outputs the zero_factor useful for projects that have a permanent mass on the scale in between power cycles.

 Setup your scale and start the sketch WITHOUT a weight on the scale
 Once readings are displayed place the weight on the scale
 Press +/- or a/z to adjust the calibration_factor until the output readings match the known weight
 Use this calibration_factor on the example sketch

 This example assumes pounds (lbs). If you prefer kilograms, change the Serial.print(" lbs"); line to kg. The
 calibration factor will be significantly different but it will be linearly related to lbs (1 lbs = 0.453592 kg).

 Your calibration factor may be very positive or very negative. It all depends on the setup of your scale system
 and the direction the sensors deflect from zero state
 This example code uses bogde's excellent library: https://github.com/bogde/HX711
 bogde's library is released under a GNU GENERAL PUBLIC LICENSE
 Arduino pin 2 -> HX711 CLK
 3 -> DOUT
 5V -> VCC
 GND -> GND

 Most any pin on the Arduino Uno will be compatible with DOUT/CLK.

 The HX711 board can be powered from 2.7V to 5V so the Arduino 5V power should be fine.

*/

//This library can be obtained here http://librarymanager/All#Avia_HX711
#include "HX711.h"
```

```
#define LOADCELL_DOUT_PIN 6
#define LOADCELL_SCK_PIN 2

HX711 scale;

//-7050 worked for my 440lb max scale setup
float calibration_factor = 200000;

void setup()
{
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale();
  //Reset the scale to 0
  scale.tare();

  //Get a baseline reading
  long zero_factor = scale.read_average();
  //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.print("Zero factor: ");
  Serial.println(zero_factor);
}

void loop()
{
  //Adjust to this calibration factor
  scale.set_scale(calibration_factor);

  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1);
  //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person
  //기존 예제가 파운드(lbs) 기준이지만 우리는 킬로그램(kg)을 쓸것이므로 'kg'로 바꿉시다.
  Serial.print(" kg");
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if(Serial.available())
```

```
{
  char temp = Serial.read();

  //변경 : 보정값 범위 설정 가능하도록 변경
  switch(temp)
  {
    case '1':
      calibration_factor += 10;
      break;
    case '2':
      calibration_factor += 50;
      break;
    case '3':
      calibration_factor += 100;
      break;
    case '4':
      calibration_factor += 1000;
      break;
    case '5':
      calibration_factor += 10000;
      break;

    case 'a':
      calibration_factor -= 10;
      break;
    case 's':
      calibration_factor -= 50;
      break;
    case 'd':
      calibration_factor -= 100;
      break;
    case 'f':ㅊ
      calibration_factor -= 1000;
      break;
    case 'g':
      calibration_factor -= 10000;
      break;
  }
}
```

2) 무게측정 및 블루투스 통신 코드 (아두이노 c 코드)

```
//This library can be obtained here http://librarymanager/All#Avia_HX711
#include "HX711.h"

//찾은 캘리브레이션값을 넣어 줍니다.
#define calibration_factor -20300.0

#include <ArduinoBLE.h>

//DT(DOUT)로 사용하는 핀
#define LOADCELL_DOUT_PIN 6
//SCK(CLK)로 사용하는 핀
#define LOADCELL_SCK_PIN 2


BLEService WeightService("58:BF:25:3C:26:CE");

BLEFloatCharacteristic WeightChar("19B10001-E8F2-537E-4F6C-D104768A1214", BLENotify);

HX711 scale;


long previousMillis = 0;   // last time the battery level was checked, in ms
long prevtime;


//최신값을 몇개 저장해 두는지 개수.
int nValueCount = 30;
//최신값을 저장해 두는 배열. 위의 변수 값과 같은 크기를 설정한다.
float fValue[30];


void setup()
{
  Serial.begin(9600);

  if (!BLE.begin()) {

    Serial.println("starting BLE failed!");

    while (1);

  }
  BLE.setLocalName("weight");
```

```
  BLE.setAdvertisedService(WeightService); // add the service UUID

  WeightService.addCharacteristic(WeightChar); // add the battery level characteristic

  BLE.addService(WeightService);

  WeightChar.writeValue(0);

  // start advertising

  BLE.advertise();

  Serial.println("BLE LED Peripheral");



