졸업프로젝트 보고서

-Petsitter(강아지 케어 시스템)-

12050054 임운택
12050034 오세준
15050064 한희연

Abstract

바빠지는 현대 사회 속에서 반려동물을 키우는 인구가 늘어날 것으로 예상된다. 그에 따라 반려동물을 키우는 사람들의 고민과 요구가 다양해 질 것이다. 반려동물의 영양 상태에 관심을 가지는 인구가 늘어날 것이고, 반려동물을 돌보기 어려운 인구 또한 생겨날 것이다. Petsitter로 명명된 이번 프로젝트에서는 Arduino를 기반으로 자동 사료 급식 기기를 개발하고, Web page에서 반려동물의 상태를 파악할 수 있게 한다. 반려동물에게 필요한 영양을 적시에 공급하여 건강을 유지하고, 주인은 언제 어디서나 상태를 파악할 수 있을 것이다.

1. 서론

세계미래학회는 '21세기 10대 미래 전망'에서 "세계 인구는 2035년부터 증가세를 멈추는 대신, 반려동물 수가 급증하게 될 것" 이라고 말한다. 그에 따라 '가족의 일원'으로 반려동물을 기르는 인구가 늘어나면서, 그를 키우는 사람들의 고민과 니즈도 다양해지고 있다. 반려동물의 영양 상태에 관심을 가지는 인구가 늘어날 것으로 예상된다. 또한 바빠지는 현대 일상 속에서 반려동물을 돌보기 어려운 인구도 늘어나고 있다. Petsitter로 명명한 이번 프로젝트에서는 이러한 고민과 문제점을 해결하기 위한 방법으로 자동화 된 사료 급식 시스템을 개발한다. 반려동물의 식사량과 몸무게를 지속적으로 감지하여 별도의 웹페이지에서 체크할 수 있게 한다. 반려동물의 필요 섭취량을 계산하여 매일 두 번 사료를 공급한다.

2. 배경지식

1) Arduino

Arduino는 오픈 소스 전자 기술 플랫폼으로 쉬운 하드웨어와 소프트웨어 사용을 기반으로 한다. Arduino board로 전기 입력을 받거나 출력을 내보낼 수 있다. 다수의 스위치나 센서로부터 값을 받아 연결 된 외부 기기를 통제한다. 별도의 통합개발환경을 제공하여, 소프트웨어 개발을 할 수 있다.

2) MQTT

구독/발행 작동방식을 기반으로 한 경량화 된 메시지 프로토콜이다. 기기간/IoT 통신을 위한 프로토콜로 낮은 대역폭이나 높은 지연시간, 낮은 신뢰성의 네트워크 환경을 극복하기 위하여 디자인 되었다. 구독/발행 방식의 특성으로 인해 다수의 기기 연결에 유리하다. 2013년 OASIS에 표준으로 제정되었고 지속적으로 업데이트 되고 있다.

3) Nodejs

Google chrome 웹 브라우저의 Javascript 렌더링 엔진으로 개발 된 V8 engine에서 동작하는 Javascript 런타임이다. Non-blocking I/O와 Single thread event loop를 통하여 높은 처리 성능을 가지고 있다. 또한 네트워크 라이브러리를 내장하고 있어 주로 네트워크 프로그래밍에 사용된다. NPM(Node package manager)이라는 프로젝트 관리 툴을 사용하여 다른 개발자가 구현한 라이브러리를 직접 자신의 프로젝트에 쉽게 사용할 수 있는 장점이 있다.

