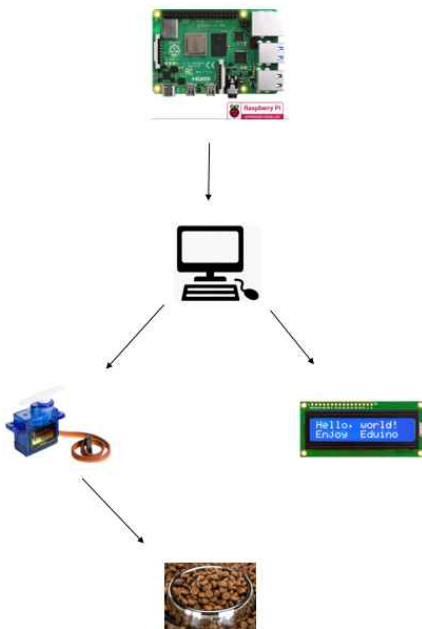


[12조] 완료 보고서

프로젝트 과목	IoT시스템응용(02)	프로젝트 기간	2023.11.01 ~ 12.05
12조	김주혁/20182229, 김기현/20214934, 최규동/20213045		
프로젝트명	강아지 사료 자동 급식기		
프로젝트 개요 및 목표	<p>■ 사용자가 웹 페이지에 사료 급여 횟수와 각 횟수별 배급 시간들을 입력하면 해당 시간에 맞추어 작동해 사료를 배급하는 자동 급식기</p> <ul style="list-style-type: none"> - 매일 정해진 급여시간으로 강아지의 체중과 단식을 관리 - 외출 또는 여행 시 강아지의 사료 자동 급여 가능 - 시중의 자동 급식기를 구매 시 드는 높은 비용을 피해 직접 제작 		
개발 과정	<p>■ 계획 단계</p> <ul style="list-style-type: none"> - 프로젝트 계획서 작성 <p>■ 설계 단계</p> <ul style="list-style-type: none"> - 서보모터, LCD 작동 모듈 및 웹 페이지 설계 - 사료통 설계 <p>■ 구현 단계</p> <ul style="list-style-type: none"> - 서보모터, LCD 작동 모듈 및 웹 페이지 구현 - 사료통 구현 <p>■ 테스트 단계</p> <ul style="list-style-type: none"> - 단위 테스트 수행 - 통합 테스트 수행 		
시스템 구축 환경	<p>■ 하드웨어 구성도</p> <p>1. 전체 구성도</p> 		

	<p>2. 하드웨어 구성</p> <ul style="list-style-type: none"> - 라즈베리 파이 - 모니터, 마우스, 키보드 - 서보모터 - 1602 LCD Display - 사료통 <p>■ 소프트웨어 구성도</p> <p>1. 전체 구성도</p> <pre> graph LR W1[웹 페이지1 (index.html) count 입력] --> W2[웹 페이지2 (submit.html) count 수 만큼 feeding_times 입력] W2 --> W3[웹 페이지3 (result.html) current time, feeding_time 웹 페이지 화면에 출력] W3 -- "Feeding_times 전달" --> SM[servo_motor (index.py) current time과 feeding_times가 일치하게 되면 servo_motor 작동] SM --> AF[사료 자동 급식기 작동 => 사료 배식] W3 -- "Feeding_times 전달" --> LCD[lcd (index.py) 몇 골에 년/월/일/시간/분/초를 아랫줄에 feeding_times를 각각 3초씩 출력] </pre>
역할 및 책임	<p>■ 김주혁(팀장)</p> <ul style="list-style-type: none"> - Flask를 활용해 웹 서버를 구축한 후 웹 페이지 구현 - 보고서 작성 - 급식기 제작 <p>■ 김기현</p> <ul style="list-style-type: none"> - 1602 Lcd Display 코드 구현 - 급식기 제작 <p>■ 최규동</p> <ul style="list-style-type: none"> - 서보 모터 코드 구현 - 급식기 제작
코드 구현	<p>1. index.py</p> <p>: 크게 5부분으로 구성 - 선언부, GPIO pin 설정과 초기화, 서보모터 함수, LCD Display 함수, Flask application에서 들어오는 HTTP 요청 처리</p> <p>(1) 라이브러리와 모듈 선언</p> <pre> from flask import Flask, request, render_template, jsonify import datetime import threading import time from RPLCD.i2c import CharLCD import board import busio import RPi.GPIO as GPIO </pre>

(2) LCD와 서보모터 설정

- LCD는 CharLCD 라이브러리(I2C 통신을 사용하여 문자 LCD를 제어하기 위한 라이브러리)를 사용하여 초기화

```
# LCD 설정
lcd_columns = 16
lcd_rows = 2
i2c = busio.I2C(board.SCL, board.SDA)
lcd = CharLCD(i2c_expander='PCF8574', address=0x27, port=1, cols=lcd_columns, rows=lcd_rows)

# 서보모터 설정
SERVO_PIN = 18
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(SERVO_PIN, GPIO.OUT)
servo = GPIO.PWM(SERVO_PIN, 50)
servo.start(0)
```

