캡스톤 디자인 (2)

(프로젝트 보고서)

담당 교수님	박상오 교수님
팀 이름	You require more city gas (2조)
팀 원	20145034 홍성현 20146290 김성민 20142921 이승현

목 차

1. 프로젝트 주제 2

2. 팀이름 및 팀원 및 깃허브 주소 2

3. 프로젝트 개요 2

4. 프로젝트를 시작한 동기 3

5. 프로젝트의 목표 3

6. 프로젝트의 차별성 3

7. 개발 및 구현 내용 5

8. 업무 분담 5

9. 주간 진행 계획 6

10. 개발 및 구현 세부 내용 7

1. 프로젝트 주제

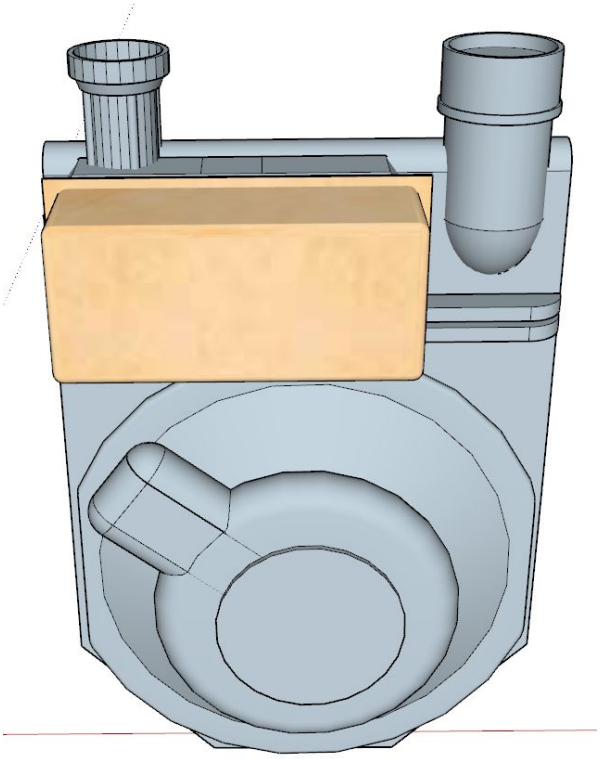
광학 문자 인식 기술을 이용한 가스 원격 검침 모듈 제작

2. 팀이름 및 팀원 및 깃허브 주소

- You require more city gas.
- 김성민(20146290), 이승현(20142921), 홍성현(20145034)
- https://github.com/yourequiremorecitygas

3. 프로젝트 개요

도시 가스 검침을 검침원이 일일이 하지 않아도, 자동으로 사진을 찍어 검침하는 모듈을 개발하고자 한다.



4. 프로젝트를 시작한 동기

우리나라 가정집들의 가스 계량기는 검침원이 매 달 직접 검침하는 경우가 많다. 계량기가 옥외에 설치되어 있다면 검침원이 검침할 수 있지만 옥내에 설치되어 있으면 거주자가 직접 확인 후 매 월 기재하거나 검침원에게 통보해야 한다. 하지만 매 월마다 자가 검침을 하다보면 때때로 검침을 잊어버리는 경우가 생기기 때문에, 검침원이 재방문하거나 연락을 하는 경우가 발생한다.

또한, 가스 계량기가 외진 장소나 주택 사이처럼 접근하기 어려운 위치에 있으면 검침원이 부상을 입기도 한다. 뿐만 아니라 가정집을 방문할 때, 검침원들은 성범죄에 노출되기도 하며, 반대로 가스 검침 사칭 범죄도 발생한 적 있다.

이와 같은 문제들을 해결하기 위해 계량기에 자동화 모듈을 부착해 원격 가스 검침을 수행하여 검침 효율을 높이고, 검침원들의 근로 환경을 개선해보고자 한다.

5. 프로젝트의 목표

- 자가 검침을 하던 거주자들이 가스 검침에 대해 신경을 쓰지 않을 수 있도록 돕는다.
- 검침원들이 검침을 위해 일일이 돌아다닐 필요가 없게 한다.

매 월 검침일 마다 검침원이 직접 검침하지 않아도 계량기 앞에 부착된 자동화 모듈을 통해 가스 계량기의 사진을 처리 서버에 전송한다. 처리 서버에서는 광학 문자 인식(OCR, Optical Character Recognition) 기술을 사용해 검침 값을 파악한다. 그 후, 이 정보들을 데이터베이스에 저장하고 결과 서버(사업체 서버)로 데이터를 전송하는 서비스를 제공한다. 세부적인 구현 목표는 다음과 같다.

- OCR의 목표 정확도를 100%에 근접하게 만든다.

기존에 하던 검침 방식은 사람이 수행하기 때문에 실수에 의해 오차가 생기기도 했다. 실제로 약 250만 세대를 관리하는 가스도시회사가 2018년도에 2개월 간 받은 총 20만 525건 가스 요금 관련 민원 또는 문의 중 가스 요금 확인 약 4만건(약 1.588%)으로 가장 높았다. 이 프로젝트는 검침 과정에 사람의 실수가 발생하지 않도록 검침을 자동화하고, 신뢰성을 얻기 위해 100%에 근접한 정확도를 목표로 한다.

6. 프로젝트의 차별성

차별성을 기술하기에 앞서, 우리 프로젝트의 예상 소비자는 '개인'이 아니라 '기업'이다. 즉, 각 지자체와 계약을 맺은 국내 34개의 도시가스회사이다. 최근 1~2인 가구들이 증가함에 따라 검침해야 하는 가구 수도

증가하고 있기 때문에 도시가스회사들은 원격 가스 검침의 필요성을 느끼고 있다. 실제로 몇몇 회사들은 이미 원격 가스미터 시스템 구축에 투자를 하고 있다.

- 호환성

우리의 프로젝트는 일반적인 기계식 계량기에 부착하는 모듈을 개발하는 것이므로 계량기를 교체하더라도 교체된 계량기에 부착하면 그대로 사용가능하다. 또한 OCR 기술을 사용해 검침값을 읽기 때문에 특정 가스 계량기 회사의 모델에 종속적이지 않다. 즉, 기존 시스템에 대해 호환성이 뛰어나다.

- 유효성 검증 및 원활한 민원 해결

사진 정보를 저장한다는 점 역시 차별성이다. 다른 원격 검침용 가스 계량기는 값만 수집 하기 때문에 가스 계량기의 값과 원격 검침 데이터가 동일한지 유효성을 검증하기 어렵다. 이는 실제로 2016년 산업통산자원부가 발표한 보고서에서 기술된 가스 AMI 보급 기술 측면 장애 요인 중 하나로서, 이 프로젝트는 사진 데이터를 통해 유효성 검증이 가능하다. 뿐만 아니라 소비자 민원 발생 시 사진 정보를 근거로 삼을 수 있기 때문에 원활한 민원 해결이 가능하다.

- 현 시스템과 지능형원격검침 시스템의 과도기에서 발생하는 문제 해결

스마트 미터는 기술에 따라 단계적으로 자동원격검침(AMR)과 지능형원격검침(AMI)로 분류된다. 자동원격검침은 무선원격검침만 가능하도록 하는 것이고, 지능형원격검침은 계량기 내에 다양한 기능을 추가해 원격 검침, 차단, 가스 안전, 에너지 효율 제고, 클라우드를 통한 빅데이터 활용 등을 위한 지능형 계량 인프라를 의미한다. 스마트 계량기로 불리기도 한다.

우리나라 정부 사업으로 전국 가스 AMI 보급확산 사업이 2016년부터 시행중이다. 한 번에 AMI를 보급하는 것이 불가능하기 때문에 단계적인 보급 확산 전략을 추진 중이다. 단계는 미적용 → 기초수준(AMR)[보급 초기] → 중간 수준(AMI Ready)[보급 확대기] → 고도화(AMI)[보급 고도화]로 나뉜다. 우리의 프로젝트는 보급 초기 단계에서 발생하는 문제점들을 해결하여 장기적인 관점에서 보급 고도화로 갈 수 있도록 징검다리 역할을 수행하여 이바지한다.

