МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Нарантуяагийн Дөлгөөн

Зоорины халаалт, агааржуулалт, чийгшүүлэлтийн автомат систем хөгжүүлэлт

(Developing automated system for cellar)

Мэдээллийн технологи (D061303)
Бакалаврын судалгааны ажил

Улаанбаатар

2025 оны 05 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Зоорины халаалт, агааржуулалт, чийгшүүлэлтийн автомат систем хөгжүүлэлт

(Developing automated system for cellar)

Мэдээллийн технологи (D061303)
Бакалаврын судалгааны ажил

Удирдагч: _____ Дэд профессор Э.Нямлхагва

Хамтран удирдагч: _____

Гүйцэтгэсэн: _____ Н.Дөлгөөн (19B1NUM0011)

Улаанбаатар

2025 оны 05 сар

Зохиогчийн баталгаа

Миний бие Нарантуяагийн Дөлгөөн ”Зоорины халаалт, агааржуулалт, чийгшүүлэлтийн автомат систем хөгжүүлэлт” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол эшлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

ЗУРГИЙН ЖАГСААЛТ

ХҮСНЭГТИЙН ЖАГСААЛТ

Кодын жагсаалт

УДИРТГАЛ

Орчин үеийн хөдөө аж ахуй болон хүнсний хадгалалтын салбарт автоматжуулалтын хэрэгцээ өсөн нэмэгдэж байгаа бөгөөд зоорины орчны хяналт, удирдлагын систем нь бүтээгдэхүүний чанарыг хадгалах, эрчим хүчний хэрэглээг оновчлох, гар ажиллагаанаас үүдэх алдааг багасгах чухал ач холбогдолтой болж байна. Гэвч уламжлалт аргаар зоорийг хянах, тохируулах нь цаг хугацаа их шаарддаг, тогтвортой бус, алдаатай байх магадлал өндөртэй тул автоматжуулсан, ухаалаг систем хөгжүүлэх хэрэгцээ гарч байна. Энэхүү судалгааны ажлын зорилго нь зоорины дотоод орчныг (температур, чийгшил, агаарын солилцоо) бодит цагийн горимд хянах, шаардлагатай үед автоматаар тохируулах боломжтой микроконтроллер болон мэдрэгчүүд дээр суурилсан ухаалаг системийг хөгжүүлэх явдал юм. Систем нь веб интерфэйсээр дамжуулан хэрэглэгчдэд мэдээлэл өгөх, алсын зайнаас удирдах боломжийг бүрдүүлнэ. Судалгааны явцад зоорины орчны стандарт шаардлагыг судалж, тохиромжтой мэдрэгч, хяналтын төхөөрөмжүүдийг сонгон авч, програмчлагдсан автомат ажиллагааг хөгжүүлэх бөгөөд үр дүнд нь зоорины хадгалалтын нөхцөлийг сайжруулах, бүтээгдэхүүний чанар алдагдахаас сэргийлэх, эрчим хүчний үр ашгийг дээшлүүлэх, хэрэглэгчийн оролцоог багасгах боломж бүрдэнэ.

1. СИСТЕМИЙН ТАНИЛЦУУЛГА

1.1 Оршил

Орчин үед технологийн хөгжлийн дэвшил нь хөдөө аж ахуй, хүнсний үйлдвэрлэл болон агуулахын системд ихээхэн нөлөө үзүүлж байна. Зоорины орчны тогтвортой байдлыг хадгалах нь хадгалалтын хугацааг уртасгаж, бүтээгдэхүүний чанарыг сайжруулах чухал хүчин зүйлсийн нэг юм. Гэвч уламжлалт зоорины хяналтын систем нь ихэвчлэн гар ажиллагаанд тулгуурласан, алдаатай, цаг хугацаа шаардсан байдгаас үр ашиг багатай хэвээр байна.

1.2 Үндэслэл

Хүнсний бүтээгдэхүүний хадгалалт нь температур, чийгшил, агаарын урсгал зэргээс шууд хамааралтай байдаг. Зоорины орчны өөрчлөлтийг бодит цаг хугацаанд хянах, шаардлагатай үед автоматаар тохируулах боломжтой систем байхгүй тохиолдолд дараах хүндрэлүүд үүсдэг:

- Цахилгаан эрчим хүчийг үр ашиггүй зарцуулах
- Хэт халуун эсвэл хэт хүйтэн нөхцөл үүсгэснээр хадгалагдаж буй бүтээгдэхүүний чанар алдагдах.
- Чийгшлийн тохиргоо буруу байснаас хөгц, мөөгөнцөр үүсэх.
- Чийгшүүлэгч, халаагчийг шаардлагатай үед асааж, унтраахаа мартсанаас болж орчны тохиргоо алдагдах.
- Байнгын хараа хяналт шаардлагатайгаас үүдэн хувь хүн, цаг хугацаа алдах

1.3 Зорилго

Энэхүү судалгааны ажлын гол зорилго нь зоорины орчныг хянах, удирдах автоматжуулсан системийг хөгжүүлэх явдал юм. Уг систем нь зоорины дотоод орчны температур, чийгшил, агаарын чанарыг бодит хугацаанд хэмжиж, шаардлагатай үед халаагч болон агааржуулагчийг автоматаар удирдах боломжтой байна. Мөн хэрэглэгч веб интерфэйсээр дамжуулан мэдээлэл авах, системийн ажиллагааг хянах, удирдах боломжтой байх бөгөөд ингэснээр зоорины дотоод орчныг тохиромжтой нөхцөлд байлгах, бүтээгдэхүүний хадгалах хугацааг уртасгах боломжийг бүрдүүлнэ.

1.4 Зорилт

Зорилгод хүрэхийн тулд дараах зорилтуудыг хэрэгжүүлнэ:

1. Зоорины орчны хяналтын шаардлага, стандартуудыг судлах
2. Мэдрэгчүүд болон микроконтроллер ашиглан өгөгдөл цуглуулах,
3. Цуглуулсан өгөгдлийг өгөгдлийн санд хадгалах
4. Өгөгдлийг веб интерфейст дамжуулах, харуулах
5. Веб интерфэйс болон удирдлагын системийг хөгжүүлэх
6. Өгөгдлийг боловсруулах алгоритмыг хөгжүүлэх
7. Системийн туршилт, алдааг олж засварлах

2. СИСТЕМИЙН СУДАЛГАА

Энэхүү судалгааны хэсэгт ESP32 микроконтроллер, DHT11 мэдрэгч, MQTT, LED гэрэл, болон программ хангамжийн технологиуд болох Spring Boot, Vue.js, MySQL-ийн онцлог, хэрэглээ, сонгох болсон шалтгааныг тайлбарлана. Зоорины орчны автоматжуулалт, хяналтын системийг хөгжүүлэхийн тулд эдгээр технологиудыг сонгосон бөгөөд тэдгээрийн ажиллагаа, давуу талуудыг нарийвчлан судалсан болно. Энэхүү системийг хийхийн тулд аль болох хямд өртгөөр үр бүтээл өндөртэйгөөр байхаар сонгосон.

2.1 Технологийн судалгаа

Системийн хөгжүүлэлтэд ашиглах технологиудын судалгаа болон сонголтын үндэслэлийг дэлгэрэнгүй тайлбарлана.

2.1.1 ESP32

ESP32 нь Wi-Fi болон Bluetooth холболттой, өндөр хүчин чадалтай 32-bit микроконтроллер юм. Энэ нь бага эрчим хүч зарцуулдаг, олон төрлийн мэдрэгч болон холболтын боломжийг дэмждэгээрээ давуу талтай. IoT болон автоматжуулалтын шийдлүүдэд өргөн хэрэглэгддэг.

ESP32-г сонгосон шалтгаан:

- Wi-Fi болон Bluetooth-ийг дэмждэг тул дата дамжуулах уян хатан сонголттой.
- Олон төрлийн GPIO (Оролт, Гаралтын Порт)-той тул мэдрэгч, релей гэх мэт нэмэлт төхөөрөмжүүдийг хялбархан холбох боломжтой.
- Дотоод санах ой (RAM: 512KB хүртэл, Flash: 4MB+ хүртэл) ихтэй тул илүү боловсронгуй код, алгоритм хэрэгжүүлэх боломжтой.
- Хямд үнэ, Arduino болон Raspberry Pi-тай харьцуулахад илүү хэмнэлттэй.

- Бага эрчим хүчний горимтой (Deep Sleep Mode: 10μA хүртэл бага хэрэглээ) учир зооринд удаан ажиллахад тохиромжтой.

2.1.2 DHT11 (Температур, Чийгшил Мэдрэгч)

DHT11 нь хүнсний ногооны хадгалалтад шаардлагатай чийгшил болон температурыг тогтмол хянах зориулалттай мэдрэгч юм.

DHT11 мэдрэгчийн давуу талууд:

- Температур хэмжих нарийвчлал: ±2°C
- Чийгшил хэмжих нарийвчлал: ±5%
- Температур хэмжих хязгаар: -0°C-аас +50°C хүртэл
- Чийгшил хэмжих хязгаар: 20% - 90% RH
- 3.3V болон 5V тэжээлтэй ажиллах боломжтой, ESP32-тэй шууд холбогдож ажиллах чадвартай.
- Бэлэн Arduino болон ESP32-ийн программчлалын сангуудтай, хөгжүүлэлт хийхэд хялбар.
- 1-wire интерфейс ашигладаг тул код болон холболт энгийн.

2.1.3 MQTT

MQTT нь Message Queuing Telemetry Transport гэсэн үгийн товчлол бөгөөд бага зурвасын өргөнтэй, сүлжээнд зориулагдсан хөнгөн, нийтлэх, захиалах үндсэн дээр суурилсан мессежийн протокол юм. Энэ нь IOT-д өргөн хэрэглэгддэг.

- Бодит цагт ажилладаг, Publish/Subscribe механизмаар төхөөрөмжүүдийн хооронд өгөгдлийг бодит цагийн горимд хурдан солилцоно.
- Хөнгөн жинтэй, бага нөөц зарцуулдаг тул Сүлжээний хурд багатай, CPU, RAM хязгаарлагдмал IoT төхөөрөмжүүдэд тохиромжтой.

- Холболт тасарсан үед буцаан илгээх боломжтой (Last Will and Retain Messages) – Хэрэглэгчийн холболт тасарсан тохиолдолд сервер автоматаар ”сүүлчийн хүсэл”-ийг (Last Will message) илгээж, сүлжээний алдаа эсвэл төхөөрөмжийн унтралт зэргийг илүү найдвартай хянах боломжтой.

2.1.4 LED гэрлүүд

Системд ашиглагдах LED гэрлүүд нь жинхэнэ төхөөрөмжүүдийг төлөөлөх туршилтын элементүүд юм. Жинхэнэ хувилбарт эдгээр LED-үүдийг халаагч, хөргүүр, агааржуулалтын хавхлагууд, сэнс гэх мэт тоног төхөөрөмжөөр орлуулах боломжтой.

2.1.5 Программ хангамж

Системийн программ хангамжийн хөгжүүлэлтэд дараах технологиудыг ашиглана.

Backend (Серверийн хэсэг)

- Spring Boot:
  - Java дээр суурилсан, орчин үеийн веб үйлчилгээ хөгжүүлэхэд зориулсан framework.
  - Модульчлагдсан бүтэцтэй тул хөгжүүлэлт, өргөтгөл хийхэд хялбар.
  - MSSQL өгөгдлийн сантай холбогдож, мэдээлэл боловсруулах, хадгалах үүрэгтэй.
  - REST API ашиглан ESP32 болон веб интерфейстэй холбогдоно.

