
모바일&스마트 시스템 미니프로젝트

스마트 도서 관리 결과 보고서

과 목	모바일&스마트시스템
분 반	B
담당교수	황기태
제 출 일	2021.12.01
트 랙	모바일소프트웨어
학 번	1771059
이 름	김준수

1. 작품 개요

종이로 되어있는 책은 온도와 습도에 민감하여 관리에 주의가 필요하다. 책을 보관하기에 가장 이상적인 환경은 섭씨 18~21도, 습도는 30~50%의 직사광선이 들어오지 않는 곳으로 알려져 있지만, 일반 가정에서 온도와 습도를 완벽하게 확인하여 책을 관리하기는 어렵다. 이에 따라 책의 관리를 쉽게 할 수 있는 스마트 도서 관리 시스템을 제안한다.

기본적인 베이스인 책의 최적인 환경인 온도와 습도를 조정하기 위해서는 현재 환경의 값을 알아야 한다. 해당 값을 측정하기 위해 HTU21D 센서를 활용한다. 해당 값들은 웹 사이트의 그래프에 시각적으로 표기되어 사용자가 현재 환경을 책 관리에 최적이 되도록 조정할 수 있게 도움이 된다. 자체 난방 시스템 등을 구현하기는 어려움으로 표기하는 수준까지 구현하였다.

부가적으로 책장의 거리를 수시로 측정하도록 초음파 센서와 표지 촬영을 위한 카메라가 활용된다. 스위치를 초기화 버튼으로 활용하여, 첫번째로 책장의 거리를 측정한 뒤, 새로운 책이 책장에 들어선다면 거리 값이 줄어드는 것을 활용한다. 그리고 해당 값을 계산하고 책의 두께와 표지를 저장하여 라즈베리파이에 저장하도록 한다. 해당 값을 다시 재가공하여 캔버스에 시각적으로 표기되고, 원격에서 현재 서고의 현황을 대략적으로 가늠할 수 있게 데이터를 제공한다. 그리고 해당 데이터에 대한 세부 정보 (책 제목, 지은이, 출판사 등)를 웹사이트에서 수정 및 입력하여 라즈베리파이에 저장할 수 있게 하여 관리에 도움이 되게 한다.

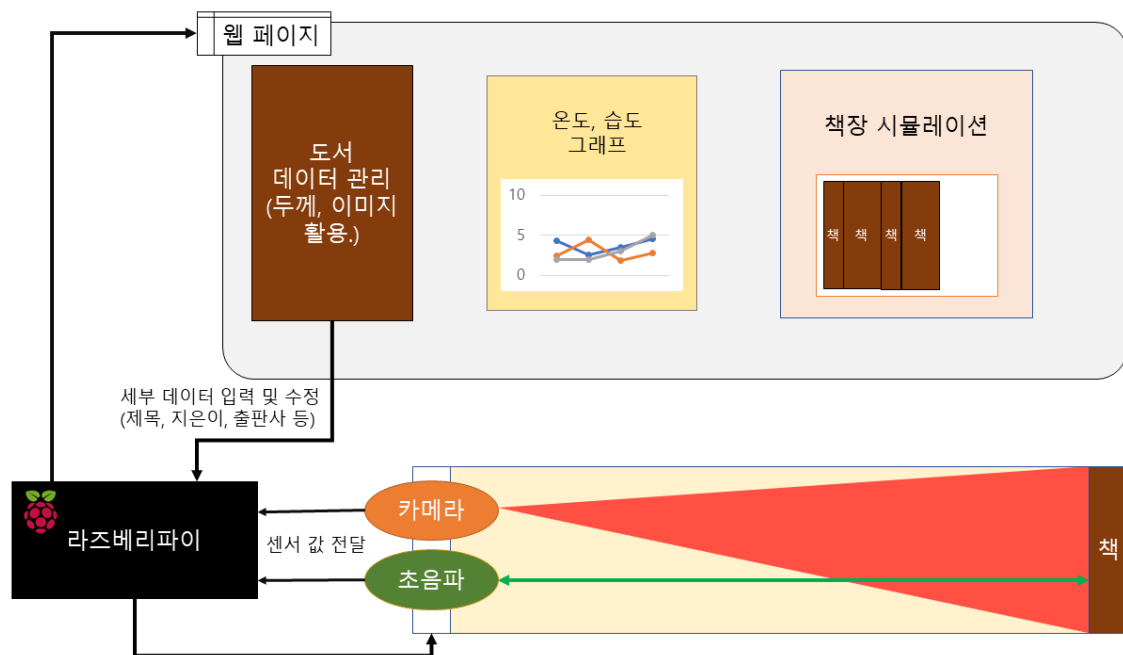


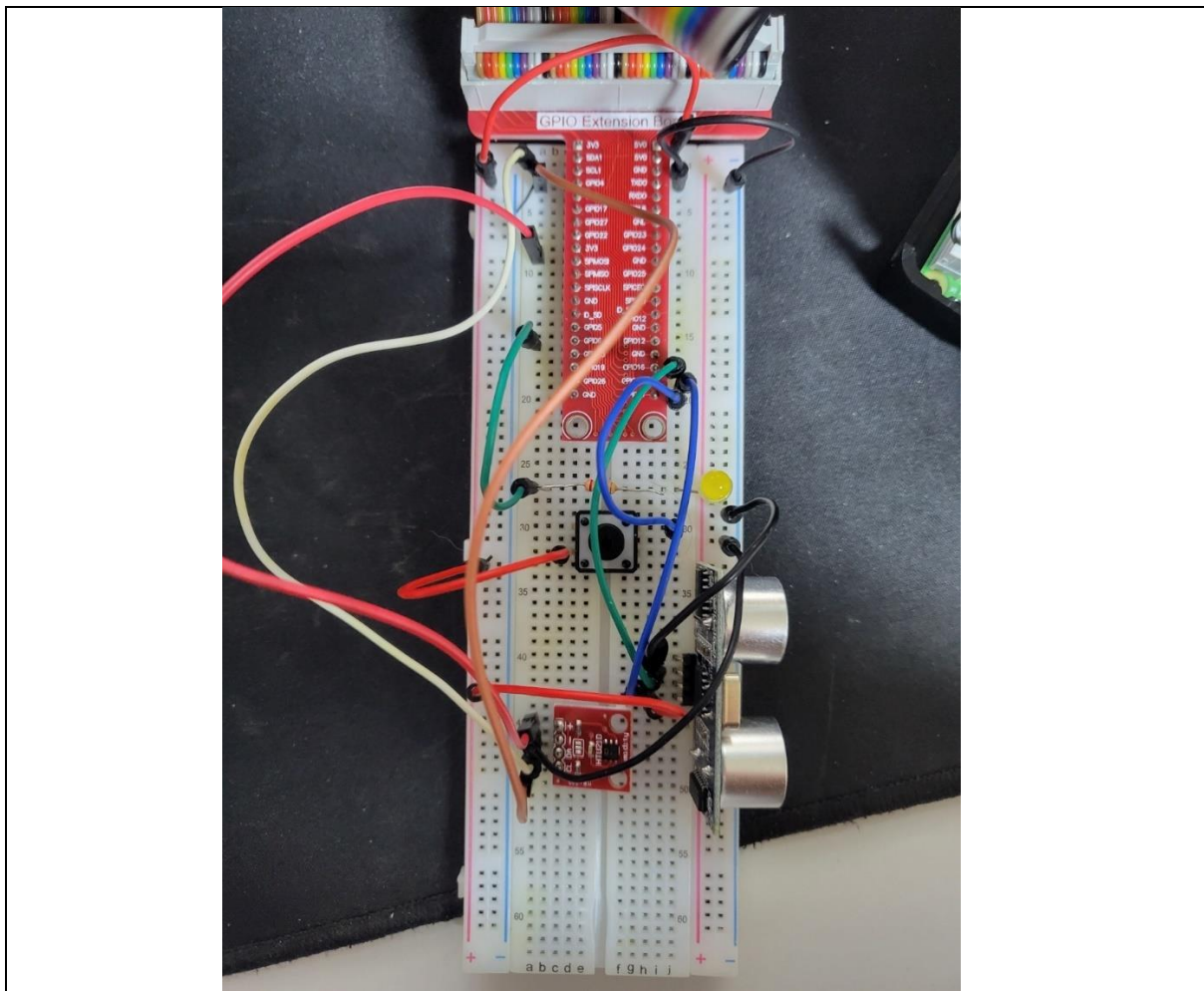
사진1. 이해를 돕기 위한 시스템 구조.