과도기에서 발생하는 문제점 중 하나는 가스 계량기는 5년 마다 교체하는데, 기계식 계량기의 경우 도시가스회사가 비용을 부담하지만 스마트 계량기의 경우 일반 사용자가 부담해야 한다는 것이다. 우리의 프로젝트는 계량기와 검침 모듈을 분리함으로써 이를 해결한다. 계량기는 마모가 상대적으로 많이 되므로 교체해야 하지만 검침 모듈은 그렇지 않다.

- 도시가스회사의 선택지 증가

도시가스회사 입장에서 기존 시스템을 유지하거나 스마트 계량기를 바로 적용한다는 선택지 이외에 원격 검침만 가능한 시스템이라는 선택지가 추가된다. 기존 기계식 계량기 사용을 유지하면서 원격검침이 가능하게 하는 것은 소비자의 선택지를 늘려주는 일이다.

7. 개발 및 구현 내용

1) 하드웨어

- 프로젝트에 필요한 하드웨어 파악 및 구매
- 하드웨어 (기계/회로) 설계 및 하드웨어 코딩
- 비동기식 통신
- 사진 촬영 및 전송

2) 처리 서버

- 하드웨어와 통신
- 하드웨어로부터 사진 수신 후 저장
- 머신러닝 이용하여 사진에서 숫자 추출(OCR 탑재)

3) 결과 서버

- 처리 서버로부터 데이터 수신 확인 및 표출
- 관리자용 웹 UI 구현

4) 광학 문자 인식(OCR)

- 데이터 전처리
- OCR을 사용한 계량기 숫자 인식
- 학습 데이터 수집 및 전처리

8. 업무 분담

- 공통 : 학습 데이터 수집 및 전처리
- 김성민 : 처리 서버 및 결과 서버 구현
- 홍성현 : 광학 문자 인식(OCR)
- 이승현 : 하드웨어 설계 및 제작, 하드웨어 코딩

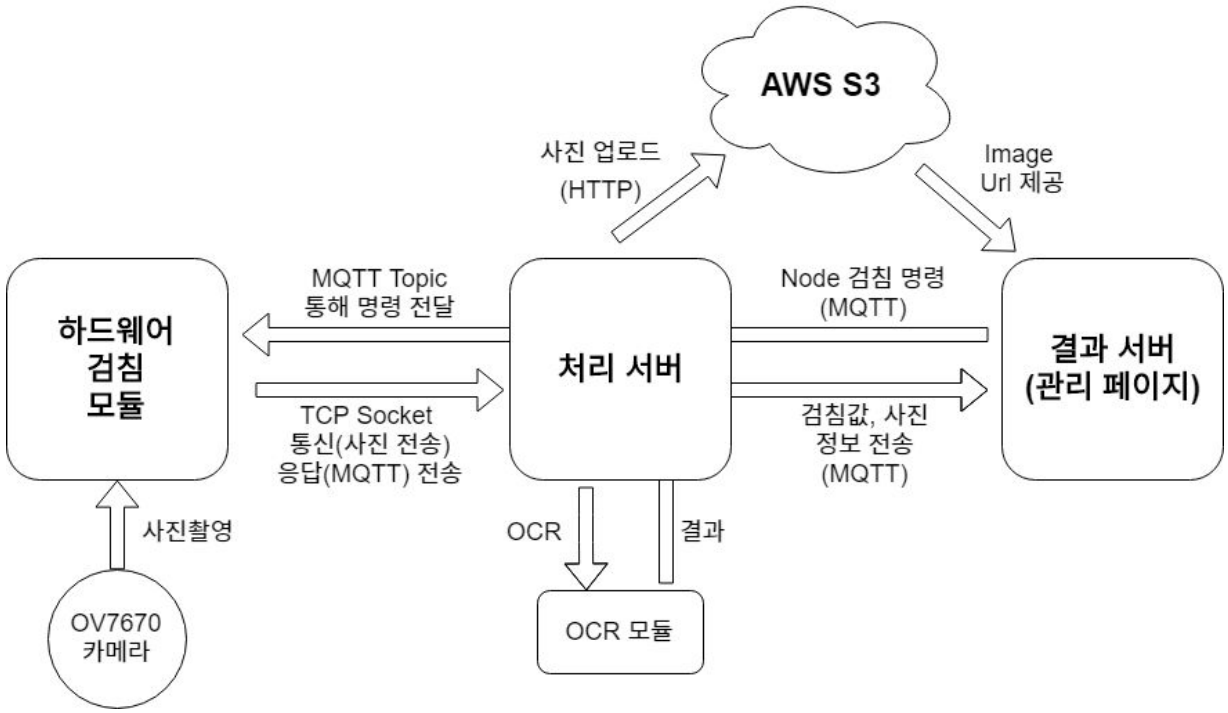
9. 주간 진행 계획

공통 : 초록색 / 이승현 : 하늘색 / 김성민 : 노란색 / 홍성현 : 분홍색

	3월				4월				5월					6월		
	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12	19
제안서 작성 및 발표							중간데모	중간고사						최종데모	최종리포트 제출	기말고사
프로젝트 계획 개선 및 제안서 수정																
학습 데이터 수집																
학습 데이터 전처리																
하드웨어 설계 및 부품 구매 (리비전 포함)																
결과서버와 처리서버 통신 프로토콜 설계																
결과서버 구축 및 결과 표시 기능 구현																
문자인식(OCR) 구현																
카메라 기능(Camera + LED) 구현 및 테스트																
처리서버 구축																
하드웨어-서버간 비동기 통신 구현 및 테스트																
문자인식 정확도 향상																
하드웨어 개선, 마감																
서버에 OCR기능 탑재																
관리페이지 구축																
결과서버-처리서버간 통신 구현																
최종발표 준비 및 통합 테스트																

10. 개발 및 구현 세부 내용

(1) 프로젝트 전체 구성



(2) 하드웨어

- **프로젝트에 필요한 하드웨어 파악 및 구매**

이번 캡스톤 프로젝트에서 사용한 하드웨어는 다음과 같다.

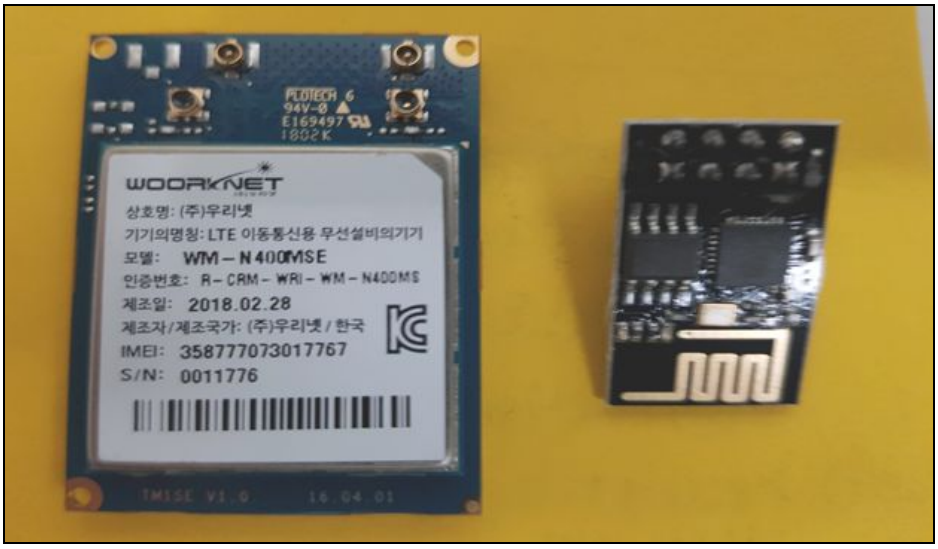
종류	부품명	단가(KRW)
MCU	Mega 2560 Pro Embeded (Arduino 호환)	11000
Camera	OV7670 (with FIFO Buffer)	3500
통신모듈	WM-N400MS(Cat.M1), sx1278(LoRa), **ESP-01(WiFi, 최종선택)**	**4000**
LED	KY-009	600
기타	3D 프린팅용 필라멘트 (750g)	69000
기타	9V 알카라인 건전지 (다이소 구매)	2000
기타	테스트용 가스계량기 (검침부)	33000

우선, 우리가 모듈제작에 있어서 가장 중요하게 생각한 요소는 모듈의 단가와 크기였다. 그 이유는 우리가 제작할 모듈의 단가가 일반 가스계량기보다 비싸다면 우리의 모듈을 구매할 메리트가 많이 떨어질 것이라고 생각했었고, 모듈의 크기 역시 가스 검침기에 붙일 수 있어야 하기 때문에, 일반적인 Arduino Uno와 같은 MCU를 사용할 수는 없었다.

