

스마트 IoT 도마뱀 사육장

< LIZH >



KOSTA 225기

강사명: 이재성

제출일: 2021.12.10

참가인원: 김강현, 박성범, 박지호,

송연석, 오정민, 조석훈

목 차

Ⅰ. 서론 8

1. 프로젝트 개요 8
   1. 개발동기 및 목적 8
   2. 역할 분담 9
   3. 개발 일정 10
   4. 개발 환경 10

Ⅱ. 본론 11

1. 기능 정의 11

2.1 기능 개요 11

1. 인식 및 제어 13
   1. Logic Process 13
   2. 온/습도 14
      1. 온/습도부 하드웨어 14
      2. 문제 해결 17
   3. 급수기 18
      1. 급수 제어 18
      2. 수위 조절 19
      3. 문제 해결 20
   4. 급여기 22
      1. 급여기 하드웨어 22
      2. 급여 제어 22
      3. 문제 해결 23
2. Server 25
   1. Logic Process 25
   2. 서버 프로그램 개요 25
      1. 서버 프로그램 구상안 25
   3. 기능정의 28
      1. 서버 프로그램 구성 28
      2. 변수 및 통신 프로토콜 31
      3. 프로그램 실행 시작 32
      4. Pi to Server 33
      5. Android to Server 41
      6. 서버 프로그램 종료 43
3. 어플리케이션 44
   1. Logic Process 44
   2. Socket 구성 44
   3. Layout 구성 및 속성 46
   4. App 프로그램 구성 47
4. 카메라 58
   1. 카메라 하드웨어 58
   2. 카메라 기능개요 60
   3. 카메라 스트리밍 및 결론 61

Ⅲ. 결론 63

Ⅳ. 참고문헌 64

표 목차

[표 1] Fan 스펙 15

[표 2] 가습기 모듈 스펙 16

[표 3] 4채널 릴레이 모듈 스펙 16

[표 4] 워터펌프 스펙 19

[표 5] 수위센서 스펙 20

[표 6] 분석 결과 39

[표 7] 파이 카메라 상세 사양표 59

그림 목차

[그림 1] 반려동물이 집에 혼자 있는지 여부와 시간 9

[그림 2] 기능 개요 11

[그림 3] 온습도 자동 제어 11

[그림 4] 온습도 수동 제어 12

[그림 5] 자동 급여, 급수 12

[그림 6] 수동급여 12

[그림 7] 센서 Logic Process 13

[그림 8] 온/습도 센서(DHT 11) 14

[그림 9] 온/습도 센서(DHT 11) 스펙 14

[그림 10] 스팟 램프 15

[그림 11] 스팟 램프 스펙 15

[그림 12] USB 쿨링 Fan 15

[그림 13] 가습기 모듈 16

[그림 14] 4채널 릴레이 모듈 16

[그림 15] Timing Diagram(HIGH, LOW신호) 메커니즘 18

[그림 16] 워터펌프 19

[그림 17] 수위센서 20

[그림 18] 급양기 하드웨어 22

[그림 19] 스텝모터 구동원리 23

[그림 20] 스텝모터 23

[그림 21] 변경 전 24

[그림 22] 변경 후 24

[그림 23] Server Logic Process 25

[그림 24] 서버 최초설계 25

[그림 25] 서버 1차 수정 설계 26

[그림 26] 서버 최종 설계 27

[그림 27] 서버 메뉴와 상태 표시줄 29

[그림 28] 온습도 메뉴 29

[그림 29] 설정 메뉴 30

[그림 30] 변수 목록 31

[그림 31] 통신 프로토콜 32

[그림 32] Pi to Server 개요도 33

[그림 33] Temperature 테이블 디자인 35

[그림 34] Temperature 테이블 SQL문 35

[그림 35] Moisture 테이블 디자인 36

[그림 36] Moisture 테이블 SQL문 36

[그림 37] InsertDB 함수 37

[그림 38] 온습도 데이터베이스 조회 37

[그림 39] 온도 데이터베이스 팝업창 38

[그림 40] 습도 데이터베이스 팝업창 38

[그림 41] 온도, 습도 데이터 일부 39

[그림 42] 위험 경보 40

[그림 43] 온습도 설정 40

[그림 44] Android to Server 개요도 41

[그림 45] Android Logic Process 44

[그림 46] Socket Read&Write 45

[그림 47] MainActivitiy 47

[그림 48] SplashActiviy 49

[그림 49] TemActivity 50

[그림 50] EatActivity 51

[그림 51] 생명주기 개요도 52

[그림 52] 생명주기 메서드 53

[그림 53] CCTVActivity 54

[그림 54] SettingActivity 55

[그림 55] AscenterActivity 56

[그림 56] PiCameraV2\_1 58

[그림 57] PiCameraV2\_2 58

[그림 58] 피사계심도 응용 60

[그림 59] OpenCV 응용: 크기 추정 60

[그림 60] 배경 차분 방식 예제1 61

[그림 61] 배경 차분 방식 예제2 61

[그림 62] 스트리밍 샘플 스크린샷 62

[그림 63] 카메라 스레드 자동 중단 기능 62

**Ⅰ. 서 론**

1. ***프로젝트 개요***
   1. 개발동기 및 목적

‘나는 편한데, 우리집 뚱이는?’

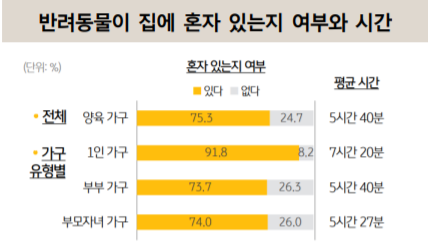
IoT 기반의 도마뱀 사육장 프로젝트는 이 의문으로 시작되었다. 기술의 진보에 따라 우리 주변의 많은 것들이 변화해 왔는데, 그 중 하나인 주거 환경의 변화는 사람들에게 더 크게 다가온다고 할 수 있다. 사람들은 이제 온기 없는 방 안에 들어서지 않아도 되며, 깜빡 잊고 끄지 못한 보일러를 확인하기 위해 집으로 다시 들어가지 않아도 된다. 냉장고 화면으로 장을 보고, 원격으로 전등을 제어하는 일이 가능한 것이 작금의 세상이다. IoT 기술에 기반한 스마트 홈은 우리의 삶을 더 편안하게 만들어 주고 있다.

그러나 ‘스마트 사육장’이라는 단어는 꽤나 이질적이다. 반려동물 관련물품의 시장규모는 증가세를 보여 왔지만, 아직 반려동물을 위한 사육장에 IoT 기술이 적용된 사례는 찾기 힘들었다. 핵가족화의 심화로 반려동물이 가족으로 받아들여지는 경우가 많아지는 가운데, 이들을 좀 더 편안하게 해줄 수 있는 집이 필요하겠다는 생각에 반려동물을 위한 스마트 홈을 개발하고자 하였다.

개나 고양이와 같이 자유로이 자신의 활동반경을 조정할 수 있는 동물보다는 도마뱀이나 거북이처럼 사육장 내부에서 생활하는 파충류에게 이와 같은 사육장이 더 필요할 것이라 생각했고, 그 중에서도 사육을 위해 온도와 습도를 조절해 줄 필요가 있는 도마뱀이 프로젝트 진행을 위해서 가장 적합하다고 판단하였다.

외온성 변온동물인 도마뱀의 특성상 주위 환경 및 온도변화에 매우 민감하다. 반려동물로 도마뱀을 키울 때 불가피하게 며칠간 집을 비워야 할 상황이 온다면, 도마뱀이 원하는 환경을 맞춰주지 못할 가능성이 높다. 도마뱀 사육자에게 가장 큰 불편함이 될 이런 부분을 해소할 수 있는 스마트 사육장을 개발하는 것이 목적이다.

• 개발 필요성

 [그림 1] 반려동물이 집에 혼자 있는지 여부와 시간

KB 금융그룹 경영연구소가 조사한 바에 따르면, 양육가구의 75.3%가 반려동물을 집에 혼자 남겨두는 경우가 있다고 응답했고 그 시간은 평균 하루 5시간 40분인 것으로 나타났다. 또한 집에 혼자 남겨두는 경우 반려동물을 위한 조치 중 가장 높은 비율이 TV 또는 조명을 켜고 외출하는 것이었다. 집 안 온도가 떨어지거나 오를 경우 어느정도 체온을 조절할 수 있는 강아지나 고양이는 혼자 남겨져도 체온을 유지할 수 있지만, 도마뱀과 같은 변온동물의 경우 자체적으로 체온을 유지하는 데에 어려움이 있다.

스마트 IoT 도마뱀 사육장은 사육자가 없는 환경에서의 온도 조절을 기본 목표으로 하며, 이 외에도 생활에 필수적인 습도 조절, 먹이와 물 급여 등이 자동으로 이루어지는 것을 목표로 한다.

* 1. 역할 분담

|  |  |  |
| --- | --- | --- |
| 역할 | 이름 | 담당 업무 |
| 팀장 | 박성범 | 카메라 스트리밍 |
| 팀원 | 김강현 | PI-Server 통신, 센서 인식 및 제어 |
| 팀원 | 박지호 | 안드로이드 앱 개발 |
| 팀원 | 송연석 | 서버 및 DB 구축 |
| 팀원 | 오정민 | 센서 인식 및 제어 |
| 팀원 | 조석훈 | 3D Print 제작, 센서 인식 및 제어 |

* 1. 개발 일정

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 일정 | 11/5 | 11/12 | 11/19 | 11/26 | 12/3 | 12/10 | 12/15 |
| 기획 |  |  |  |  |  |  |  |
| 분석 |  |  |  |  |  |  |  |
| 설계 |  |  |  |  |  |  |  |
| 개발 |  |  |  |  |  |  |  |
| 테스트 |  |  |  |  |  |  |  |
| 프로젝트 안정화  및 발표 |  |  |  |  |  |  |  |