  Serial.println("HX711 kg demo");

  //HX711 객체를 초기화 한다.
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  //설정된 캘리브레이션 값을 설정 한다.
  scale.set_scale(calibration_factor);
  //영점 잡기. 현재 값을 0으로 둔다.
  scale.tare();

  Serial.println("Readings:");
}

float weight(){
  float w;
  Serial.print("Reading: ");

  //값 임시 저장
  float fValueTemp = 0.0;
  //값 합산
  float fValueSum = 0.0;



  //scale.get_units() returns a float
  fValueTemp = scale.get_units();
  //읽은 값을 합산값에 넣어 준다.
  fValueSum = fValueTemp;
```

```
  int i;
  for(i = 0; i < nValueCount; i = i + 1)
  {
    if(i > 0)
    {//0보다 클때만 계산
      //앞번호일수록 오래된 데이터이다.

      //기존에 저장된 데이터 합산
      fValueSum = fValueSum + fValue[i];

      //값을 앞으로 한칸씩 민다.
      fValue[i - 1] = fValue[i];
    }
  }

  //맨 마지막에 최신값을 넣어 준다.
  fValue[nValueCount - 1] = fValueTemp;

  //합산값을 평균을 낸다.
  w = fValueSum / nValueCount;
  Serial.print((fValueSum / nValueCount), 1);

  //You can change this to kg but you'll need to refactor the calibration_factor
  Serial.print(" kg");
  Serial.println();
  return w;

}

void loop()
{


  // listen for BLE peripherals to connect:

  BLEDevice central = BLE.central();

  // if a central is connected to peripheral:

  if (central) {
    Serial.print("Connected to central: ");

    Serial.println(central.address());
```

```
    // while the central is still connected to peripheral:
    while (central.connected()) {
      float weights;
      long currentMillis = millis();
      long currenttime = millis();
      // if 200ms have passed, check the battery level:

      if (currentMillis - previousMillis >= 50) {

        previousMillis = currentMillis;

        weights = weight();

      }
      if (currenttime - prevtime >= 500) {

        prevtime = currenttime;

        WeightChar.writeValue(weights);   // and update the battery level characteristic

      }

      //if (WeightChar.written()){
        //int val = WeightChar.value();
        //Serial.println(val);
      //}
    }

    Serial.print(F("Disconnected from central: "));

    Serial.println(central.address());

  }



}
```

3) 블루투스 연결 및 무게 수신(PC 파이썬 코드)

```
import os
import asyncio # 비동기화 통신을 위한 라이브러리
import bleak   # bleak 라이브러리
from bleak import BleakClient
import time
import struct
import datetime
read_data = bytearray()
filename = "C:/project/weight.txt"
# ESP32 맥 주소
address = "58:BF:25:3C:26:CE"
# ESP32 BLE_notity 예제에 있는 캐릭터리스틱 주소
# notity_charcteristic_uuid = "19B10001-E8F2-537E-4F6C-D104768A1214"
notity_charcteristic_uuid = "19b10001-e8f2-537e-4f6c-d104768a1214"


# ESP32가 notify로 보낸 데이터를 받는 콜백함수
def notify_callback(sender: int, data: bytearray):
    global read_data
    read_data = data

async def run(address):
    global read_data
    # BleakClient 클래스 생성 및 바로 연결 시작
    # address: ESP32 맥주소
    # timeout: 연결 제한 시간 5초가 넘어가면 더 이상 연결하지 말고 종료
    async with BleakClient(address, timeout=50) as client:
        # 연결을 성공함
        print('connected')
        # 연결된 BLE 장치의 서비스 요청
        services = await client.get_services()
        # 서비스들을 루프돌려 내부 캐릭터리스틱 정보 조회
        for service in services:
            print('service uuid:', service.uuid)
            # 각 서비스들에 있는 캐릭터리스틱을 루프 돌려 속성들 파악하기
            for characteristic in service.characteristics:
                print('  uuid:', characteristic.uuid)
                # handle 정보도 함께 확인
                print('  handle:', characteristic.handle)
                print('  properties: ', characteristic.properties)
                # 캐릭터리스틱 UUID가 우리가 찾는 UUID인지 먼저 확인
                if characteristic.uuid == notity_charcteristic_uuid:
                    # 우리가 찾던 UUID가 맞다면
                    # 해당 캐릭터리스틱에 notify 속성이 있는지 확인
                    if 'notify' in characteristic.properties:
                        # notify 속성이 있다면 BLE 장치의 notify 속성을
                        # 활성화 작업 후 notify_callback 함수 연결
```