Frontend (Хэрэглэгчийн веб интерфейс)

- Vue.js:
  - Хэрэглэгчийн веб интерфейсийг хөгжүүлэхэд ашиглана.
  - Компонент суурьтай учир дахин ашиглагдах UI элементүүд үүсгэх боломжтой.
  - Бодит цагийн дата харуулах боломжтой.

Өгөгдлийн сан

- MySQL
  - Өндөр найдвартай, баталгаатай өгөгдлийн сан учир их хэмжээний мэдээлэл боловсруулахад тохиромжтой.
  - Үнэгүй ашиглах боломжтой бөгөөд олон нийтийн дэмжлэг сайн.
  - SQL хэл дээр суурилсан тул сурахад ойлгомжтой, түгээмэл хэрэглэгддэг.

2.2 Ижил төстэй системийн судалгаа

Энэ хэсэгт бид дараах ижил төрлийн системийг авч үзэн, онцлог болон өөрийн хөгжүүлсэн системтэй харьцуулалт хийнэ.

2.2.1 Brewmation Cellar Control

Brewmation Cellar Controls нь мэргэжлийн пиво, дарсны үйлдвэр, сүүний фермерүүдэд зориулсан бүрэн автомат удирдлагын систем юм. Уг систем нь танк болон өрөөний дулааныг тогтмол хянаж, халаалт, хөргөлтийн төхөөрөмжүүдийг автоматаар удирдана. Мөн интернетээр хянах боломжтой бөгөөд өгөгдлийг хадгалах, хэрэглэгчийн тохиргоо хийх боломжтой.

Давуу талууд:

- Бүхэл системийн бүрдэлтэй (мэдрэгч, хяналтын самбар, веб UI)
- Аж үйлдвэрийн зориулалттай найдвартай ажиллагаатай
- Програмчлагдах автомат ажиллагаа
- SCADA системтэй холбогдох чадвартай

Сул талууд:

- Маш өндөр өртөгтэй

- Суулгах, тохируулах ажиллагаа төвөгтэй
- Хувийн хэрэглээнд зориулагдаагүй, уян хатан бус

2.2.2 Өөрийн хөгжүүлсэн системийн харьцуулалт

Давуу талууд:

- Бодит цагийн температур, чийгшлийн хяналт болон 4 релейн удирдлага.
- Автомат горим ба хэрэглэгчийн гараар удирдах боломжтой UI бүхий вэб систем.
- MQTT ашиглан хөнгөн жинтэй, хурдан холболттой өгөгдөл дамжуулалттай.
- Config тохиргоог backend-ээс бодит цагт шинэчилдэг.
- Уян хатан архитектуртай тул цаашид IoT төхөөрөмж нэмэхэд амархан.

Сул талууд:

- Интерфэйс болон хэрэглэгчийн тохиргоо UI-г сайжруулах шаардлагатай.
- Сүлжээ тасарсан тохиолдолд найдвартай байдлыг хангах нэмэлт шийдэл хэрэгтэй.
- Мэдээллийн аюулгүй байдал (authentication, encryption) бүрэн хийгдээгүй.
- Мобайл дэмжлэг сул (responsive дизайн эсвэл апп шаардлагатай).

Үзүүлэлт	Brewmation Cellar Controls	Өөрийн систем
Релей удирдлага	Тийм	Тийм
Интернэтээр хянах	Тийм	Тийм
Өгөгдөл хадгалах	Тийм	Тийм
Бодит цагийн өгөгдөл дамжуулалт	Тийм	Тийм
Хэрэглэгчийн UI	SCADA болон Web	Веб интерфейс
Систем өргөтгөх боломж	Хязгаарлагдмал	Өндөр
Үнийн хувьд	Өндөр	Хямд
Зориулалт	Аж үйлдвэр	Гэр ахуй, жижиг фермер

Хүснэгт 2.1: Brewmation систем ба өөрийн хөгжүүлсэн системийн харьцуулалт

2.3 Бүлгийн дүгнэлт

Судалгаанаас харахад ESP32 нь бага эрчим хүч зарцуулалт, утасгүй холболт, олон төрлийн мэдрэгчтэй нийцэх чадвар зэргээрээ хамгийн тохиромжтой сонголт болсон. DHT11 мэдрэгч нь зоорины чийг, температурыг өндөр нарийвчлалтай хэмжих боломжтой тул сонгогдсон. LED гэрлүүд нь системийн ажиллагааг төлөөлөхөд ашиглагдана. Программ хангамжийн хувьд, Spring Boot нь серверийн хэсэгт, Vue.js хэрэглэгчийн интерфейст, MySQL өгөгдөл хадгалах, хэрэглэгчийн бүртгэл хийхэд тус тус ашиглагдана. Миний хөгжүүлсэн систем нь үнийн хувьд хямд, энгийн хэрэглээнд тохиромжтой, интернетээр хянах, өөрчлөх боломжтой веб UI, релей хяналт, бодит цагийн өгөгдөл дамжуулалт, өгөгдлийн сангийн хадгалалт зэрэг боломжуудыг агуулсан. Мөн хэрэглэгч өөрөө тохиргоо үүсгэх, сонгох, горим солих уян хатан байдалтай тул микро түвшний автоматжуулалт хэрэгтэй айл өрх, фермерүүдэд илүү нийцтэй шийдэл юм.

3. СИСТЕМИЙН ШИНЖИЛГЭЭ

Энэхүү хэсэгт зоорины автоматжуулалтын системийн шаардлагуудыг тодорхойлох зорилготой. Энэ нь системийн үндсэн үйл ажиллагаа, хэрэглэгчийн хүсэлт, хүлээгдэж буй үр дүнг тусгана.

3.1 Төхөөрөмжийн шаардлага

Системийн ажиллах горимыг тохируулахдаа Монгол улсын хүнсний ногоо хадгалах стандарт, нөхцөлүүдийг харгалзан дараах үзүүлэлтүүдийг баримтална:

Ногооны төрөл	**Температур (°C)**	**Чийгшил (%)**
Төмс	4 - 15	80 - 95
Лууван, манжин	0 - 3	90 - 95
Байцаа	0 - 3	90 - 95
Сонгино, сармис	10 - 15	50 - 70
Улаан лооль	7 - 21	80 - 95
Өргөст хэмх	10 - 14	80 - 95
Шош, вандуй	7 - 10	80 - 95

Хүснэгт 3.1: Ногооны хадгалах тохиромжтой нөхцөл

3.2 Функциональ шаардлага

1. Систем нь температур, чийгшлийн өгөгдлийг бодит цаг хугацаанд хянах ёстой.
2. Систем нь ESP32-оос өгөгдөл хүлээн авч, өгөгдлийн санд дамжуулдаг байх ёстой.
3. Систем нь өгөгдлийн санг ашиглаж, өгөгдөл хадгалдаг байх ёстой.
4. Систем нь температур, чийгшлийн утгууд тохирсон хязгаараас хэтэрсэн үед автоматаар халаагч эсвэл сэнсийг асаах ёстой.

5. Систем нь зоорины орчны тохиргоог хэрэглэгчийн хүссэнээр өөрчлөх боломжтой байх ёстой.

6. Систем нь хэрэглэгчид хадгалалтын горим тохируулах, урьдчилан сэргийлэх сэрэмжлүүлэг хүлээн авах боломжийг олгох ёстой.

7. Систем нь лог хөтлөлт хийх ба хэрэглэгч өөрийн түүхэн өгөгдлийг харах боломжтой байх ёстой.

8. Систем нь сүлжээ тасарсан тохиолдолд хэрэглэгчид мэдэгдэх ёстой.

9. Систем нь гар утас болон компьютероос веб интерфейсээр дамжуулан удирдах боломжтой байх ёстой.

10. Систем нь ашиглалтын явцад хэрэглэгчдэд мэдэгдэл илгээх боломжтой байх ёстой (температур хэтэрсэн, чийгшил тохиромжгүй гэх мэт).

11. Систем нь “гараар” болон “автомат” горимоор ажиллах сонголттой байх ёстой. Гараар горимд хэрэглэгч төхөөрөмжүүдийг шууд удирдах боломжтой байх ба автомат горимд тохиргооны дагуу төхөөрөмжүүд өөрсдөө ажиллах ёстой.

12. Систем нь урьдчилан тогтоосон болон хэрэглэгчийн өөрийн тохиргооны профайлуудыг хадгалах, тэдгээрээс сонгож хэрэглэх боломжтой байх ёстой. Жишээ нь: төмс, сонгино, улаан лооль гэх мэт.

3.3 Функциональ бус шаардлага

1. Систем нь 24/7 тасралтгүй ажиллах боломжтой байх ёстой.

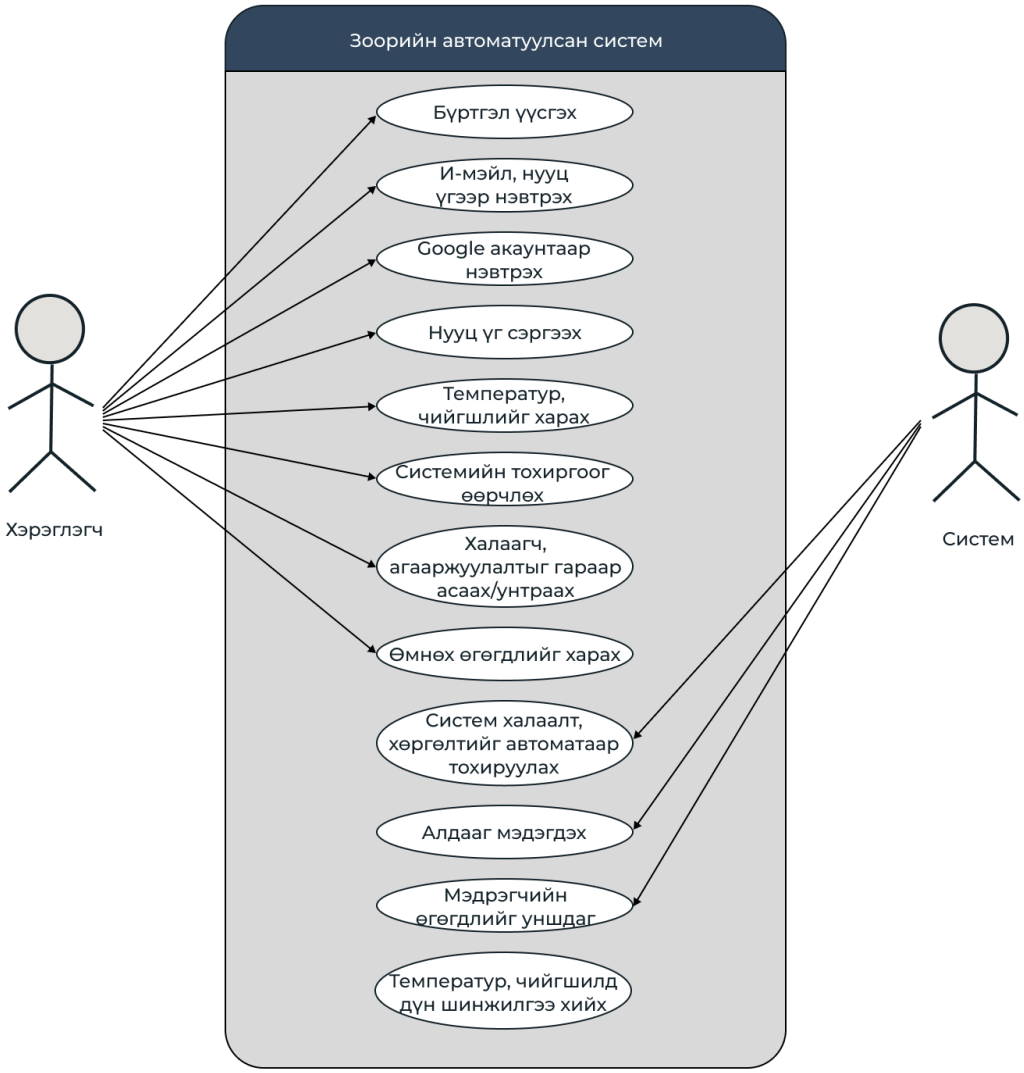
2. Систем нь 5 секунд тутамд шинэ өгөгдлийг авах хүчин чадалтай байх ёстой.