* 1. 개발 환경
* **OS**

Windows 10

Raspbian Linux

Android

* **개발 도구**

Visual Studio code

Visual Studio 2019

Android Studio

Geany

Thinkercad

* **개발 언어**

C/C++

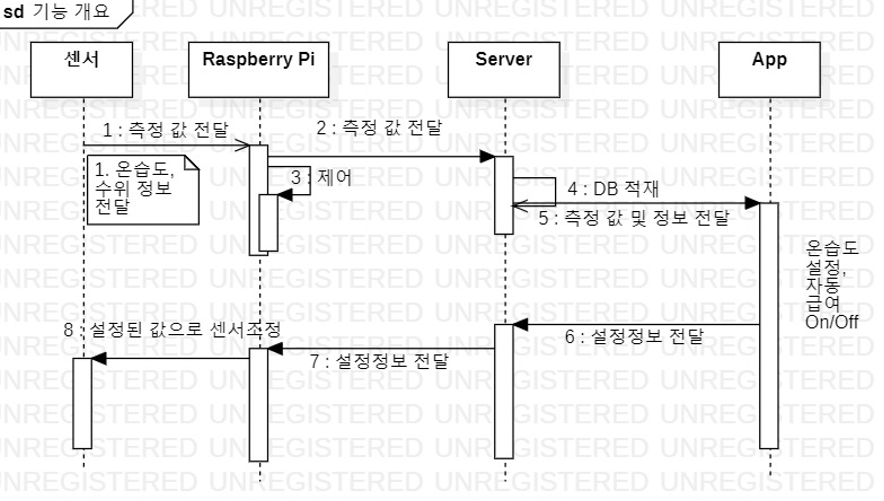
C#

Android Java

Python

**Ⅱ. 본 론**

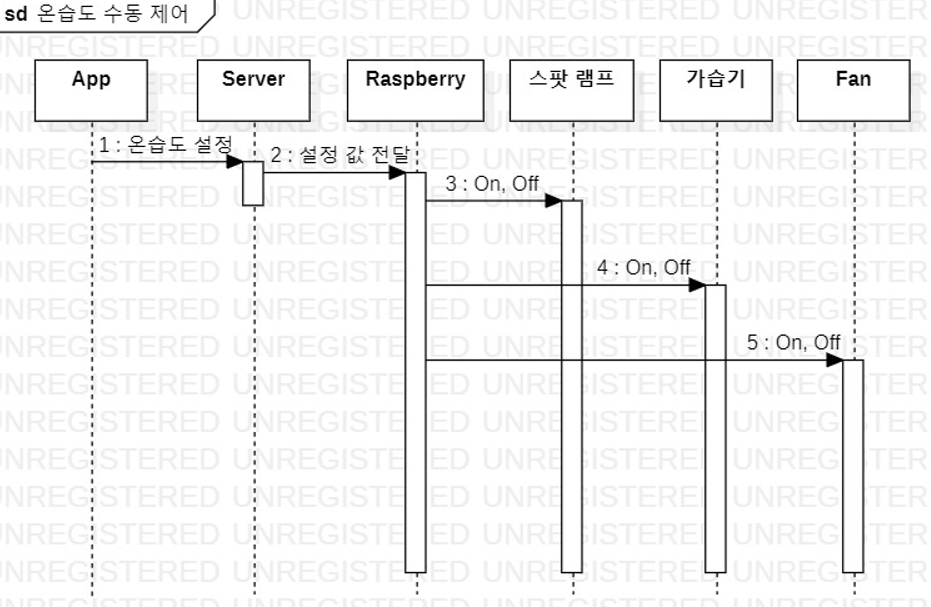
1. ***기능 정의***
   1. 기능 개요



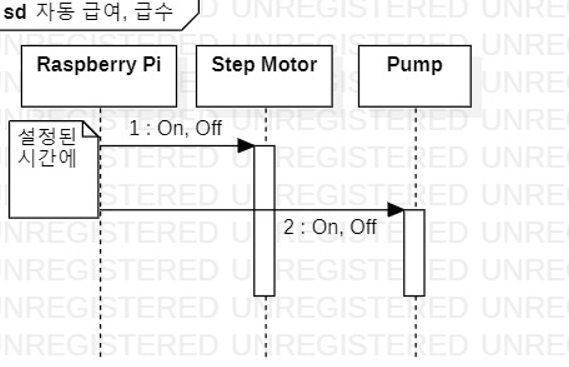
[그림 2] 기능 개요



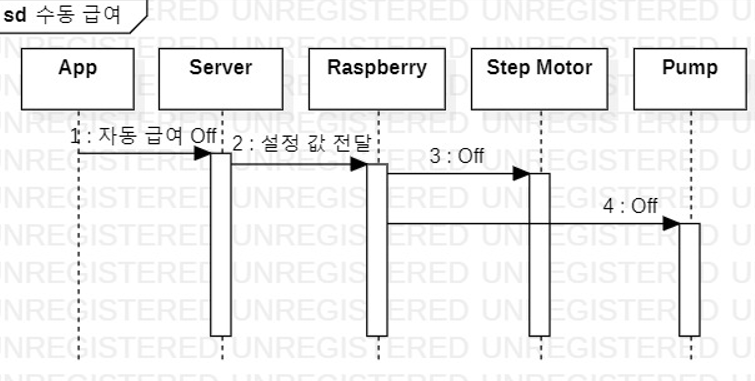
[그림 3] 온습도 자동 제어

******

[그림 4] 온습도 수동 제어

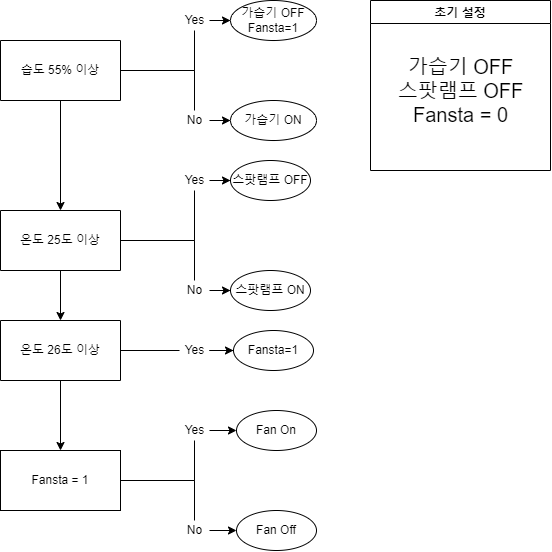
******

[그림 5] 자동 급여, 급수

******

[그림 6] 수동급여

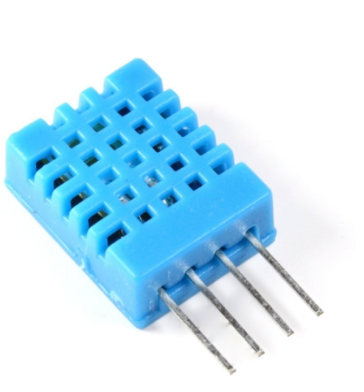
1. ***인식 및 제어***
   1. Logic Process



[그림 7] 센서 Logic Process

* 1. 온/습도
     1. 온/습도부 하드웨어
* 온/습도 센서(DHT-11)

사육장 내부의 온도를 측정하기 위해 DHT-11 센서를 사용한다. DHT-11 센서는 0~50 ℃의 온도를 측정하고 ±2%의 오차범위를 가지며, 20~95%의 습도를 측정하고 ±5%의 오차범위를 갖는다. 도마뱀의 서식조건이 25~35 ℃의 온도와 40~55%의 습도이기 때문에 DHT11의 센서로 충분히 인식이 가능하다.

 테이블이(가) 표시된 사진

자동 생성된 설명

[그림 8] 온/습도 센서(DHT 11) [그림 9] 온/습도 센서(DHT 11) 스펙

* DHT-11 기능

DHT-11 센서 2개를 사육장 양쪽 끝에 배치하여 각각의 온도, 습도 값을 받아오고 온도와 습도의 평균값을 구해 저장한다. 추후 온/습도 제어 시 이 평균값들이 사용된다.

* 스팟램프

사육장 내부의 온도 제어를 위해 히팅 스팟램프를 사용한다. 스팟 램프는 빛과 열이 나는 램프로, 일정 지역을 따뜻하게 데워주거나 전체 공간을 서서히 따뜻하게 해서 온도를 유지시킨다.

해당 스팟 램프는 레드 스팟으로 적외선을 쐬어 온도를 조절하며, 일반 스팟 램프와 다르게 사육 동물의 눈의 피로를 덜게 하고 안정감을 줄 수 있다. 밝은 환경을 피해야 하는 야행성 파충류에게 사용한다.

 텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 10] 스팟 램프 [그림 11] 스팟 램프 스펙

* Fan

사육장 내부의 온도와 습도를 제어하기 위해 Fan을 사용한다. Fan으로 내부의 공기를 외부로 내보내 줌으로서 높아진 온도를 낮추거나, 습기를 외부로 배출해 사육장 내부의 습도를 낮춘다.

|  |  |
| --- | --- |
| 항목 | 사양 |
| 크기 | 60X60X10mm 케이블 :35cm |
| 무게 | 36g |
| RPM | 3500 RPM |
| CFM | 14.3 CFM |

텍스트, 부채, 검은색이(가) 표시된 사진

자동 생성된 설명

[그림 12] USB 쿨링 Fan [표 1] Fan 스펙

* 가습기

|  |  |
| --- | --- |
| 항목 | 사양 |
| 전압 | DC 5V |
| 전류 | 80 mA |
| 진동자 주파수 | 108 kHz |
| 크기 | 진동자 지름 16mm  진동자 전선 길이 18cm |

사육장 내부의 습도를 증가해주기 위해 가습기 모듈을 사용한다.

전자기기이(가) 표시된 사진

자동 생성된 설명

[그림 13] 가습기 모듈 [표 2] 가습기 모듈 스펙

* 릴레이 모듈

릴레이는 전자석의 원리를 사용해 전류가 흐르면 자기장을 형성하여 자기력으로 자석을 끌어당겼다가 전류가 흐르지 않으면 자석을 놓는 일종의 스위치이다. 전자기기의 On/Off를 진행할 때 필수적으로 사용되는 부품으로 파충류를 위한 스마트 하우스를 만들기 위해 스팟램프, 가습기, Fan, 워터펌프에 사용되었다.

|  |  |
| --- | --- |
| 항목 | 사양 |
| 전압 | DC 5V |
| 전류 | 10A |
| 크기 | 2.7X5.4X1.7cm |

전자기기, 회로이(가) 표시된 사진

자동 생성된 설명

[그림 14] 4채널 릴레이 모듈 [표 3] 4채널 릴레이 모듈 스펙

* 온도조절 기능

DHT 11 센서로부터 값을 받고 온도의 평균값이 25℃보다 작으면 스팟램프를 켜놓고 넘으면 스팟램프를 꺼준다.

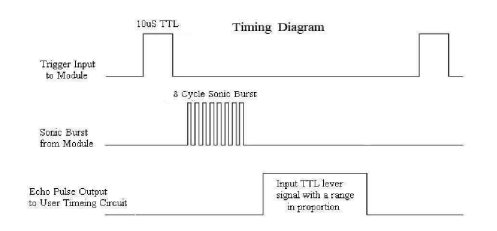
온도의 평균값이 26℃를 넘으면 2.5초간 Fan을 켜주고 15초동안 꺼준다. (이런 구동방식이 되는 이유는 2가지이다. 1. Thread가 도는 동안 main함수의 while loop가 돌지 않기 때문 2. Fan이 짧은 시간만 돌아도 습도가 순식간에 내려가기 때문)

* 가습조절 기능

DHT 11 센서로부터 값을 받고 습도의 평균값이 55%보다 작으면 가습기를 켜고 넘으면 가습기를 꺼준다.

습도의 평균값이 55%를 넘으면 2.5초간 Fan을 켜주고 15초동안 꺼준다. (이런 구동방식이 되는 이유는 2가지이다. 1. Thread가 도는 동안 main함수의 while loop가 돌지 않기 때문 2. Fan이 짧은 시간만 돌아도 습도가 순식간에 내려가기 때문)

* + 1. 문제 해결
* 최초 설계
* DHT11 센서 2개를 양쪽 끝에 배치하여 각각의 온도, 습도 값을 받아오고 평균 값을 구해 저장한다. 저장된 Data를 바탕으로 사육장에 설치된 스팟 램프, 가습기, Fan을 조절
* 문제
* DHT11 센서의 값을 끊임없이 받아와야 하는데 최대 50회(1분가량) 받아오면 알 수 없는 이유로 프로그램이 멈추며 더 이상 센서 값을 받을 수 없음
* 1차 문제 해결
* DHT-11과 브레드보드, DHT-11과 전선 사이에 틈이 생겨 신호를 못 받아온다고 판단. DHT와 전선을 납땜하여 밀착시켰으나 여전히 1분 내로 프로그램이 멈추며 값을 받을 수 없음
* 2차 문제 해결
* ‘프로그램은 최초 HIGH신호에서 LOW신호가 오기를 기다리는데, 센서는 LOW신호를 넘어 다음 HIGH신호에 가 있어서 만날 수 없다’라는 타이밍상의 문제라고 판단했다. 따라서 코드 안에 있는 무한 Loop를 수정했다. GetRespons, ReadData 함수에서 HIGH와 LOW인 상태의 대기시간이 일정 시간 이상 길어질 경우 음수의 값을 받고 Run 함수의 해당 while(for)문에서 Continue하게 된다.



[그림 15] Timing Diagram(HIGH, LOW신호) 메커니즘

* 1. 급수기
     1. 급수 제어
* 워터펌프

물통 속에서 펌프 바닥에 있는 흡입구로 들어간 물이 모터에 의해 회전하면서 모터안의 물은 원심력에 의해 압력이 증가한다. 축의 바깥쪽은 압력이 증가하고 중심은 압력이 낮아지는데 압력이 낮은 중심부에서 압력이 높은 배출구로 물이 이동하면서 물을 배출한다.

|  |  |
| --- | --- |
| 항목 | 사양 |
| 전압 | DC 3~5V |
| 물 방출구멍 | 지름 7mm |
| 크기 | 42X22 mm |
| 전선 길이 | 178mm |

시험관이(가) 표시된 사진

자동 생성된 설명

[그림 16] 워터펌프 [표 4] 워터펌프 스펙

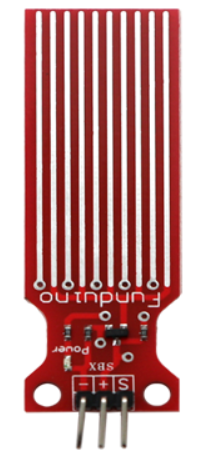
* 워터펌프 기능

지정된 시간이 되면(1일 1회, 지정된 시각) 릴레이 모듈을 사용하여 도마뱀의 물통에 5초동안 워터펌프를 가동시켜 물을 급여한다.