(3) 서보모터 함수

- 전역 함수인 servo_motor_running이 web page3(result.html)로부터 feeding_times를 받으면 True가 되고 while문 실행됨. 현재 시간(00초 일 때)과 feeding_times를 비교하여 같으면 서보모터가 90도 -> 0도로 작동함.

```
def run_servo_motor():
    global servo_motor_running, feeding_times

    while servo_motor_running:
        current_time = datetime.datetime.now().strftime("%H:%M:%S")
        if current_time.endswith(":00"):
            if current_time[:-3] in feeding_times:
                servo.ChangeDutyCycle(7.5)
                time.sleep(1)
                servo.ChangeDutyCycle(2.5)
                time.sleep(1)

        time.sleep(1) # 1초마다 확인

    print("Servo motor stopped.")
    servo.stop()
    GPIO.cleanup()
```

(4) LCD Display 함수

- 무한 루프를 실행, LCD 디스플레이 clear 후 윗 줄에 현재 시간 표시, (1,0): 아래줄로 이동 후 feeding times 출력 디스플레이 clear 후 현재 시간 표시하고 2번째 줄로 이동 후 다시 for문으로 feeding times 출력

```
def display_lcd():
    global feeding_times

    while True:
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M")
        lcd.clear()
        lcd.write_string(current_time)
        lcd.cursor_pos = (1, 0)
        if feeding_times:
            for time_val in feeding_times:
                lcd.write_string(time_val)
                time.sleep(3) # 2개 feeding times을 3초간 표시
            lcd.clear()
            lcd.write_string(current_time)
            lcd.cursor_pos = (1, 0)
        else:
            lcd.write_string("No feeding times")
            time.sleep(5) # 5초 동안 메시지 표시
            lcd.clear()
```

(5) HTTP 요청 처리

- 각 경로의 POST 또는 GET 요청을 처리하고, result() 부분에서 전역 변수 servo_motor_running가 False일 때, 실행되고 사용자가 입력한 feeding_times를 가져오고 True 바뀌서 위의 서보모터 함수의 while문이 돌아가게 한다.

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    count = int(request.form['count'])
    return render_template('submit.html', count=count)

@app.route('/result', methods=['POST'])
def result():
    global feeding_times, servo_motor_running

    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    count = int(request.form['count'])

    new_feeding_times = []
    for i in range(1, count + 1):
        time_val = request.form.get(f'time{i}', '')
        new_feeding_times.append(time_val)

    if not servo_motor_running:
        feeding_times = new_feeding_times
        servo_motor_running = True
        servo_thread = threading.Thread(target=run_servo_motor)
        lcd_thread = threading.Thread(target=display_lcd)
        servo_thread.start()
        lcd_thread.start()

    return render_template('result.html', current_time=current_time, feeding_times=feeding_times)

@app.route('/get_current_time')
def get_current_time():
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    return jsonify(current_time=current_time)

@app.route('/get_feeding_times')
def get_feeding_times():
    global feeding_times
    return jsonify(feeding_times=feeding_times)

if __name__ == '__main__':
    app.run(debug=True)
```

2. index.html

- 급식 횟수를 입력하는 폼을 생성하는 코드로 입력한 feed count는 POST방식으로 “/submit”에 전송한다.

```
<html>
<head>
    <title>Enter feed count</title>
    <style>
        /* css code */
    </style>
</head>
<body>
    <h2>Enter feed count</h2>
    <form action="/submit" method="post">
        <label type="count">Feed count:</label>
        <input type="number" id="count" name="count" min="1" required>
        <input type="submit" value="Enter">
    </form>
</body>
</html>
```

3. submit.html

- 사용자에게 급식 시간을 입력하도록 하는 폼을 생성하는 코드로 count만큼 반복되는 시간 입력 폼을 만들고 입력한 시간들은 POST 방식으로 “/result”에 전송된다

```
<html>
<head>
  <title>Enter feed time</title>
</head>
<body>
  <h2>Enter feed time</h2>
  <p>Example: 9:30, 12:00, 21:30</p>
  <form action="/result" method="post">
    <input type="hidden" name="count" value="{{ count }}">
    {% for i in range(1, count +1) %}
      <label for="time{{ i }}">Count {{ i }} time (h m):</label>
      <input type="text" name="time{{ i }}" id="time{{ i }}"><br>
    {% endfor %}
    <input type="submit" value="Enter">
  </form>
</body>
</html>
```

4. result.html

- current time과 feeding times를 업데이트하고 표시하는 부분이다

```
<html>
<head>
  <title>Feeding time results</title>
</head>
<body>
  <h2>Feeding time results</h2>
  <p>Current time: <span id="current_time"></span></p>
  <p>Feeding times:</p>
  <ul id="feeding_times">
    {% for time in feeding_times %}
      <li>{{ time }}</li>
    {% endfor %}
  </ul>