물론, 크기와 가격적인 측면에서 Arduino Nano 계열의 MCU역시 사용할 수는 있었으나, 우리가 MCU에 추가로 장착해야하는 카메라, 통신 모듈의 경우, I2C 통신과 Serial 통신을 추가로 요구했으므로 필요한 핀의 개수가 최소 20개 이상이었다. 따라서 하드웨어간의 호환성을 고려하여, 아두이노 계열중에서도 가장 성능이 좋고 지원하는 핀의 개수가 많은 Mega 계열의 MCU를 선택하게 되었다. 또한, 국내에서 구입할 경우 하드웨어 모듈 단품에 대한 비용이 더 들었기 때문에, 중국의 타오바오를 이용하여 모듈의 단가를 더 줄이는데 노력했다.

〈 LoRa 통신모듈 , E32-915T30D(sx1278) 〉

통신모듈의 경우, 원래 우리의 생각은 LoRa 네트워크를 이용해서 이미지를 전송해볼 생각이었다. 그래서 필요한 LoRa transceiver 모듈과 사설 LoRa Gateway를 구축하기 위한 MCU를 구매했었다. 하지만 LoRa의 대역폭이 낮아 이미지를 보내는데 부적합했고, 대안으로 우리는 SKT가 국내에 구축한 저전력 LTE(Cat.M1) 네트워크를 이용해보고자 했다.



〈 WM-N400MSE (LTE 모듈 (좌)), ESP-01(WiFi모듈(우)) 〉

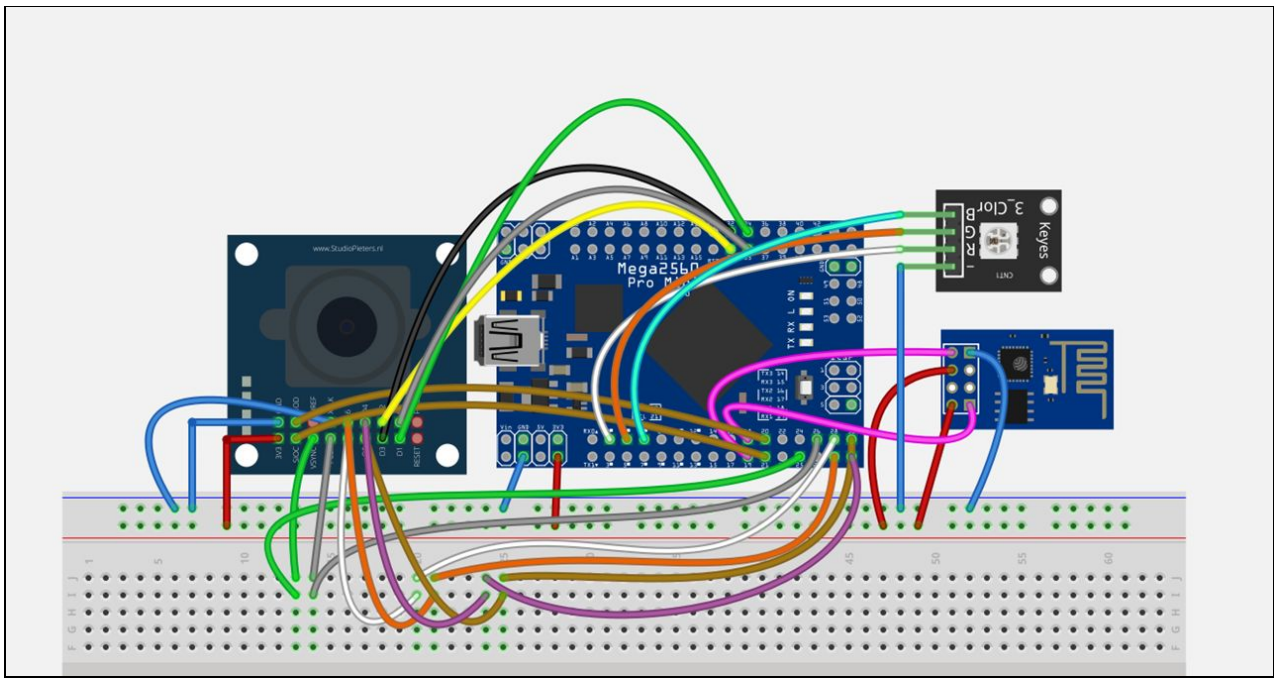
SKT의 Cat.M1 네트워크망의 경우, 면허 대역으로 SKT의 사전승인이 필요했다. 다행이도, SKT에서는 자사 네트워크 홍보의 일환으로 LTE모듈(좌)을 무료로 나눠주고 있었고 우리는 해당 모듈을 신청했다. 하지만, 실제로 중요한 것은 승인 받은 유심을 해당 모듈에 끼워 사용해야 한다는 것이었다. 그래서 우리는 통신모듈을 사용하기 위해, SKT에 Cat.M1 유심 개통을 요구했다.

하지만 SKT로부터 받은 답장은 '개통거절' 이었다. 이유는 Cat.M1 도 현재 운영중인 LTE(4G) 인프라를 이용하므로 해당 망 위에서 테스트를 하다가 망부하가 걸릴 경우, SKT 서비스 운영에 지장이 있을 수 있다는 이유였고, 추가로 학생에게 개통 승인해주는 경우는 자사의 해커톤(대회)에 참가했을 때 뿐이라고 답장을 받았다.

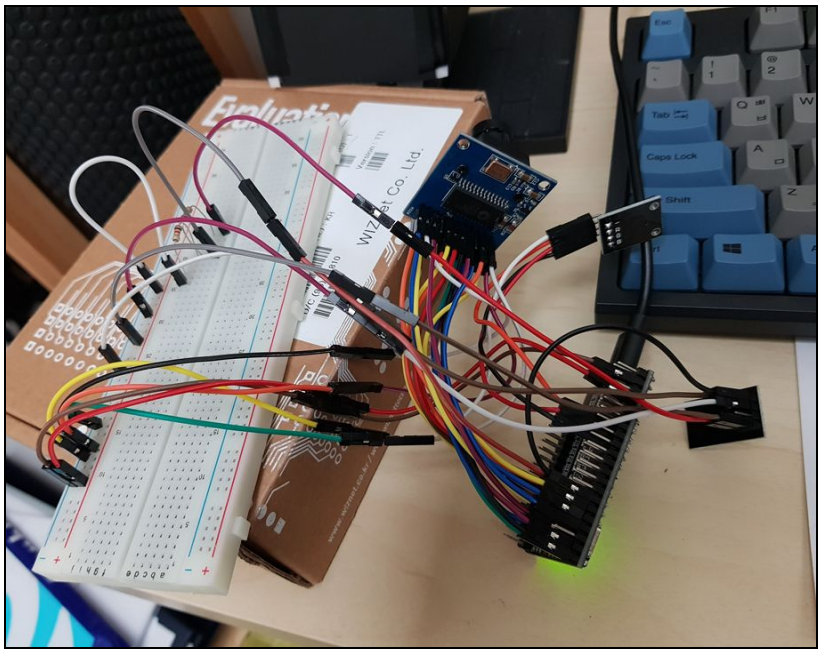
결국 우리는 LoRa와 Cat.M1 네트워크 대신에 WiFi 네트워크를 선택했다. 이를 이용하기 위해서 ESP-01(우) 모듈을 추가로 구매했고 해당 모듈을 위해 하드웨어에 코딩을 했다.

