Functionality:

Synchronization Test (Frontend + Backend):

* Time from when sheep1 “gets lost” until it displays as lost on the monitor

Reliability Test (Frontend):

* Testing the whole system, by making one of the follower sheep missing. (60 trials)
* Testing the whole system, by making one of the leader sheep missing. (60 trials?)

Backend Reliability Results I did ten trials with 100% success, both missing leader sheep and follower sheep are successfully detected.

Procedures:

1. Collect the information from the front end, put them into the required string format for each leader sheep. (See Frond end follower to leader testing video in the Sharepoint folder Design Milestones → Prototype- → Testing Results)

2. Manually send each string to the backend through Bluefruit UART (this step would be automated if we have the GSM module).

3. Backend polls from the Arduino serial monitor and write the information into the main database.

4. From the main database, we check missing sheep by the end of each update, once missing sheep detected, the program will write the missing sheep information to the MISSING database.

5. UI displays the information from the missing database, we have to manually run the UI in order to see it for now, but this could be easily improved if we upgrade our product using firebase or React.

A screenshot of a computer

Description automatically generated

Figure 1 Console output reading from the serial monitor

A picture containing text

Description automatically generated

Figure 2 Send Frontend info through Bluefruit UART

Graphical user interface, application

Description automatically generated

Figure 3 Maindatabase

Graphical user interface, application, table

Description automatically generated

Figure 4 UI displaying the MISSING database information

Graphical user interface

Description automatically generated with medium confidence

Figure 5 English version

Location Accuracy Test (Frontend):

* Bluetooth RSSI range vs distance
  + Leader sheep activity range: 2m, distributed evenly
  + “RSSI is a negative value where the more negative it is, the further away the Bluetooth device. Close devices are usually in the range -10 dBm to -30 dBm while devices at the limit of detection give values less than -90 dBm”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| -45 dBm | -55dBm | -65dBm | -75dBm | -80 dBm | -90 dBm |
| 0m | 0.5m | 1m | 5m | 12m | 30m |

* 5 trials, record the displacement and the GPS recorded, make a table? Or a plot? (Build a calibration curve)
* Residual plot: plot the delay in time versus the predicted location uncertainty

Durability:

Waterproof/Dustproof Test (Collar):

* Put tissue inside and soak in water for 2 hours: Does it get wet?
  + We used rubber bands and tape to secure the openings of the collar, mimicking a seal. When implemented in real life, the seal should be developed such that it is more seamless
  + Tissue did not get wet: Waterproof & Dustproof lv IP6



Portability:

Weight of Collar (Collar):

|  |  |
| --- | --- |
| Functionality |  |
| Synchronization |  |
|  |  |
|  |  |
| Reliability |  |
|  |  |
|  |  |
| Location Accuracy |  |
|  |  |
|  |  |

\* colour code for frontend backend collar