```
50)                     print('try to activate notify.')
51)                     # await client.start_notify(characteristic,
   notify_callback)
52)
53)         while client.is_connected() :
54)             try :
55)
56)                 # read_data = await
   client.read_gatt_char(notity_charcteristic_uuid)
57)                 await client.start_notify(characteristic, notify_callback)
58)                 await asyncio.sleep(3)
59)

60)             except :
61)                 print("read date failed")
62)                 await client.disconnect()
63)                 exit()
64)#                 args = readCommand(sys.argv[1:])
65)# if runGames(**args) == -1:
66)#     os.execl(sys.executable, sys.executable, *sys.argv)
67)             # await client.write_gatt_char(notity_charcteristic_uuid,
   bytes(b'hello world'))
68)             await client.stop_notify(characteristic)
69)             weight = struct.unpack('f', read_data)
70)
71)             weight = round(weight[0], 2)
72)             print(weight)
73)             now = datetime.datetime.now()
74)             print(now)
75)             if (os.path.isdir("C:/project") == True) and
   (os.path.isfile('C:/project/weight.txt') == False) :
76)                 f = open('C:/project/weight.txt', 'w')
77)                 f.close()
78)             elif (os.path.isdir("C:/project") == False) and
   (os.path.isfile('C:/project/weight.txt') == False) :
79)                 os.makedirs('C:/project')
80)                 f = open('C:/project/weight.txt', 'w')
81)                 f.close()
82)             f = open('C:/project/weight.txt', 'a')
83)             f.write(str(now) + ", " + str(weight) + "\n")
84)
85)             f.close()
86)
87)             # await asyncio.sleep(2.5)
88)         '''
89)         # client 가 연결된 상태라면
90)         if client.is_connected:
91)             # 1 초간 대기
92)             await asyncio.sleep(1)
93)             print('try to deactivate notify.')
94)             # 활성시켰단 notify 를 중지 시킨다.
95)             await client.stop_notify(notity_charcteristic_uuid)
96)         '''
97)     #with 문을 빠져나오면 장치는 알아서 disconnect 가 된다.
```

```
    print('disconnect')

        loop = asyncio.get_event_loop()
        loop.run_until_complete(run(address))

        # loop.run_in_executor(run(address))
        print('done')
```

4) 메인 프로그램 (PC 파이썬 코드)

```
import tkinter
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import time
import random
import threading
import os
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import pyperclip
import csv
import datetime as dt
import pandas as pd

#제품 관련 정보 저장을 위한 txt 파일 생성
if (os.path.isdir("C:/project") == True) and (os.path.isfile('C:/project/info.txt')
== False) :
    f = open('C:/project/info.txt', 'w')
    f.close()
elif (os.path.isdir("C:/project") == False) and
(os.path.isfile('C:/project/info.txt') == False) :
    os.makedirs('C:/project')
    f = open('C:/project/info.txt', 'w')
    f.close()

mainpage = Tk() #메인화면 생성

mainpage.title("Loadcell Project") #메인화면 이름
mainpage.geometry("400x720+1400+200") #메인화면 크기 결정
mainpage.resizable(False, False) #메인화면 크기 조절 여부 불가로 설정

page = Canvas(mainpage, width=1080, height= 720) #메인화면에 선 배치를 위한 캔버스
설정
page.pack() #캔버스 배치

#page.create_line(400,0,400,720, fill = "black") #세로선 배치
page.create_line(0,200,400,200, fill = "black") #가로선 배치
page.create_line(0,500,400,500, fill = "black") #가로선 배치
####### 무게 읽는 함수 새로 추가
####### 마지막 줄 = now, 끝에서 두번재 줄 = prev
def readweight() :

    format = '%Y-%m-%d %H:%M:%S.%f'

    f = open("C:/project/weight.txt", 'r')

    lines = f.readlines()
    if not lines :
        return [(-1, -1), (-1, -1)]
    if len(lines) < 10 :
        return [(-1, -1), (-1, -1)]
```

