Collector (Arduino)

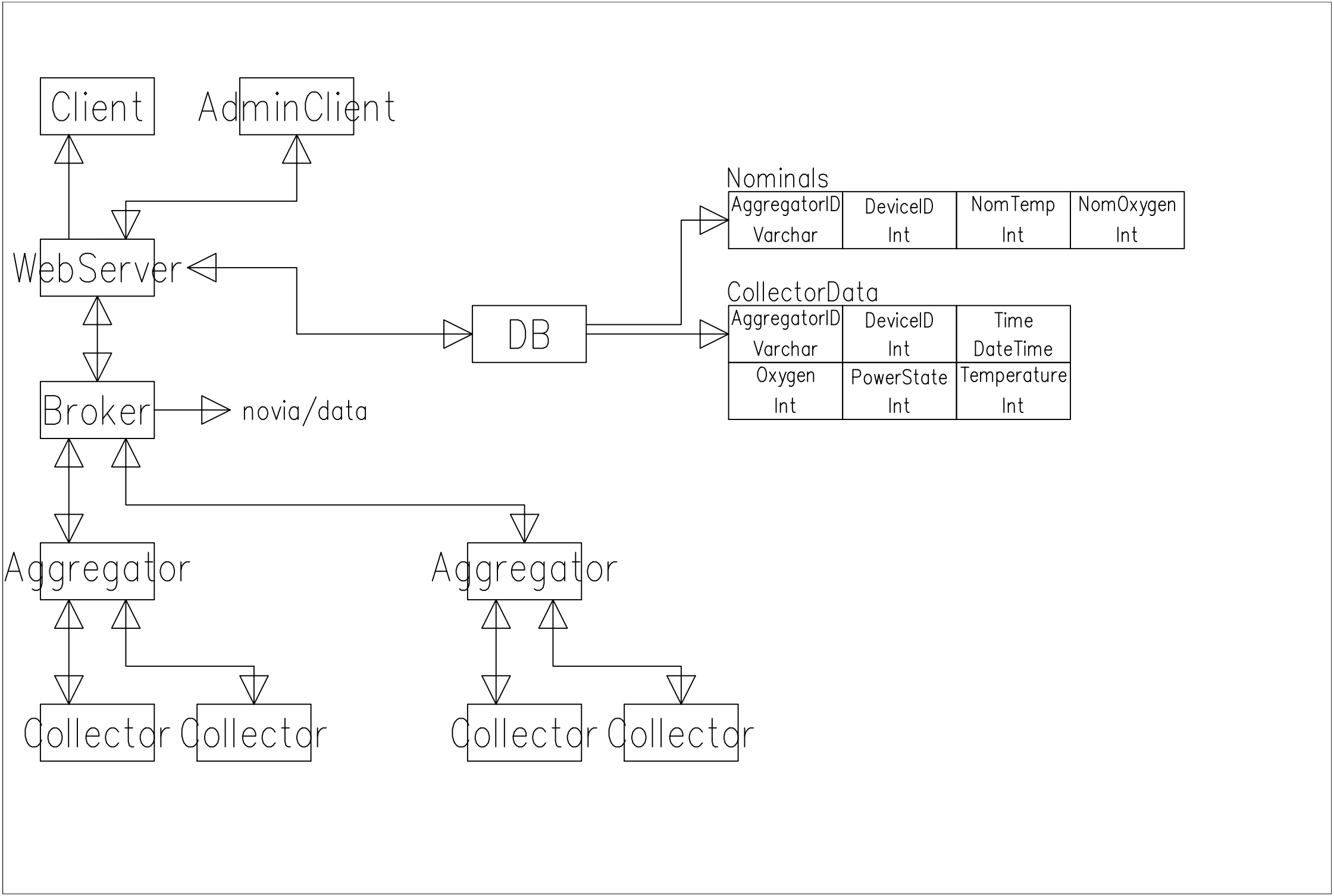
- Panic button (Living quarter)
  - Sends a message to the Aggregator (Raspberry)
    - Aggregator
      - Cuts the electricity
      - Led turns on in the Aggregators panel
      - Send message to collector
- Fan (Living quarter) (Controlled by the collector (Arduino))
  - Temperature is >= 10% above nominal