추가적으로, 우리가 WiFi 보다 LoRa, Cat.M1 네트워크에 집중한 이유는 iot 를 위한 저비용, 저전력 네트워크였기 때문이다. 실제로 LoRa나 Cat.M1 네트워크를 이용할 경우, WiFi보다 전력소모가 더 적을뿐만 아니라 사용자 입장에서 네트워크 망이용에 대한 가격 부담이 적기 때문이다. (WiFi의 경우 회선 가입비와 무선라우터 구매 비용이 필요하고 통신거리 역시 짧다)

- 하드웨어 (회로) 설계

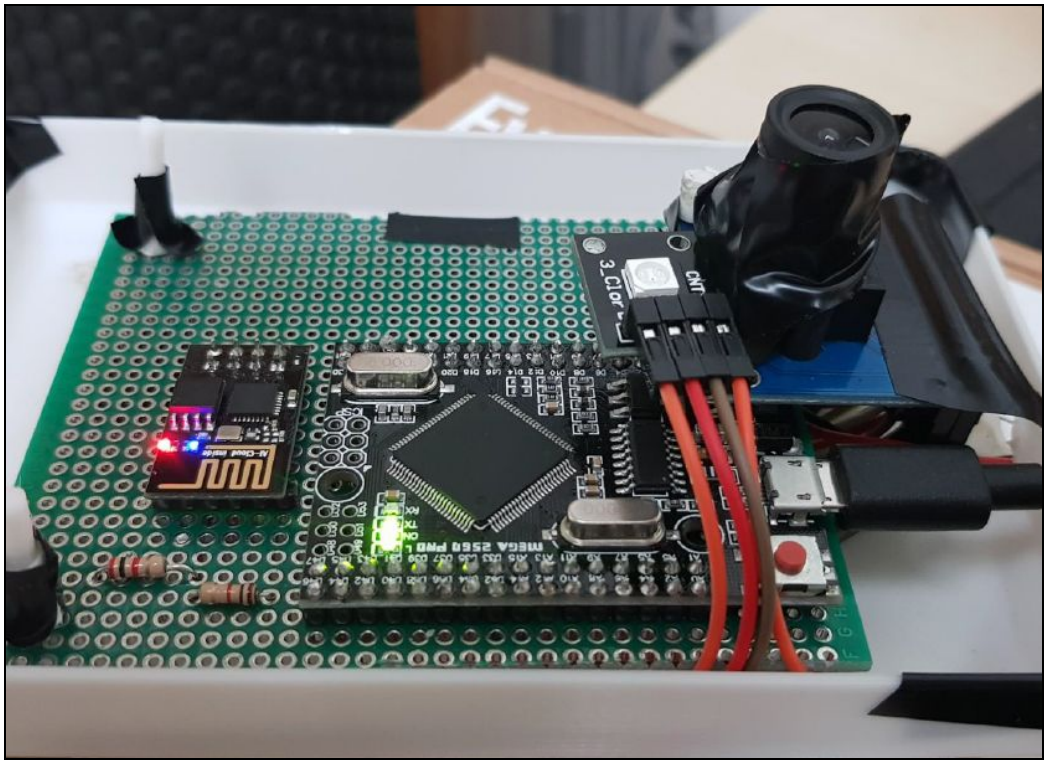


〈 검침모듈 핀 다이어그램 〉



〈 브레드보드를 이용한 작동 테스트 〉

앞서 언급한 내용을 토대로 하드웨어끼리의 핀 다이어그램을 설계했고, 브레드보드를 이용해서 정상작동하는지 확인 해보았다.



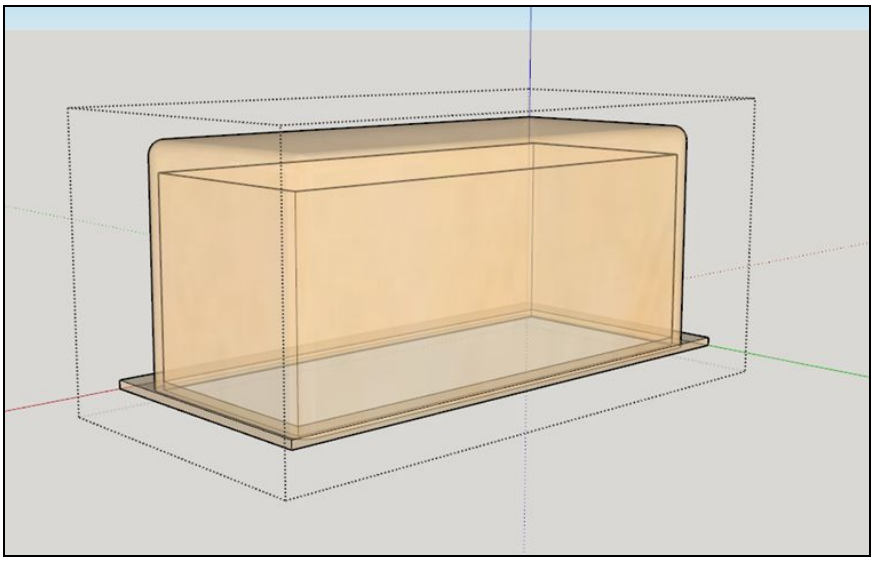
〈 만능기판에 납땜한 모습 〉

이후 크기를 최소화 하기 위해서 만능기판에 납땜을 했고, 역시 정상적으로 동작함을 확인 할 수 있었다.

추가적으로, 우리는 해당 회로를 기반으로 PCB제작까지 하려고 생각을 했었고 실제로 PCB 발주 기업에 문의를 했었으나, 시간과 금전적인 한계로 인해서 PCB까지는 제작하지 못하였다. 하지만 최대한 하드웨어의 완성도를 높히기 위해서 납땜할 수 있는 MCU를 이용했고 만능기판에 직접 납땜을 진행함으로써 생각보다 괜찮은 수준의 완성도를 만족시킬 수 있었다.

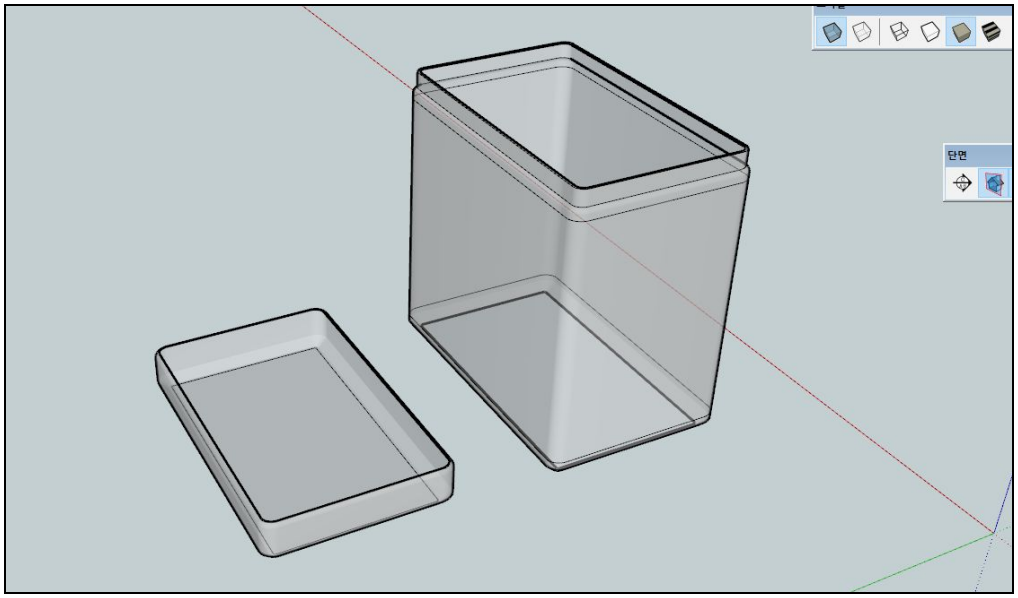
- 하드웨어 (기계) 설계

가스 검침기의 설치된 위치 특성상 실외의 어두운 곳에 설치될 가능성이 높다. 이러한 환경속에서도 각 모듈이 정상작동 할 수 있도록 보호해줄 케이스가 필요했고 어두운 환경속에서도 사진을 찍기 위해서는 LED를 이용한 조도(빛)제어가 필요했다. 따라서, 우리조에서는 케이스를 직접 모델링해서 3D프린팅 하기로 결정했다.