4) Mysql

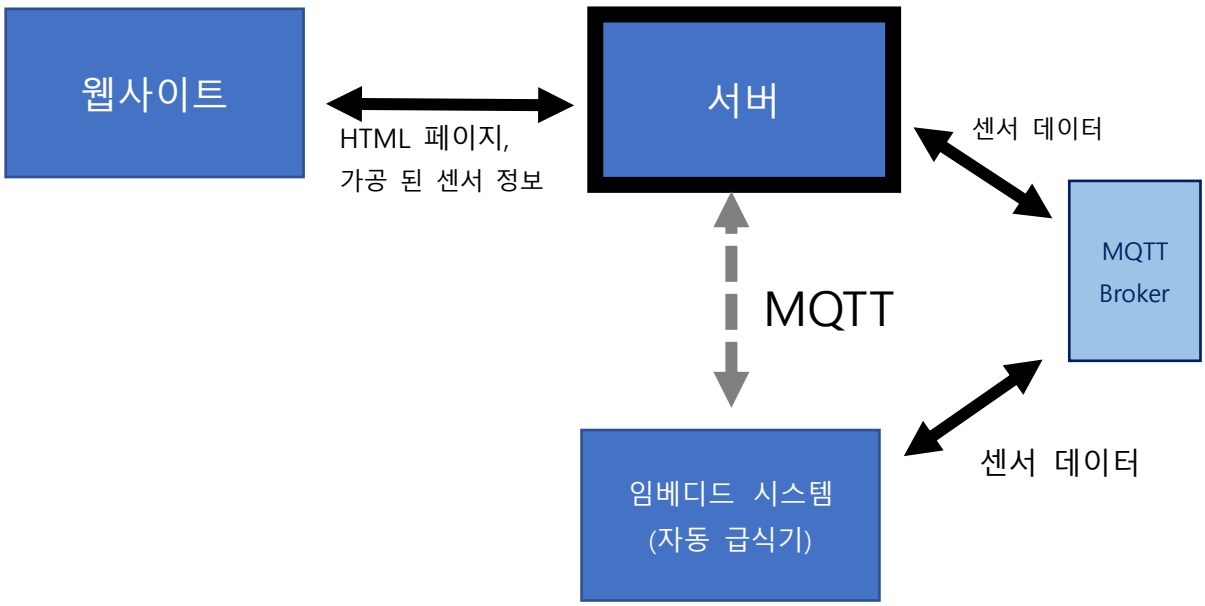
처음에 오픈소스 관계형 데이터베이스로 개발, 관리 되었다. ACID 특성을 지원하기 때문에 데이터의 무결성과 신뢰성을 유지할 수 있다. 스키마가 고정 되었기 때문에 구조 변화에 어려울 수 있다. 현재는 Oracle에서 관리하고 있다.

5) Couchbase Server

JSON 모델을 사용한 스키마의 제약이 없는 NoSQL 데이터 베이스이다. 인덱스를 사용하고, 관계를 사용하지 않기 때문에 데이터 처리 속도에 큰 이점을 가지고 있다. 그리고 이러한 특성으로 인해 데이터베이스의 수평적 확장에 유리하고, 별도의 서버 장애 처리 매커니즘을 가지고 있어 높은 유효성(Availability)를 가지고 있다. 또한 N1QL 이라는 Couchbase 만의 SQL과 흡사한 질의 언어를 지원하기 때문에 기존 관계형 데이터베이스처럼 사용 할 수 있는 장점을 가지고 있다.