3. Систем нь 1 секундээс бага хугацаанд хэрэглэгчийн хүсэлтэд хариу өгөх ёстой.

4. Систем нь өгөгдлийн сантай найдвартай ажиллах, өгөгдлийн бүрэн бүтэн байдлыг хангах ёстой.
5. Системийн веб интерфейс нь хурдан бөгөөд хэрэглэгчид ойлгомжтой байх ёстой.
6. Систем нь шалгаж баталгаажсан хэрэглэгчдэд л хандах боломж олгох, аюулгүй байдлыг хангах ёстой.
7. Систем нь нэмэлт тоног төхөөрөмж шаардалгүйгээр интернэт хөтөч дээр ажиллах ёстой.
8. Систем нь монгол хэл дээр ажиллах боломжтой байх ёстой.
9. Системийн ихэнх тохиргоог хэрэглэгч өөрийн хэрэгцээнд тохируулан өөрчлөх боломжтой байх ёстой.
10. Систем нь хэрэглэгчийн хувийн мэдээллийг шифрлэх, зөвшөөрөлгүй хандалтаас хамгаалах ёстой.
11. Систем нь бүх өгөгдлийг нууцлалтайгаар дамжуулах (HTTPS, TLS 1.2 ба түүнээс дээш) шаардлагатай.

3.4 Use-Case диаграмм

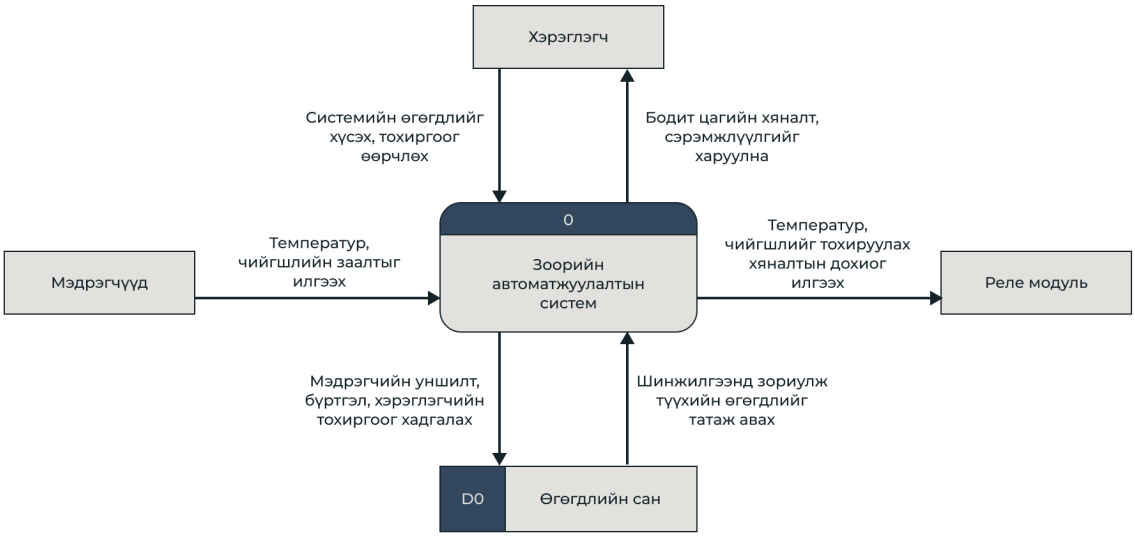
Энэхүү диаграмм нь хэрэглэгч болон систем хоорондын харилцааг харуулсан бөгөөд үндсэн функцүүд болон автоматжуулалтын үйл явцыг харуулсан болно. Энэ нь жүжигчдийн үүрэг, тэдгээрийн системийн янз бүрийн функцтэй харилцах харилцааг харуулна.



Зураг 3.1: Use Case диаграмм

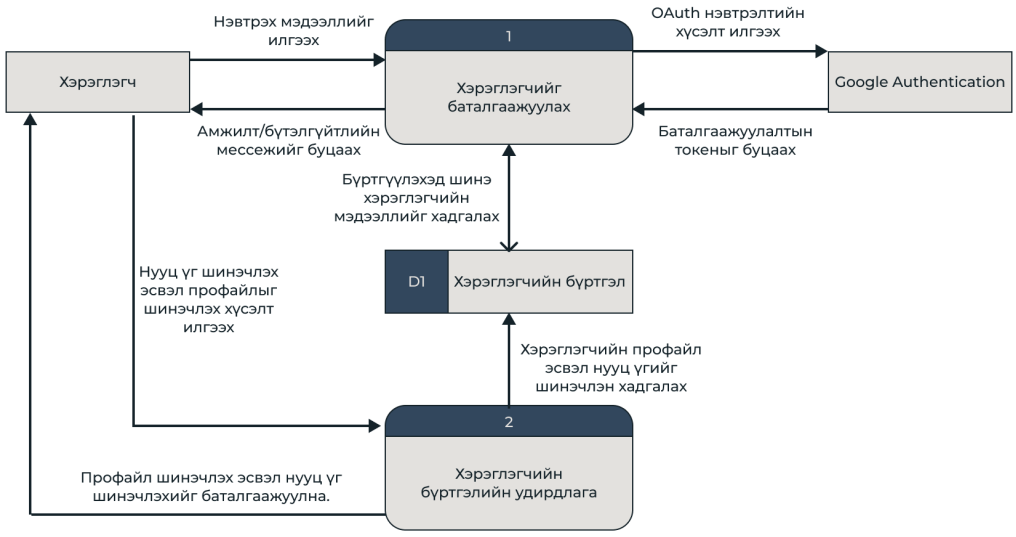
3.5 Өгөгдлийн урсгалын диаграмм

Энэхүү диаграмм нь бүхэл системийн өндөр түвшний тоймыг гаргаж, хэрэглэгчид түүнтэй хэрхэн харилцаж, гадаад нэгж, процесс, хадгалалтын хооронд өгөгдөл хэрхэн урсаж байгааг харуулна.



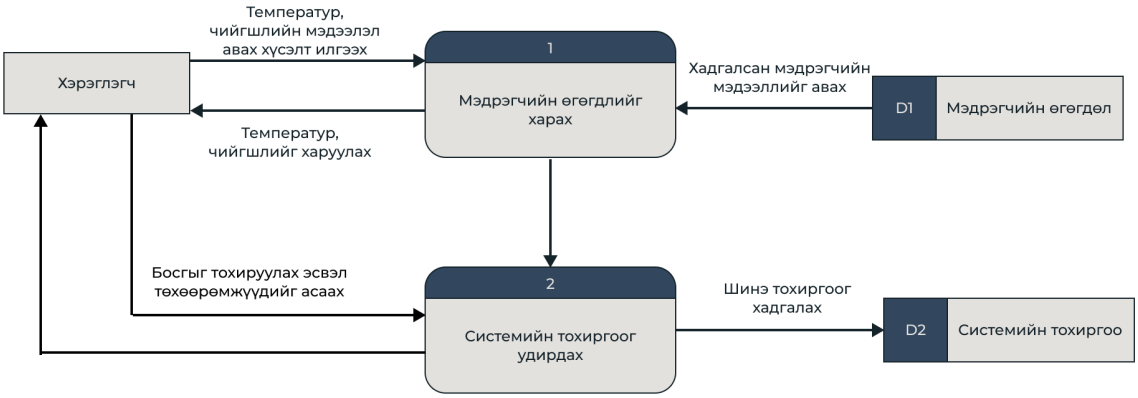
Зураг 3.2: 0-р түвшний ӨУД Зоорины автоматжуулалтын систем

Энэ диаграммд хэрэглэгчид хэрхэн бүртгүүлэх, нэвтэрч орох, бүртгэлээ удирдах, тухайлбал нэвтрэлт танилт, профайлын шинэчлэлтүүдийг дэлгэрэнгүй тайлбарлана.



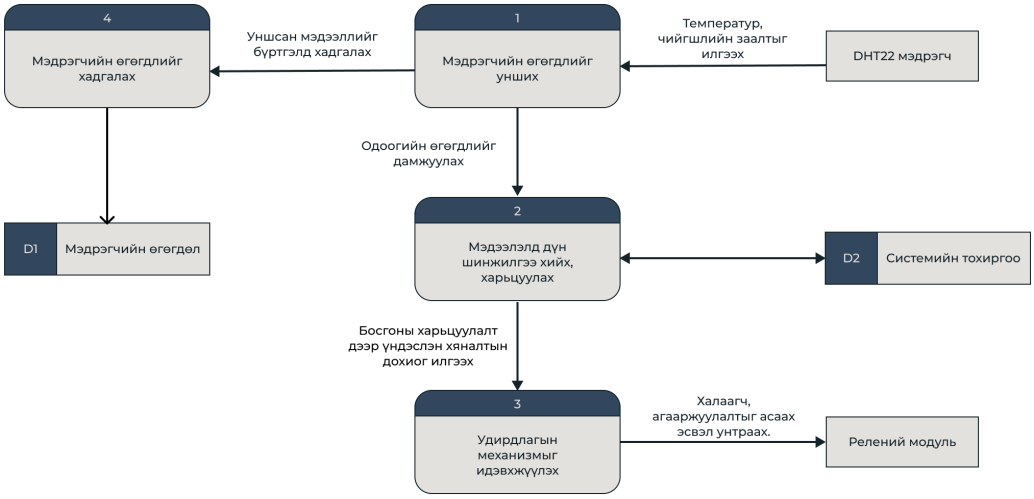
Зураг 3.3: 1-р түвшний ӨУД Хэрэглэгчийн удирдлага

Энэ диаграмм нь хэрэглэгчид системийн интерфейсээр дамжуулан зоорины нөхцөлийг (жишээлбэл, халаалт эсвэл агааржуулалтыг асаах/унтраах) гараар хэрхэн удирдаж байгааг харуулж байна.



Зураг 3.4: 1-р түвшний ӨУД Хэрэглэгчийн тохиргоо ба хяналт

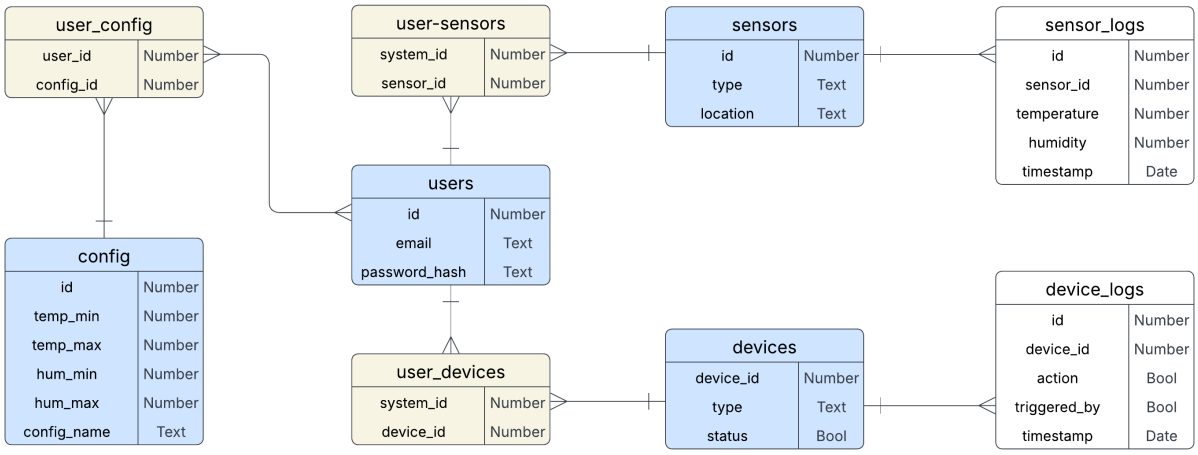
Энэхүү диаграмм нь систем нь мэдрэгчээс өгөгдлийг хэрхэн цуглуулж, боловсруулж, автомат удирдлагад зориулж хадгалдаг болохыг харуулж байна.