```
    now = lines[-1]
    prev = lines[-10]

    now_date, now_weight = now.split(", ")
    prev_date, prev_weight = prev.split(", ")

    now_weight = float(now_weight)
    now_date = dt.datetime.strptime(now_date,format)

    prev_weight = float(prev_weight)
    prev_date = dt.datetime.strptime(prev_date,format)
    f.close()

    return [(now_date, now_weight), (prev_date, prev_weight)]

#테스트를 위한 랜덤 무게 생성 및 출력
def weight_change():
    # weight = random.uniform(0,50)
    #측정된 무게를 화면에 표시
    weights = readweight()
    weight = weights[0][1]

    weight_label = tkinter.Label(mainpage, text = "%.1f"%weight,
font=("arial",40,"normal"))
    weight_label.place(x=130, y=60)
    print("%.1f" % weight)

    # return weight
    threading.Timer(10, weight_change).start()

weight_change()

#자동 주문 함수
def auto_order():
    print("start auto_order")
    driver = webdriver.Chrome('C:/project//chromedriver')

    # 네이버 웹페이지 접속)
    driver.get('https://naver.com')
    time.sleep(1)

    # 로그인 버튼 클릭
    driver.find_element_by_xpath("//a[@class='link_login']").click()
    time.sleep(1)

    # 아이디 입력
    tag_id = driver.find_element_by_name('id')
    tag_id.clear()
    tag_id.click()
    pyperclip.copy('Naver_ID')
    # pyperclip.copy('Naver_ID')
    tag_id.send_keys(Keys.CONTROL, 'v')
```

```
    # 비밀번호 입력
    tag_pw = driver.find_element_by_name('pw')
    tag_pw.clear()
    tag_pw.click()
    pyperclip.copy('Naver_PW')
    tag_pw.send_keys(Keys.CONTROL, 'v')

    # 로그인 버튼 클릭
    driver.find_element_by_id('log.login').click()
    time.sleep(1)

    # 상품 구매 페이지 접속
    url = 'https://smartstore.naver.com/auto_test/products/6732771139'

    driver.get(url)
    driver.set_window_size(960, 1200)

    # 구매 수량 확인
    #count = 1
    #while count < 4:
    #    btn_plus =
driver.find_element_by_xpath('//*[@id="content"]/div/div[2]/div[2]/fieldset/div[5]/
ul/li/div/div/div/button[2]')
    #    btn_plus.send_keys(Keys.ENTER)
    #    count = count + 1
    #    if count == 3:
            # 구매버튼 클릭
    xpath = '//*[@id="content"]/div/div[2]/div[2]/fieldset/div[7]/div[1]/div/a'
    btn = driver.find_element_by_xpath(xpath)
    btn.send_keys(Keys.ENTER)
    time.sleep(3)

    driver.quit()

#자동주문 날짜, 시간 저장을 위한 파일 생성
if os.path.isfile('C:/project/order_date.txt') == False:
    f = open('C:/project/order_date.txt', 'w')
    f.close()


#현재 날짜 및 시간 저장
def order_date():
    now = dt.datetime.now()

    f = open('C:/project/order_date.txt', 'a', newline='')
    f.write(now.strftime('%Y-%m-%d %H:%M\n'))
    f.close()


#현재 무게 확인 후 기준 무게 이하면 자동 주문
def weight_order():
    # weight_change()

    print("weight_order")
```

```
    weights = readweight()
    now_weight = weights[0][1]
    prev_weight = weights[1][1]
    print("now weight : {0}, prev_weight = {1}".format(now_weight, prev_weight))

    if now_weight < 10 and prev_weight > 10:
        time.sleep(3)
        auto_order()
        order_date()

    #30초마다 반복 수행
    threading.Timer(30, weight_order).start()

weight_order()


#무게 단위 표시
tkinter.Label(mainpage, text = "Kg", font=("arial",24,"normal")).place(x=240,y=80)

#제품 등록/변경 페이지
def product_page():
    p_page = Toplevel()
    p_page.title("제품 등록/변경")
    p_page.geometry('200x250+450+250')
    p_page.resizable(False, False)

    title = Label(p_page)
    title['text'] = '제품 등록/변경'
    title.place(x = 55, y = 10)