〈 처음 생각했던 하드웨어 케이스 〉

처음에 생각했던 모양은 위와 같다. 한쪽 면만 뚫린 모자모양으로, 닫힌 면에 회로와 카메라를 붙여 사진을 찍는 방식이었다. 하지만, 이러한 설계의 단점으로 검침원이 눈으로 검침값을 확인하기 위해서는 모듈 전체를 탈착해야 하는 불편함이 있었다.



〈 개선한 케이스(모델링) 〉

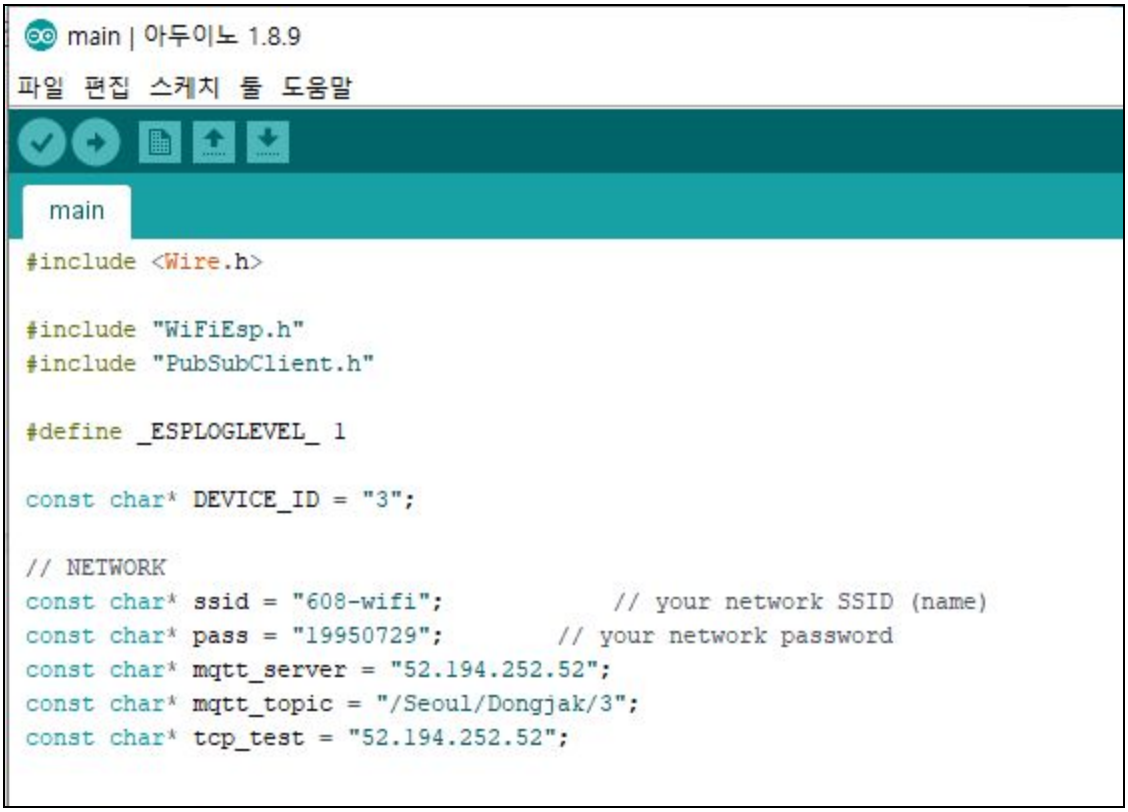
〈 3D프린팅된 케이스 〉

이를 해결하기 위해서 뚜껑과 몸체가 분리된 형태의 케이스 모델링을 진행했다. 실제로 뚜껑 부분에 회로와 카메라를 장착시키고 뚜겅을 닫아 모듈을 완성하는 방식으로 개선했다. 이를 통해 검침원이 검침값을 눈으로 확인하고 싶을 경우 뚜껑만 열면 되는 것이다.

하지만 케이스 설계에 있어서 가장 큰 문제는 모듈의 크기가 크다는 점이었다. 사실 하드웨어의 회로(MCU)는 매우 작은편으로 소형화를 하기에는 충분했다. 하지만 가장 문제가 되는 부분은 카메라의 화각이었다.

우리가 이용한 카메라(OV7670) 모듈의 경우 VGA(640*480) 해상도를 지원하지만 화각이 25° 밖에 되지 않아 검침부의 5자리 숫자를 모두 찍기 위해서는 최소 13cm 이상의 거리를 필요로 했다. 따라서 이를 위해서 모듈의 몸통이 길어질 수 밖에 없었다. 나중에 우리가 만든 모듈을 개선할 수 있는 기회가 있다면, 더 좋은 화각의 카메라를 장착하여 모듈을 소형화 해보겠다.

- 하드웨어 코딩



〈 아두이노IDE를 이용한 하드웨어 코딩, 전체소스는 여기[1]서 확인할 수 있다. 〉

우리가 이용한 MCU의 경우 Mega 2560 Pro로, 정품 아두이노[2]는 아니었다. 하지만 Arduino Mega 2560의 회로도를 사용하고 있어 Arduino IDE를 이용해 소스를 업로드 할 수 있었다.

[1] https://github.com/yourequiremorecitygas/hardware
[2] 아두이노는 회로도가 공개된 하드웨어이므로, 다른 제조사에서 약간 커스텀하여 보드를 생산하기도 한다

하드웨어의 펌웨어를 개발하는데 있어 가장 어려웠던 점은 각 모듈이 제공하는 Mega2560용 라이브러리가 없던 것이었다. 이는 곧 하드웨어의 데이터시트를 참고하여 바닥부터 코딩을 해야 한다는 것이었다. 하지만 다행이도 다른 보드(ESP-8266) 를 위한 라이브러리가 존재하여 해당 라이브러리를 우리가 사용하는 보드(Mega2560)에 맞게 수정하여 사용하였다.

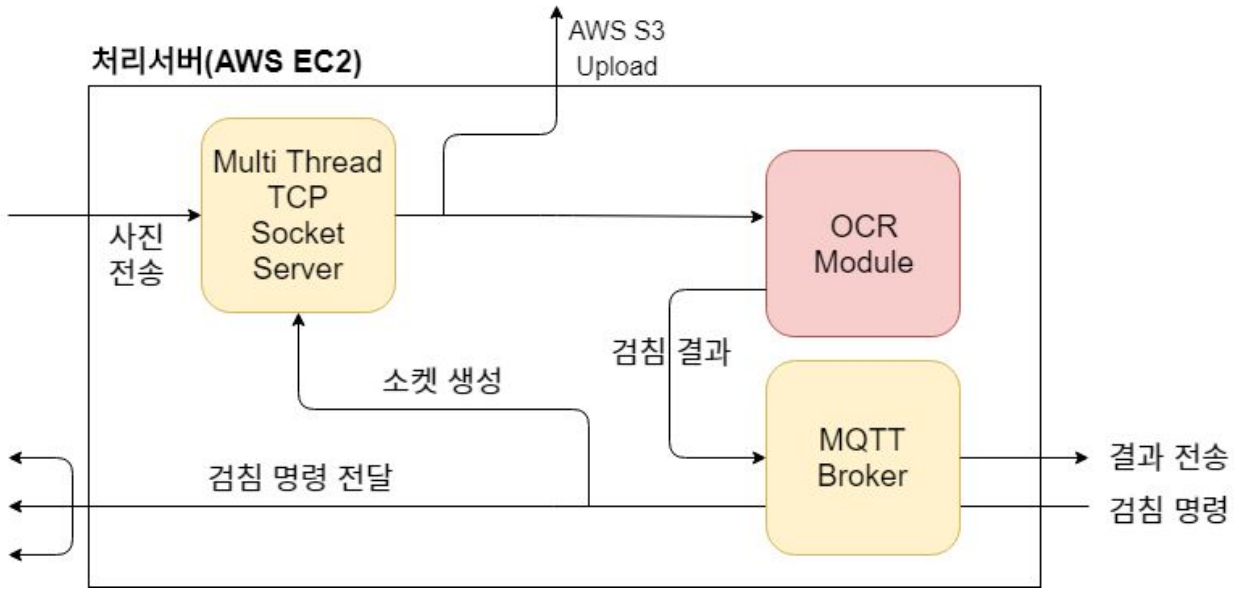
모듈	라이브러리	비고
ESP-01 (WiFi, Mqtt)	WiFiEsp, PubSubClient	ESP8266 보드용 라이브러리를 Mega2560에서 동작하도록 수정 (WiFiEsp)
OV7670 (Camera)	없음	Arduino Mega 에서 작동시킨 Github 소스와 데이터시트 참고하여 Mega2560에서 동작하도록 코딩