* + 1. 수위 조절
* 수위센서

물통 속에서 물의 수위를 인식하기 위해 사용하는 센서이다. 세로로 길게 있는 전극부분이 수위(물과의 접촉)에 따라 저항 값이 변한다. 전압이 일정한 상황이기 때문에 저항이 변함에 따라 전류도 변하게 되어 수위의 정도를 인식한다.

|  |  |
| --- | --- |
| 항목 | 사양 |
| 전압 | DC 5V |
| 작동 전류 | 20mA 이하 |
| 크기 | 센서 감지보드 42X16mm  센서 보드 60X20X8mm |
| 센서 무게 | 3.5g |



[그림 17] 수위센서 [표 5] 수위센서 스펙

* 수위센서 기능

제작된 물통으로부터 배출되는 물은 가습용도의 물, 급수용도의 물로 2가지 종류이다. 이러한 물통의 물의 양을 충만해야 할 때 사용자가 인식할 수 있어야 하므로 수위가 10%미만으로 지속될 경우 서버로 물 부족 신호를 보내 어플리케이션에서 팝업 알림을 뜨게 한다.

* + 1. 문제 해결
* 워터펌프
* 최초 설계
* 물 공급 관에 구멍을 2개 뚫어서(윗구멍: 공기유입구, 아래구멍: 물 배출구) 물이 윗구멍을 막으면 자동으로 물이 더 이상 나오지 않는 물의 압력차를 활용하는 방식으로 자동 급수기를 구현하고자 했다. 즉 도마뱀이 물을 마셔서 물의 높이가 낮아지면 지정된 위치(윗구멍)까지 자동으로 물 급수되는 방식이다.
* 문제
* 물의 급수를 해주는 물통과 관의 높이 차이에 따른 수압 차 때문에 일정 높이까지 물을 채워주는 형식이 불가능
* 문제 해결
* 워터 펌프로 일정 시간(1일 1회, 00시)이 되면 자동으로 워터펌프를 사용하여 물을 급수하는 방식으로 변경.
* 수위센서
* 최초 설계
* 가습기와 급수기의 필요한 물을 충만하는 물통의 물이 부족해지면 어플리케이션에서 사육자에게 알림을 주는 형식 고안
* 문제
* 라즈베리파이의 GPIO는 아두이노와 다르게 디지털 입력만 가능. ADC가 내장되어있지 않아 analogread()함수를 사용할 수 없음
* 문제해결
* 별도의 ADC장치를 사용하여 문제 해결
  1. 급여기
     1. 급여기 하드웨어

급여기는 회전부와 고정부로 이루어져 있다. 회전부의 칸 수가 총 8칸이고 고정부에는 45도 각도로 한 칸 크기 정도 구멍이 뚫려 있다. 회전부가 회전하며 이 구멍으로 먹이가 떨어지는 메커니즘이다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

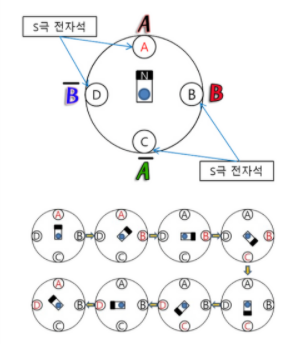
[그림 18] 급양기 하드웨어

[정면도] [배면도]

[평면도] [밑면도]

* + 1. 급여제어
* 스텝모터

S극 전자석 A, B, C, D에 전류를 흘러주면 자기력이 발생하여 중앙에 있는 N극 전자석이 전류에 따른 자기력을 따라 정렬한다. 예를 들어 A와 B에 전류를 흘려준다면 두번째 그림의 2번처럼 N극 전자석이 A와 B 사이에 위치한다. 이런 식으로 전류를 360도 돌려주면 모터는 360도 회전하게 되고, 흘려주는 전류의 양을 바꿔 줌으로써 특정 각도로도 설정이 가능하다. 스텝모터와 회전부를 연결하여 설정한 시각이 되면 급여기를 45\*n도(n은 설정한 시각이 되었을 때의 count이며, 해당 시각이 돌아올 때마다 1만큼 더해준다.)만큼 돌려준다.

[그림 19 ]스텝모터 구동원리 [그림 20] 스텝모터

* + 1. 문제 해결
* 최초 설계
* 스텝모터를 활용한 자동 급여기 구현하여 일정 시간이 되면 통이 돌아가면서 먹이(귀뚜라미) 떨어뜨려 줌
* 문제
* 스텝모터와 연결하여 구동 시, 마찰과 유격 문제로 의도대로 작동 실패
* 문제 해결
* 3D 프린터로 제작 (ThikerCad, Cura 프로그램 사용, Sindoh 3DWOX 프린터기 사용)

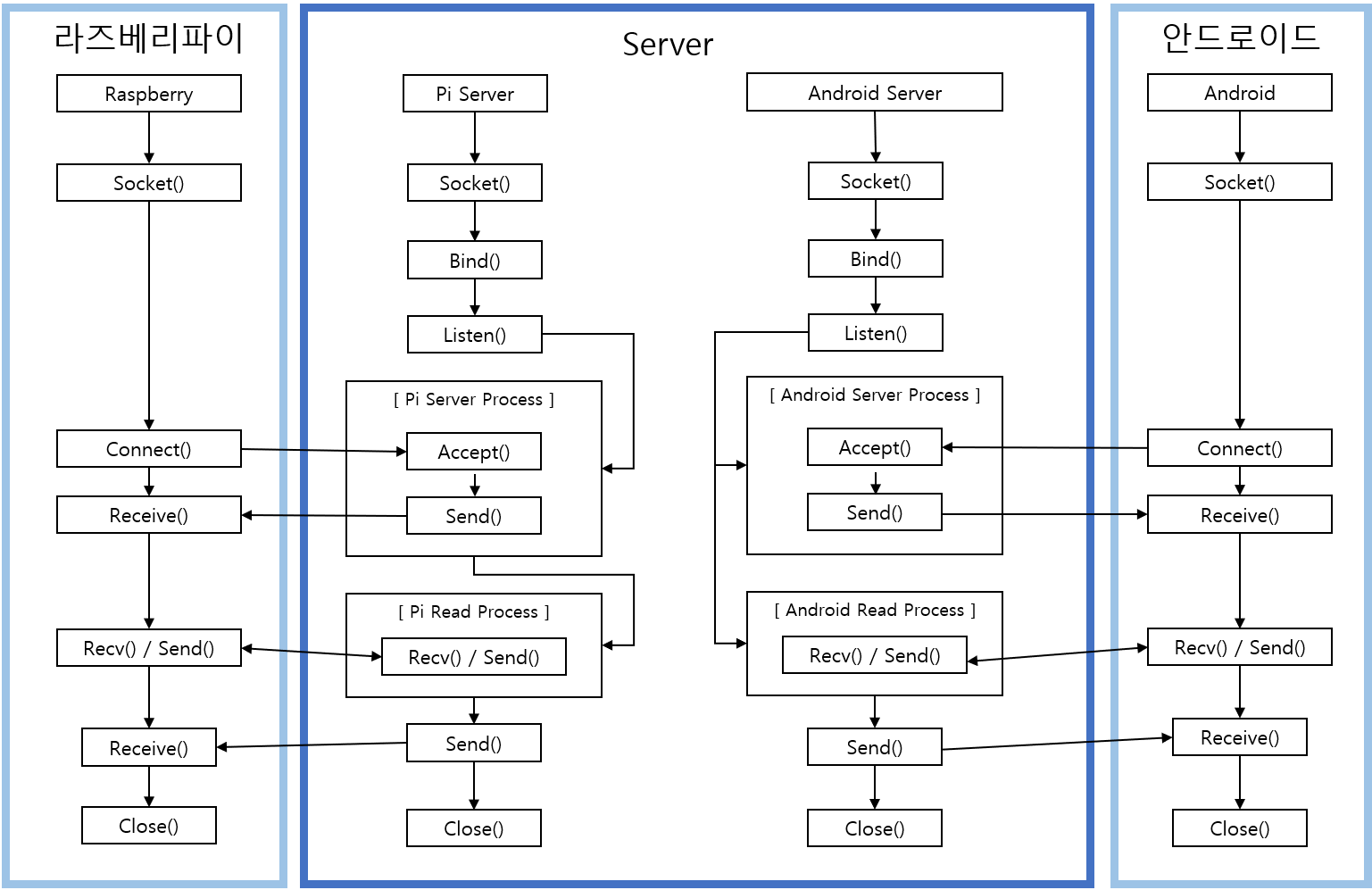
텍스트이(가) 표시된 사진

자동 생성된 설명 하얀색이(가) 표시된 사진

자동 생성된 설명

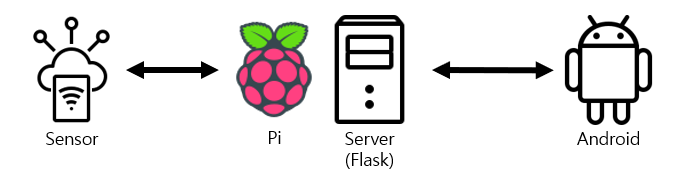
[그림 21] 변경 전 [그림 22] 변경 후

1. ***서버*** 
   1. Logic Process



[그림 23] Server Logic Process

* 1. 서버 프로그램 개요
     1. 서버 프로그램 구상안
* 최초 설계



[그림 24] 서버 최초설계

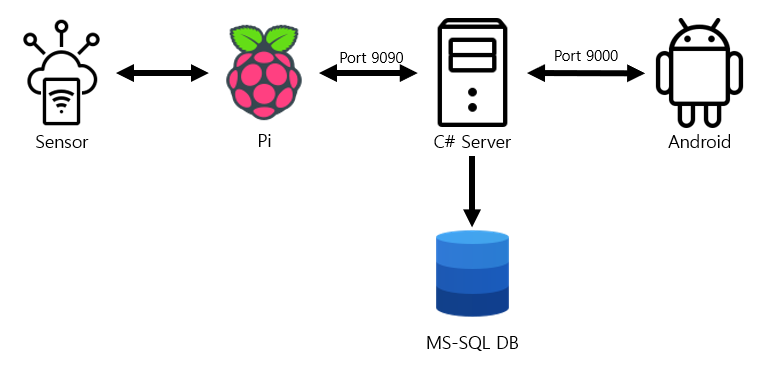
* Python의 웹 프레임워크인 Flask를 이용한 웹 서버를 구축하여 안드로이드와 라즈베리파이 간 통신이 가능하도록 계획하였다.
* Flask는 Python의 마이크로 웹 프레임워크로 가볍고 코드도 비교적 단순하며, 특히 API 서버를 만들기에 편리하다는 장점이 있어 선택하였다.
* 라즈베리파이에 Flask 웹 서버를 구축하고, 기능별로 도메인 주소를 나눠 모바일 앱의 기능과 연동시켜 안드로이드와 라즈베리파이가 Flask 웹 서버를 통해 데이터 통신하는 것을 목표로 하였다.
* Flask 웹서버를 사용하여 라즈베리파이의 GPIO를 제어할 수 있었지만, 이미 센서 제어 프로그램을 C 언어로 작성하고 있었으므로 Flask 웹서버에서 제어 프로그램과 웹 서버간의 TCP/IP 소켓 통신이 필요하다고 판단하였다.
* 문제
* Flask 웹서버와의 소켓 통신을 위해 Python의 socket 패키지와 Flask의 socketio 웹 소켓 패키지를 적용하였으나 소켓 통신이 이뤄지지 않는 문제가 발생하였다.
* 1차 수정 설계