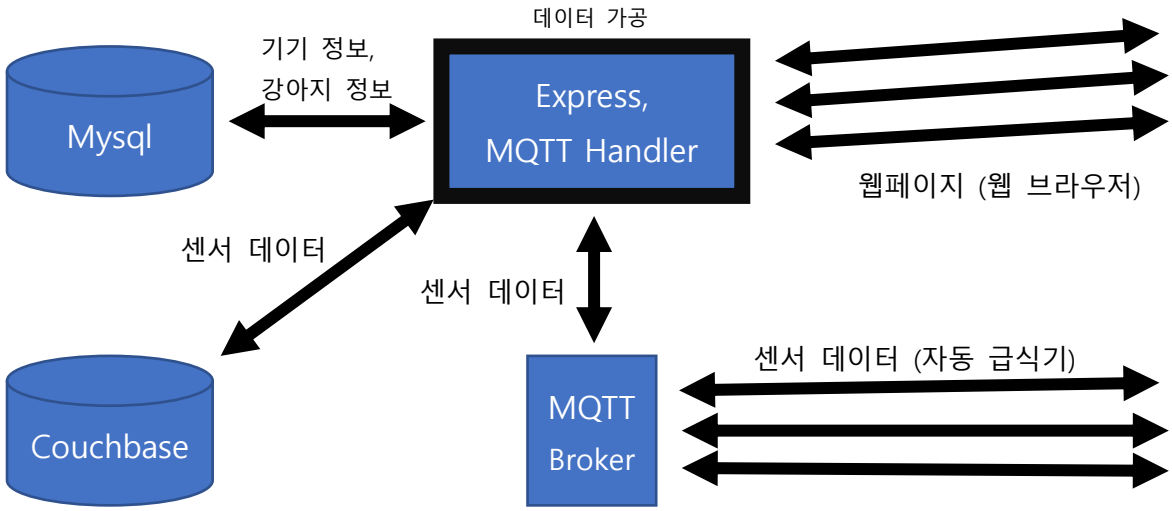
3. 구현 시스템

1) 개괄



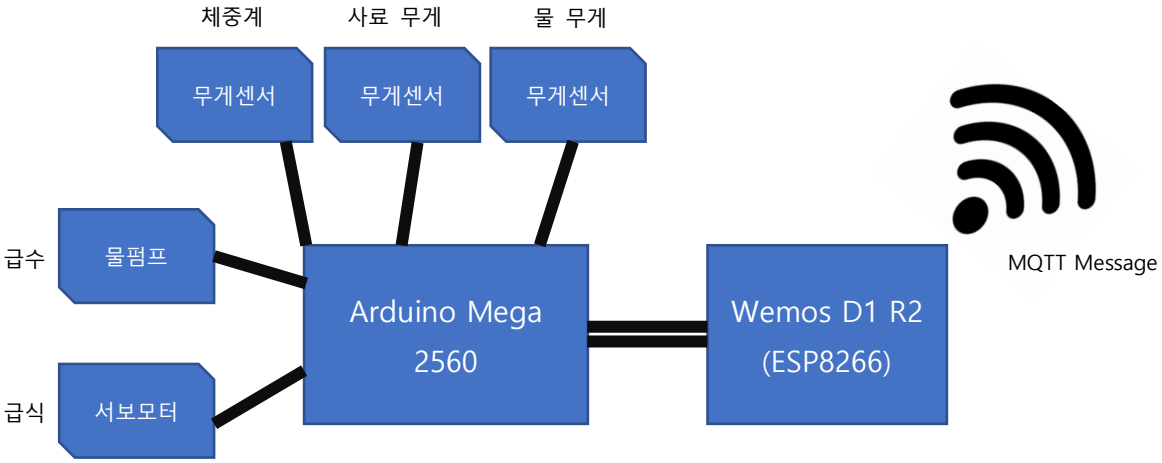
서버와 자동 급식기는 MQTT 프로토콜로 센서 데이터와 통제 명령을 주고 받는다. MQTT 메시지를 구독/발행 하기 위해 중간에 MQTT Broker를 둔다. MQTT Broker로는 Mosca를 사용한다. Mosca는 적은 리소스로 실행되며 최신 프로토콜을 지원한다. 서버는 웹사이트 접속자에게 HTML 문서와 데이터베이스에 축적된 센서 데이터를 바탕으로 반려동물의 현재 상태를 볼 수 있게 한다. 웹사이트를 제공하기 위하여 Node.js의 웹 애플리케이션 프레임워크인 Express를 사용한다.

2) 서버



기기는 인터넷에 연결되면 자동으로 서버에 등록이 되며 Mysql 데이터베이스에 저장된다. 그리고 사용자는 웹사이트를 통해 자신의 기기의 일련번호를 등록 후, 반려동물을 등록 하면 반려동물의 정보가 Mysql에 저장이 되며 기본적인 정보를 웹사이트에서 볼 수 있다. 사료 급식기로부터 읽어 들이는 센서 데이터는 모두 Couchbase에 저장이 되며, 서버는 저장 된 센서 데이터를 바탕으로 사료량을 계산 후 MQTT Broker를 통해 기기를 통제한다. 그리고 웹사이트 접속자는 센서 데이터를 통해 일 주일간의 몸무게, 칼로리 섭취량, 물 섭취량 등을 알 수 있다.