Зураг 3.5: 1-р түвшний ӨУД Мэдрэгчийн өгөгдөл боловсруулалт

3.6 Entity-Relation диаграмм

Энэхүү Entity-Relationship Diagram (ERD) нь зоорины орчны хяналт, удирдлагын автоматжуулсан системийн өгөгдлийн сангийн бүтэц, түүний бүрэлдэхүүн хэсгүүдийн хоорондын харилцааг дүрсэлнэ.



Зураг 3.6: Зоорины автоматжуулалтын системийн ERD

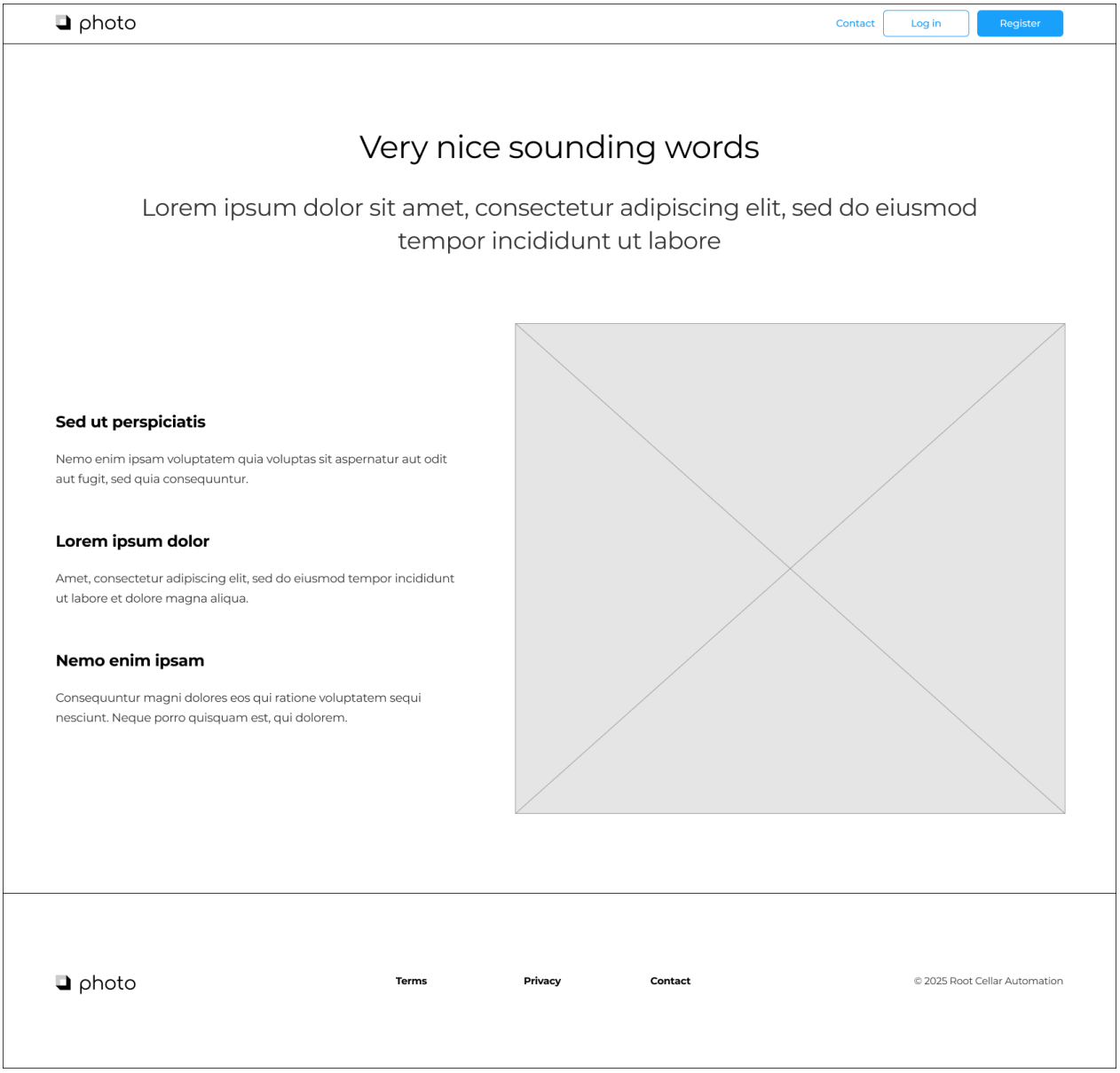
3.7 Бүлгийн дүгнэлт

Энэхүү бүлэгт системийн бүтэц, үйл ажиллагааны шаардлагууд болон өгөгдлийн урсгал, харилцааг тодорхойлж, цаашдын хөгжүүлэлтийн суурь судалгааг бүрэн гүйцэтгэлээ. Энэ нь системийн хөгжүүлэлтийн дараагийн шат болох архитектурын төлөвлөлт, хэрэгжилтийн ажилд үндэс суурь болж өгнө.

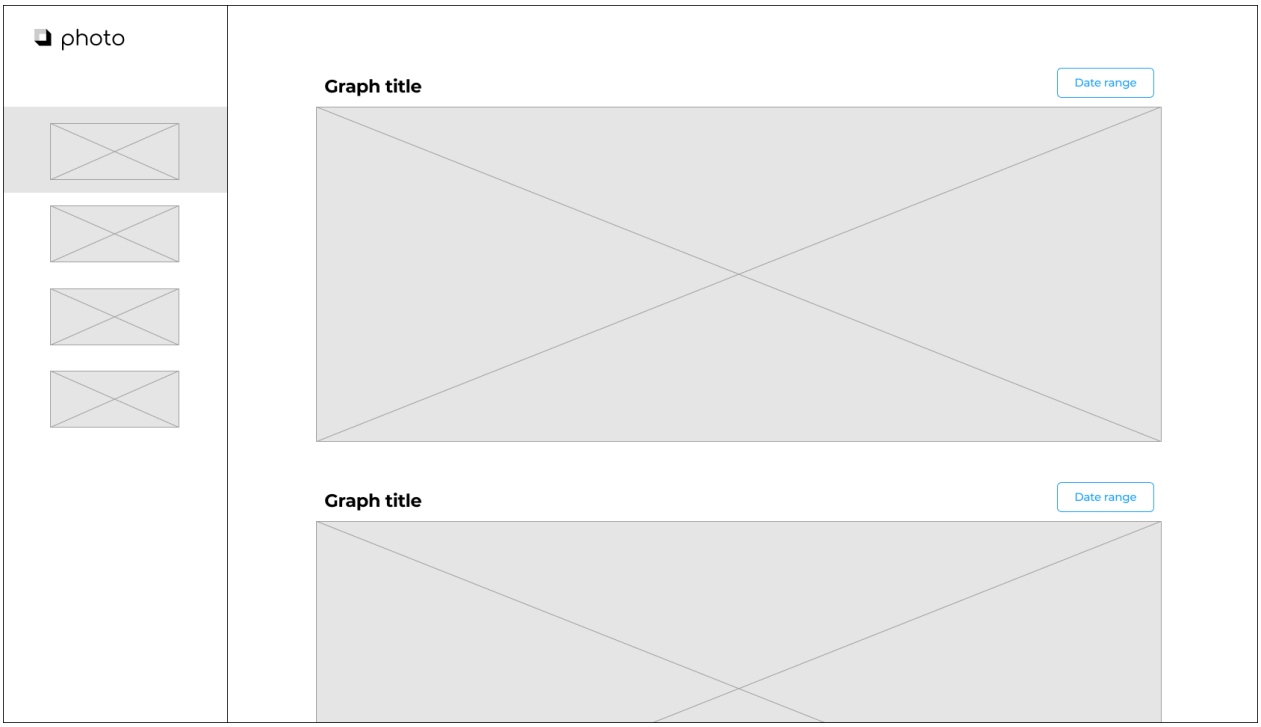
4. СИСТЕМИЙН ЗОХИОМЖ

4.1 Дэлгэцийн зохиомж

Энэхүү хэсэгт дэлгэцийн зохиомжийн хар зургийг үзүүлнэ.



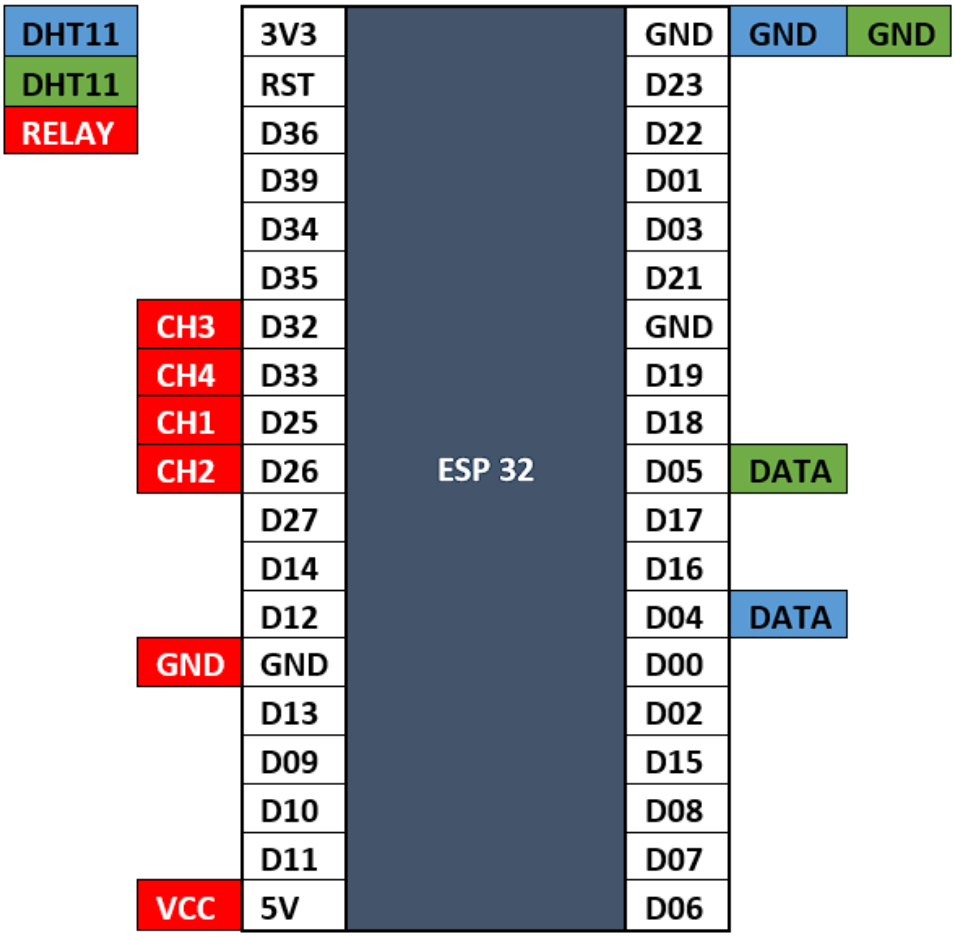
Зураг 4.1: ”Нүүр” хуудас



Зураг 4.2: ”Удирдлагын самбар” хуудас

4.2 Техник хангамжийн зохиомж

Доорх диаграмм нь esp32-н хөлүүд дээрх бусад модулиудын аль хөл холбогдохыг харуулсан диаграмм юм. Үүнд 2 чийгшил мэдрэгч DHT11 мэдрэгч, 4 сувагт 1 релей модуль багтсан. Үүн дээр цаашид 2 ширхэг DHT11 мэдрэгч, халаагч, хөргүүрийг төлөөлүүлэх LED гэрлүүд нэмэгдэнэ.