2. 구현 방법

2.1. 하드웨어

순번	하드웨어	핀 번호 (GPIO)	비고
1	라즈베리파이	-	서버, 데이터, 센서 관리 등 종합 관리.
2	파이 카메라	-	책 표지 촬영.
3	초음파 센서	Echo: 16 Trig: 20	센서와 책 사이의 거리를 측정하여, 새로운 책이 들어왔는지 판단.
4	스위치	21	책장 사이의 거리를 측정함. 초기 설정 세팅용.
5	HTU21D (온습도 센서)	-	책 관리 환경에 필요한 온도와 습도를 측 정.
6	LED	6	초음파 측정 중 확인용.

2.1.1 회로도 사진




2.2 소프트웨어

순번	이름	작동 방식	비고
1	웹 페이지	HTML	데이터를 시각적으로 표시.
2	자바 스크립트	JavaScript	데이터 가공, 웹 페이지 동작, 그래프 표시, 캔버스 표시, MQTT 관련 함수 실행 등.
3	CSS	CSS	그냥 웹페이지 디자인.
4	플라스크	Python (Flask)	웹 브라우저 접속 및 제어.
5	장치 관리	Python	센서 장치 제어 및 데이터 관리.
6	MQTT	Python, JS	온도, 습도 값 전송.

2.2.1. 웹 페이지

Login


로그인

로그인용 웹 페이지, 간단하게 post 방식을 사용하여 처리한다.

CSS

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}  
a {  
    text-decoration: none;  
    color: black;  
}  
li {  
    list-style: none;  
}
```

```
.login {
width: 50%;
height: 600px;
background: white;
border-radius: 20px;
display: flex;
justify-content: center;
align-items: center;
flex-direction: column;
position: fixed;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
}
form{
width: 50%;
}
h2 {
color: #191970;
font-size: 2em;
}
.login_logo {
text-align: left;
}
.login_id {
margin-top: 20px;
width: 80%;
}
.login_id input {
width: 100%;
height: 50px;
border-radius: 30px;
margin-top: 10px;
padding: 0px 20px;
border: 1px solid lightgray;
outline: none;
}
.login_pw {
margin-top: 20px;
width: 80%;
}
.login_pw input {
width: 100%;
height: 50px;
border-radius: 30px;
margin-top: 10px;
padding: 0px 20px;
border: 1px solid lightgray;
outline: none;
}
.submit {
margin-top: 50px;
width: 80%;
}
.submit input {
width: 100%;
height: 50px;
border: 0;
outline: none;
border-radius: 40px;
background: linear-gradient(to left, rgb(25, 25, 112), rgb(100, 100, 180));
color: white;
font-size: 1.2em;
letter-spacing: 2px;
```

```
}
```

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <link href="static/login.css" type="text/css" rel="stylesheet">
  </head>
  <body>
    <div class="login">
      <form action="/main" method="post">
        <div class="login_logo">
          
        </div>
        <h2>로그인</h2>
        <div class="login_id">
          <input type="text" name="id" placeholder="아이디">
        </div>
        <div class="login_pw">
          <input type="password" name="pw" placeholder="비밀번호">
        </div>
        <div class="submit">
          <input type="submit" value="로그인">
        </div>
      </form>
    </div>
  </body>
</html>
```

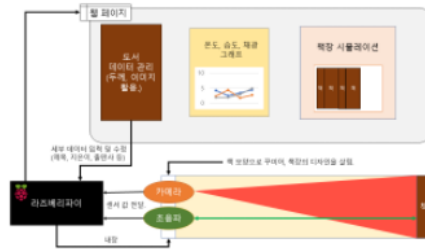


초기화면

내 책장 확인

책장 환경 확인

책 데이터 수정



스마트 도서 관리

스마트 도서 관리는 사용자들의 소중한 자산인 책을 보호 및 관리하게 편하게 해주는 서비스입니다.

메뉴 설명

초기화면: 해당 화면입니다.

내 책장 확인: 책장의 현 상황을 시각적으로 확인합니다.

책장 환경 확인: 책장의 환경을 그래프로 확인합니다.

책 데이터 수정: 저장된 데이터를 수정합니다.

1771059 김준수 제작.

설명용 index용도의 웹 페이지.

CSS

```

        @font-face {
            font-family: 'InfinitySans-RegularA1';
            src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/InfinitySans-RegularA1.woff') format('woff');
            font-weight: normal;
            font-style: normal;
        }
        @font-face {
            font-family: 'NanumBarunGothic';
            font-style: normal;
            font-weight: 400;
            src: url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.eot');
            src: url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.eot?#iefix') format('embedded-opentype'),
            url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.woff') format('woff'),
            url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.ttf') format('truetype');
        }
        * {
            font-family: 'NanumBarunGothic';
        }
        nav{
            width: 200px;
            height: 100%;
            background-color: #eee;
            border-right: 1px solid #ddd;
            position: fixed;
            font-family: 'InfinitySans-RegularA1';
        }
        ul{
            list-style: none;
            padding-inline-start: 10px;
        }
        .menu img{
            text-align: center;

```

```

        width: 100%;
    }
    .menu li a {
        height: 30px;
        line-height: 30px;
        display: block;
        padding: 10px 20px;
        font-size: 20px;
        text-decoration: none;
        color: #111;
    }
    .menu li a:hover{
        background-color: #191970;
        color: white;
    }
    section{
        padding: 20px 20px 20px 220px;
    }

```

HTML

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>초기화면</title>
    <nav>
        <ul class = "menu">
            <li></li>
            <li><a href="{{url_for('main', select='intro')}}">초기화면</a></li>
            <li><a href="{{url_for('main', select='book')}}">내 책장 확인</a></li>
            <li><a href="{{url_for('main', select='graph')}}">책장 환경 확인</a></li>
            <li><a href="{{url_for('main', select='data')}}">책 데이터 수정</a></li>
        </ul>
    </nav>
    <link href="static/index.css" type="text/css" rel="stylesheet">
</head>
<body>
<section>
    