    #제품명 입력
    label_name = Label(p_page)
    label_name['text'] = '제품이름 : '
    label_name.place(x = 30, y = 50)
    p_name = Entry(p_page, width=10)
    p_name.place(x = 90, y = 50)

    #기준무게 입력
    label_weight = Label(p_page)
    label_weight['text'] = '기준무게 : '
    label_weight.place(x = 30, y = 90)
    p_weight = Entry(p_page, width=10)
    p_weight.place(x = 90, y = 90)

    #구매링크 입력
    label_link = Label(p_page)
    label_link['text'] = '구매링크 : '
    label_link.place(x = 30, y = 130)
    p_link = Entry(p_page, width=10)
    p_link.place(x = 90, y = 130)

    #구매수량 입력
    label_count = Label(p_page)
    label_count['text'] = '구매수량 : '
```

```
    label_count.place(x = 30, y = 170)
    p_count = Entry(p_page, width=10)
    p_count.place(x = 90, y = 170)

    def save():
        f = open('C:/project/info.txt', 'w', encoding="UTF8")
        f.write('%s,' % p_name.get())
        f.write('%s,' % p_weight.get())
        f.write('%s,' % p_link.get())
        f.write('%s' % p_count.get())
        f.close()
        p_page.destroy()


    #저장 버튼, 클릭 시 창 닫힘
    b_save = Button(p_page, width=10, height=1, text = "저장", command=save)
    b_save.place(x = 55, y = 200)

    p_page.mainloop()

#제품 등록/변경 버튼
product = Button(mainpage, width=20, height=5, text = "제품 등록/변경",
command=product_page)
product.place(x = 30, y = 250)

#제품 정보 확인 페이지
def info_page():
    i_page = Toplevel()
    i_page.title("제품 정보 확인")
    i_page.geometry('200x230+450+250')
    i_page.resizable(True, True)

    title = Label(i_page)
    title['text'] = '제품 정보 확인'
    title.place(x = 55, y = 10)

    #저장된 제품 정보를 불러옴
    f = open('C:/project/info.txt', 'r', encoding='UTF8')
    p_info = f.readline()
    info_list = p_info.split(',')
    p_name = info_list[0]
    p_weight = info_list[1]
    p_link = info_list[2]
    p_count = info_list[3]

    #제품명 출력
    label_name = Label(i_page)
    label_name['text'] = '제품이름 : '
    label_name.place(x = 30, y = 50)
    i_name = Label(i_page, text = p_name)
    i_name.place(x = 90, y = 50)

    #제품 기준무게 출력
    label_weight = Label(i_page)
```

```
    label_weight['text'] = '기준무게 : '
    label_weight.place(x = 30, y = 90)
    i_weight = Label(i_page, text = p_weight)
    i_weight.place(x = 90, y = 90)

    #제품 구매링크 출력
    label_link = Label(i_page)
    label_link['text'] = '구매링크 : '
    label_link.place(x = 30, y = 130)
    i_link = Label(i_page, text = p_link)
    i_link.place(x = 90, y = 130)

    #구매수량 입력
    label_count = Label(i_page)
    label_count['text'] = '구매수량 : '
    label_count.place(x = 30, y = 170)
    i_count = Label(i_page, text = p_count)
    i_count.place(x = 90, y = 170)

    i_page.mainloop()

#제품 정보 확인 버튼
info = Button(mainpage, width=20, height=5, text = "제품 정보 확인",
command=info_page)
info.place(x = 220, y = 250)

#자동 주문 내역 불러오기
def order_list():
    list_show = Toplevel()
    list_show.title("최근 주문 내역")
    list_show.geometry('300x200+450+250')
    list_show.resizable(False, False)

    title = Label(list_show)
    title['text'] = '최근 주문 내역'
    title.place(x = 110, y = 10)

    f = open('C:/project/order_date.txt', 'r')

    order_dates = f.readlines()
    count = len(order_dates)

    #최근 5번의 자동주문 내역을 보여줌
    if count >= 5:
        j = 5
        for i in range(0, 5):
            num = Label(list_show)
            num['text'] = i+1
            num.place(x = 90, y = 40 + (30 * i))

            order_show = Label(list_show)
            order_show['text'] = order_dates[count-j].strip()
            order_show.place(x = 110, y = 40 + (30 * i))
        j -= 1
```