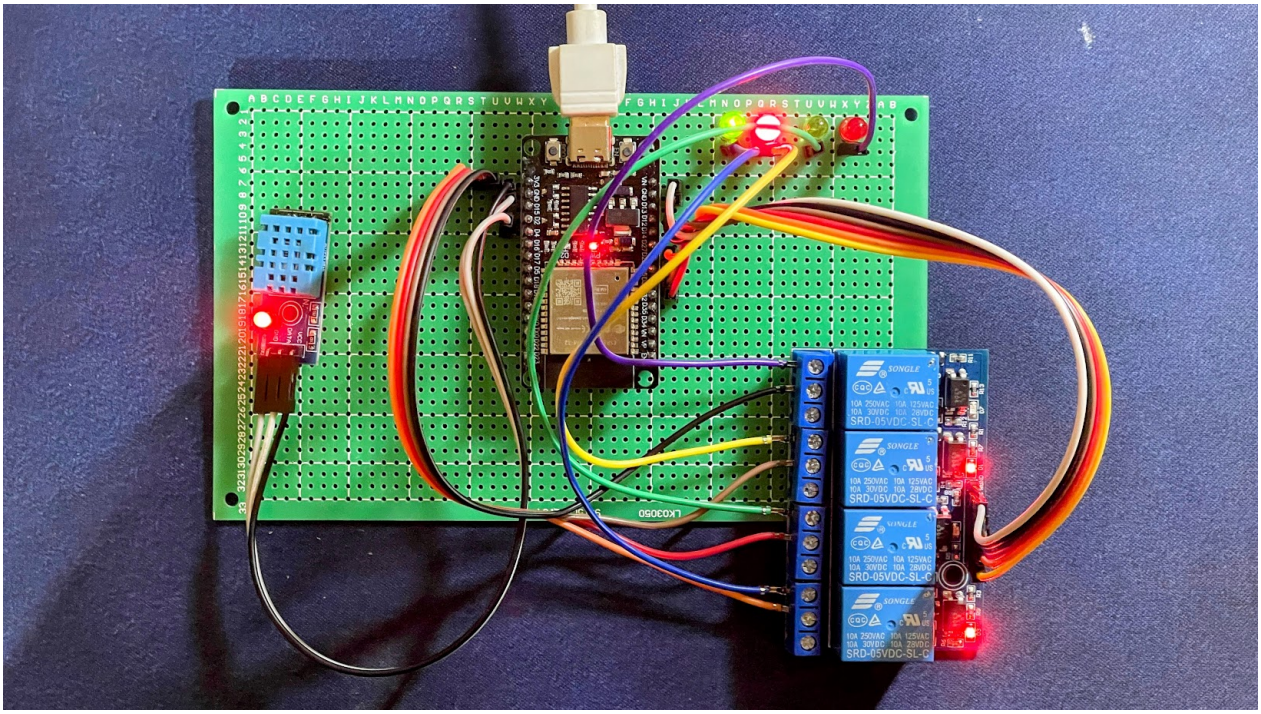


Зураг 4.3: ESP32 нэгдсэн хэлхээний хөлүүдийн байрлалыг харуулсан диаграмм

5. ХЭРЭГЖҮҮЛЭЛТ

Техник хангамжийн зохиомжид үзүүлснийг бодитоор угсран бэлдсэн байгаа. Эдгээр релей нь хөргүүр, халаагуур, агаар чийгшүүлэгч, агааржуулагч зэрэг төхөөрөмжүүдийн унтраалга болон ажиллах юм.

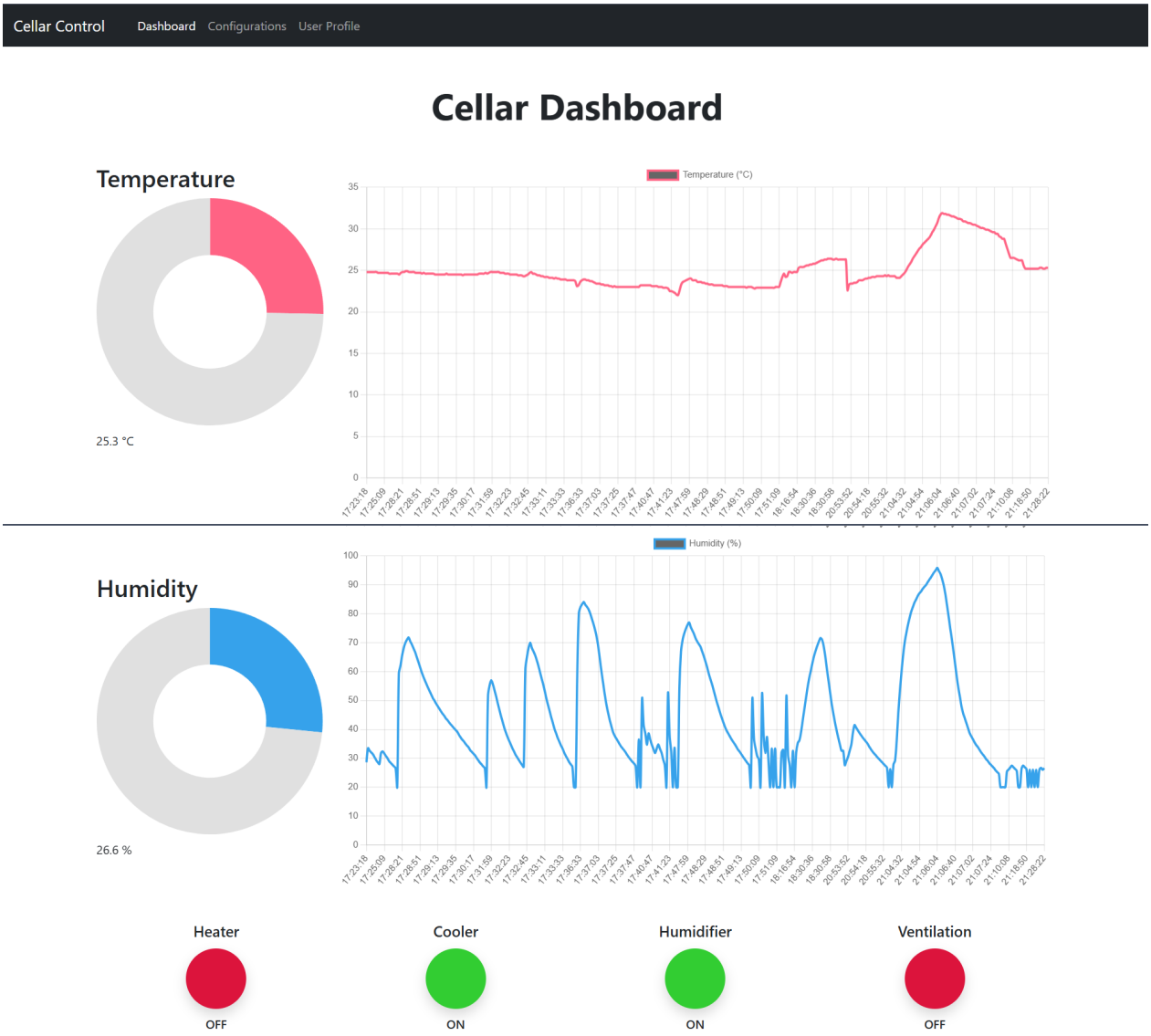
ESP32 нь өгөгдөл өөрчлөгдөхөд үүнийгээ MQTT протокол ашиглан MQTT Broker-рүү илгээдэг. ESP32 нь өмнөх релей төлөвийг хадгалж, хэрэв тухайн релейний төлөв өөрчлөгдсөн бол л ‘esp32/relay/status‘ сэдвээр илгээдэг. Энэ нь сүлжээг дэмий ачаалуулахгүй, зөвхөн шаардлагатай үед мэдээлэл явуулах давуу талтай. Ийм төрлийн оптимизаци нь IoT системүүдийн үндсэн шаардлагуудын нэг юм.



Зураг 5.1: ESP32-н угсарсан байдал

MQTT протокол ашиглан ESP32-н илгээж буй мэлээллийг MQTT broker-с авч backend-с өгөгдлийн сан руу хадгалж байгаа.

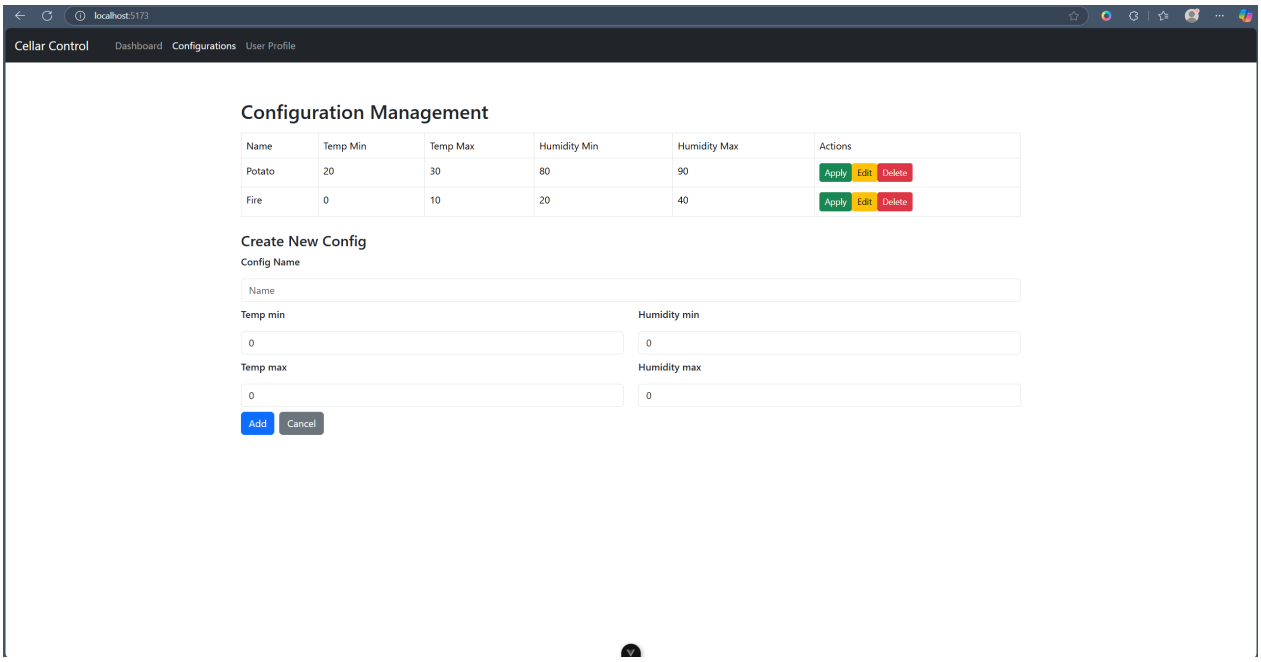
Frontend Backend-с өгөгдлийг авч харагдацыг шинэчилнэ. Ингэснээр бодит цагийн мэдээлэл авна. Бодит цагийн мэдээллийг харагдуулахдаа WebSocket эсвэл SSE ашиглахаас татгалзаж, polling буюу тодорхой хугацаанд нэг удаа асуулт тавих аргыг сонгосон. Үүнд frontend секунд тутам backend-ээс хамгийн сүүлийн мэдрэгчийн мэдээлэл болон релей төлөвийг асууж авна.