일반적인 PC에서 하는 코딩과 하드웨어(임베디드)에서 하는 코딩은 다소 차이가 있었는데, 가장 큰 차이는 아두이노의 성능이 생각보다 좋지 못하다는 점이었다.

우리가 겪은 가장 큰 문제는 카메라로 찍은 사진을 TCP를 이용해 처리서버로 보내는데 있어 한번에 1560byte 이상을 보내지 못한다는 점이었다. 이를 해결 하기 위해서, 1460 바이트 크기의 byte 배열을 선언 후에 매번 1460바이트씩 끊어 보내는 방법으로 구현을 했다. 또한 아두이노의 펌웨어에서 제공하는 소켓이 생각보다 잘 열고 닫히지 않아 소켓과 관련된 에러가 자주 발생 했었다.

하지만 처음해보는 임베디드 코딩 치고는 잘 동작하는 수준의 코드가 나왔고, 한가지 아쉬운 점이 있다면 코드를 깔끔하게 정리하지 못해 코드의 가독성이 많이 떨어진다는 점이다.

(3) 처리 서버

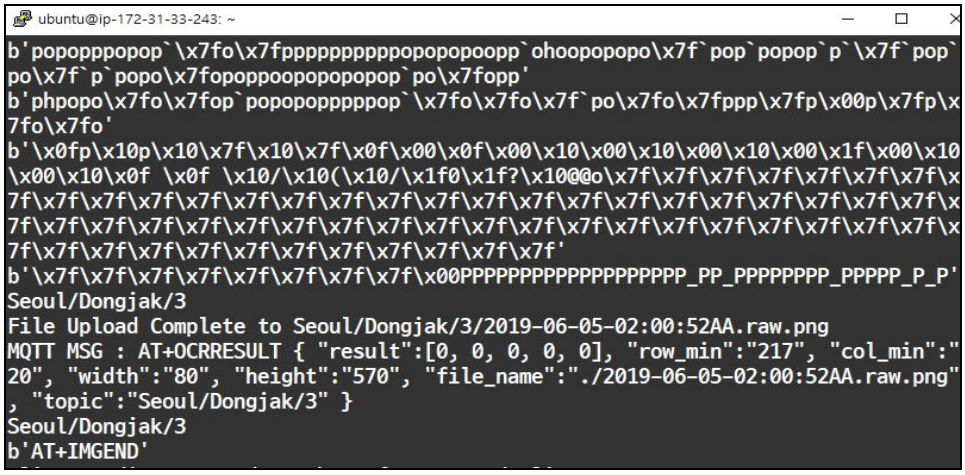


처리서버는 멀티쓰레드 TCP Socket 서버, 각 Thread마다 독립적으로 실행되는 OCR Module 그리고 각 Node마다 할당된 MQTT Topic을 중계해주는 MQTT Broker로 구성되어있다. MQTT Broker은 메세지가 지나는 커다란 통로라고 할 수 있는데, 각 node들은 client로, 특정 Topic의 메세지만 들을 수 있다. 각 Node마다 고유한 topic을 구독하고 있기 때문에 모든 기기에 독립적으로 명령을 전달할 수 있다.

- 하드웨어와 통신

하드웨어와 통신은 Payload의 크기가 작고, 특정 Node만 지정해서 메세지를 보내도 안정적으로 전달되어야 하므로 MQTT Protocol이 적당하다고 판단했다. 그래서 처리서버를 MQTT Broker로 두고, 각 node에 client(수신 기능)와 publisher(발신 기능)을 담당하는 MQTT 모듈을 탑재했다.

하드웨어와 통신은 간단한 명령어를 전달하고 받는 형식으로 진행 된다. 예를 들어, 검침을 하는 경우엔 “AT+START”라는 메세지를 받고, 검침이 끝나면 “AT+IMGEND”라는 메세지를 자신이 구독하고 있는 topic으로 발송한다.



그럼 TCP Server와 node는 Socket통신을 종료하고, OCR을 진행한 후 node가 진행하는 topic에 AWS S3에 업로드 된 파일명과 검침 결과가 담긴 메세지를 전송한다.

- 소켓 통신

node로부터 검침값을 전송 받는 TCP소켓 통신 서버는 처음에 단일 쓰레드 서버로 디자인 되었다. 하지만, 이렇게 하게 되면 python socket server가 기본적으로 제공하는 통신 프로토콜이 blocking(busy-waiting) 방식이기 때문에 한 node가 사진을 전송하는 도중에는 다른 node가 사진을 전송할 수 없다. 그래서 non-blocking 방식으로 사진 전송을 시도해보려고 했지만, 잘 되지 않고 패킷이 깨져서(4KB 패킷 1개씩 손상) 다른 방법을 통해 사진을 전송 받기로 했다.

<(손상된 사진)>

그래서 채택한 방식이 멀티 쓰레드 소켓인데, 파이썬에는 멀티쓰레드와 소켓 통신 라이브러리가 잘 되어 있어서 각 쓰레드마다 Unique한 소켓을 할당하는 작업을 원활하게 할 수 있었다. 따라서 여러 곳에서 파일을

보내도 OCR 모듈이나 AWS S3에 올리는 boto3 library를 탑재한 파이썬 모듈도 독립적으로 작동하기 때문에 안정적인 TCP 통신을 제공할 수 있었다

- 하드웨어로부터 사진 수신 후 저장

하드웨어로부터 TCP Socket 통신을 통해 받은 사진들은, Local Storage에 저장하기엔 너무 양이 많아질 수 있고, 초기의 설계대로 Local Storage에 저장한 파일을 다시 결과서버에(사용자에게 표시하기 위해) TCP Socket 통신을 이용해 전송하는 것은 매우 비효율적이라고 생각했다. 그래서 AWS S3 인스턴스를 하나 더 구축해 그 곳에서 사진 파일을 관리하기로 했다. 그러므로 소켓 통신을 통해 받은 raw파일을 png파일로 컨버팅 한 후 수신시간을 파일명으로 특정 node를 위한 directory에 저장되게 했다. 따라서, 모든 node에 대한 검침 사진을 독립적으로, 시간 순서대로 저장할 수 있는 안정적인 파일 저장 환경을 구축할 수 있었다.

- 개발된 OCR 모듈을 이용하여 사진에서 숫자 추출(OCR 탑재)

소켓 서버에서 전달 받은 파일을 처리하고 업로드 한 후, 검침값을 OCR 모듈로 읽는다. 이는 매우 단순하다. 왜냐 하면, ocr.py에 탑재된 ocr(..) 함수를 실행한 후, 결과값을 json파일로 파싱해서 MQTT topic으로 전송해주면 되기 때문이다.

(4) 결과 서버

- 처리 서버로부터 데이터 수신 확인 및 표출

처리 서버로부터 전송받은 데이터는 각각 검침 결과와 사진 파일로 나뉘어 MQTT, AWS S3으로 전송된다. 결과 서버는 모든 topic(#)을 구독 하고 있고 모든 MQTT 메세지는 그 안에 각각 자신의 topic을 포함하고 있으므로 어느 node에서 보낸 메세지인지 쉽게 구분이 가능하다. 따라서, 각 메세지에 담긴 AWS S3의 파일 디렉토리를 이용해서 mqtt 서버에 있는 사진 파일을 html을 사용해 웹에 표시해준다.