    <h1>스마트 도서 관리</h1>
    <p>스마트 도서 관리는 사용자들의 소중한 자산인 책을 보호 및 관리하게 편하게 해주는 서비스
    입니다.</p>
    <h3>메뉴 설명</h3>
    <p>초기화면: 해당 화면입니다.</p>
    <p>내 책장 확인: 책장의 현 상황을 시각적으로 확인합니다.</p>
    <p>책장 환경 확인: 책장의 환경을 그래프로 확인합니다.</p>
    <p>책 데이터 수정: 저장된 데이터를 수정합니다.</p>
    <p style="text-align: right;">1771059 김준수 제작.</p>
</section>
</body>
</html>

```




초기화면

내 책장 확인

책장 환경 확인

책 데이터 수정

내 책장 현황

책 현황을 확인할 수 있습니다.



캔버스 갱신

번호	제목	출판사	지은이	두께	메모	사진 확인
1	테스트로로로로	테스트출판사	김테스트	20	테스트222	사진
2	명동 C++ Programming	생능출판	황기태	3	객체지향언어1 교재임시3	사진
3	임시3	출판사	지은이	3.95	수정이 필요합니다.	사진

책장을 캔버스로 그려서 보여주는 웹 페이지. flask에서 json 형식으로 데이터를 받아와서 캔버스에 표기한다. 사진은 책 제목과 함께 저장되기 때문에, 해당열의 제목을 가져와서 사진을 띄운다.

CSS

```

    @font-face {
      font-family: 'InfinitySans-RegularA1';
      src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/InfinitySans-RegularA1.woff') format('woff');
      font-weight: normal;
      font-style: normal;
    }
    @font-face {
      font-family: 'NanumBarunGothic';
      font-style: normal;
      font-weight: 400;
      src: url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.eot');
      src: url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.eot?#iefix') format('embedded-opentype'),
      url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.woff') format('woff'),
      url('//cdn.jsdelivr.net/font-nanumlight/1.0/NanumBarunGothicWeb.ttf') format('truetype');
    }
    * {
      font-family: 'NanumBarunGothic';
    }
    nav{
      width: 200px;
      height: 100%;
      background-color: #eee;
      border-right: 1px solid #ddd;
      position: fixed;
      font-family: 'InfinitySans-RegularA1';
    }
    ul{
      list-style: none;
      padding-inline-start: 10px;
  
```

```

}
.menu img{
    text-align: center;
    width: 100%;
}
.menu li a {
    height: 30px;
    line-height: 30px;
    display: block;
    padding: 10px 20px;
    font-size: 20px;
    text-decoration: none;
    color: #111;
}
.menu li a:hover{
    background-color: #191970;
    color: white;
}
section{
    padding: 20px 20px 20px 220px;
}
#modal {
    display: none;
    position: relative;
    width: 100%;
    height: 100%;
    z-index: 1;
}

#modal h2 {
    margin: 0;
}

#modal button {
    display: inline-block;
    width: 100%;
    /* margin-left: calc(100% - 100px - 10px); */
}

#modal table{
    width: 100%;
}
#modal input[type="text"]{
    width: 95%;
}
#modal .modal_content {
    width: 300px;
    margin: 100px auto;
    background: #fff;
}

#modal .modal_layer {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: -1;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin: 25px 0;
}

```

```

        font-size: 0.9em;
        font-family: sans-serif;
        box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
    }
    table thead tr {
        background-color: #191970;
        color: #ffffff;
        text-align: left;
    }
    table th,
    table td {
        padding: 12px 15px;
    }
    table tbody tr {
        border-bottom: 1px solid #dddddd;
    }
    table tbody tr:nth-of-type(even) {
        background-color: #f3f3f3;
    }
    table tbody tr:last-of-type {
        border-bottom: 2px solid #191970;
    }
    table tbody tr.active-row {
        font-weight: bold;
        color: #191970;
    }
}

```

HTML

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>내 책장 확인</title>
    <nav>
        <ul class = "menu">
            <li></li>
            <li><a href="{{url_for('main', select='intro')}}">초기화면</a></li>
            <li><a href="{{url_for('main', select='book')}}">내 책장 확인</a></li>
            <li><a href="{{url_for('main', select='graph')}}">책장 환경 확인</a></li>
            <li><a href="{{url_for('main', select='data')}}">책 데이터 수정</a></li>
        </ul>
    </nav>
    <link href="static/bookData.css" type="text/css" rel="stylesheet">
    <script type="text/javascript">
        const getRandomRGB = () => `rgb( ${new Array(3).fill().map(v => Math.random() *
255).join(", ")} )`;
        function openModal(button){
            tr = button.parentNode.parentNode;
            modal = document.getElementById("modal");
            modal.style.display="block";
            canvas = document.getElementById("imgCanvas");
            context = canvas.getContext("2d");
            img = new Image();
            img.onload = function () {
                context.drawImage(img, 0, 0, 360, 640);
            }
            img.src = 'static/pic/' + tr.children[1].innerText + '.jpg';
        }
        function closeModal(){
            document.getElementById("modal").style.display="none";
        }
        function draw() {
            var canvas = document.getElementById("canvas");

```



```
    </table>
</section>
</body>
</html>
```



초기화면

내 책장 확인

책장 환경 확인

책 데이터 수정

책 데이터 편집

책 데이터 편집 및 추가를 할 수 있습니다.

번호	제목	출판사	지은이	두께	메모	수정
1	테스트로로로로	테스트출판사	김테스트	20	테스트222	수정
2	명품 C++ Programming	생능출판	황기태	3	객체지향언어1 교재임시3	수정
3	임시3	출판사	지은이	3.95	수정이 필요합니다.	수정

책 데이터 확인 및 수정. Get 방식으로 flask에 보내고 해당 id 값을 바탕으로 저장한다.

CSS

Book과 같음.

HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>책 데이터 수정</title>
  <nav>
    <ul class = "menu">
      <li></li>
      <li><a href="{{url_for('main', select='intro')}}">초기 화면</a></li>
      <li><a href="{{url_for('main', select='book')}}">내 책장 확인</a></li>
      <li><a href="{{url_for('main', select='graph')}}">책장 환경 확인</a></li>
      <li><a href="{{url_for('main', select='data')}}">책 데이터 수정</a></li>
    </ul>
  </nav>
  <link href="static/bookData.css" type="text/css" rel="stylesheet">
  <script type="text/javascript">
    function openModal(button){
      tr = button.parentNode.parentNode;
      modal = document.getElementById("modal");
      modal.style.display="block";
      document.getElementById("title").innerText = tr.children[1].innerText + "의
내용 수정";
      document.getElementById("modal_id").value = button.id;
      document.getElementById("modal_title").value = tr.children[1].innerText
      document.getElementById("modal_publisher").value = tr.children[2].innerText
      document.getElementById("modal_writer").value = tr.children[3].innerText;
      document.getElementById("modal_len").value = tr.children[4].innerText;
      document.getElementById("modal_memo").value = tr.children[5].innerText;
    }
    function closeModal(){
      document.getElementById("modal").style.display="none";
    }
  </script>
</head>
<body>
<section>
  <h1>책 데이터 편집</h1>
  <p>책 데이터 편집 및 추가를 할 수 있습니다.</p>
```

```

<div id="modal">
  <div class="modal_content">
    <form action="/modify" method="get">
      <input type="hidden" id="modal_id" name="id" value="">
      <table>
        <thead>
          <td colspan="2"><h2 id = "title">수정</h2></td>
        </thead>
        <tbody>
          <tr>
            <td>제목</td>
            <td><input type="text" id="modal_title" name="title"
value=""></td>
          </tr>
          <tr>
            <td>출판사</td>
            <td><input type="text" id="modal_publisher" name="publisher"
value=""></td>
          </tr>
          <tr>
            <td>지은이</td>
            <td><input type="text" id="modal_writer" name="writer"
value=""></td>
          </tr>
          <tr>
            <td>두께</td>
            <td><input type="text" id="modal_len" name="len" value=""></td>
          </tr>
          <tr>
            <td>메모</td>
            <td><input type="text" id="modal_memo" name="memo"
value=""></td>
          </tr>
          <tr>
            <td colspan="2"><button type="submit" name="type"
value="save">저장</button></td>
          </tr>
          <tr>
            <td colspan="2"><button type="submit" name="type"
value="del">삭제</button></td>
          </tr>
          <tr>
            <td colspan="2"><button type="button" id="modal_close_btn"
onclick="closeModal();">닫기</button></td>
          </tr>
        </tbody>
      </table>
    </form>
  </div>
  <div class="modal_layer"></div>
</div>
<table>
  <thead>
    <td>번호</td>
    <td>제목</td>
    <td>출판사</td>
    <td>지은이</td>
    <td>두께</td>
    <td>메모</td>
    <td>수정</td>
  </thead>
  <tbody>
    {% for name in title : %}

```