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

[그림 25] 서버 1차 수정 설계

* Flask 웹 서버 프로그램 설계 실패로 회의를 진행하였다. IoT 개발자 양성과정을 통해 C# 언어에 대한 친밀성이 높았다는 점과 C#의 Windows Form 라이브러리로 TCP/IP 소켓 프로그래밍을 사용한 프로그램을 개발한 경험이 있다는 점에 근거하여 C# Windows Form 라이브러리를 사용하여 서버 프로그램을 개발하기로 결정하였다.
* 외온성 변온동물인 도마뱀의 특성상 주위 환경 및 온도변화에 매우 민감하다. 만약 온습도 센서 DHT-11가 고장이 나서 정상적으로 온도와 습도를 측정하지 못한다면 온/습도를 제어하는 데에 큰 문제점이 발생할 것이라 예상하였다.
* DHT-11 센서 불량 여부를 분석하기 위해 온도, DHT-11 센서 상태, 날짜로 이루어진 온도 데이터베이스와 습도, DHT-11 센서 상태, 날짜로 이루어진 습도 데이터베이스를 구축하는 게 좋겠다는 결론을 내렸고, Visual Studio 2019에서 제공되는 MS-SQL을 사용하여 데이터베이스를 구축하였다
* 문제
* 라즈베리파이와 안드로이드, 서버를 모두 연결하여 데이터를 주고받는 과정에서 데이터 충돌이 발생하여 소켓 통신 오류가 발생하였다.
* 오류가 발생하게 된 원인은 하나의 포트로 서버를 구성했기 때문이었다. 안정적인 소켓 통신을 위해 서버 포트를 안드로이드 전용 포트(9000)와 라즈베리파이 전용 포트(9090)로 나눠서, 안드로이드 전용 서버와 라즈베리파이 전용 서버를 구성하였다.
* 최종 설계



[그림 26] 서버 최종 설계

* 서버 프로그램을 개발하기 위해 C#의 Windows Form 라이브러리를 사용하기로 결정하였다.
* 안드로이드 전용 포트(9000)와 라즈베리파이 전용 포트(9090)로 나눠서 서버 프로그램 안에 안드로이드 전용 서버와 라즈베리파이 전용 서버를 구축하였다.
* C# 서버 프로그램에 Metro UI를 적용하여 사용자들이 사용하기 쉽고 정보를 한 눈에 알아볼 수 있도록 구성하였다.
* C# Windows Form 오픈소스 UI 중 하나인 Metro UI는 마이크로소프트가 만든 ‘타이포그래피’ 기반 UI로, 복잡한 구성을 피하고 사용자에게 보다 인지하기 쉬운 인터렉션을 제공하는 특징을 가지고 있다.
* [Metroframework (thielj.github.io)](https://thielj.github.io/MetroFramework/) (사용된 Framework)
  1. 기능 정의

4.3.1 서버 프로그램 구성

* 서버 메뉴와 상태 표시줄
  + 서버 메뉴는 서버 시작과 프로토콜 확인 텍스트 박스, 서버 기록 확인 텍스트 박스로 이루어져 있다.
  + 서버 시작 버튼을 클릭하면 라즈베리파이와 안드로이드 서버가 생성되어 라즈베리파이와 안드로이드에서 연결 요청이 올 때까지 기다린다. 연결 요청이 들어온다면 연결을 수락한다.
* 상태 표시줄은 최근 연결 IP, 연결 클라이언트 기기를 알 수 있고, DB 목록을 클릭하여 습도 DB와 온도 DB를 확인할 수 있다. 또한 도마뱀의 종류가 다양하여 도마뱀에 따른 사육장 적정 온습도 범위가 다르므로, 온습도 설정 버튼을 클릭하여 사육장 적정 온습도 범위를 설정할 수 있도록 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 27] 서버 메뉴와 상태 표시줄

* 온습도 메뉴: 현재 온습도, 설정한 온습도를 확인할 수 있는 메뉴이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 28] 온습도 메뉴

* 설정 메뉴: 서버 프로그램의 유지보수를 위한 기능으로 이루어진 메뉴이다.



[그림 29] 설정 메뉴

4.3.2 변수 및 통신 프로토콜

* 변수 목록

- 다음 변수들은 전역 변수로 처리하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 30] 변수 목록

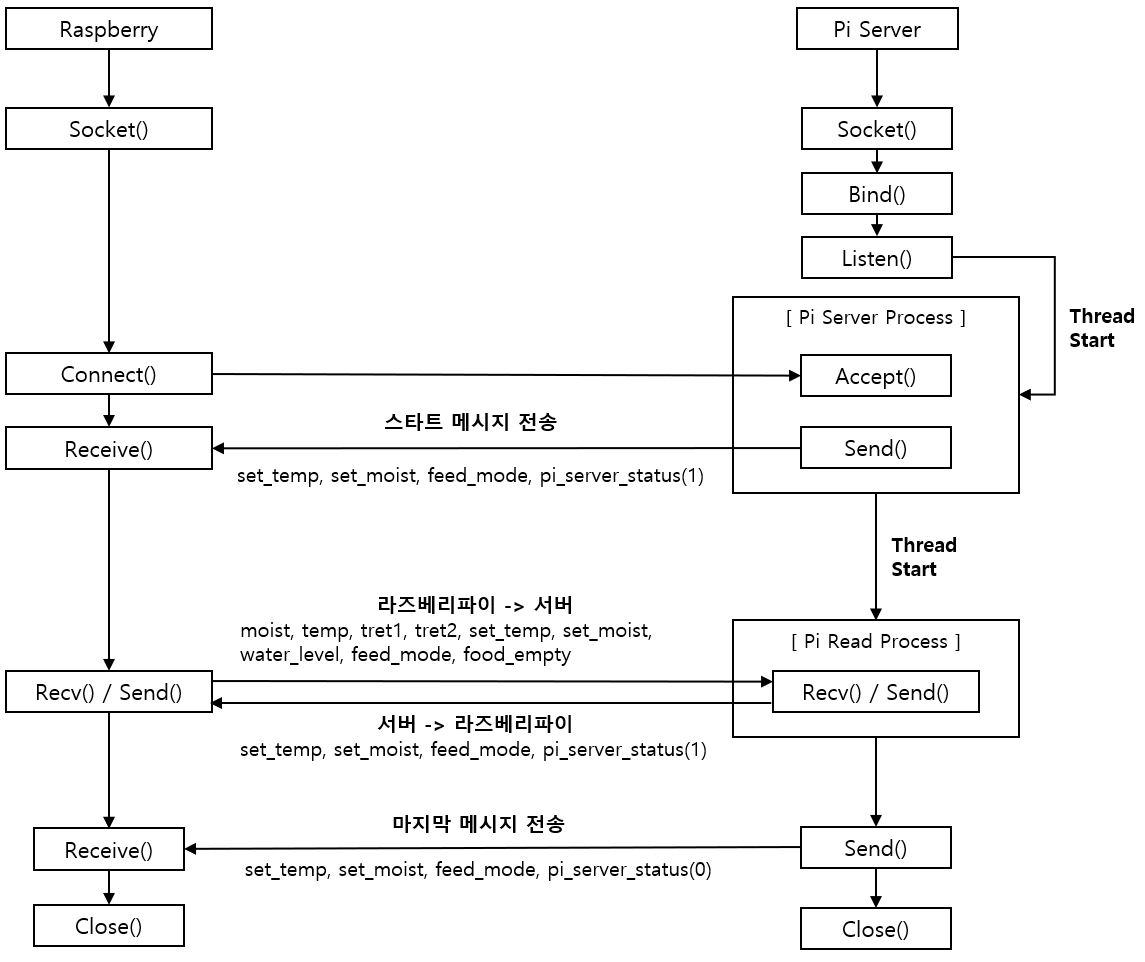
* 통신 프로토콜
* 안드로이드와 라즈베리파이, 서버 간의 데이터를 통신하기 위해서 표준화된 통신 규칙이 필요하다. 이를 위해 안드로이드 앱 개발 팀원과 라즈베리파이와 서버 통신을 담당하는 팀원과의 회의를 거쳐 표준화된 규칙을 만들었다.
* 데이터를 전송할 때, 데이터의 종류를 구분하기 위해 콤마(,)를 사용하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

[그림 31] 통신 프로토콜

* + 1. 프로그램 실행 시작
* mylibrary.dll 파일에 있는 iniFile 함수를 사용하여 최근 데이터를 저장하고, 추후에 불러올 currentData.ini 파일을 생성한다.
* 프로그램을 실행할 때 DB와 연결되고, currentData.ini 파일에 저장되어 있는 온습도, 설정 온습도, 물통 수위, 자동 급여 On/Off 여부, 급여기 현황, 사육장 적정 최대/최소 온습도 데이터를 불러온다.
* 불러온 온습도, 설정 온습도 데이터는 온습도 메뉴의 현재 온습도, 설정 온습도에 입력된다.
  + 1. Pi to Server



[그림 32] Pi to Server 개요도

* 상세 설명 (Pi Server Start: 라즈베리파이 서버 생성)
* 서버 시작 버튼을 누르면 라즈베리파이 서버 소켓을 생성하고 소켓에 포트번호 9090을 부여한다. 그 후 listen을 시작하여 라즈베리파이가 서버와 연결이 될 때까지 접속을 기다린다.
* 만약 라즈베리파이 서버가 이미 생성되어 있다면 “라즈베리파이 서버를 다시 시작하시겠습니까?”라는 메시지를 띄운다. OK를 클릭하면 라즈베리파이 서버 listen을 종료하고 라즈베리파이 스레드를 모두 종료시킨 후에, 다시 라즈베리파이 서버를 생성한다.
* 이후 라즈베리파이의 연결 요청를 처리할 수 있는 스레드 PiThreadServer를 시작한다.

* 상세 설명 (PiThreadServer: 라즈베리파이 Connect 요구 처리 스레드)
* 라즈베리파이에서 연결 요청이 들어오면, 연결 요청을 수락하는 기능을 가진 AcceptTcpClient() 함수를 호출하여 라즈베리파이 연결에 대한 새로운 소켓을 생성한다.
* 라즈베리파이와 서버가 연결이 되었으므로, 서버와 라즈베리파이의 연결 상태 변수의 값을 1로 설정하여 서버와 라즈베리파이는 연결이 되었음을 표시한다.
* 이후 연결 확인을 위해 최근에 저장된 데이터와 서버-라즈베리파이 연결 상태 변수의 값을 라즈베리파이로 전송하여 라즈베리파이와의 연결이 성공적으로 이루어져 있음을 알린다.
* 이후 라즈베리파이에서 보낸 데이터를 처리할 수 있는 PiThreadRead 스레드를 시작한다.
* 상세 설명 (PiThreadRead 스레드)
* 라즈베리파이로부터 받은 온습도, DHT-11 에러 유무, 설정 온습도, 물통 수위, 자동급여 On/Off 여부, 급여기 현황 데이터를 저장하고 온습도 메뉴에 있는 현재 온도와 현재 습도 현황판에 입력시킨다.
* 라즈베리파이로부터 받은 온습도, DHT-11 에러 유무 데이터와 날짜와 시간 데이터를 데이터베이스에 적재한다.
* 서버에서 라즈베리파이로 설정 온습도, 자동급여 On/Off 여부, 서버와 라즈베리파이 연결 상태 변수를 전송한다. 서버와 라즈베리파이는 서로 연결 중인 상태이므로 서버-라즈베리파이 연결 상태 변수의 값으로 1을 전송한다.
* 상세 설명 (Pi Server Off: 라즈베리파이 서버 종료)
* 라즈베리파이와 서버의 연결이 종료되므로 서버와 라즈베리파이의 연결 상태를 나타내는 변수의 값을 0으로 설정한다.
* 서버를 종료하기 전에 마지막으로 저장된 설정 온습도, 자동급여 On/Off 여부, 서버와 라즈베리파이 연결 상태 변수 값을 라즈베리파이에 전송한다.
* 이후 스레드를 모두 종료하여 라즈베리파이 서버를 종료하고, “라즈베리파이 서버를 종료합니다.” 메시지 박스를 띄워 라즈베리파이 서버가 종료되었음을 알린다.
* 데이터베이스 적재
* MS-SQL 데이터베이스 파일인 SensorDB.mdf을 Visual Studio 2019에 연동하여 Temperature 테이블과 Moisture 테이블을 생성하였다.