- 관리자용 웹 UI 구현

관리자용 웹 UI는 vue.js를 사용해, 대쉬 보드를 구현했다. 거기에 MQTT 웹 소켓을 탑재해 웹 서버에서 MQTT topic을 구독하면서, 명령을 각 node에 전달할 수 있다.

(5) 광학 문자 인식(OCR)

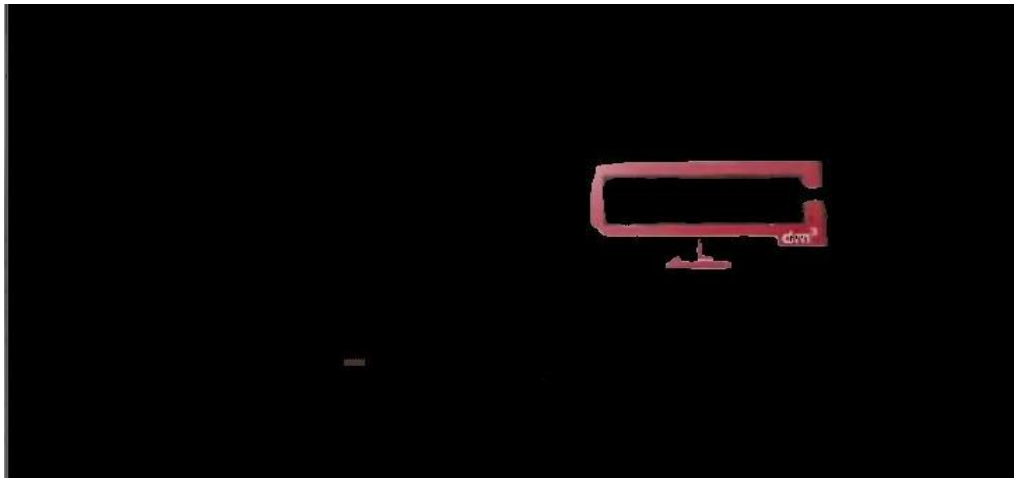
- 데이터 전처리

OCR을 통해서 인식하기 위해, 관심 영역(ROI)를 검출하고, 조명과 노이즈의 영향을 줄인다.

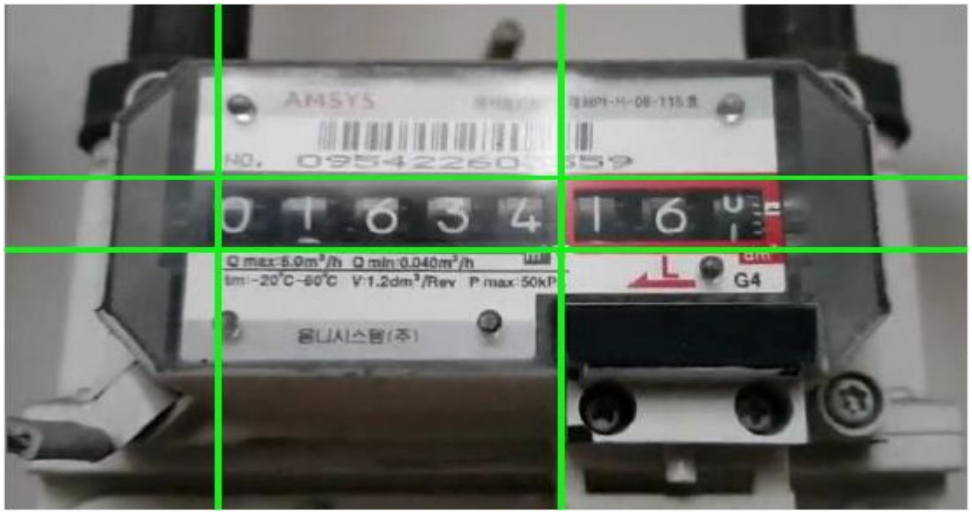
- 관심 영역(ROI) 검출

가스 계량기는 제조사와 모델마다 외형의 차이는 존재하지만, 모두가 공통으로 따르고 있는 규약이 존재한다. 검은색에 하얀 숫자로 된 부분이 실제 검침을 위해 읽어야 하는 부분이고, 그 오른쪽에 있는 빨간 사각형이 있는 부분의 숫자는 읽지 않는 부분이라는 것이다. 가스 계량기의 모델에 관계 없이, 읽어야 할 부분의 위치를 찾기 위해 이 점을 이용했다. 가스 계량기에서 빨간 사각형이 있는 위치를 찾은 후, 이를 이용해 읽어야 할 부분의 위치를 찾아낸다.

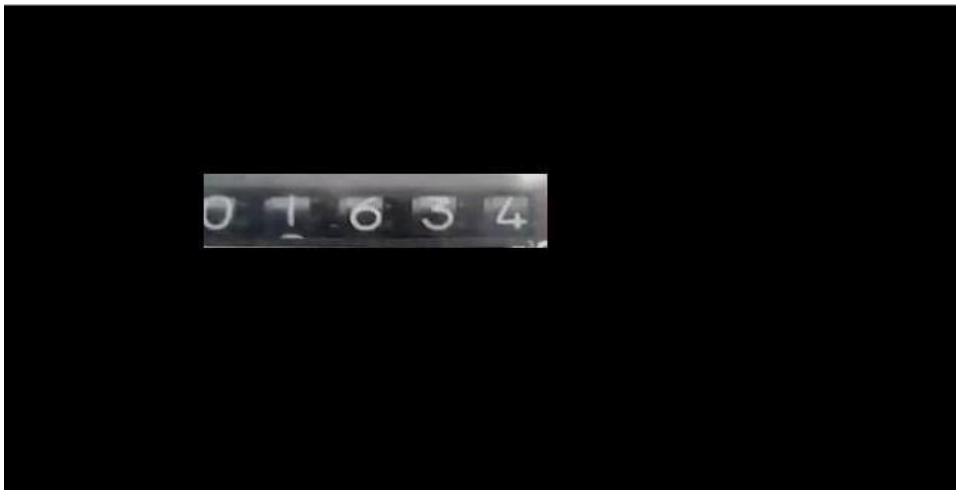
〈테스트에 이용된 사진 원본〉



〈빨간색 영역(읽지 않는 부분)을 검출한 모습〉



〈빨간색 영역을 기준으로 평행선을 그은 모습(개념도)〉



〈관심 영역(ROI)를 검출한 모습(검은색 이미지와 합친 후, 경계선 찾음)〉

- 조명과 노이즈 영향 감소

다양한 방법들을 조합하며 실험(mean 필터링, median 필터링, 가우시안 필터링, 양자화, 이진화(지역 가변 이진화, 전역 고정 이진화), homomorphic 필터링, morphology 필터링)