```
<tr>
<td>{% print(loop.index) %}</td>
<td>{% print(name) %}</td>
<td>{% print(publisher[loop.index0]) %}</td>
<td>{% print(writer[loop.index0]) %}</td>
<td>{% print(len[loop.index0]) %}</td>
<td>{% print(memo[loop.index0]) %}</td>
<td>
    <button type="button" id="{{loop.index0}}"
onclick="openModal(this);">수정</button>
</td>
</tr>
{% endfor %}
</tbody>
</table>
</section>
</body>
</html>
```


Chart

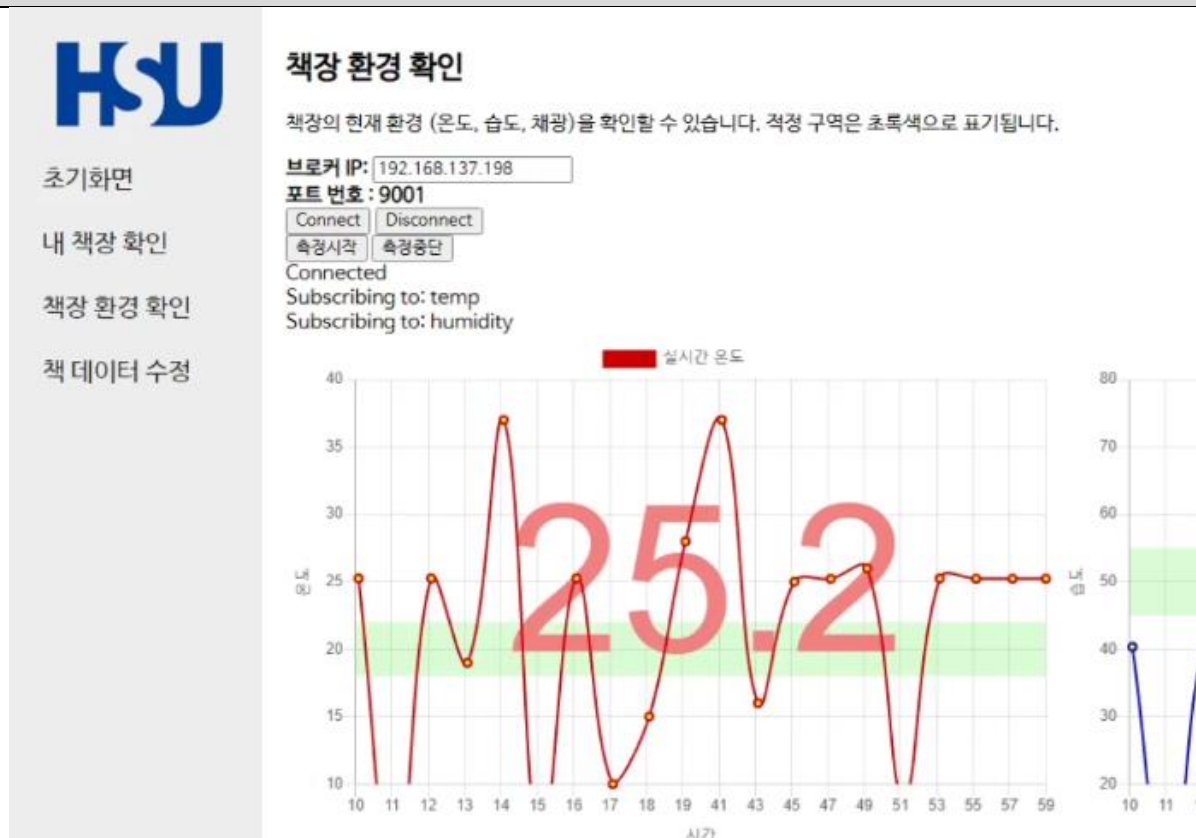


차트 표시하는 웹페이지. Mqtt를 통하여 센서 데이터를 받아온다.

CSS

Index와 같음

HTML

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>책장 환경 확인</title>
  <nav>
    <ul class = "menu">
      <li></li>
      <li><a href="{{url_for('main', select='intro')}}">초기화면</a></li>
      <li><a href="{{url_for('main', select='book')}}">내 책장 확인</a></li>
      <li><a href="{{url_for('main', select='graph')}}">책장 환경 확인</a></li>
      <li><a href="{{url_for('main', select='data')}}">책 데이터 수정</a></li>
    </ul>
  </nav>
  <link href="static/index.css" type="text/css" rel="stylesheet">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.2/mqtts31.min.js"
type="text/javascript"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"
type="text/javascript"></script>
  <script src="static/chart.js" type="text/javascript"></script>
  <script src="static/mqtt.js" type="text/javascript"></script>
  <script type="text/javascript">
    window.addEventListener("load", drawChart); // load 이벤트가 발생하면 drawChart()
    호출하도록 등록
    window.addEventListener("load", function () {
```

```

        var url = new String(document.location);
        ip = (url.split("/")[1]); // ip = "224...:8080/"
        ip = (ip.split(":")[0]); // ip = "224..."
        document.getElementById("broker").value = ip
    });
</script>
</head>
<body>
<section>
    <h1>책장 환경 확인</h1>
    <p>책장의 현재 환경 (온도, 습도, 채광)을 확인할 수 있습니다. 적정 구역은 초록색으로
    표기됩니다.</p>
    <form id="connection-form">
        <b>브로커 IP:</b>
        <input id="broker" type="text" name="broker" value=""><br>
        <b>포트 번호 : 9001</b><br>
        <input type="button" onclick="startConnect()" value="Connect">
        <input type="button" onclick="startDisconnect()" value="Disconnect">
    </form>
    <form id="subscribe-form">
        <input type="button" onclick="subscribe('temp');
        subscribe('humidity');" value="측정시작">
        <input type="button" onclick="unsubscribe('temp');
        unsubscribe('humidity');" value="측정중단">
    </form>
    <div id="messages"></div>
    <table>
    <tr>
    <td>
        <canvas id="temp" width="600" height="400"></canvas>
    </td>
    <td>
        <canvas id="humidity" width="600" height="400"></canvas>
    </td>
    </tr>
    </table>
    <br>
    <button id="addData" onclick="addChartData(0, Math.floor(Math.random()*40));
    addChartData(1, Math.floor(Math.random()*60));">Add Data</button>
</section>
</body>
</html>

```

Chart Java Script

```

var config_temp = {
    type: 'line',
    data: {
        labels: [],
        datasets: [
            {
                label: '실시간 온도',
                backgroundColor: 'yellow',
                borderColor: 'rgb(204, 0, 0)',
                borderWidth: 2,
                data: [],
                fontColor: '#f08080',
                fill : false
            }
        ]
    },
    yHighlightRange: {
        begin: [18],
        end: [22],
        color: ['rgba(77, 237, 48, 0.2)']
    }
}