텍스트, 점수판, 스크린샷이(가) 표시된 사진

자동 생성된 설명

[그림 33] Temperature 테이블 디자인

* id는 자동으로 1씩 증가하도록 지정해두었다.

테이블이(가) 표시된 사진

자동 생성된 설명

[그림 34] Temperature 테이블 SQL문

텍스트, 점수판, 스크린샷이(가) 표시된 사진

자동 생성된 설명

[그림 35] Moisture 테이블 디자인

* id는 자동으로 1씩 증가하도록 지정해두었다.

테이블이(가) 표시된 사진

자동 생성된 설명

[그림 36] Moisture 테이블 SQL문

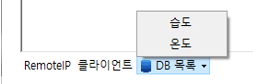
* PiThreadRead 스레드에서 라즈베리파이에서 받은 데이터 중 온도, DHT-11 에러 유무, 날짜와 시간 데이터를 데이터베이스에 저장하기 위해 myLibrary.dll 파일에 있는 함수를 사용하였다.
* 먼저 데이터베이스 접속 경로 변수를 생성하고, 프로그램을 실행할 때 데이터베이스를 연결한다.
* InsertDB 함수를 사용하여 온도, DHT-11 에러 유무, 날짜와 시간을 Temperature 테이블에 입력하는 insert SQL문을 입력하고 실행시킨다.
* 마찬가지로 InsertDB 함수를 사용하여 습도, DHT-11 에러 유무, 날짜와 시간을 Moisture 테이블에 입력하는 insert SQL문을 입력하고 실행시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 37] InsertDB 함수

* 온도 데이터베이스와 습도 데이터베이스를 확인하기 위해 아래와 같이 상태 표시줄에 DB 목록 메뉴 아이템을 생성한 후, 온도 메뉴와 습도 메뉴를 만들었다.
* 온도 메뉴와 습도 메뉴를 클릭하면 데이터베이스를 확인할 수 있도록 온도 데이터베이스 조회 클릭 함수와 습도 데이터베이스 조회 클릭 함수를 생성한다.



텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 38] 온습도 데이터베이스 조회

테이블이(가) 표시된 사진

자동 생성된 설명

[그림 39] 온도 데이터베이스 팝업창

테이블이(가) 표시된 사진

자동 생성된 설명

[그림 40] 습도 데이터베이스 팝업창

* 정보 분석
* DB에 적재한 DHT-11 에러 유무 데이터를 활용하여 센서 정확도를 측정하고자 한다.
* 1034개 중 579개 데이터를 사용하였는데, 11월 23일 14시 49분 이전 데이터와 25일 16시 25분 38초부터 30일 10시 4분 3초 데이터는 테스트를 위해 입력된 데이터이므로 제외하였다.
* 정보 분석을 하기 위해서 MS-SQL의 SensorDB.mdf 파일에 있는 온도 DB와 습도 DB 자료를 검토한 후, 탭 키로 구분하여 텍스트 파일로 자료를 처리하였고, R 프로그램의 dplyr 패키지를 활용하여 데이터를 분석하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

[그림 41] 온도, 습도 데이터 일부

|  |  |  |  |
| --- | --- | --- | --- |
| Tret1  Tret2 | 1 (정상) | -1 (오류) | 합계 |
| 1 (정상) | 543 | 20 | 563 |
| -1 (오류) | 13 | 3 | 16 |
| 합계 | 556 | 23 | 579 |

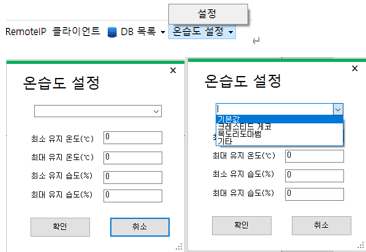
[표 6] 분석 결과

* 579개 데이터를 분석한 결과, 두 센서가 모두 정상인 데이터는 543개, 한 센서만 정상인 데이터는 33개, 모두 오류인 경우는 3개로, 정확도는 543/579 = 0.938로 높은 정확도를 가지고 있는 것을 확인하였다.
* 위험 경보
* 도마뱀 중 크레스티드 게코 도마뱀의 사육 적정 온도는 18도 ~ 27도이고, 사육 적정 습도는 60% ~ 80%이다. 목도리도마뱀의 경우 사육 적정 온도는 23.9도 ~ 37.7도이고, 사육 적정 습도는 55 ~ 65%이므로 키우는 파충류의 특성에 맞게 최소 온도와 최대 온도, 최소 습도와 최대 습도를 설정할 수 있도록 구현하였다.
* 사육 적정 온도와 습도에서 벗어나면 tbServer 텍스트박스와 tbServerLog 텍스트 박스를 빨갛게 표시하여 위험 경보를 알리도록 구현하였다.



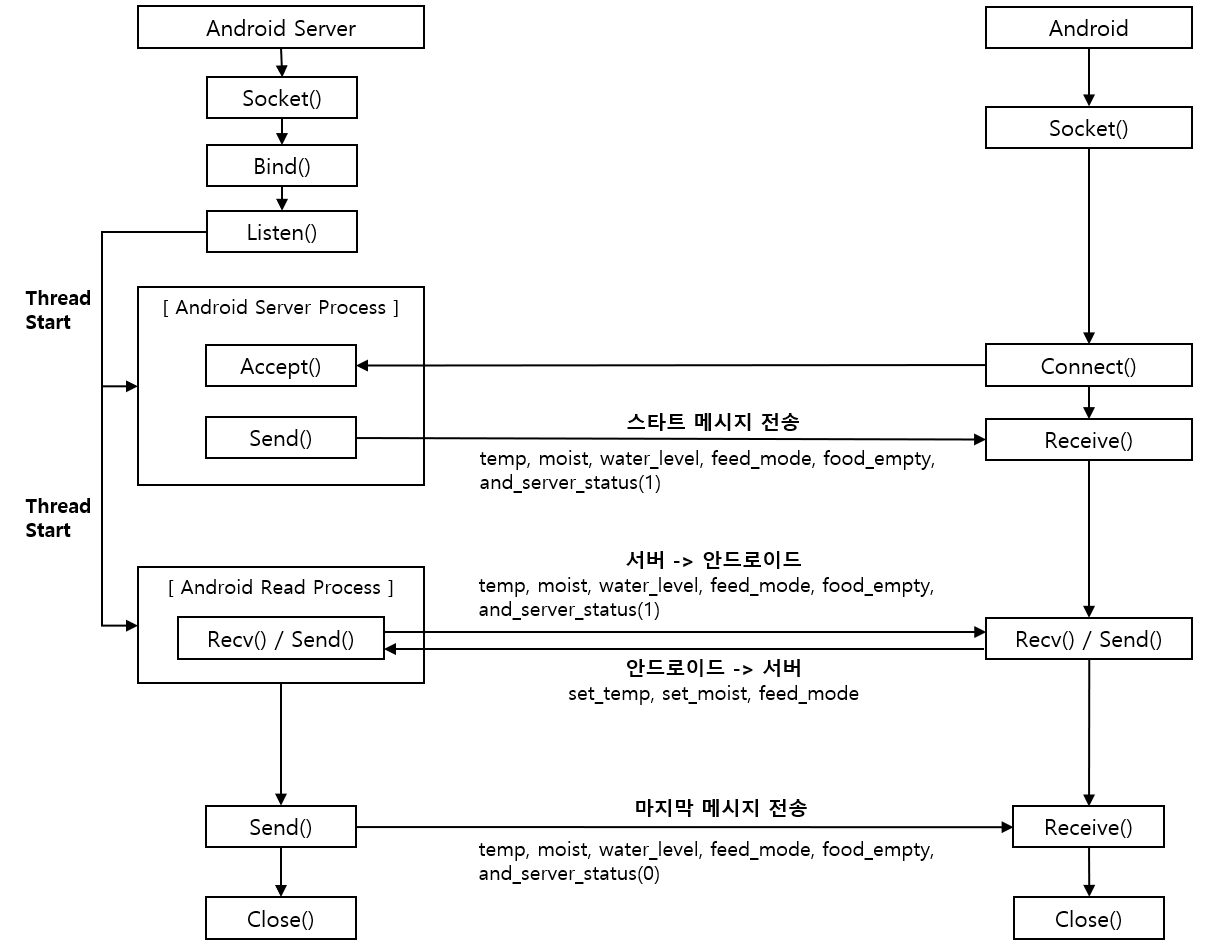
[그림 42] 위험 경보

* 사육 적정 온도 기본 값은 25 ~ 35 ℃, 적정 습도 기본 값 40% ~ 55% 범위를 기준으로 설정하였다.
* 만약 키우는 도마뱀의 사육 적정 온습도가 다르다면, 상태 표시줄에 있는 온습도 설정에 들어가서 ‘기타’를 클릭하고 사육 적정 온습도 범위를 지정할 수 있다.



[그림 43] 온습도 설정

* + 1. Android to Server



[그림 44] Android to Server 개요도

* 상세 설명(Android Server Start: 안드로이드 서버 생성)
* 서버 시작 버튼을 누르면 안드로이드 서버 소켓을 생성하고 소켓에 포트번호 9000을 부여한다. 그 후 listen을 시작하여 안드로이드와 서버가 연결이 될 때까지 접속을 기다린다.
* 만약 안드로이드 서버가 이미 생성되어 있다면 “안드로이드 서버를 다시 시작하시겠습니까?”라는 메시지 팝업을 띄운다. OK를 클릭하면 안드로이드 서버 listen을 종료하고 안드로이드 스레드를 모두 종료시킨 후에 다시 안드로이드 서버를 생성한다.
* 이후 안드로이드의 연결 요청을 처리할 수 있는 스레드 AndroidThreadServer와 안드로이드와 데이터 교환을 처리할 수 있는 스레드 AndroidThreadRead를 시작한다.

* 상세 설명(AndroidThreadServer: 안드로이드 Connect 요구 처리 스레드)
* 안드로이드에서 연결 요청이 들어오면, 연결 요청을 수락하는 기능을 가진 AcceptTcpClient() 함수를 호출하여 안드로이드 연결에 대한 새로운 소켓을 생성한다.
* 안드로이드와 서버가 연결이 되었으므로, 안드로이드와 서버의 연결 상태를 나타내는 변수의 값을 1로 설정하여 안드로이드와 서버는 연결이 되었음을 저장하고, 이를 전송한다.
* 이후 연결 확인을 위해 최근에 저장된 데이터와 안드로이드-서버 연결 상태 변수의 값을 안드로이드로 전송하여 안드로이드와 연결이 성공적으로 이루어져 있음을 알린다.

* 상세 설명(AndroidThreadRead 스레드)
* 안드로이드로부터 받은 설정 온습도, 자동급여 On/Off 여부 데이터를 저장하고 온습도 메뉴의 설정 온도와 설정 습도 현황판에 입력시킨다.
* 서버에서 안드로이드로 온습도, 물통 수위, 자동급여 On/Off 여부, 급여기 현황, 서버와 안드로이드 연결 상태 데이터를 전송한다. 안드로이드와 서버는 서로 연결 중인 상태이므로 안드로이드-서버 연결 상태 변수에 1을 입력한다.
* 상세 설명(Android Server Off: 안드로이드 서버 종료)
* 안드로이드와 서버의 연결이 종료되므로 안드로이드와 서버의 연결 상태를 나타내는 변수 값을 0으로 설정한다.
* 서버를 종료하기 전에 마지막 설정 값인 온습도, 물통 수위, 자동급여 On/Off 여부, 급여기 현황, 서버와 안드로이드 연결 상태 데이터를 전송한다.
* 이후 스레드를 모두 종료하여 안드로이드 서버를 종료하고, “안드로이드 서버를 종료합니다.” 메시지 박스를 띄워 안드로이드 서버가 종료되었음을 알린다.

4.3.6 서버 프로그램 종료

- 실행 중인 스레드를 모두 종료한다.