→ **실험적으로, 양자화와 morphology 필터링을 병행하는 것이 가장 좋았다.**

〈원래 이미지〉

〈지역 가변 이진화 적용(밝기 변화에 유연하지만, 노이즈 제거는 잘 되지 않는다)〉

〈지역 가변 이진화 후 모폴로지 필터 적용(노이즈가 좀 더 제거 된 모습)〉

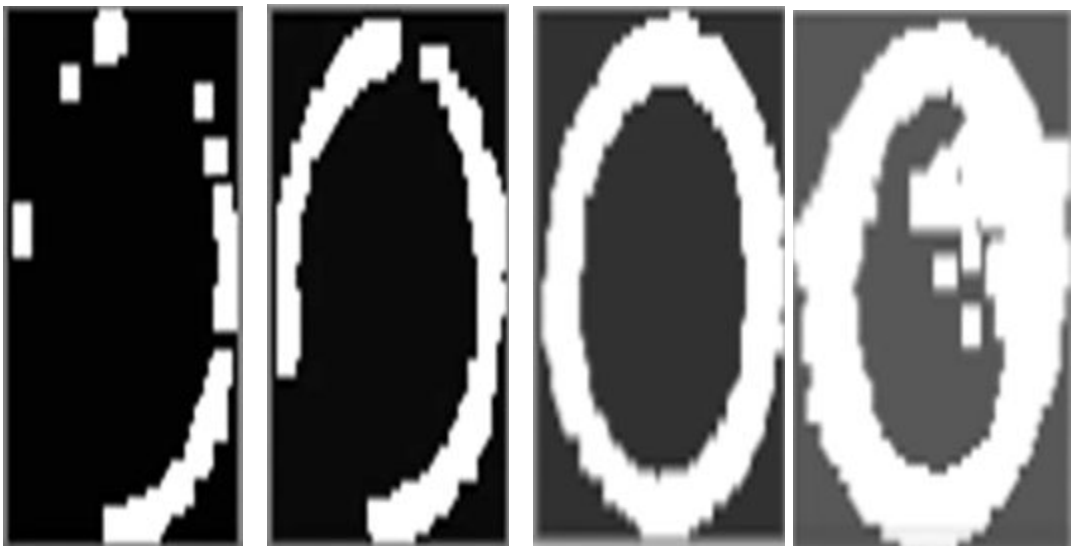
〈전역 고정 이진화 후 모폴로지 필터 적용(노이즈가 깔끔하게 제거된 모습)〉

단점 : 나눌 임계 값 설정을 정하는 것이 어렵다. (값이 낮을 경우 노이즈 제거가 되지 않고, 높을 경우 데이터가 손실된다)

해결책 : 높은 임계값에서 노이즈를 제거시킨 후 위치를 찾고, 임계값을 낮춰 가며 데이터를 복원한다.(이진화와 모폴로지 필터링 동시 적용)

- 검출 영역 정밀화

이진화를 수행하여 노이즈를 제거시킨 후, 해당 이미지에 대해 추가적으로 모폴로지 필터링을 적용하면 자잘한 노이즈는 제거되고, 손실된 데이터가 복구되는 효과가 있다. 이 작업을 높은 임계값에서 수행하면 일부의 데이터와 노이즈가 제거되기 때문에, 위치를 대략적으로 잡을 수 있고, 임계값을 낮춰갈 수록 데이터와 노이즈가 복원된다. 각 digit 영역에서 노이즈인 부분을 휴리스틱하게 제거해나가면서 최대한 digit 영역만 검출하게 만든다.



〈예시1. 높은 임계값 예시2. 중간 임계값1 예시3. 중간 임계값2 예시4. 낮은 임계값〉

(회색으로 보이는 이유는 데모를 위해 찍은 영상에서 캡쳐를 해서 그렇습니다. 원래는 흰검)

- OCR을 사용한 계량기 숫자 인식

- 숫자 영역에 필요한 feature를 제작

〈4에 대한 featrue 예시〉

임계값을 바꿔가면서 모폴로지 필터를 적용하고, 노이즈를 휴리스틱하게 제거하는 과정에서 위와 같은 이미지들이 형성된다. 4의 오른쪽 아래 부분처럼 작게 튀어나온 부분들이 지워져가는 모습을 볼 수 있다. 두꺼운 이미지, 얇은 이미지 등을 각 숫자에 대해 전처리 과정에서 얻어지는 중간 단계의 이미지들을 획득한다.

- 휴리스틱 적용

이미지 마다 노이즈의 영향이 다르기 때문에 이러한 변수를 제어하기 위해 관찰 결과를 토대로 휴리스틱을 적용했다.

A. 모든 digit에 대해 특정 높이 이하가 되면, 원본 데이터가 많이 상실된 순간이었기 때문에 인식하지 않는다.
B. 1이라고 인식하는 경우를 제외하고 digit에 대해 특정 너비 이하면 마찬가지로 원본 데이터가 많이 상실된 순간이므로 인식하지 않는다.
C. 전체 이미지에 대해 임계 값 이상의 비율이 검은색일 경우, 마찬가지로 원본 데이터가 많이 상실된 것이므로 인식하지 않는다.
D. 모든 digit에 대해 유사도가 임계 값 이하일 경우, 온전히 그 숫자라고 하기 힘들기 때문에 인식하지 않는다.
E. 각 단계에서 digit 영역 정밀 검출을 위해 노이즈라고 인식할 범위는 실험적으로 임계 값을 얻었다
F. 모폴로지 필터링의 커널 크기, 이터레이션 횟수 등의 파라미터 역시 실험적으로 임계 값을 얻었다. digit image가 평균적으로 너비 60px, 높이 80px 정도였는데, 이 경우에 대해 kernel의 크기를 (2,2)를 사용하고 iteration 횟수를 3으로 사용하는 경우가 데이터 손실이 적으면서도, 노이즈 제거가 가장 우수했다.

- 유사도 측정

각 digit들(높은 이진화 임계값 부터 임계값을 낮춰 가는 각 단계의 digit)을 feature들에 대해 유사도를 측정한다. 유사도 측정은, L2-norm, L2-norm square, L1-norm, SSIM, mahalanobis distance,

minkowski distance, cosine similarity 등을 이용해봤고, 그 중 L1-norm을 이용했을 때 가장 성능이 우수했다. 각 단계의 digit들을 가장 유사도가 높은 숫자로 mapping 해두고, 모든 단계가 끝나면 확률적으로 가장 높은 숫자를 결과로 반환한다.

- 학습 데이터 수집 및 전처리

- 학습 데이터 수집

초기에는 실제 이미지들을 직접 찍으러 돌아다녔지만, 시간이 경과한 후, 딥러닝 모델을 사용하지 않게 되면서 이 때 찍었던 사진들을 별로 쓰지 못했다. 관심 영역 설정을 평행하게 찾기 때문에 각도의 변화에 대해 유연하지 않다. 그래서 최종적으로는 과거에 찍은 사진들 중 각도가 맞는 사진만 우리 모듈의 화각에 맞는 범위를 잘라서 사용했다.

- 데이터 전처리

초기에는 테서렉트나 딥러닝 모델을 사용할 것이라고 판단하여 데이터 전처리 과정이 필요했으나, 추후 딥러닝 모델을 쓰지 않게 되면서 전처리가 필요하지 않게 되었다. 툴을 사용해서 직접 레이블링을 해줬었다.

〈레이블링을 하여 학습 데이터 전처리를 해준 모습〉

- OCR 결과

- 실제 데이터에 대한 인식 결과

데이터	결과	데이터	결과	데이터	결과
00148	00148	00501	00301	00666	00636
00222	00222	00506	00006	00686	00686
00290	00290	00612	00612	00734	00730
00392	00392	00615	00613	00961	00961
00482	00482	00660	00660	01261	01061

이상적인 조건(각도, 위치 등)을 맞춘 상태에서는 정확도 92%

→ 실제 데이터를 촬영할 때 조명의 영향을 생각하지 못 하고 찍었기 때문에 정확도가 떨어지는 원인이 되었다. 또한, 페인트가 튀어 있는 등 노이즈가 더 다양하게 존재했다.

- 가지고 있는 검침기로 직접 얻은 데이터에 대한 인식 결과

이상적인 조건(각도, 위치 등)을 맞춘 상태에서 정확도 100%를 얻었다. (00003 ~ 00049까지 총 47장(235개의 digit)의 이미지에 대해 실험) 에 대해 실험하였으며, 밝기는 사람이 눈으로 읽기도 어려울 정도로 어두운 이미지가 아니라면 모두 인식했다. (다양한 밝기에 대해 테스트를 진행했다.)

- 이상적인 조건이 아닌 경우(각도나 위치가 맞지 않는 경우)

각도가 맞지 않는 경우, 관심 영역 검출을 빨간 선에 대해 수평으로 찾기 때문에 영역 검출을 제대로 하지 못해 인식을 하지 못 했다.

위치가 맞지 않는 경우, 관심 영역 검출을 위해 빨간 선을 이용하는데, 그 부분을 찾지 못해 관심 영역 검출에 실패하여 인식을 하지 못 했다.