```

```

    }
  },
  // 차트의 속성 지정
  options: {
    responsive : false, // 크기 조절 금지
    scales: { /* x 축과 y 축 정보 */
      xAxes: [{
        display: true,
        scaleLabel: { display: true, labelString: '시간' },
      }],
      yAxes: [{
        display: true,
        scaleLabel: { display: true, labelString: '온도' },
        ticks: {
          min: 10,
          max: 40,
        }
      }
    ]
  }
};

var config_humidity = {
  type: 'line',
  data: {
    labels: [],
    datasets: [
      {
        label: '실시간 습도',
        backgroundColor: 'yellow',
        borderColor: 'rgb(0, 0, 204)',
        borderWidth: 2,
        data: [],
        fontColor: '#6495ed',
        fill : false
      }
    ],
    yHighlightRange: {
      begin: [45],
      end: [55],
      color: ['rgba(77, 237, 48, 0.2)']
    }
  },
  // 차트의 속성 지정
  options: {
    responsive : false, // 크기 조절 금지
    scales: { /* x 축과 y 축 정보 */
      xAxes: [{
        display: true,
        scaleLabel: { display: true, labelString: '시간' },
      }],
      yAxes: [{
        display: true,
        scaleLabel: { display: true, labelString: '습도' },
        ticks: {
          min: 20,
          max: 80,
        }
      }
    ]
  }
};

var ctx = []
var chart = []

```

```

var LABEL_SIZE = 20; // 차트에 그려지는 데이터의 개수
var tick = 0; // 도착한 데이터의 개수임, tick의 범위는 0에서 99까지만
var originalLineDraw = Chart.controllers.line.prototype.draw;
Chart.helpers.extend(Chart.controllers.line.prototype, {
  draw: function() {
    var chart = this.chart;
    var yHighlightRange = chart.config.data.yHighlightRange;
    if (yHighlightRange !== undefined) {
      if (yHighlightRange.begin.length == yHighlightRange.end.length) {
        for (yCount = 0; yCount < yHighlightRange.begin.length; yCount++) {
          var ctx = chart.chart.ctx;
          var yRangeBegin = yHighlightRange.begin[yCount];
          var yRangeEnd = yHighlightRange.end[yCount];
          var yRangeColor = yHighlightRange.color[yCount];
          var xaxis = chart.scales['x-axis-0'];
          var yaxis = chart.scales['y-axis-0'];
          var yRangeBeginPixel = yaxis.getPixelForValue(yRangeBegin);
          var yRangeEndPixel = yaxis.getPixelForValue(yRangeEnd);
          ctx.save();
          ctx.fillStyle = yRangeColor;
          ctx.fillRect(xaxis.left, Math.min(yRangeBeginPixel, yRangeEndPixel),
            xaxis.right - xaxis.left, Math.max(yRangeBeginPixel, yRangeEndPixel) -
            Math.min(yRangeBeginPixel, yRangeEndPixel));

          var n = chart.data.datasets[0].data.length;
          var data = chart.data.datasets[0].data[n - 1];
          ctx.fillStyle = chart.data.datasets[0].fontColor;
          ctx.font = '120pt Arial';
          ctx.textAlign = 'center';
          if (n != 0) ctx.fillText(data.toFixed(1), 325, 200);
          ctx.restore();
        }
      } else {
        console.warn("yHighlightRange.begin and yHighlightRange.end are not the
        same length");
      }
    }
    originalLineDraw.apply(this, arguments);
  }
});
function drawChart() {
  ctx[0] = document.getElementById('temp').getContext('2d');
  ctx[1] = document.getElementById('humidity').getContext('2d');
  chart[0] = new Chart(ctx[0], config_temp);
  chart[1] = new Chart(ctx[1], config_humidity);
  init();
}
// chart의 차트에 labels의 크기를 LABEL_SIZE로 만들고 0~19까지 레이블 붙이기
function init() {
  for(let chartNum = 0; chartNum < chart.length; chartNum++){
    for(let i = 0; i < LABEL_SIZE; i++) {
      chart[chartNum].data.labels[i] = i;
    }
    chart[chartNum].update();
  }
}
function addChartData(i, value) {
  tick++; // 도착한 데이터의 개수 증가
  tick %= 100; // tick의 범위는 0에서 99까지만. 100보다 크면 다시 0부터 시작
  let n = chart[i].data.datasets[0].data.length; // 현재 데이터의 개수
  if(n < LABEL_SIZE)
    chart[i].data.datasets[0].data.push(value);
  else {

```

```

        // 새 데이터 value 삽입
        chart[i].data.datasets[0].data.push(value);
        chart[i].data.datasets[0].data.shift();
        // 레이블 삽입
        chart[i].data.labels.push(tick);
        chart[i].data.labels.shift();
    }
    chart[i].update();
}

```

MQTT Java Script

```

var port = 9001 // mosquitto 의 디폴트 웹 포트
var client = null; // null 이면 연결되지 않았음
function startConnect() { // 접속을 시도하는 함수
    clientID = "clientID-" + parseInt(Math.random() * 100); // 랜덤한 사용자 ID 생성
    // 사용자가 입력한 브로커의 IP 주소와 포트 번호 알아내기
    broker = document.getElementById("broker").value; // 브로커의 IP 주소
    // id 가 message 인 DIV 객체에 브로커의 IP 와 포트 번호 출력
    // MQTT 메시지 전송 기능을 모두 가진 Paho client 객체 생성
    client = new Paho.MQTT.Client(broker, Number(port), clientID);
    // client 객체에 콜백 함수 등록
    client.onConnectionLost = onConnectionLost; // 접속이 끊어졌을 때 실행되는 함수
    client.onMessageArrived = onMessageArrived; // 메시지가 도착하였을 때 실행되는 함수
    // 브로커에 접속. 매개변수는 객체 {onSuccess : onConnect}로서, 객체의 프로퍼티는
    onSuccess 이고 그 값이 onConnect.
    // 접속에 성공하면 onConnect 함수를 실행하라는 지시
    client.connect({
        onSuccess: onConnect,
    });
}
var isConnected = false;
function onConnect() { // 브로커로의 접속이 성공할 때 호출되는 함수
    isConnected = true;
    document.getElementById("messages").innerHTML += '<span>Connected</span><br/>';
}
var topicSave;
function subscribe(topic) {
    if(client == null) return;
    if(isConnected != true) {
        topicSave = topic;
        window.setTimeout("subscribe(topicSave)", 500);
        return
    }
    // 토픽으로 subscribe 하고 있음을 id 가 message 인 DIV 에 출력
    document.getElementById("messages").innerHTML += '<span>Subscribing to: ' +
topic + '</span><br/>';
    client.subscribe(topic); // 브로커에 subscribe
}
function publish(topic, msg) {
    if(client == null) return; // 연결되지 않았음
    client.send(topic, msg, 0, false);
}
function unsubscribe(topic) {
    if(client == null || isConnected != true) return;
    // 토픽으로 subscribe 하고 있음을 id 가 message 인 DIV 에 출력
    document.getElementById("messages").innerHTML += '<span>Unsubscribing to: ' +
topic + '</span><br/>';
    client.unsubscribe(topic, null); // 브로커에 unsubscribe
}
// 접속이 끊어졌을 때 호출되는 함수

```

```

function onConnectionLost(responseObject) { // 매개변수인 responseObject 는 응답 패킷의
정보를 담은 개체
    document.getElementById("messages").innerHTML += '<span>오류 : 접속
끊어짐</span><br/>';
    if (responseObject.errorCode !== 0) {
        document.getElementById("messages").innerHTML += '<span>오류 : ' + +
responseObject.errorMessage + '</span><br/>';
    }
}
// 메시지가 도착할 때 호출되는 함수
function onMessageArrived(msg) { // 매개변수 msg 는 도착한 MQTT 메시지를 담고 있는 객체
    var topic = msg.destinationName;
    var value = msg.payloadString;

    console.log("onMessageArrived: " + '(' + topic + ')' + value);
    switch(topic){
        case "temp":
            addChartData(0, parseFloat(value));
            break;
        case "humidity":
            addChartData(1, parseFloat(value));
            break;
    }
}
// disconnection 버튼이 선택되었을 때 호출되는 함수
function startDisconnect() {
    client.disconnect(); // 브로커에 접속 해제
    document.getElementById("messages").innerHTML +=
'<span>Disconnected</span><br/>';
}

```

2.2.2. 파이썬 (MQTT, FLASK, SENSOR)

book

웹 페이지 flask. Book.txt 파일에 있는 데이터를 웹 페이지에 적절하게 보내준다.