Зураг 5.2: Өгөгдөл болон релэйн мэдээллийг харуулсан байдал

Цагираг графикаар одоогийн байгаа дулаан болон чийгшилийн мэдээллийг, зураасан графикар нийт дулаан болон чийгшлийн өөрчлөлтийг дүрслэн харуулсан. Доор байх 4 гэрэл нь халаагуур,

хөргүүр, чийгшүүлэгч, агааржуулагчийг тус тус төлөөлөнө. Ногоон байвал ассан, улаан байвал унтарсан гэдгийг илтгэнэ.



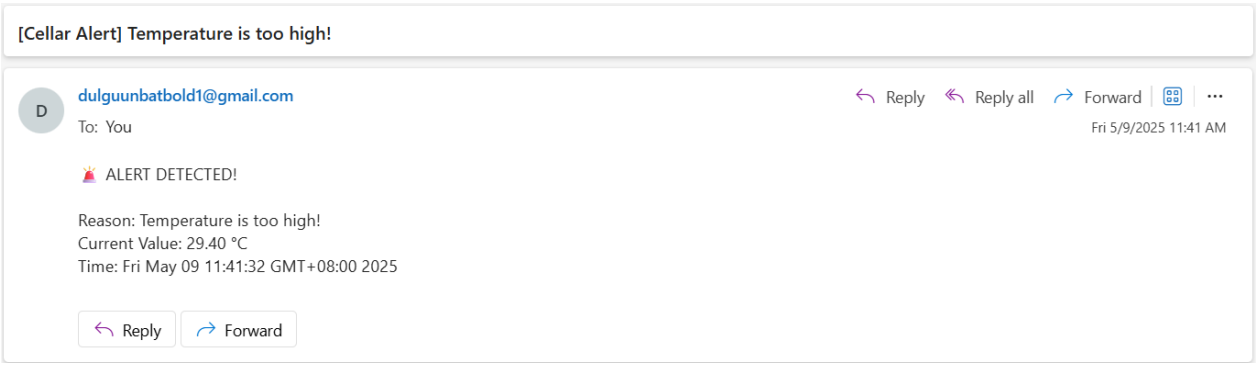
Зураг 5.3: Тохиргооны хуудас

Систем нь хэрэглэгчид тохиргооны профайл үүсгэх, устгах, засах, болон аль нэгийг идэвхжүүлэх боломжийг олгоно. Тухайн тохиргоонд ‘minTemp‘, ‘maxTemp‘, ‘minHum‘, ‘maxHum‘ утгууд оршино.

Frontend хэсэгт ‘Config.vue‘ хуудас үүсгэгдсэн бөгөөд хэрэглэгч дараах боломжтой:

- Бүх тохиргооны жагсаалтыг харах
- Шинэ тохиргоо нэмэх
- Одоогийн идэвхтэй тохиргоог өөрчлөх
- Тохиргоо устгах

Backend нь ‘/api/config‘ маршрутаар ажиллаж, REST API хэлбэрээр тохиргооны өгөгдлийг CRUD хэлбэрээр удирдана. Хэрэглэгчийн сонгосон тохиргоо нь MQTT ‘esp32/config‘ сэдвээр ESP32 рүү бодит цагийн бадлаар илгээгддэг.



Зураг 5.4: Мэдэгдэл ирсэн байдал

Систем нь орчны нөхцөл байдал тохиргоонд заасан хязгаараас хэтэрсэн тохиолдолд хэрэглэгчид анхааруулах и-мэйл мэдэгдэл илгээдэг.

Мэдэгдэл илгээх үйл явц дараах алхмаар хийгдэнэ:

1. ESP32 нь температур болон чийгшлийн мэдрэгчээс өгөгдлийг уншина.
2. Уг өгөгдлийг MQTT сэдвээр backend рүү илгээдэг.
3. Backend систем нь тухайн өгөгдлийг шалгаж:
4. Хэрвээ утгууд тохиргооны хязгаараас хэтэрсэн байвал
5. Тохирлын логик шалгасны дараа, мэдэгдэл илгээх үйлчилгээ (EmailService) ажиллаж, хэрэглэгчийн бүртгэлтэй и-мэйл хаяг руу анхааруулга илгээнэ.

Дүгнэлт

Энэхүү төслийн хүрээнд орчны температур болон чийгшлийг хянаж, автомат удирдлагатай зоорийн системийг боловсруулан амжилттай хэрэгжүүллээ. Төслийн үндсэн зорилго нь зоорийн орчинд хадгалагдаж буй бүтээгдэхүүний чанар, хадгалалтын нөхцлийг сайжруулах, хүний оролцоог багасгах, хяналтын процессыг ухаалаг болгоход чиглэсэн байв.

Систем нь ESP32 микроконтроллер, DHT11 мэдрэгч, релей модуль зэрэг хямд өртөгтэй төхөөрөмжүүдийг ашиглан орчны мэдээллийг цуглуулж, MQTT протокол ашиглан өгөгдлийг сервер рүү дамжуулдаг. Backend нь өгөгдлийг хадгалах, боловсруулж хариу үйлдэл хийх, тохиргоог хэрэглэгчээс хүлээн авах, relay хяналт хийх, бодит цагийн өгөгдлийг түгээх зэрэг олон үүрэг гүйцэтгэдэг. Frontend хэсэг нь Vue.js ашиглан бүтээсэн бөгөөд хэрэглэгчид өгөгдлийг графикаар харах, төхөөрөмжийн төлвийг хянах, тохиргоог удирдах боломжийг олгодог.

Системийн үндсэн хэсгүүд нь:

- Температур, чийгшил хэмжих DHT11 мэдрэгч
- Дулааны хяналт, агааржуулалтын удирдлага хийх 4 сувгийн релей
- Хяналтын дохио үзүүлэх LED гэрлүүд
- ESP32 микроконтроллер ашиглан мэдээллийг боловсруулах, хянах
- Технологийн судалгааны үр дүнд ESP32, MySQL, Spring Boot, Vue.js ашиглах боломжийг судалж, хамгийн тохиромжтой шийдлүүдийг сонгов. Системийн шинжилгээний үе шатанд төхөөрөмж болон программ хангамжийн шаардлагыг тодорхойлж, Use-Case, ERD болон өгөгдлийн урсгалын диаграммуудыг гаргав.

Хэрэгжүүлэлтийн явцад:

- Ашигласан MQTT topic-ууд:

  - `esp32/data` – мэдрэгчийн мэдээлэл
  - `esp32/config` – серверээс ирсэн тохиргоо
  - `esp32/relay/status` – одоогийн релейний төлөв
  - `esp32/alert` – анхааруулга мессежийн мэдээлэл
- ESP32
  - WiFi болон MQTT брокерт холбогдоно.
  - DHT11 мэдрэгчээс температурыг болон чийгшлийг уншина.
  - `minTemp`, `maxTemp`, `minHum`, `maxHum` утгууд дээр үндэслэн хяналтын логик ажиллуулна.
  - MQTT-ээр ирж буй тохиргооны мэдээллийг (`esp32/config`) хүлээн авч, утгуудыг шинэчилнэ.
  - Релейн төлвийг хадгалж, зөвхөн өөрчлөгдсөн үед нь MQTT-д илгээнэ.
  - Температур, чийгшлийн утга өөрчлөгдөх үед л мэдрэгчийн мэдээллийг илгээнэ.
  - Температур, чийгшлийн утга дээд, доод хязгаараас хэтэрсэн тохиолдолд мэдэгдэл явуулна.
- Backend дээр
  - `esp32/data`, `esp32/relay/status`, `esp32/alert` сэдвүүдэд subscribe хийнэ.
  - JSON мэдээллийг задлаж, мэдрэгч болон релейний мэдээллийг өгөгдлийн санд хадгална.
  - `SensorData` модель, репозитор, үйлчилгээ, контроллер бичигдсэн.
  - `RelayStatus` модель, репозитор, үйлчилгээ, контроллер бичигдсэн.
  - `SensorConfig` модель, репозитор, үйлчилгээ, контроллер бичигдсэн.
  - Мэдрэгч болон релейний өөрчлөлтүүдийг орж ирсэн цагтай нь бүртгэнэ.

  - esp32/alert topic дээр мэдэгдэл орж ирвэл хэрэглэгчид майлээр мэдэгдэнэ.

- Frontend дээр
  - Хамгийн сүүлийн мэдрэгчийн мэдээллийг графикаар харуулна.
  - Релейний төлөвийг бодит цагийн байдлаар харуулна.
  - Шинээр тохиргоо үүсгэх, устгах, өөрчлөх, сонгох боломжтой.
  - Халаагуур, хөргүүр зэрэг төхөөрөмжүүдийг автоматаар болон гараар удирдах боломжтой.

Систем нь гэр ахуйн болон жижиг фермийн орчинд хэрэгжүүлэх боломжтой, уян хатан, өргөтгөх боломжтой, хямд өртөгтэй шийдэл болж чадсан. Уг төсөл нь IoT системийн зохион байгуулалт, өгөгдөл дамжуулах протоколууд, микроконтроллер програмчлал, backend хөгжүүлэлт, мөн хэрэглэгчийн интерфейсийн шийдэл зэрэг олон чиглэлийг хамарч, оюутны хувьд цогц мэдлэг олгох бодит туршлага болсон.

Bibliography

[1] Espressif Systems. (2020). *ESP32 Technical Reference Manual*. https://www.espressif.com/en/products/socs/esp32/resources

[2] Aosong Electronics. (2018). *DHT11 Temperature & Humidity Sensor Datasheet*. https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT11.pdf

[3] Spring.io. (2024). *Spring Boot Documentation*. https://docs.spring.io/spring-boot/docs/current/reference/html/

[4] Vue.js. (2024). *Vue.js Documentation*. https://vuejs.org/guide/introduction.html

[5] MySQL. (2024). *MySQL Documentation*. https://dev.mysql.com/doc/

[6] GitHub. (n.d.). *Cellar Control System by Dulguun*. https://github.com/Dulguun06/cellar-control-system

А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <ArduinoJson.h>
#include <WiFiClientSecure.h>

// WiFi credentials
const char *ssid = "XXXXXXXXX";
const char *password = "XXXXXXX";

// MQTT broker
const char *mqttServer = "XXX.XXX.XXX.XXX";
const int mqttPort = 1883;

const char *mqtt_user = "XXXX";
const char *mqtt_pass = "XXXXXX";

const char *root_ca = R"EOF(
-----BEGIN CERTIFICATE-----
MIIETTCCAzWgAwIBAgIUSI6cDJLEz3r0VuIgKhH5QzBXlmowDQYJKoZIhvcNAQEL
BQAwgbUxCzAJBgNVBAYTAk1OMRQwEgYDVQQIDAtVbGFhbmJhYXRhcjENMAsGA1UE
BwwEY2l0eTEoMCYGA1UECgwfTmF0aW9uYWwgVW5pdmVyc2l0eSBvZiBNb25nb2xp
YTEbMBkGA1UECwwSRW5naW5lZXJpbmcgU2Nob29sMRAwDgYDVQQDDAdEdWxndXVu
MSgwJgYJKoZIhvcNAQkBFhlkdWxndXVuYmF0Ym9sZDFAZ21haWwuY29tMB4XDTI1
MDUwNTEzMTI0M1oXDTI2MDUwNTEzMTI0M1owgbUxCzAJBgNVBAYTAk1OMRQwEgYD
VQQIDA|.....................................................
.....CLEARED THIS PART BECAUSE OF SECURITY REASONS.....
.....................................................|sxvBn11vH
QxbGquZIuCzCTl1hoQkw7uJp4OBUv65MS4wnbxbrG0jcgozAS64OF9nt6XjPD1O1
e1r0/zLT336Gto5xZmpK1ybix6tCpGtBtPbgV+x6bDCkDgJ9QcI+aFAGFM30MhSs
CXcpIA9C1PeR4cIn9jfLsOK7gl4/odyyiUKsxWhomwGXLTOAypk6ITN7h0eII8lR
8dQsASq9wjN2CyaimCsDSLzfMvKf6KFPaeHvTkEN3g6jSP/6x2oKqCuQd8ufl/5f
3croBvAsKlh1JIwpoCMikI81cbVWzvyK1ydRGY5lc2y6VynDd79enHb+t4shBcPG
aw==
-----END CERTIFICATE-----
)EOF";