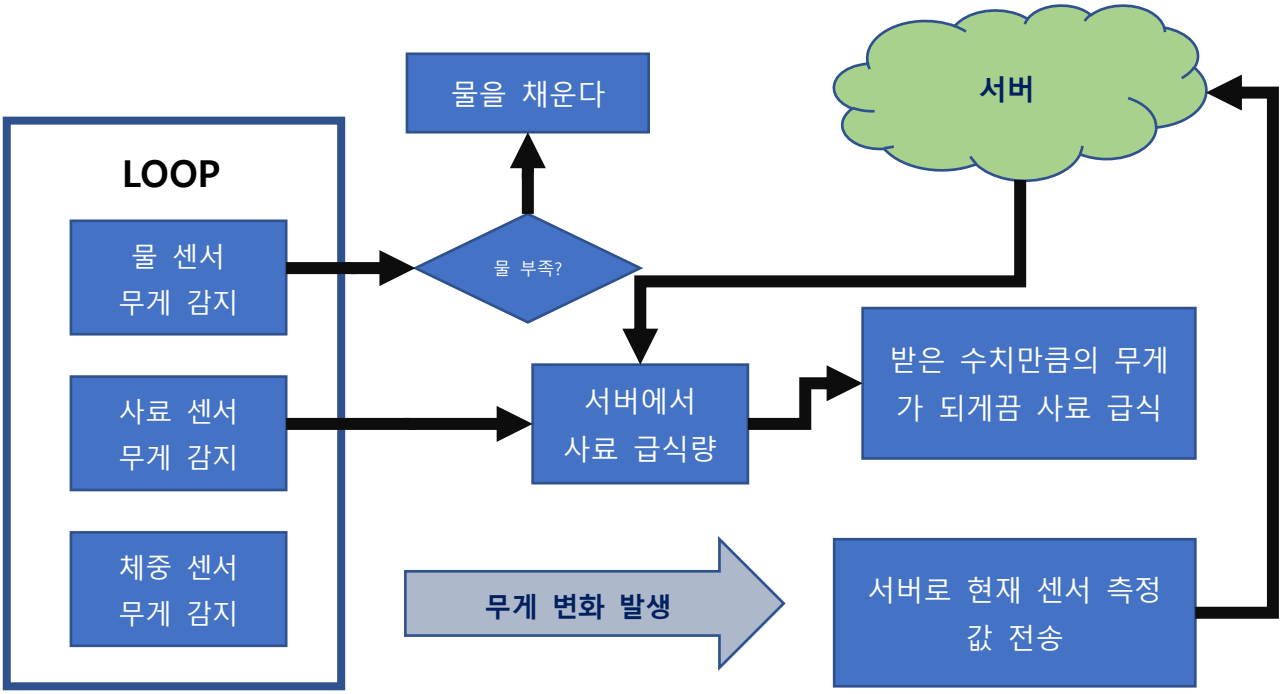
3) 임베디드 시스템(자동 급식기)



Arduino board로는 Arduino Mega 2560과 Wemos D1 R2를 사용한다. 장착 된 외부 기기들이 입출력을 많이 사용하는데, 성능이 우수하고 사용 가능한 포트가 많은 Arduino Mega 2560에 모든 장치를 연결 하고, WiFi를 이용해 서버와 통신을 위하여 Wemos D1 R2에 내장 된 ESP8266 칩셋을 사용한다. 이 두 기기는 서로 유선으로 연결 되어있다. 그림과 같이 반려동물의 몸무게를 알 수 있는 무게 센서와, 사료와 물의 양을 알 수 있는 무게 센서를 사용한다. 변화하는 센서 데이터를 지속적으로 서버로 보내고 서버가 보내오는 통제 명령을 받아 물펌프나 서보모터를 조종해 물과 사료를 제공한다.

4. 구현 방법

1) 알고리즘



시스템이 동작하는 과정을 나타낸 플로우 차트이다. 사료 급식기에 부착 된 3가지 무게 센서는 무한 루프를 돌며 지속적으로 무게를 감지한다. 모든 무게 감지 센서는 무게의 변화를 감지하면 센서 데이터를 서버로 전송하고 데이터는 서버 데이터베이스에 저장된다. 첫 번째로 물 무게의 감소가 감지 되면, 곧바로 물 펌프를 작동해 적당한 수치까지 물을 채운다. 두 번째로 사료 무게의 감소가 감지 되면, 센서 데이터를 서버로 보내고, 서버에서는 가장 최근에 받은 사료 무게 센서 데이터를 바탕으로 앞으로 공급해야 할 사료량을 조절한다. 서버에서는 하루 두 번, 계산 한 사료량을 수치로 하여 사료 급식기에 전송하며 이를 전해 받은 사료 급식기는 서보 모터를 작동하여 사료 통의 입구를 열어 사료 공급 후 정해진 양이 채워지면 입구를 다시 닫는다. 이러한 과정이 계속적으로 반복하여 진행된다.