  <script>
    function updateCurrentTime() {
      var currentTimeSpan = document.getElementById('current_time');

      function displayTime() {
        fetch('/get_current_time')
          .then(response => response.json())
          .then(data => currentTimeSpan.textContent = data.current_time);
      }

      displayTime();
      setInterval(displayTime, 1000); // 1초마다 현재 시간 업데이트
    }

    function displayFeedingTimes() {
      fetch('/get_feeding_times')
        .then(response => response.json())
        .then(data => {
          const feedingTimes = data.feeding_times || [];
          const feedingTimesList = document.getElementById('feeding_times');

          // 기존 리스트 초기화
          feedingTimesList.innerHTML = '';

          // 새로운 시간들로 리스트 업데이트
          feedingTimes.forEach(time => {
            const listItem = document.createElement('li');
            listItem.textContent = time;
            feedingTimesList.appendChild(listItem);
          });
        });
    }

    setInterval(displayFeedingTimes, 1000); // 1초마다 먹이 시간 업데이트
    displayFeedingTimes(); // 페이지 로드 시 먹이 시간 업데이트
    updateCurrentTime(); // 페이지 로드 시 현재 시간 업데이트
  </script>
</body>
</html>
```

시연

1. 데이터 입력 (웹 페이지)

- (1) count = 3
- (2) feeding times
 - time 1 : 9:00
 - time 2 : 12:00
 - time 3 : 18:00

①

Enter feed count

Feed count:

②

Enter feed time

Example: 9:30, 12:00, 21:30

Count 1 time (h m):
Count 2 time (h m):
Count 3 time (h m):

③

Feeding time results

Current time: 2023-12-05 15:35:51

Feeding times:

- 9:00
- 12:00
- 18:00

2. LCD Display 작동

(* 시연한 LCD 사진을 찾지 못해 다른 테스트 사진으로 대체함)



3. 서보모터 동작

- 급식기 내부 모습으로 서보모터가 화살표 방향으로 동작하여 사료 배식



4. 사료 배식

- (* 시연한 사진을 찾지 못해 다른 테스트 사진으로 대체함)



추가 및
변경 사항

1. Web Page2 입력 예시 추가 (11/15)

- Web Page2에서 feeding times 입력 예시로 “Example: 9:30, 12:00, 21:30”을 추가

2. LCD Display 화면 출력 방법 변경 (11/22)

- 기존 : LCD Display의 feeding times 출력이 한 칸씩 왼쪽으로 이동하면서 보여줌

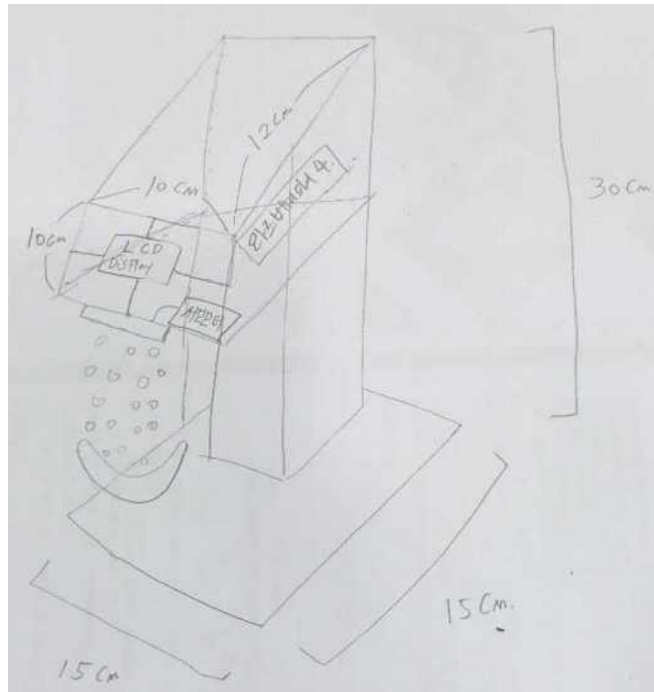
=> feeding times를 한 개씩 1개씩 보여줌

=> (11/28 변경) 1초->3초

3. 사료 급식기 변경

■ 기존

- 사료 급식기 전체를 직접 제작



■ 변경

- 박스를 이용



<p>수행 내역 (주차별)</p>	<ul style="list-style-type: none"> - 9주차 : 프로젝트 계획서 작성 및 조원간 역할 분담 - 10주차 : Flask를 활용하여 웹 서버 구축한 후 웹페이지 구현, 서보모터 코드 구현 시작 - 11주차 : 서보모터, LCD Display 코드 구현 - 12주차 : 단위/통합 테스트 진행, 코드 수정 및 최종 확인 - 13주차 : 사료 급식기 제작 및 기기 장착 - 14주차 : 프로젝트 발표 및 시연
<p>재료</p>	<ul style="list-style-type: none"> - 라즈베리파이 - 암암/암수/수수 점퍼선 - 1602 LCD display - 서보 모터 - 우드락 - 맥심 모카골드 마일드 커피믹스 230T 박스 - 글루건