CODE

```
import os
from flask import Flask, render_template, request, redirect
from collections import namedtuple
app = Flask(__name__)
BOOK = 'static/book.txt'
USERDATA = 'static/userData.txt'
LOGIN_HTML = 'login.html'
INDEX_HTML = 'index.html'
CHART_HTML = 'chart.html'
BOOK_HTML = 'book.html'
BOOKDATA_HTML = 'bookdata.html'
@app.route('/') # / url 요청이 들어온다면 해당 함수를 실행
def first(): # index.html 을 반환함.
    return render_template(LOGIN_HTML)
@app.route('/main', methods=['post'])
def login(): # index.html 을 반환함.
    id = request.form['id']
    pw = request.form['pw']
    with open(USERDATA, 'r') as file:
        for line in file:
            line = line.rstrip().split(',')
            if line[0] == id and line[1] == pw:
                return render_template(INDEX_HTML)
    return redirect('/')

@app.route('/main', methods=['get'])
def main(): # index.html 을 반환함.
    menu = request.args.get('select')
    if menu == 'intro':
        return render_template(INDEX_HTML)
    elif menu == 'book':
        bookTitle = []
        bookPublisher = []
        bookWriter = []
        bookLen = []
        bookMemo = []

        with open(BOOK, 'r') as file:
            for line in file:
                line = line.rstrip().split(',')
                bookTitle.append(line[0])
                bookPublisher.append(line[1])
                bookWriter.append(line[2])
                bookLen.append(line[3])
                bookMemo.append(line[4])

        return render_template(BOOK_HTML, title = bookTitle, publisher = bookPublisher,
writer = bookWriter, len = bookLen, memo = bookMemo)

    elif menu == 'graph':
        return render_template(CHART_HTML)

    elif menu == 'data':
        bookTitle = []
        bookPublisher = []
        bookWriter = []
        bookLen = []
```

```

bookMemo = []

with open(BOOK, 'r') as file:
    for line in file:
        line = line.rstrip().split(',')
        bookTitle.append(line[0])
        bookPublisher.append(line[1])
        bookWriter.append(line[2])
        bookLen.append(line[3])
        bookMemo.append(line[4])

    return render_template(BOOKDATA_HTML, title = bookTitle, publisher =
bookPublisher, writer = bookWriter, len = bookLen, memo = bookMemo)

else:
    return redirect('/')
@app.route('/modify', methods=['get'])
def modify(): # 전화번호 검색
    bookTitle = []
    bookPublisher = []
    bookWriter = []
    bookLength = []
    bookMemo = []
    i = 0
    type = request.args.get('type')
    id = request.args.get('id')
    title = request.args.get('title')
    publisher = request.args.get('publisher')
    writer = request.args.get('writer')
    length = request.args.get('len')
    memo = request.args.get('memo')

    with open(BOOK, 'r') as file:
        for line in file:
            line = line.rstrip().split(',')
            if i == int(id):
                if type == 'save':
                    bookTitle.append(title)
                    bookPublisher.append(publisher)
                    bookWriter.append(writer)
                    bookLength.append(length)
                    bookMemo.append(memo)
                    oldName = os.path.join("static/pic", line[0] + ".jpg")
                    newName = os.path.join("static/pic", title + ".jpg")
                    os.rename(oldName, newName)

                else:
                    bookTitle.append(line[0])
                    bookPublisher.append(line[1])
                    bookWriter.append(line[2])
                    bookLength.append(line[3])
                    bookMemo.append(line[4])
            i = int(i) + 1

    file = open(BOOK, 'w')
    for i in range(len(bookTitle)):
        data = '%s,%s,%s,%s,%s\n' % (bookTitle[i], bookPublisher[i], bookWriter[i],
bookLength[i], bookMemo[i])
        file.write(data)
    file.close()
    return render_template(BOOKDATA_HTML, title = bookTitle, publisher = bookPublisher,
writer = bookWriter, len = bookLength, memo = bookMemo)
if __name__ == '__main__': # 실행
    app.run(host='0.0.0.0', port=8080, debug=True)

```


Sensor

초음파 센서를 통해 책이 들어오는지 판단하는 파이썬 프로그램.

CODE

```
import time
import picamera
import RPi.GPIO as GPIO
from PIL import Image
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
led = 6
echo = 16
trig = 20
button = 21
bookLen = []
bookCaseLen = 0
GPIO.setup(led, GPIO.OUT)
GPIO.setup(button, GPIO.IN, GPIO.PUD_DOWN)
GPIO.setup(trig, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)
GPIO.output(trig, False)
def ledOnOff(onOff):
    global led
    GPIO.output(led, onOff)
def measureDistance():
    global trig, echo
    GPIO.output(trig, True) # 신호 1 발생
    time.sleep(0.00001) # 짧은시간후 0으로 떨어뜨려 falling edge를 만들기 위한
    GPIO.output(trig, False) # 신호 0 발생(falling 에지)
    while(GPIO.input(echo) == 0):
        pass
    pulse_start = time.time() # 신호 1. 초음파 발생이 시작되었음을 알림
    while(GPIO.input(echo) == 1):
        pass
    pulse_end = time.time() # 신호 0. 초음파 수신 완료함을 알림
    pulse_duration = pulse_end - pulse_start
    return 340 * 100 / 2 * pulse_duration
def measureDistanceSecond(n):
    distance = []
    for i in range(n):
        distance.append(measureDistance())
        ledOnOff(True)
        time.sleep(0.5)
        ledOnOff(False)
        time.sleep(0.5)
    avg = sum(distance) / len(distance)
    return (round(avg) == round(distance[1]), avg)
def buttonPressed(pin):
    global bookCaseLen
    checkDistance = measureDistanceSecond(5)
    if checkDistance[0]:
        ledOnOff(True)
        bookCaseLen = checkDistance[1]
        print("측정 완료")
        with open('static/bookLen.txt', 'w') as file:
            file.write(str(bookCaseLen))
    else:
        ledOnOff(False)
        print("에러값 발생")
        time.sleep(5)
        ledOnOff(False)
def getBookLen():
    with open('static/book.txt', 'r') as file:
```

```

        for line in file:
            line = line.rstrip().split(',')
            bookLen.append(float(line[3]))
def writeBook(lens):
    with open('static/book.txt', 'a') as file:
        lens = round(lens, 2)
        data = '%s,%s,%s,%s,%s\n' % ("임시" + str(len(bookLen)+1), "출판사", "지은이",
lens, "수정이 필요합니다.")
        file.write(data)
    with picamera.PiCamera() as camera:
        camera.resolution = (640, 480)
        src = 'static/pic/임시' + str(len(bookLen)+1) + '.jpg'
        print(src)
        '''
        time.sleep(1)
        camera.capture(src)
        img = Image.open(src)
        img = img.transpose(Image.ROTATE_270)
        img = img.transpose(Image.FLIP_LEFT_RIGHT)
        img.save(src)
        '''

    bookLen.append(lens)
def inRange(a, b): #a 가 b +- 0.5 안에 있나?
    return b-1 <= a <= b+1
GPIO.add_event_detect(button, GPIO.RISING, buttonPressed, 200)
print("기존 데이터가 있는지 확인 중 입니다.")
while True :
    with open('static/bookLen.txt', 'r+') as file:
        line = file.readline()
        if line != '':
            getBookLen()
            bookCaseLen = float(line)
            print("기존 데이터가 확인 되었습니다.")
            break
        else:
            print("기존 데이터가 없습니다.")
nowDistance = measureDistance()
while True :
    beforeDistance = nowDistance
    nowDistance = measureDistance()
    dataDistance = bookCaseLen - sum(bookLen)
    if nowDistance > bookCaseLen:
        continue
    if inRange(dataDistance, nowDistance):
        print("정상 범위")
        print("nowDistance", nowDistance)
        print("beforeDistance", beforeDistance)
        print("bookCaseLen", bookCaseLen)
        print("bookLen", sum(bookLen))
    elif not(inRange(nowDistance, beforeDistance)) and nowDistance < beforeDistance:
        print("새 책 발견됨")
        checkDistacne = measureDistanceSecond(5)
        if inRange(checkDistacne[1], nowDistance):
            print("책을 추가합니다.")
            writeBook(beforeDistance - nowDistance)
        else:
            print("단순 에러")
            print("checkDistacne", checkDistacne[1])
            print("nowDistance", nowDistance)
            print("beforeDistance", beforeDistance)
            print("bookCaseLen", bookCaseLen)
            print("bookLen", sum(bookLen))