2) 데이터베이스 구성

본 프로젝트에서 사용 한 데이터베이스는 Mysql, Couchbase Server로 Docker Container의 형태로 실행한다. Docker는 가상 환경으로써, 응용 소프트웨어를 소프트웨어의 실행에 필요한 모든 것을 감싸는 파일 시스템이다. 각각의 소프트웨어는 Image의 형태로 배포되며, Container의 형태로 실행 된다. 이는 소프트웨어가 실행중인 환경에 관계 없이 언제나 동일하게 실행될 것을 보증한다. 분산 시스템의 구성이나 빠른 설치와 실행에 이점을 준다. 첫 째로 Mysql의 테이블로 Device, Dog, Breed 총 3가지 테이블이 존재한다. DDL(Data definition language)로 나타내면 다음과 같다.

```
CREATE TABLE `tbl_device` (
  `serial` varchar(255) NOT NULL,
  `ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `registered` tinyint(1) DEFAULT '0',
```

```
  PRIMARY KEY (`serial`)
)
CREATE TABLE `tbl_breed` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  `minweight` float NOT NULL,
  `maxweight` float NOT NULL,
  PRIMARY KEY (`id`)
)

CREATE TABLE `tbl_dog` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `name` varchar(100) NOT NULL,
  `birth` datetime NOT NULL,
  `breed` int(11) NOT NULL,
  `device_serial` varchar(255) NOT NULL,
  `comment` varchar(100) DEFAULT NULL,
  `picture` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `device_serial` (`device_serial`),
  KEY `breed` (`breed`),
  CONSTRAINT `tbl_dog_ibfk_1` FOREIGN KEY (`device_serial`) REFERENCES `tbl_device` (`serial`),
  CONSTRAINT `tbl_dog_ibfk_2` FOREIGN KEY (`breed`) REFERENCES `tbl_breed` (`id`)
)
```

Device 테이블은 기기가 인터넷에 연결 되었을 때 서버에 등록을 위한 테이블이다. 기기의 일련번호와 등록 날짜, 사용자가 웹사이트에서 기기를 사용하고자 하는지에 대한 registered 항목이 있다.

Breed 테이블은 반려동물의 종에 관한 정보가 저장 되어있고, 종의 적정 체중에 관한 정보가 저장 되어 있어 사용자는 본인의 반려동물의 체중이 적정 수치인지를 파악할 수 있다.

Dog 테이블은 사용자가 등록한 반려동물의 정보가 저장 된다.

두 번째로 센서데이터의 JSON 데이터 형식은 다음과 같다.

```
{
  "device_id": String,
  "meal": Integer Number,
  "water": Integer Number,
  "weight": Float Number,
  "time_stamp": Datetime Millis
}
```

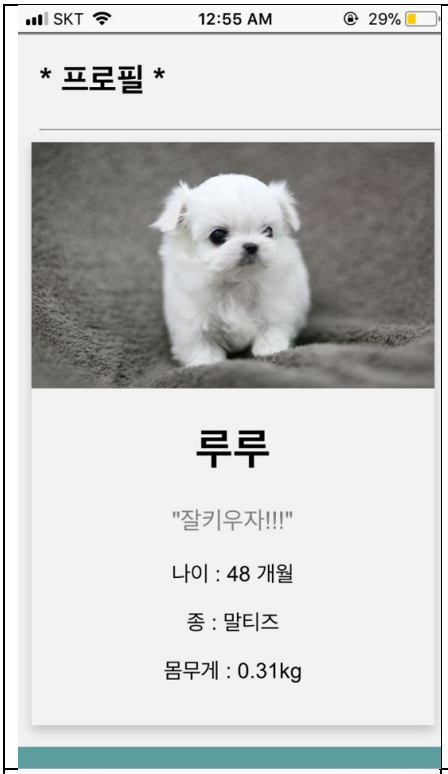
센서 정보를 읽어 들인 일련번호는 device_id에, 남은 사료량, 물량, 현재 체중이 각각 meal, water, weight에, time_stamp에 센서 데이터 감지 시간이 저장된다.