- 프로그램 종료 시, 라즈베리파이에 설정 온습도, 자동급여 On/Off 여부, 서버와 라즈베리 파이 연결 상태 데이터를 전송한다. 연결이 끊어지므로 서버-라즈베리파이 연결 상태 값은 0으로 전송된다.

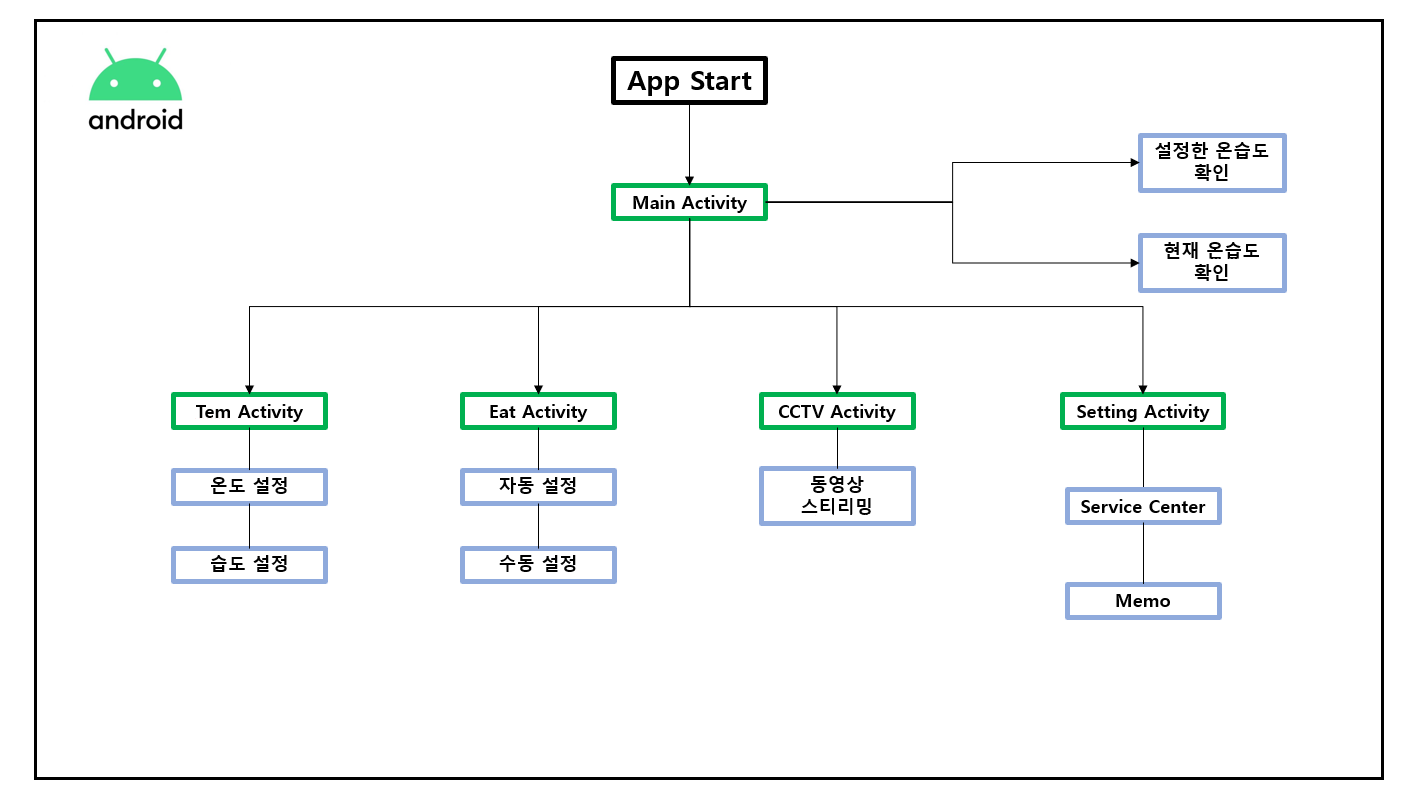
- 마찬가지로 안드로이드에 온습도, 물통 수위, 자동급여 On/Off 여부, 급여기 현황, 안드로이드와 서버 연결 상태 데이터를 전송한다. 연결이 끊어지므로 안드로이드-서버 연결 상태 값은 0으로 전송된다.

- currentData.ini 파일에 온습도, 설정 온습도, 물통 수위, 자동급여 On/Off 여부, 급여기 현황, 사육장 적정 최대/최소 온습도 데이터를 저장한다.

1. ***어플리케이션***



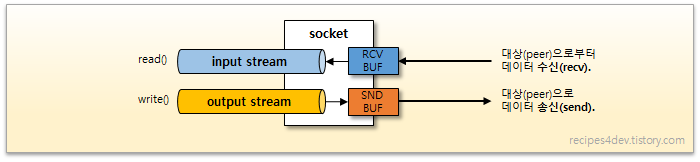
5.1 Logic process



[그림 45] Android Logic Process

5.2 Socket 구성

* Server와 통신하기 위해 Socket을 사용한다.
* 자바에서는 소켓(Socket)을 통해 데이터를 주고받을 때, 자바의 다른 입출력 방법과 마찬가지로, 스트림(Stream)을 사용한다.
* 자바의 입출력(I/O)에 사용되는 스트림(Stream)은 입력(Input) 또는 출력(Output)에 대한 단일 방향으로 동작한다. 즉, 입력된 데이터를 처리하기 위한 입력 스트림(Input Stream)과 출력할 데이터를 전달할 출력 스트림(Output Stream)이 각각 따로 존재한다.
* 소켓(Socket)을 통해 주고받는 데이터는 소켓(Socket)에 할당된 시스템 버퍼를 통해 순서대로 처리되는데, 수신된 데이터를 위한 버퍼와 송신을 위한 버퍼가 각각 따로 존재한다. 네트워크를 통해 수신된 데이터는 수신 버퍼(RCV\_BUF)에 쌓이고, 이는 입력 스트림(Input Stream)을 사용해 프로그램에서 가져올 수 있다. 반대로 출력 스트림(Output Stream)을 통해 쓰여진 데이터는 송신 버퍼(SND\_BUF)를 통해 순서대로 네트워크를 통해 전송된다.



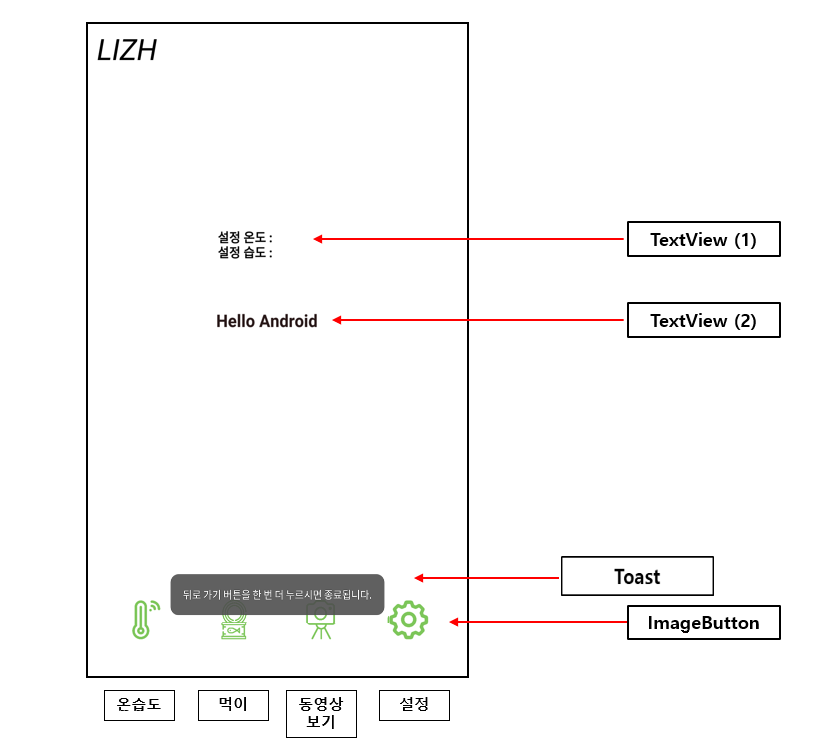
[그림 46] Socket Read&Write

* + 입력 스트림(Input Stream)과 출력 스트림(Output Stream)을 사용하려면, 소켓 인스턴스로부터 스트림에 대한 참조를 가져오면 된다. getInputStream() 메서드와 getOutputStream() 메서드를 통해 사용된다.
  + Server 와 통신하기 위해 첫 실행화면인 main 화면에 Socket Thread를 구성했다.
* 문제
* 다른 Activity에서 다시 main 화면으로 돌아오면 Server에 새로운 Session으로 반복 접속하는 문제가 발생하였다,
* Socket Thread를 Service에서 시작하여 메인으로 불러오기 구현했다.

5.3 Layout 종류 및 속성

* 리니어 레이아웃 (LinearLayout)
  + 방향을 수평 또는 수직 둘 중 결정하여 배치하는 레이아웃으로써 가장 기본적으로 사용되는 레이아웃이다.
  + Vertical 로 설정하게 되면 가로방향으로 요소들을 채워 내려가는 형태가 되고, Horizontal의 경우 세로방향으로 요소를 채워 가는 형태이다.
* 렐리티브 레이아웃 (RelativeLayout)
  + 사전적인 의미로 상대적인이라는 의미를 담고 있는 레이아웃으로써 다른 요소들에 의해 상대적으로 위치가 결정되는 레이아웃이다.
  + 이 레이아웃은 서로 뷰간 연결고리가 있어 위치를 결정함에 있어 서로 연결되어 위치가 결정된다.
* 프레임 레이아웃 (FrameLayout)
  + 프레임, 액자라는 의미로 컨텐츠들이 겹겹이 쌓여 제일 앞에 놓여있는 컨텐츠가 사용자에게 보이는 형태로 구성하는 레이아웃이다
* 테이블 레이아웃 (TableLayout)
  + 일반적으로 알고 있는 테이블의 규칙과 같다, 행과 열로 구분하여 문서의 표 작성하는 구조와 같으며 행을 추가하거나, 열을 추가하고 위젯을 넣어 화면을 구성한다.

5.4 App 프로그램 구성



[그림 47] MainActivitiy

* xml파일 구성

ImageButton, Textview, LinerLayout으로 구성

* TextView (1)

TemActivity에서 보내주는 설정 온습도 값을 getIntent와 getStringExtra를 사용하여 받아서 TextView에서 확인할 수 있게 구현하였다.

* TextView (2)

Raspberry Pi로부터 App으로 보내주는 현재 온습도 값을 TextView에서 받아서 확인할 수 있게 구현하였다. (첫 Server에 연결 성공했을 때 Hello Android 값을 받게 구현)

* ImageButton

자신이 원하는 이미지로 구현할 수 있는 ImageButton으로 구현했다.