```

```

else:
    print("비정상 범위 값 초기화가 필요합니다.")
    print("nowDistance", nowDistance)
    print("beforeDistance", beforeDistance)
    print("bookCaseLen", bookCaseLen)
    print("bookLen", sum(bookLen))
time.sleep(1)

```

MQTT

Mqtt 실행하는 프로그램.

CODE

```

# publisher
import time
import paho.mqtt.client as mqtt
import mqttsensor
broker_ip = "localhost" # 현재 이 컴퓨터를 브로커로 설정
client = mqtt.Client()
client.connect(broker_ip, 1883)
client.loop_start()
while(True):
    temp = circuit.getTemperature()
    humidity = circuit.getHumidity()
    client.publish("temp", temp, qos=0)
    client.publish("humidity", humidity, qos=0)
    time.sleep(1)
client.loop_stop()
client.disconnect()

```

Mqttsensor

온습도 센서 mqtt 전송용 프로그램.

CODE

```

import time
import RPi.GPIO as GPIO
from adafruit_htu21d import HTU21D
import busio
# 전역 변수 선언 및 초기화
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
sda = 2 # GPIO 핀 번호, sda 라고 이름이 보이는 핀
scl = 3 # GPIO 핀 번호, scl 이라고 이름이 보이는 핀
i2c = busio.I2C(scl, sda)
sensor = HTU21D(i2c)
def getTemperature() :
    return float(sensor.temperature) # HTU21D 장치로부터 온도 값 읽기
def getHumidity() :
    return float(sensor.relative_humidity) # HTU21D 장치로부터 습도 값 읽기

```




2.2.3. 기타 데이터 형식

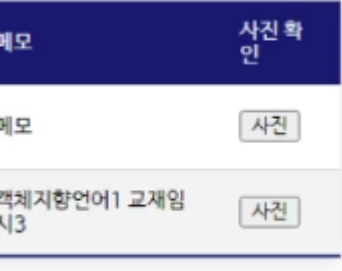
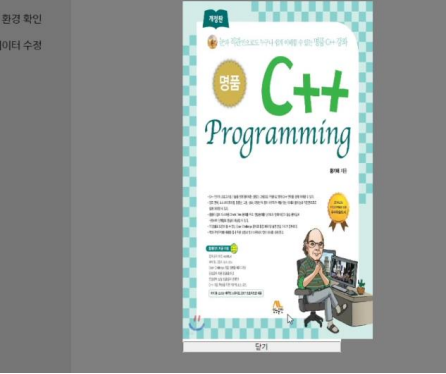

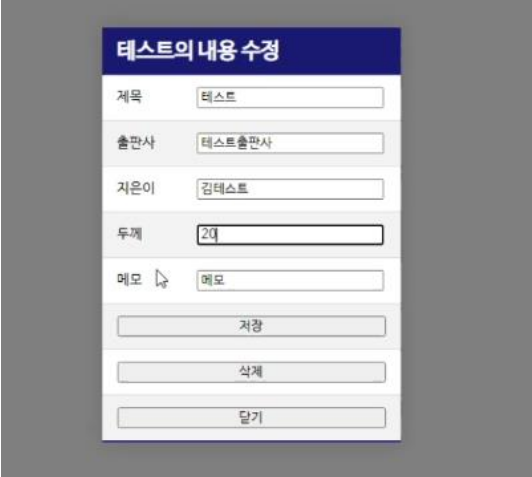
book
책 데이터
CODE
테스트ㄹㄴㅇㄴㄹㅇ, 테스트출판사, 김테스트, 20, 테스트 222 명품 C++ Programming, 생능출판, 황기태, 3, 객체지향언어 1 교재임시 3 임시3, 출판사, 지은이, 3.95, 수정이 필요합니다.

bookLen
책장 거리 측정용
CODE
35

userData
유저 계정 데이터
CODE
test,test

3. 실행 결과

사진	설명
	<p>로그인 페이지에 userdata.txt에 있는 값과 일치하는 정보를 입력하면 index.html 페이지로 연결된다.</p> <p>로그인 인증 방식이 따로 없고 post 방식으로 적당히 흉내만 냈기에 장식용이다.</p>
	<p>메인 화면에서는 간단한 설명을 볼 수 있다.</p>
	<p>내 책장 확인에서는 book.txt 데이터를 json으로 받아와 캔버스에 그린다.</p> <p>Json은 페이지 갱신될 때 받아 오기 때문에 캔버스 갱신을 눌러도 값이 추가되지 않는다. 누르면 색만 바뀐다.</p>

	<p>사진은 table의 책 제목을 바탕으로 static/pic에서 가져온다.</p>
	<p>사진을 불러온 모습. 원래는 책 저장과 동시에 사진을 찍음으로 저렇게 뚜렷한 사진은 안 찍힌다. 테스트용으로 넣은 임시 이미지이다.</p>
	<p>책장 환경 확인 에서는 MQTT를 통해 온도, 습도 센서 값을 받아와서 차트에 그린다.</p> <p>적정 온도에는 초록색으로 표시하였다. y축을 동적으로 설정하니 초록색이 안보이고 정적으로 설정하니 값이 잘 안보여서 배경에 숫자를 추가했다.</p>
	<p>책 데이터 수정에서는 book.txt 내용을 수정할 수 있다.</p> <p>초음파 센서의 이상으로 이상한 값이 입력되는 경우가 많기에 삭제 버튼도 존재한다.</p>