// MQTT topics
const char *dataTopic = "esp32/data";
const char *configTopic = "esp32/config";
const char *relayStatusTopic = "esp32/relay/status";

// DHT settings
#define DHTPIN 4
#define DHTTYPE DHT11

// Relay pins
#define HEATER 25
```

```
#define COOLER 26
#define HUMIDIFIER 32
#define VENTILATION 33

WiFiClientSecure espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

// Default thresholds (can be updated by MQTT config)
float minTemp = 4.0, maxTemp = 15.0;
float minHum = 80.0, maxHum = 95.0;

unsigned long lastSendTime = 0;
const unsigned long sendInterval = 1000;

// Store previous relay states
bool lastHeater = false;
bool lastCooler = false;
bool lastHumidifier = false;
bool lastVentilation = false;

//Sensor Data
float lastTemp = NAN;
float lastHum = NAN;
const float changeThreshold = 0.5;

void setup() {
  Serial.begin(115200);
  dht.begin();

  pinMode(HEATER, OUTPUT);
  pinMode(COOLER, OUTPUT);
  pinMode(HUMIDIFIER, OUTPUT);
  pinMode(VENTILATION, OUTPUT);

  // Turn everything off initially
  digitalWrite(HEATER, LOW);
  digitalWrite(COOLER, LOW);
  digitalWrite(HUMIDIFIER, LOW);
  digitalWrite(VENTILATION, LOW);

  connectToWiFi();

  configTime(0, 0, "pool.ntp.org");  // Set time via NTP
  while (time(nullptr) < 8 * 3600 * 2) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Time synchronized.");

  espClient.setCACert(root_ca);
  client.setServer(mqttServer, mqttPort);
```

```
  client.setCallback([](char *topic, byte *payload, unsigned int length) {
    if (String(topic) == configTopic) {
      payload[length] = '\0';
      handleConfigUpdate((char *)payload);
    }
  });

  connectToMQTT();
  client.subscribe(configTopic);
}

void loop() {
  if (!client.connected()) connectToMQTT();
  client.loop();

  unsigned long now = millis();
  if (now - lastSendTime >= sendInterval) {
    sendSensorData();
    lastSendTime = now;
  }
}

void connectToWiFi() {
  Serial.print("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected!");
}

void connectToMQTT() {
  while (!client.connected()) {
    Serial.print("Connecting to MQTT...");
    if (client.connect("ESP32Client", mqtt_user, mqtt_pass)) {
      Serial.println(" connected!");
      client.subscribe(configTopic);
    } else {
      Serial.print(" failed, rc=");
      Serial.println(client.state());
      delay(5000);
    }
  }
}

void handleConfigUpdate(char *payload) {
  StaticJsonDocument<256> doc;
  DeserializationError error = deserializeJson(doc, payload);
  if (error) {
    Serial.println("Failed to parse config JSON");
```

```
    return;
  }

  minTemp = doc["minTemp"] | minTemp;
  maxTemp = doc["maxTemp"] | maxTemp;
  minHum = doc["minHum"] | minHum;
  maxHum = doc["maxHum"] | maxHum;

  Serial.println("Updated config:");
  Serial.printf("Temp: %.2f - %.2f | Hum: %.2f - %.2f\n", minTemp,
     maxTemp, minHum, maxHum);
}

void applyControl(float temp, float hum) {
  bool isCold = temp < minTemp;
  bool isHot = temp > maxTemp;
  bool isHumid = hum > maxHum;
  bool isDry = hum < minHum;

  bool heater = isCold && !isHumid;
  bool cooler = isHot;
  bool humidifier = isDry && !isHot;
  bool ventilation = isHumid || (isCold && isHumid);

  digitalWrite(HEATER, heater ? HIGH : LOW);
  digitalWrite(COOLER, cooler ? HIGH : LOW);
  digitalWrite(HUMIDIFIER, humidifier ? HIGH : LOW);
  digitalWrite(VENTILATION, ventilation ? HIGH : LOW);

  // Send MQTT only if any status has changed
  if (heater != lastHeater || cooler != lastCooler || humidifier !=
     lastHumidifier || ventilation != lastVentilation) {
    sendRelayStatus(heater, cooler, humidifier, ventilation);
    lastHeater = heater;
    lastCooler = cooler;
    lastHumidifier = humidifier;
    lastVentilation = ventilation;
  }
}

void sendSensorData() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();

  Serial.printf("Sensor: %.2f°C, %.2f%%\n", temp, hum);

  applyControl(temp, hum);

  // Only send if temp or hum changed more than the threshold
  if (isnan(lastTemp) || isnan(lastHum) || abs(temp - lastTemp) >
     changeThreshold || abs(hum - lastHum) > changeThreshold) {

```

```
    StaticJsonDocument<256> doc;
    JsonObject sensor = doc.createNestedObject("sensor");
    sensor["temp"] = temp;
    sensor["hum"] = hum;

    char buffer[256];
    size_t len = serializeJson(doc, buffer);
    client.publish(dataTopic, buffer, len);

    Serial.println("Sensor data changed → sent to MQTT.");

    // Update previous values
    lastTemp = temp;
    lastHum = hum;
  } else {
    Serial.println("No change in sensor data → MQTT not sent.");
  }
}


void sendRelayStatus(bool heater, bool cooler, bool humidifier, bool
   ventilation) {
  StaticJsonDocument<128> doc;
  doc["heater"] = heater;
  doc["cooler"] = cooler;
  doc["humidifier"] = humidifier;
  doc["ventilation"] = ventilation;

  char buffer[128];
  size_t len = serializeJson(doc, buffer);
  client.publish(relayStatusTopic, buffer, len);

  Serial.println("Relay status updated:");
  Serial.println(buffer);
}


void checkAndSendAlerts(float temp, float hum) {
  StaticJsonDocument<128> alertDoc;
  bool alertTriggered = false;

  if (temp > maxTemp && !tempHighAlertSent) {
    alertDoc["type"] = "TEMP_HIGH";
    alertDoc["message"] = "Temperature is too high!";
    alertDoc["value"] = temp;
    tempHighAlertSent = true;
    alertTriggered = true;
  } else if (temp <= maxTemp) {
    tempHighAlertSent = false;
  }

  if (temp < minTemp && !tempLowAlertSent) {
```

```
    alertDoc["type"] = "TEMP_LOW";
    alertDoc["message"] = "Temperature is too low!";
    alertDoc["value"] = temp;
    tempLowAlertSent = true;
    alertTriggered = true;
  } else if (temp >= minTemp) {
    tempLowAlertSent = false;
  }

  if (hum > maxHum && !humHighAlertSent) {
    alertDoc["type"] = "HUM_HIGH";
    alertDoc["message"] = "Humidity is too high!";
    alertDoc["value"] = hum;
    humHighAlertSent = true;
    alertTriggered = true;
  } else if (hum <= maxHum) {
    humHighAlertSent = false;
  }

  if (hum < minHum && !humLowAlertSent) {
    alertDoc["type"] = "HUM_LOW";
    alertDoc["message"] = "Humidity is too low!";
    alertDoc["value"] = hum;
    humLowAlertSent = true;
    alertTriggered = true;
  } else if (hum >= minHum) {
    humLowAlertSent = false;
  }

  if (alertTriggered) {
    char alertBuffer[128];
    size_t len = serializeJson(alertDoc, alertBuffer);
    client.publish(alertTopic, alertBuffer, len);
    Serial.println("Alert sent:");
    Serial.println(alertBuffer);
  }
}
```

Код А.1: ESP32-т зориулан бичсэн код

```
package num.edu.cellar.config;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MqttConfig {

    @Bean
```

```
    public MqttClient mqttClient() throws MqttException {
        String broker = "tcp://xxx.xxx.x.x:xxxx"; //MQTT Broker - ip
        String clientId = "SpringBootClient";
        MqttClient client = new MqttClient(broker, clientId);
        MqttConnectOptions options = new MqttConnectOptions();
        options.setAutomaticReconnect(true);
        options.setCleanSession(true);

        client.connect(options);
        return client;
    }
}
```

Код А.2: MqttConfig.java тохиргоо

```
package num.edu.cellar.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.*;

@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/api/**")
                .allowedOrigins("*")
                .allowedMethods("GET", "POST", "PUT", "DELETE") //
                   Allowed HTTP methods
                .allowedHeaders("*") // Allowed request headers
                .allowCredentials(false)
                .maxAge(3600);
    }
}
```

Код А.3: webconfig.java CORS тохиргоо

```
package num.edu.cellar.model;

import jakarta.persistence.*;
import lombok.*;

import java.time.LocalDateTime;
import java.util.Date;

@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class SensorData {

```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private float temperature;
    private float humidity;
    private Date timestamp;
}
```

Код А.4: SensorData.java мэдрэгчийн мэдээллийг хадгалах обьектын ашиглах класс

```
package num.edu.cellar.controller;

import lombok.RequiredArgsConstructor;
import num.edu.cellar.model.SensorData;
import num.edu.cellar.repository.SensorDataRepository;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/api/sensor")
@RequiredArgsConstructor
public class SensorDataController {

    private final SensorDataRepository repository;

    @GetMapping("/all")
    public List<SensorData> getAllData() {
        return repository.findAll();
    }

    @GetMapping("/latest")
    public SensorData getLatestData() {
        return repository.findTopByOrderByTimestampDesc();
    }
}
```