* + TemAactivity(온습도)로 이동하는 버튼
  + EatAactivity(먹이)로 이동하는 버튼
  + cctvAactivity(동영상)로 이동하는 버튼
  + SettingAactivity(설정)로 이동하는 버튼



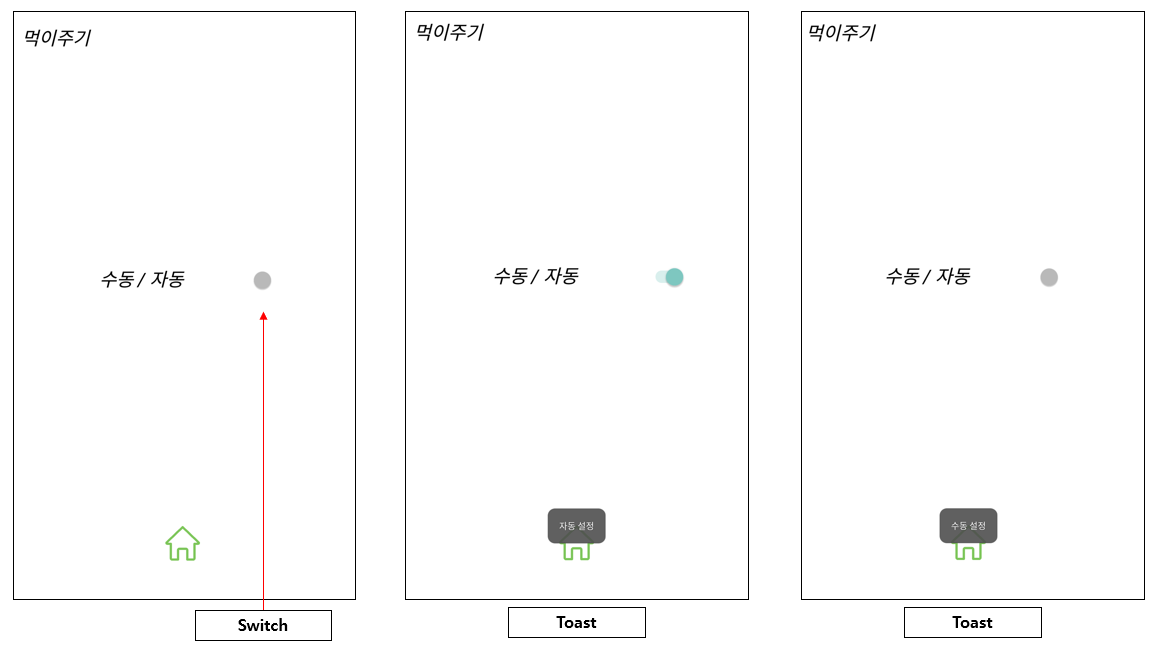
[그림 48] SplashActiviy

* 문제
  + 앱 실행 후 SplashActivity를 거쳐 MainActivity로 오게 되는데, SplashActivity는 앱 실행 처음 보여지는 화면으로 한 번만 나타나면 되지만, MainActivity에서 뒤로 가기 버튼을 누르게 되면 SplashActivity로 돌아간다.
* 문제 해결
  + 한번 누르면 Toast를 띄워 사용자에게 알림을 확인시켜 주고 두 번 누르게 되면 앱 전체가 종료되게 구성했다.
  + onBackPressed()를 사용하여 첫 Toast 메시지를 띄운 후 2.5초 안에 한 번 더 누르게 되면 앱 종료, 2.5초 후에 한 번 더 누르게 되면 초기화된다.

- 

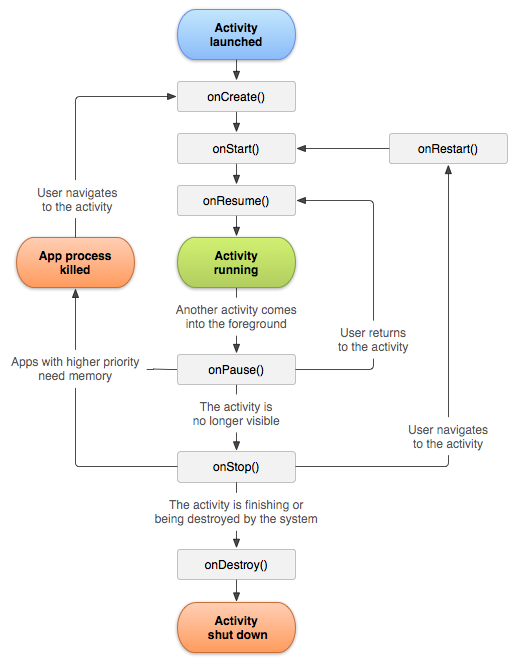
[그림 49] TemActivity

* EditText, ImageButton, LinerLayout로 xml 구성
* EditText를 사용해 온도, 습도를 입력 후 putExtra, Intent를 사용하여 설정 완료 버튼을 누르면 MainActivity의 TextView(1)로 값을 전달되는 동시 MainActivity로 자동 화면 전환 후 Toast box 설정 완료를 띄운다. 또한, 버튼 클릭 시 Tem(온도), Moi(습도) 값이 Service로 거쳐 Server로 값을 보내주는 역할을 하고 있다.



[그림 50] EatActivity

* TextView, Switch, ImageButton, LinerLayout로 xml 구성
* Switch On
* 자동 설정 Toast box를 띄운 후 Service로 1을 보낸 후 Server로 전달한다.
* Switch Off
* 수동 설정 Toast box를 띄운 후 Service로 0을 보낸 후 Server로 전달한다.



[그림 51] 생명주기 개요도

* 안드로이드는 앱이 실행된 후 다른 액티비티 화면으로 전환되거나, 스마트폰 화면이 꺼지거나 혹은 앱이 종료될 때와 같이 상태 변화가 있을 때마다 화면에 보이는 액티비티의 생명 주기 메서드를 호출해서 상태 변화를 알려준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 52] 생명주기 메서드

* 생명 주기 호출
  + 액티비티는 인스턴스 생성과 동시에 생성 관련 생명 주기 메서드가 순차적으로 호출된다.
  + 액티비티를 종료하면 소멸과 관련된 생명 주기 메서드가 순차적으로 호출된다.
  + 액티비티 생성

1. onCreate() → 생성된 화면 구성요소를 메모리에 로드

2. onStart(), onResume() → 화면의 구성요소를 나타내고 사용자와 상호작용 시작(Resumed: 실행 중)

* 액티비티 화면에서 제거

1. onPause(), onStop() → 뒤로 가기, finish()를 실행할 때 동시에 실행

2. onDestory() → 최종적으로 액티비티가 메모리에서 제거

* 액티비티를 종료하지 않고 다른 액티비티 실행

1.onPause(), onStop() → 현재 액티비티를 종료하지 않고 새로운 액티비티가 만들어질 때(Stopped)

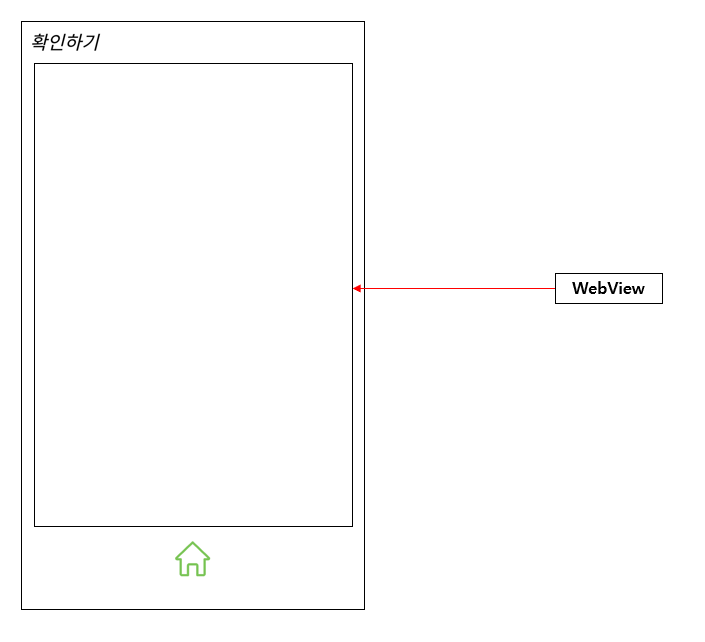
2.onStart(), onResume() → 두 메서드가 연속적으로 실행되고 Resumed 상태로 변경

* 액티비티를 종료하지 않거나, 모두 가려지지 않을 때 다른 액티비티 실행

1. onPause() → 완전히 사라진 것은 아니므로 Paused 상태로 변경

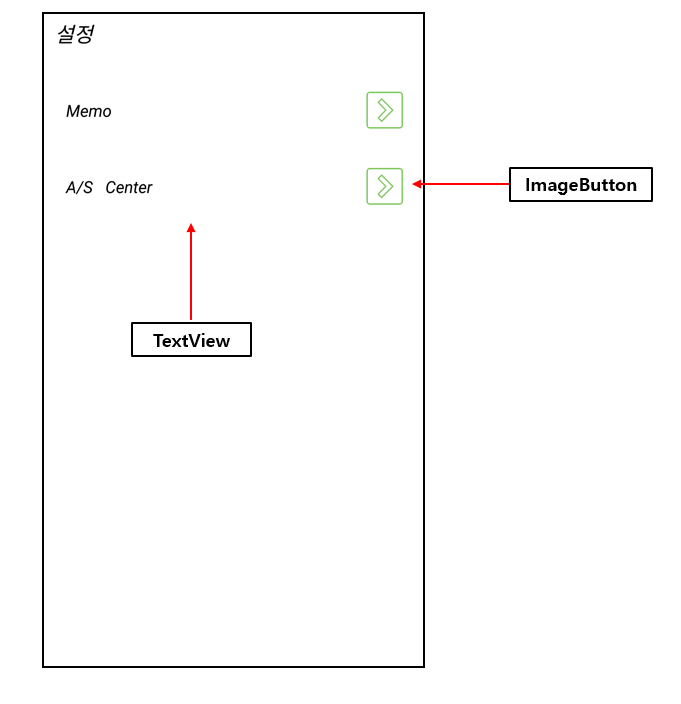
2. onResume() → 정지가 아니니 onStart를 거치지 않고 바로 onResume로 Resumed

* 문제 해결
* Switch 버튼을 ON으로 했을때 Preference에 원하는 변수를 넣은 후 SharedPreferences는 주로 어플리케이션 세팅에 쓰이는 데 이를 응용해서 액티비티가 실행될 때 넣어 놨던 Preference값을 보고 ON으로 했을 때의 변수가 나오는지 검사 후 스위치를 ON 혹은 OFF로 돌리면 된다.

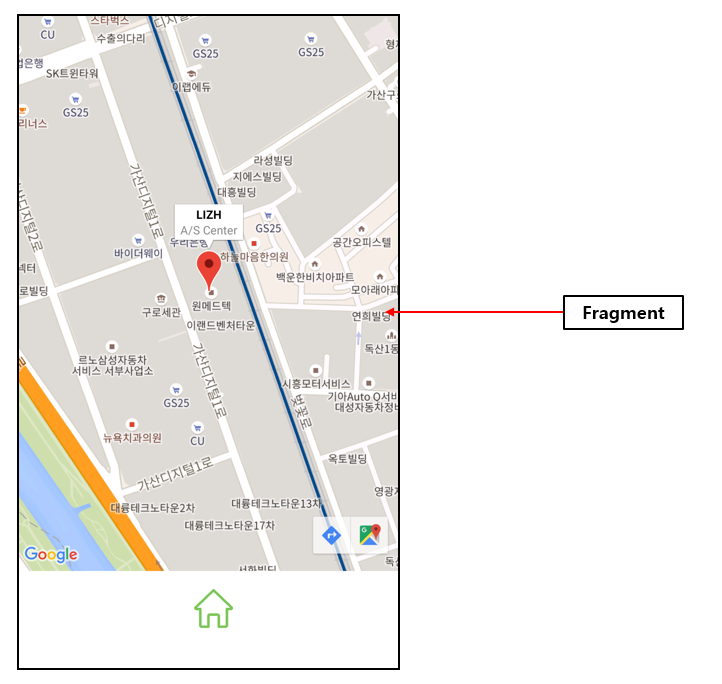


[그림 53] CCTVActivity

* ImageButton, WebView, LinerLayout로 xml 구성
* WebView 와 URL주소를 String으로 값을 넣고 특정 웹 사이트를 띄울 수 있다.
* Raspberry Pi의 카메라 URL 주소를 입력하여 App에서 직접 확인할 수 있다.



[그림 54] SettingActivity

* TextView, ImageButton, LinerLayout로 xml 구성
* TextView와 ImageButton에 setOnClickListener을 사용하여 클릭 시 다른 Activity로 넘어가게 구현했다.
* 또 다른 방법으로는 LinerLayout에 RecyclerView 또는 listView를 추가하여 Listitem.xml을 따로 넣는 방법이 있다. 이렇게 구현하게 된다면 한 xml을 Setting화면 xml에 추가하여 두 가지 클릭 옵션이 있는 방법보다 한 가지의 클릭 옵션이 있어 더 유용할 것 같다. 

[그림 55] AscenterActivity

* Fragment, ImageButton, LinerLayout로 xml 구성
* Google Map Api를 Android studio에 등록 후 불러왔습니다. 그 후 원하는 Location을 등록하기 위해 LatLng 값을 입력한다.
* Marker 와 title을 입력하여 위치를 정확하게 나타내고 설명을 추가했다.