```
    #최근 주문내역이 5개 미만이면 현재까지의 주문내역만 출력
    else:
        for k in range(0, count):
            num = Label(list_show)
            num['text'] = k+1
            num.place(x = 90, y = 40 + (30 * k))

            order_show = Label(list_show)
            order_show['text'] = order_dates[k].strip()
            order_show.place(x = 110, y = 40 + (30 * k))

    f.close()


#자동 주문 내역 버튼
order = Button(mainpage, width=20, height=5, text = "자동 주문 내역", command =
order_list)
order.place(x = 30, y = 360)

#제품 정보 초기화
def delete_info():
    if os.path.isfile('C:/project/info.txt'):
        f = open('C:/project/info.txt', 'w')
        f.write(' , , ')
        f.close()

        messagebox.showinfo("Delete", "제품 정보가 삭제되었습니다.")

#제품 정보 초기화 버튼
product_reset = Button(mainpage, width=20, height=5, text = "제품 정보 초기화",
command = delete_info)
product_reset.place(x = 220, y = 360)



#일 단위 그래프 버튼
day_graph = Button(mainpage, width=15, height=5, text = "일 단위 그래프")
day_graph.place(x = 20, y = 560)

#주 단위 그래프 버튼
week_graph = Button(mainpage, width=15, height=5, text = "주 단위 그래프")
week_graph.place(x = 140, y = 560)

#월 단위 그래프 버튼
month_graph = Button(mainpage, width=15, height=5, text = "월 단위 그래프")
month_graph.place(x = 260, y = 560)



mainpage.mainloop() #메인화면 동작
```

설계구성요소 및 제한요소

설계구성요소	과제보고서내의 항목	내용 요약
목 표 설 정	1.1 개발과제의 개요	과제의 배경, 목표, 내용을 설정함
합 성	2.1 설계사양 2.2 개념설계안 2.4 조립도 2.5 부품도	과제를 세부적으로 나누고 세부사양별 여러 개념설계안 도출, 개념설계안 채택 및 구체화
분 석	2.1 설계사양 2.3 이론적 계산 및 시뮬레이션	구체화된 개념설계안 분석
제 작	2.8 자재소요서 3.1완료작품소개 3.2 개발사업비 내역서	프로토타입 제작
시험 및 평가	3.3 완료작품의 평가	시제품 시험 및 평가

설계제한요소	과제보고서내의 항목	내용 요약
원 가	1.3 관련시장에 대한 분석 3.2 개발사업비 내역서	프로토타입의 예상 제작비는 200,000원이고, 실제 제작비는 314,220원이 소요되었다.
안정성 및 윤리성	1.1 개발과제의 개요 1.4 개발과제의 기대효과 2.1 설계사양	IoT를 활용한 재고관리는 대부분 대형 사업장을 대상으로 하고 있다. 하지만, 중소규모의 사업장에서도 IoT를 활용하면 효율적인 재고 관리가 가능하다. 따라서, 지속적으로 사용하는 제품의 무게를 측정해 자동으로 주문할 수 있다면 경쟁력이 있을 것으로 판단해 본 프로젝트를 기획했다.
신 뢰 성	2.1 설계사양	측정최대무게, 정확도, 소프트웨어 안정성 등 제품의 신뢰도에 대한 설계사양을 정하였다.
사회에 미치는 영향	1.1 개발과제의 개요 1.4 개발과제의 기대효과	사람에 의해 직접 수행되었던 재고관리를 IoT 기술을 활용하여 효율적으로 관리할 수 있고, 인적/시간적 자원의 낭비를 줄일 수 있다.
환 경 요 인	2.1 설계사양	전원 공급 방식, 하드웨어와 PC 사이의 거리, PC 프로그램의 편의성을 고려하여 설계하였다.
기 타	2.1 설계사양 3.3 완료작품의 평가	본 프로젝트에서 비중이 가장 높은 자동주문 정확도는 90%로 개발 목표치를 만족하였다. 프로토타입 제작 가격과 무게 측정 정확도는 개발 목표치를 달성하지 못했다. 하지만, 대량생산과 로드셀 센서의 변경 등을 통해 충분히 개선 가능한 부분으로 판단했다.