<p>내 책장 현황</p> <p>책 현황을 확인할 수 있습니다.</p> 	<p>일반 서비스에서 두께를 수정할 일은 없지만, 변경하면 내 책장 현황에 그대로 반영된다.</p>
	<p>책장에 책을 넣으면 거리가 줄어들고, 라즈베리 파이가 인식한다.</p>
<pre> t bookCaseLen 35.0 bookLen 5.0 f 정상 범위 nowDistance 30.365943908691406 beforeDistance 30.32541275024414 J bookCaseLen 35.0 t bookLen 5.0 새 책 발견됨 </pre>	<p>새 책 발견됨이라 뜬 모습.</p>
<pre> nowDistance 29.920101165771484 beforeDistance 30.382156372070312 bookCaseLen 35.0 bookLen 5.0 새 책 발견됨 단 순 예 러 checkDistacne 24.074697494506836 nowDistance 0.12564659118652344 beforeDistance 29.920101165771484 </pre>	<p>초음파 센서의 불안정으로 책이 추가되지 않았는데 추가되었다고 인식될 경우가 있다. 그럴 경우 5번은 재측정과 평균 값을 대조하는 과정을 통해서 해당 거리값이 정상적인 값인지 판별한다.</p>

	<p>또한 거리 검증 과정에서는 알아보기 쉽게 LED가 5번 점등한다. 해당 값이 가짜라면 5번 점등 후 꺼지고 진짜라면 5번 점등 후 켜진다.</p> <p>(잘 안보이지만, 좌측 꺼짐, 우측 켜짐.)</p>
	<p>기기에 있는 스위치는 거리값 계산 결과 정상이 아닐 경우 초기화하는 버튼이다.</p> <p>제작 중 거리를 재 측정 및 저장하는 용도로 사용되었고, 데이터 삭제하는 건 구현되지 않았다.</p>
 <pre> bookCaseLen 35.0 bookLen 5.0 새 책 발견됨 책을 추가합니다. static/pic/임시3.jpg checkDistance 26.75786018371582 nowDistance 26.531696319580078 beforeDistance 30.47943115234375 bookCaseLen 35.0 bookLen 8.95 </pre>	<p>책이 추가된다면, static/pic에 임시 + (있는 데이터 개수)의 이름으로 저장이 된다. 또한 bookLen값에 해당 값을 추가하여 다음 책도 인식할 수 있게 하였다.</p> <p>다만 파이카메라의 time out 에러로 시연 영상에서 작동 모습은 없다.</p>
<p>내 책장 현황</p> <p>책 현황을 확인할 수 있습니다.</p>  <p>캔버스 갱신</p>	<p>책이 추가하면 book.txt 파일에 추가됨으로 당연히 책장 현황의 캔버스도 변화한다.</p>

4. 결론 및 후기

미니 프로젝트를 수행하면서 어려운 점이 참 많았다. 첫번째 문제는 캔버스에 책 데이터를 그려 넣는 것이었다. Render template을 통해서 데이터 기반으로 html을 만드는 방법으로 알고 있었는데, js에 데이터를 넣는 방법을 알아야했다. 그래서 jinja2 문법을 계속 검색한 결과로 json.parse + {{tmp|json|safe}}를 검색을 통해 알았는데, “은 작동 안하고 ‘만 작동하는 것을 알아내기 위해 3시간을 소모하여 해결했다. 문제는 해결했지만, 하도 여러 방법을 시도해서 “가 아니라 ‘쓰는 게 해결 방법인지 장담이 되지 않는다.

사실 이쯤에서 jinja2 문법을 찾아보는 중에 내가 만드는 게 너무 웹페이지에 치중되어 있어서 모바일 스마트 시스템인지 웹 프로그래밍 과목인지 다른 것을 만들어야 하는지 하는 고민에 빠졌는데, 딱히 좋은 아이디어가 없어서 그냥 그대로 만들었다.

두번째 문제는 차트 관련한 부분이었다. 제안서를 만들 때 차트에 사각형 정도는 그리는 기능이 있겠지 생각했는데, 잘 나와있는 문서가 없었다. 그렇게 찾아 다니다가 Chartjs plugin Annotation.js 었는데, 버전도 안 맞고 적용도 안 되어서 stack overflow에 Chart.helpers.extend로 사각형을 그려주는 예시 코드가 있어서 해결할 수 있었다. 잘 모르는 방식을 복사 붙여넣기 해서 억지로 해결하는 방식은 싫었는데 정말 방법이 없어서 넣었다. Adapter? override? 비슷하게 함수를 추가하는 방식인 것 같은데 Js 문법을 더 공부해야 할 것 같다. 그래도 코드를 분석해서 캔버스 뒷배경에 최근 데이터를 띄워주는 방식을 추가한 것은 나름 뿌듯했다. 시각적으로도 예쁜 것 같아서 가장 마음에 든다.

가장 큰 문제는 초음파 센서였다. 고등학교 때에도 아두이노를 해 보았고, 대학교 와서도 소프트웨어 DIY와 피지컬컴퓨팅 창작을 통해 해당 센서들을 겪어봐서 조금 익숙했다. 그렇기 때문에 분명히 초음파 센서를 그렇게 민감하게 사용하면 에러가 난다고 예상하고 여러 대처 방법을 생각해 두었다. 그렇게 5초동안 재 측정하고 데이터 검증하는 과정을 만들었는데도 그래도 또 에러가 발생했다. 맨날 겪는 값이 튀는 문제는 대처 가능한데, 이 문제는 분명히 같은 회로에 같은 코드인데 갑자기 작동을 안 했다. 여러 번의 테스트 결과 원인을 추측할 수 있었는데, 책의 표지 때문이었다. 책 커버가 반짝이는 이유 때문에 초음파가 이상하게 튀는 문제였다. 정말로 책 커버의 문제인지는 모르겠지만, 시연 영상에 나오는 책과 노트를 제외하고는 죄다 커버가 코팅되어 있어서 나온 추측이다.

마지막으로 시연 영상을 찍기 전에 카메라가 작동을 안 했다. 전날만 화질이 파란 것만 제외하면 잘 찍혔고 자고 일어나서 시연하기 위해 켜는데, 파이카메라에 빨간 불이 들어오고 그대로 time out 에러를 보내 주었다. 카메라 고장은 아닌데 왜 capture에서 오류를 낼까 한참을 고치려고 검색하다가 별 소득 없이 기능을 빼고 시연 영상을 제작하였다. 그냥 빼고 소개할까 했는데 아쉬워서 갑자기 에러가 났다는 얘기로 시연을 마쳤다. 그리고 아직도 해결 못 하였다.

제안서에는 opencv를 통해 이미지 인식을 추가하려 했는데 막상 개발에 들어가니 시간이 부족해서 해당 부분은 제외하였다. 좀 더 열심히 제작하고 기획을 잘 했으면 추가할 시간이 있었을 것 같은데, 손도 못 대보고 포기한 것이 아쉽다.

최종적으로 미니 프로젝트를 통해 개발을 할 때 프로젝트 관리의 필요성을 느꼈다. 정리도 안하고 테스트용으로 너무 많이 파일을 만들다 보니까, 덤프파일이 너무 많이 생겨서 후에 막 정리하다가 중요한 파일을 지워서 다시 제작하였다. 코드 가독성을 위해 주석도 열심히 쓰고 파일도 그때 그때 정리해야겠다고 느꼈다. 또한 모바일 스마트 시스템을 통해 여러 언어, 환경, 장치 들을 통합하여 개발하는 과정에 대해 익힐 수 있었고, mqtt, flask, opencv등에 대해 대한 지식을 조금이나마 익힐 수 있었다.