1. ***카메라***

6.1 카메라 하드웨어

* 라스베리파이 카메라 모듈 2 (Raspberry Pi Camera Module 2)

 장난감이(가) 표시된 사진

자동 생성된 설명 [그림 56] PiCameraV2\_1 [그림 57] PiCameraV2\_2

* 사양 개요
  + 2016년 4월 출시
  + Sony IMX219 카메라 센서 사용
  + 최대 4k 해상도의 이미지 지원
  + 1080p30 혹은 720p30 영상 지원
  + 50cm 카메라 전용 케이블
* 상세 사양

|  | Camera Module v1 | **Camera Module v2** | HQ Camera |
| --- | --- | --- | --- |
| 크기 | 25 × 24 × 9 mm | **25 × 24 × 9 mm** | 38 x 38 x 18.4mm |
| 무게 | 3g | **3g** |  |
| 정지 해상도 | 5 Megapixels | **8 Megapixels** | 12.3 Megapixels |
| 비디오 모드 | 1080p30, 720p60 and 640 × 480p60/90 | **1080p30, 720p60 and 640 × 480p60/90** | 1080p30, 720p60 and 640 × 480p60/90 |
| 리눅스 통합 | V4L2 driver available | **V4L2 driver available** | V4L2 driver available |
| C 프로그래밍 API | OpenMAX IL and others available | **OpenMAX IL and others available** |  |
| 센서 | OmniVision OV5647 | **Sony IMX219** | [Sony IMX477](https://www.sony-semicon.co.jp/products/common/pdf/IMX477-AACK_Flyer.pdf) |
| 센서 해상도 | 2592 × 1944 pixels | **3280 × 2464 pixels** | 4056 x 3040 pixels |
| 센서 이미지 영역 | 3.76 × 2.74 mm | **3.68 x 2.76 mm (4.6 mm diagonal)** | 6.287mm x 4.712 mm (7.9mm diagonal) |
| 픽셀 크기 | 1.4 µm × 1.4 µm | **1.12 µm x 1.12 µm** | 1.55 µm x 1.55 µm |
| 광학 크기 | 1/4" | **1/4"** |  |
| 피사계 심도 | approx. 1 m to infinity | **adjustable with supplied tool** | N/A |
| 초점거리 | 3.60 mm +/- 0.01 | **3.04 mm** | Depends on lens |
| 가로 관측 시야 | 53.50 +/- 0.13 degrees | **62.2 degrees** | Depends on lens |
| 세로 관측 시야 | 41.41 +/- 0.11 degrees | **48.8 degrees** | Depends on lens |
| 초점비 (F-Stop) | 2.9 | **2.0** | Depends on lens |

[표 7] 파이 카메라 상세 사양표

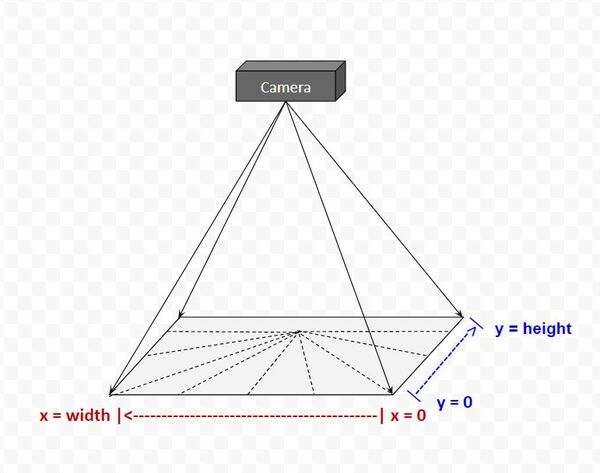
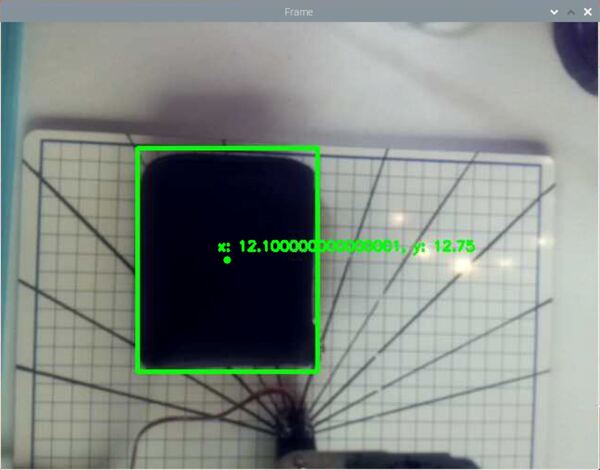
6.2 카메라 기능 개요

* 반려동물 CCTV 기능의 필요성

많은 반려인들이 집을 비우게 됐을 때 반려동물이 잘 지내는지 확인하고자 한다. 그래서 개/고양이와 같은 반려동물을 실시간으로 확인할 수 있게 해주는 CCTV가 이미 시중에 존재한다. 따라서 스마트 하우스도 파이 카메라 모듈 v2를 이용한 CCTV 기능을 구현하기로 결정했다.

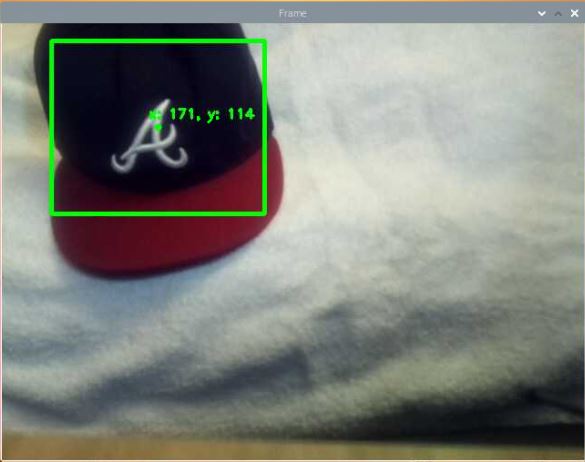
* CCTV 기능 초기 구상안
  + 도마뱀 크기 측정 기능

OpenCV 라이브러리를 이용해서 피사계심도(FOV)와 해상도 등을 이용해서 물체의 크기를 대략 추정할 수 있다. 하지만 사용자가 도마뱀의 크기를 직접 측정하는 것이 보다 정확하며, 이 기능이 정말 필요하다면 추가로 초음파 센서 등을 부착하는 것이 더 정밀하다. 따라서 카메라에 이 기능을 추가하는 것은 보류하기로 정했다.

[그림 58] 피사계심도 응용 [그림 59] OpenCV 응용: 크기 추정

* 도마뱀 움직임 감지
  + OpenCV 라이브러리를 이용해서 배경 차분 알고리즘을 이용하면 비교적 간단하게 구현 가능하다. 배경 차분 방식은 특정 배경 모델을 기반으로 현재 영상을 배경에 반영하고 배경 영상과 현재 영상의 차이를 이용하여 전경을 분리하는 방법이다. 하지만 일반적으로 도마뱀은 야행성이기 때문에 낮 시간에 이 기능의 실용성에 의문점이 존재해서 최종 기능에서는 배제하기로 결정하였다.

[그림 60] 배경 차분 방식 예제1 [그림 61] 배경 차분 방식 예제2

6.3 카메라 스트리밍 및 결론

* Motion 라이브러리와 Flask 모두 실시간 카메라 스트리밍에 유용한 기능들을 제공한다. Motion은 Raspbian Buster OS에서 daemon을 통한 자동 시작 같은 편리함을 제공해주지만, 수동으로 설정하기에 불편한 점이 있다. Flask의 경우 초기 설정이 비교적 더 까다롭지만, 확장성과 제어에 있어 용이함이 있어 최종적으로 Flask 기반의 웹 스트리밍을 하기로 결정했다. 안드로이드 앱에서 CCTV를 사용할 때 AndroidWebView를 이용해서 렌더링 해 사용자에게 표시를 하기로 했다. 또한 라스베리파이 하드웨어에 부담을 덜어주기 위해서 상시 카메라를 사용하는 것이 아니라 사용자의 요청이 있을 때만 카메라가 동작하도록 설정하였다.

텍스트, 실내, 모니터, 녹색이(가) 표시된 사진

자동 생성된 설명

[그림 62] 스트리밍 샘플 스크린샷

텍스트이(가) 표시된 사진

자동 생성된 설명

[그림 63] 카메라 스레드 자동 중단 기능

**Ⅲ. 결론**

핵가족화로 인해 반려동물에 대한 사람들의 관심도가 높아지고 있으며, 강아지와 고양이와 같은 반려동물뿐만 아니라 희귀 반려동물인 도마뱀, 거북이, 뱀과 같은 동물에 대한 사육도 증가하고 있는 추세이다. 그러나 반려인의 부재에 따른 영향이 매우 큰 변온동물들을 위한 사육환경 공급은 부족한 상황이다. 이러한 상황에 맞춰 본 프로젝트에서는 반려인의 도움 없이도 동물들에게 생존환경을 보장할 수 있는 스마트 사육장과 그와 연동된 어플리케이션을 제작하였다.

사육장의 경우 센서와 릴레이 모듈, 여러가지 제어 장치를 사용하여 온도, 습도, 먹이, 급수를 통제한다. 가습 및 급수에 필요한 물의 부족, 사육장의 온습도 이상범위 감지, 먹이 부족 등과 같이 동물의 생존에 영향을 미치는 상황일 때 이러한 정보를 서버를 통해 어플리케이션으로 보내주도록 구현하였다. 현재 센서의 오류율은 5% 이하로 유지되고 있으며, 제어 상태에 있어서 미 작동 횟수는 없을 정도로 인식 및 제어부분에서 높은 신뢰도를 보이고 있다.

어플리케이션은 사육장을 통제하는 라즈베리파이와의 통신을 통해 전송받은 데이터로 현재의 온습도를 표시하며, 특정 온도를 설정한 후 이를 라즈베리파이로 전송해 사육장의 온도를 조절할 수도 있다. 급여기의 자동, 수동모드를 변경할 수도 있으며, 먹이나 물이 부족할 때나 온습도의 이상범위를 감지했을 때 팝업경고 알림을 확인할 수 있도록 개발하였다. 반려인이 외출했을 경우, 카메라 스트리밍을 통해 외부에서 해당 반려동물의 움직임을 확인할 수도 있다.

본 프로젝트는 도마뱀, 뱀과 같은 변온동물을 키우는 반려인들과 예비 반려인들의 외출에 대한 부담과 걱정을 줄여줄 수 있다. 또한 외부에서 어플리케이션을 사용하여 반려인의 반려동물을 실시간으로 관찰할 수 있기 때문에, 반려동물에 대한 애정도의 상승도 기대된다.

**Ⅳ. 참고문헌**

[1] 윤석준 외 3명. 「파충류 사육을 위한 스마트 환경제어시스템」, 「한국산업기술대학교 학술발표대회 논문집」, 2019년, 249-252

[2] 박재현 외 5명, 「라즈베리파이와 온습도센서를 이용한 자연식가습기」, 「안동대학교 학술발표대회 논문집」, 2019년, 218-220

[3] Naver Post, “개코 도마뱀을 키우고 싶다면?”, 21.11.9

<https://m.post.naver.com/viewer/postView.nhn?volumeNo=9317240&memberNo=37963193>

[4] KEEPING ExoticPets, “Leopard Gecok Temperature: How Warm Should My Vivarium Be?”, 21.11.9

<https://www.keepingexoticpets.com/leopard-gecko-temperature/>

[5] Naver Post, “판게아 슈퍼푸드 브리딩포뮬러 454g 크레스티드게코사료 크레먹이 붙이류먹이 도마뱀사료”, 21.11.10

<https://m.blog.naver.com/peren4754/222486793377>

[6] Naver Post, “파충류 사육 시 사용하는 스팟 램프의 종류”, 21.11.10

<https://m.blog.naver.com/metal6125/221187417545>

[7] Tistroy, “피스톤 운동 방정식”, 21.11.12

<https://gammabeta.tistory.com/219>

[8] Verge, “The Automatic Reptile Feeder”, 21.11.25

<https://www.vergepets.com/>