Код А.5: SensorDataController.java мэдрэгчүүдийн мэдээллийг авахад ашиглах endpoint-ууд

```
package num.edu.cellar.service;

import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import jakarta.annotation.PostConstruct;
import lombok.RequiredArgsConstructor;
import num.edu.cellar.model.RelayStatus;
import num.edu.cellar.model.SensorData;
import num.edu.cellar.repository.RelayStatusRepository;
import num.edu.cellar.repository.SensorDataRepository;
import org.eclipse.paho.client.mqttv3.*;
import org.springframework.stereotype.Service;
```

```
13
14 import java.sql.Date;
15
16 @Service
17 @RequiredArgsConstructor
18 public class MqttListenerService {
19
20     private final MqttClient mqttClient;
21     private final SensorDataRepository repository;
22     private final RelayStatusRepository relayStatusRepository;
23     private final EmailService emailService;
24     private final ObjectMapper objectMapper = new ObjectMapper();
25
26     @PostConstruct
27     public void subscribeToTopic() throws MqttException {
28         mqttClient.subscribe("esp32/data", this::handleSensorData);
29         mqttClient.subscribe("esp32/relay/status", this::
               handleRelayStatus);
30         mqttClient.subscribe("esp32/alert", this::handleAlertMessage);
31     }
32
33     private void handleSensorData(String topic, MqttMessage message) {
34         {
35             String payload = new String(message.getPayload());
36             System.out.println("Received: " + payload);
37
38             try {
39                 JsonNode root = objectMapper.readTree(payload);
40                 JsonNode sensor = root.get("sensor");
41
42                 if (sensor != null && sensor.has("temp") && sensor.has(
                       "hum")) {
43                     SensorData data = SensorData.builder()
44                             .temperature((float) sensor.get("temp").
                                asDouble())
45                             .humidity((float) sensor.get("hum").
                                asDouble())
46                             .timestamp(new Date(System.
                                currentTimeMillis()))
47                             .build();
48
49                     repository.save(data);
50                     System.out.println("Sensor data saved to DB.");
51                 } else {
52                     System.err.println("Invalid sensor data format.");
53                 }
54             } catch (Exception e) {
55                 System.err.println("Failed to parse or save: " + e.
                       getMessage());
56             }
57         }
58     }
```

```
59
60      private void handleRelayStatus(String topic, MqttMessage message) {
61          String payload = new String(message.getPayload());
62          System.out.println("Relay Status Received: " + payload);
63
64          try {
65              JsonNode root = objectMapper.readTree(payload);
66
67              RelayStatus status = RelayStatus.builder()
68                      .heater(root.get("heater").asBoolean())
69                      .cooler(root.get("cooler").asBoolean())
70                      .humidifier(root.get("humidifier").asBoolean())
71                      .ventilation(root.get("ventilation").asBoolean())
72                      .timestamp(new Date(System.currentTimeMillis()))
73                      .build();
74              relayStatusRepository.save(status);
75              System.out.println("Relay status saved.");
76          } catch (Exception e) {
77              System.err.println("Failed to parse relay status: " + e.
                    getMessage());
78          }
79      }
80
81      public void handleAlertMessage(String topic, MqttMessage
            mqttMessage) {
82          try {
83              JsonNode node = objectMapper.readTree(mqttMessage.
                    getPayload());
84              String type = node.get("type").asText(); // TEMP_HIGH,
                    HUM_LOW, etc.
85              String message = node.get("message").asText(); // e.g. "
                    Temperature is too high!"
86              double value = node.get("value").asDouble();
87
88              // Set units based on alert type
89              String unit;
90              if (type.startsWith("TEMP")) {
91                  unit = "°C";
92              } else if (type.startsWith("HUM")) {
93                  unit = "%";
94              } else {
95                  unit = "";
96              }
97
98              String subject = "[Cellar Alert] " + message;
99              String body = String.format("""
100                       ALERT DETECTED!
101
102                     Reason: %s
103                     Current Value: %.2f %s
104                     Time: %s
105                     """, message, value, unit, new java.util.Date());
```

```
106
107             emailService.sendAlert(subject, body);
108             System.out.println("Email alert sent: " + subject);
109         } catch (Exception e) {
110             System.err.println("Failed to handle alert message: " + e.
                    getMessage());
111             e.printStackTrace();
112         }
113     }
114     // Publish relay control to MQTT
115     public void publishRelayControl(boolean heater, boolean cooler,
            boolean humidifier, boolean ventilation) {
116         try {
117             String message = String.format("{\"heater\":%b, \"cooler
                    \":%b, \"humidifier\":%b, \"ventilation\":%b}",
118                     heater, cooler, humidifier, ventilation);
119             mqttClient.publish("esp32/relay/control", message.getBytes
                    (), 0, false);
120         } catch (MqttException e) {
121             throw new RuntimeException("Error publishing relay control"
                    , e);
122         }
123     }
124
125     // Publish mode switch (manual/auto) to MQTT
126     public void publishModeSwitch(boolean isAutoMode) {
127         try {
128             String message = String.format("{\"isAutoMode\":%b}",
                    isAutoMode);
129             mqttClient.publish("esp32/mode", message.getBytes(), 0,
                    false);
130         } catch (MqttException e) {
131             throw new RuntimeException("Error publishing mode switch",
                    e);
132         }
133     }
134 }
```

Код А.6: MqttListenerService.java релей болон мэдрэгчийн мэдээллийг ESP32-с авах

```
1  package num.edu.cellar.repository;
2
3  import num.edu.cellar.model.SensorData;
4  import org.springframework.data.jpa.repository.JpaRepository;
5
6  import java.util.List;
7
8  public interface SensorDataRepository extends JpaRepository<SensorData,
       Long> {
9
10     SensorData findTopByOrderByTimestampDesc();
```

```
}
```

Код А.7: SensorDataRepository.java өгөгдлийн сангаас өгөгдөл авах

```
package num.edu.cellar.service;

import lombok.RequiredArgsConstructor;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class EmailService {
    private final JavaMailSender mailSender;
    private String recipientEmail = "dulguunbatbold1@outlook.com"; //
       Default email

    public void sendAlert(String subject, String message) {
        SimpleMailMessage mail = new SimpleMailMessage();
        mail.setTo(recipientEmail);
        mail.setSubject(subject);
        mail.setText(message);
        try {
            mailSender.send(mail);
            System.out.println("Email sent successfully to " +
               recipientEmail);
        } catch (Exception e) {
            System.err.println("Failed to send email: " + e.getMessage
               ());
            e.printStackTrace();
        }
    }

    public String getRecipientEmail() {
        return recipientEmail;
    }

    public void updateRecipientEmail(String newEmail) {
        this.recipientEmail = newEmail;
        System.out.println("Notification email updated to: " + newEmail
           );
    }
}
```

Код А.8: EmailService.java хэрэглэгчид мэдэгдэл илгээх үйлчилгээ

Frontend-н кодууд

```
import axios from 'axios'

const API_URL = 'http://localhost:8080/api'

export const getLatestSensorData = () => {
```

```
    return axios.get(`${API_URL}/sensor/latest`)
}
export const getAllSensorData = ()=>{
    return axios.get(`${API_URL}/sensor/all`)
}
export const getRelayStatus = () => {
    return axios.get(`${API_URL}/relay/latest`)
}
```

Код А.9: api.js backend-с өгөгдөл татахад ашиглагдана.

Home.vue нь vue component-уудыг нэгтгэх гол суурь нь юм. Эндээс Navbar ашиглан бусад хуудсуудыг өөр дээрээ харуулах юм.

```
<template>
  <div>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Cellar Control</a>
        <button
            class="navbar-toggler"
            type="button"
            data-bs-toggle="collapse"
            data-bs-target="#navbarNav"
            aria-controls="navbarNav"
            aria-expanded="false"
            aria-label="Toggle navigation"
        >
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav ms-3">
            <li class="nav-item">
              <a class="nav-link" :class="{ active: activeTab === '
                 realtime' }" href="#" @click="activeTab = 'realtime'">
                 Dashboard</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" :class="{ active: activeTab === '
                 config' }" href="#" @click="activeTab = 'config'">
                 Configurations</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" :class="{ active: activeTab === '
                 profile' }" href="#" @click="activeTab = 'profile'">
                 User Profile</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>

    <!-- Main Content Section -->
```

```
    <div class="container mt-4">
      <Dashboard v-if="activeTab === 'realtime'" />
      <Configs v-if="activeTab === 'config'" />
      <Profile v-if="activeTab === 'profile'" />
    </div>
  </div>
</template>

<script>

import Dashboard from './Dashboard.vue';
import Configs from './Configs.vue';
import Profile from './Profile.vue';

export default {
  components: {
    Dashboard,
    Configs,
    Profile,
  },
  data() {
    return {
      activeTab: 'realtime',
    };
  },
};
</script>
```

Код А.10: Home.vue

Dashboar.vue нь өөртөө DonutChart, SensorGraph, RelayStatus гэсэн 3 хэсгээс бүтэх ба эдгээрийг нэгтгэн харуулах үүрэгтэй

```
<template>
  <div class="container py-4">
    <h1 class="text-center mb-5 display-5 fw-bold">Cellar Dashboard</h1
       >

    <div class="row mb-3">
      <div class="col-md-3 mb-3 mb-md-0">
        <DonutChart />
      </div>
      <div class="col-md-9">
        <SensorGraph />
      </div>
    </div>

    <div class="row">
      <div class="col">
        <RelayStatus />
      </div>
    </div>
  </div>
```

```
</template>

<script setup>
import DonutChart from "@/components/DonutChart.vue";
import SensorGraph from "@/components/SensorGraph.vue";
import RelayStatus from "@/components/RelayStatus.vue";
</script>
```

Код А.11: Dashboard.vue ESP32-н мэдээллйг харуулах view.

```
<template>
  <div class="flex flex-col items-center row-gap-5">
    <!-- Temperature Donut Chart -->
    <div class="flex flex-col items-center">
      <h2 class="text-xl font-semibold mb-2">Temperature</h2>
      <div class="w-48 h-48">
        <canvas ref="tempChartRef"></canvas>
      </div>
      <p class="text-lg font-semibold mt-2">{{ sensorData.temperature
         }} °C</p>
    </div>

    <!-- Humidity Donut Chart -->
    <div class="flex flex-col items-center" id="hum">
      <h2 class="text-xl font-semibold mb-2">Humidity</h2>
      <div class="w-48 h-48">
        <canvas ref="humidityChartRef"></canvas>
      </div>
      <p class="text-lg font-semibold mt-2">{{ sensorData.humidity }}
         %</p>
    </div>
  </div>
</template>

<script setup>
import { ref, onMounted, onBeforeUnmount } from "vue";
import { Chart } from "chart.js/auto";
import { getLatestSensorData } from "@/service/api";

const tempChartRef = ref(null);
const humidityChartRef = ref(null);
const sensorData = ref({
  temperature: 0,
  humidity: 0
});
let chartInstanceTemp = null;
let chartInstanceHumidity = null;

// Function to create the donut chart
const createDonutChart = (refElement, data, label, color) => {
  return new Chart(refElement, {
    type: 'doughnut',
```

```
    data: {
      labels: [label],
      datasets: [{
        data: [data, 100 - data],
        backgroundColor: [color, '#e0e0e0'],
        borderWidth: 0,
      }]
    },
    options: {
      responsive: true,
      plugins: {
        tooltip: {
          enabled: false
        },
        legend: {
          display: false
        },
        datalabels: {
          display: true,
          color: '#333',
          font: {
            weight: 'bold',
            size: 18,
          },
          formatter: (value) => value + '%',
        }
      }
    }
  });
};

// Fetch latest sensor data
const fetchSensorData = async () => {
  try {
    const res = await getLatestSensorData();
    sensorData.value = res.data;
    if (chartInstanceTemp) {
      chartInstanceTemp.data.datasets[0].data = [sensorData.value.
         temperature, 100 - sensorData.value.temperature];
      chartInstanceTemp.update();
    }
    if (chartInstanceHumidity) {
      chartInstanceHumidity.data.datasets[0].data = [sensorData.value.
         humidity, 100 - sensorData.value.humidity];
      chartInstanceHumidity.update();
    }
  } catch (error) {
    console.error("Failed to fetch sensor data:", error);
  }
};

// Set interval to fetch sensor data every second
```

```
let intervalId = null;

onMounted(() => {
  // Create initial charts
  chartInstanceTemp = createDonutChart(tempChartRef.value, sensorData.
     value.temperature, 'Temp', 'rgba(255, 99, 132, 1)');
  chartInstanceHumidity = createDonutChart(humidityChartRef.value,
     sensorData.value.humidity, 'Humidity', 'rgba(54, 162, 235, 1)');

  // Start fetching sensor data every second
  intervalId = setInterval(fetchSensorData, 1000);
});

onBeforeUnmount(() => {
  // Clear interval when component is unmounted
  clearInterval(intervalId);

  // Destroy charts when unmounted
  if (chartInstanceTemp) chartInstanceTemp.destroy();
  if (chartInstanceHumidity) chartInstanceHumidity.destroy();
});
</script>

<style scoped>
canvas {
  max-width: 100%;
}
#hum{
  margin-top: 50%;
}
</style>
```

Код А.12: DonutChart.vue яг одоогийн өгөгдлийг харуулах