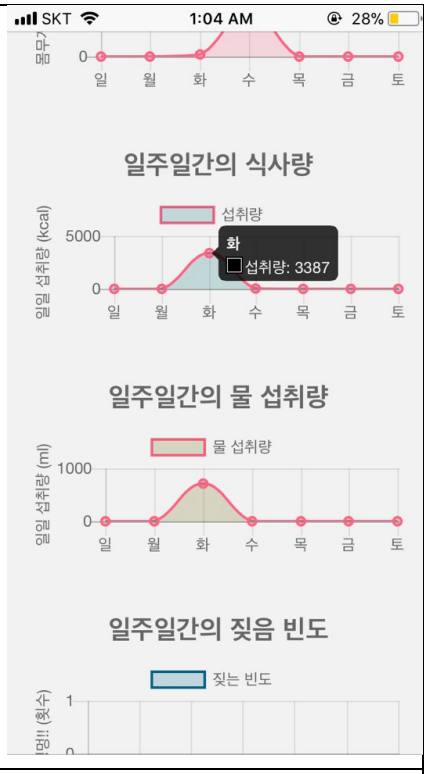
5. 실행 결과

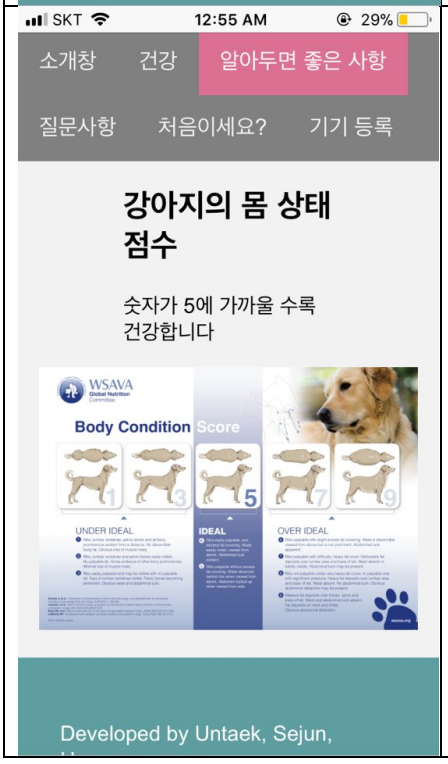
1) 자동 급식기

위 그림은 자동 급식기의 모습이다. 기본적인 급식기의 틀은 시중에서 판매하는 제품을 개조하여 만들었다.

2) 웹 페이지







스마트폰 사용자를 웹사이트를 반응형 웹페이지로 만들어 스마트폰 화면에 최적화 하였다. 등록된 반려동물의 프로필을 볼 수 있고 데이터베이스에 저장 된 센서 데이터로부터 일 주일간의 그래프를 보여준다. 또한 반려동물의 건강 상태를 알 수 있게 도와주는 정보도 제공한다.

3) 시연 연상

https://youtu.be/oUX70TAJJUU - 기계구현영상

https://youtu.be/2Xk7yQ5ae0k – 웹사이트영상

6. 결론

개의 몸무게를 바탕으로 사료량을 조절하는 것을 핵심 기능으로 하여 개발을 진행하였다. 부족했던 점과 한계점이 다수 존재한다고 판단한다. 이번 프로젝트를 진행하면서 미숙하거나 부족했던 점으로는 임베디드 시스템(사료급식기) 개발 시 부품 선택 검토의 미숙함으로 인해 제작 기간이 딜레이 되었던 점이다. 본 프로젝트의 한계점으로는 첫 째로 시중에서 실제로 판매한다고 가정하였을 때 약 2018년 6월 기준 약 12만원이 소요되어 가격이 비싸다고 소비자들이 가격이 비싸다고 판단할 수 있을 것이라는 것, 두 번째로 사료마다 모두 칼로리가 다른데 사용자가 따로 설정할 수 없고 단순히 반려동물의 몸무게로 사료량을 판단하는 점, 마지막으로 기기의 반응 속도와 무게 측정의 오차로 인해 정확한 사료 급식이 어렵다는 점이 있다. 이러한 단점을 개선한다면 충분히 시중에도 효과가 있을 것이라고 예상한다. 문제점을 해결하고 추가적인 기능을 구현해 점점 개선해 나갈 것이다.