    - Stops when temperature is < 1 % from nominal
- 2 sensors (Temperature (°C), Oxygen (%))
  - 5 Leds per collector sensor
    - Wired to Collector managed by Aggregator
    - Nominal < 1 % : Green led on
    - Nominal >= 1 % but < 2 % : Yellow led flash rate 1/0,5s
    - Nominal >= 2 % but < 3 % : Yellow led on
    - Nominal >= 3 % but < 5 % : Red led flash rate 1/0,25s
    - Nominal >= 5 % : Red led on

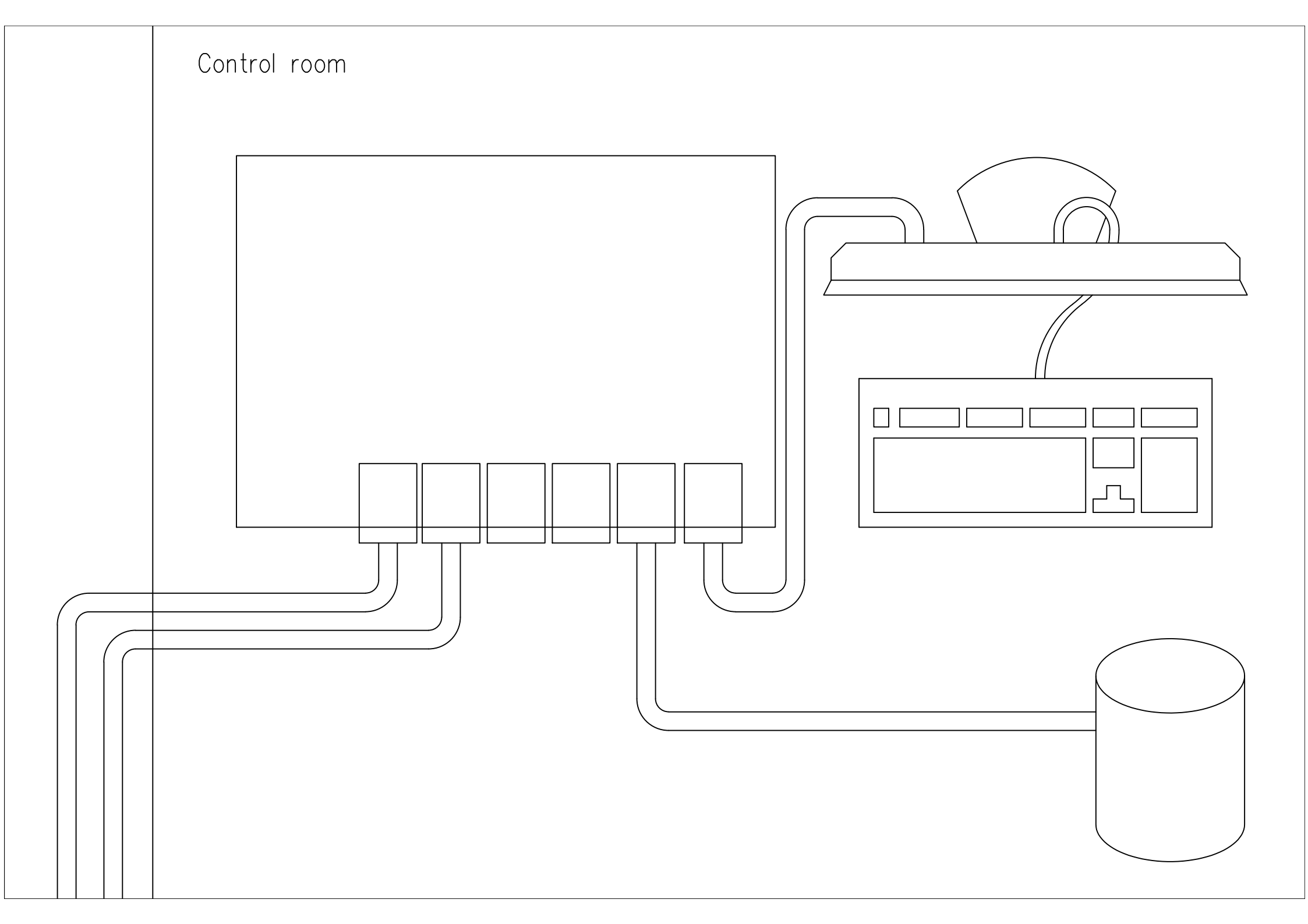
Aggregator (Raspberry)

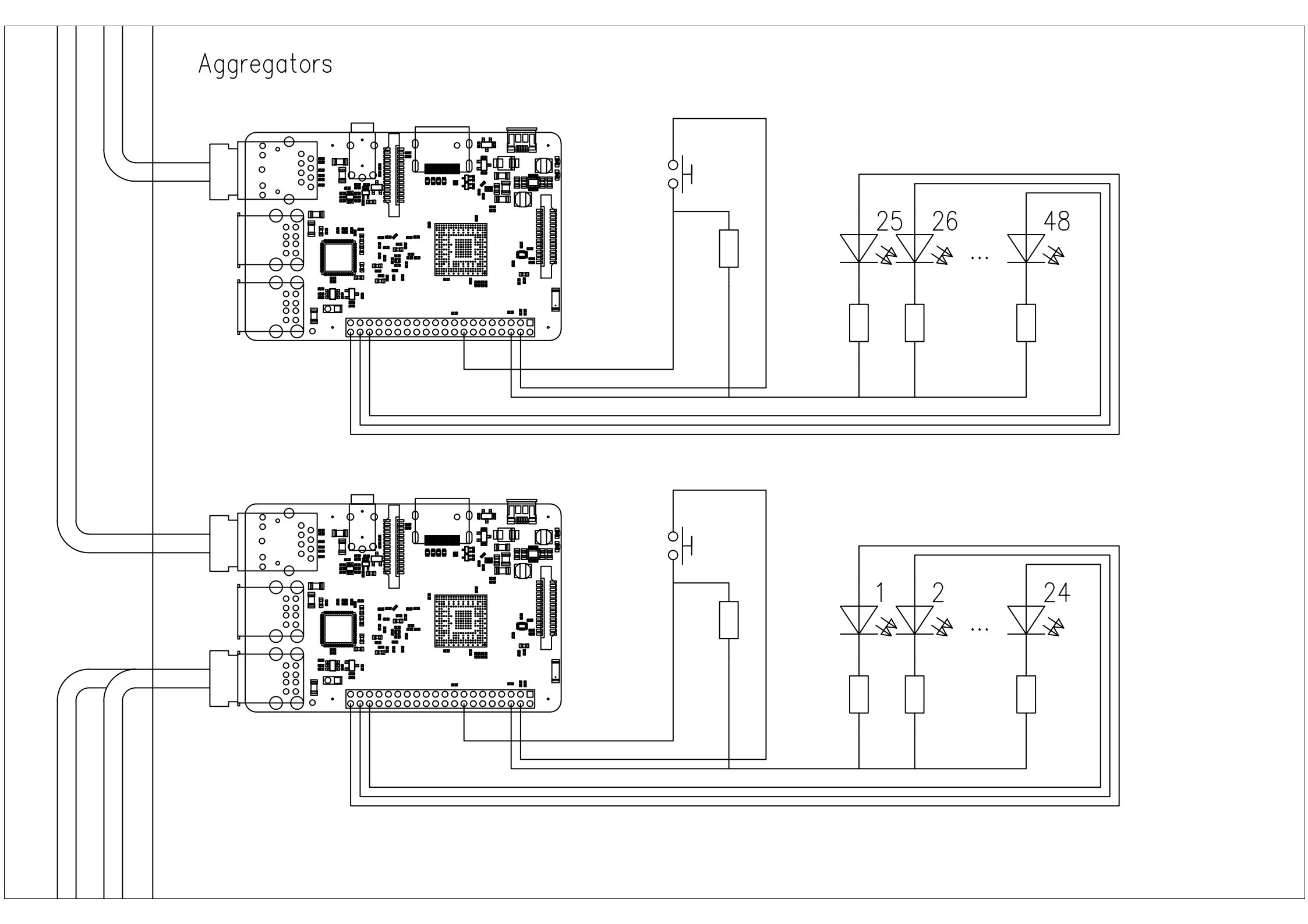
- Stop button (Aggregator)
  - Cuts the electrical feed to all collectors connected to it
  - Sends message to Control room
- 1 Led per collector (Panic status)
- (In debug mode the Aggregator can turn of panic/stop feature)

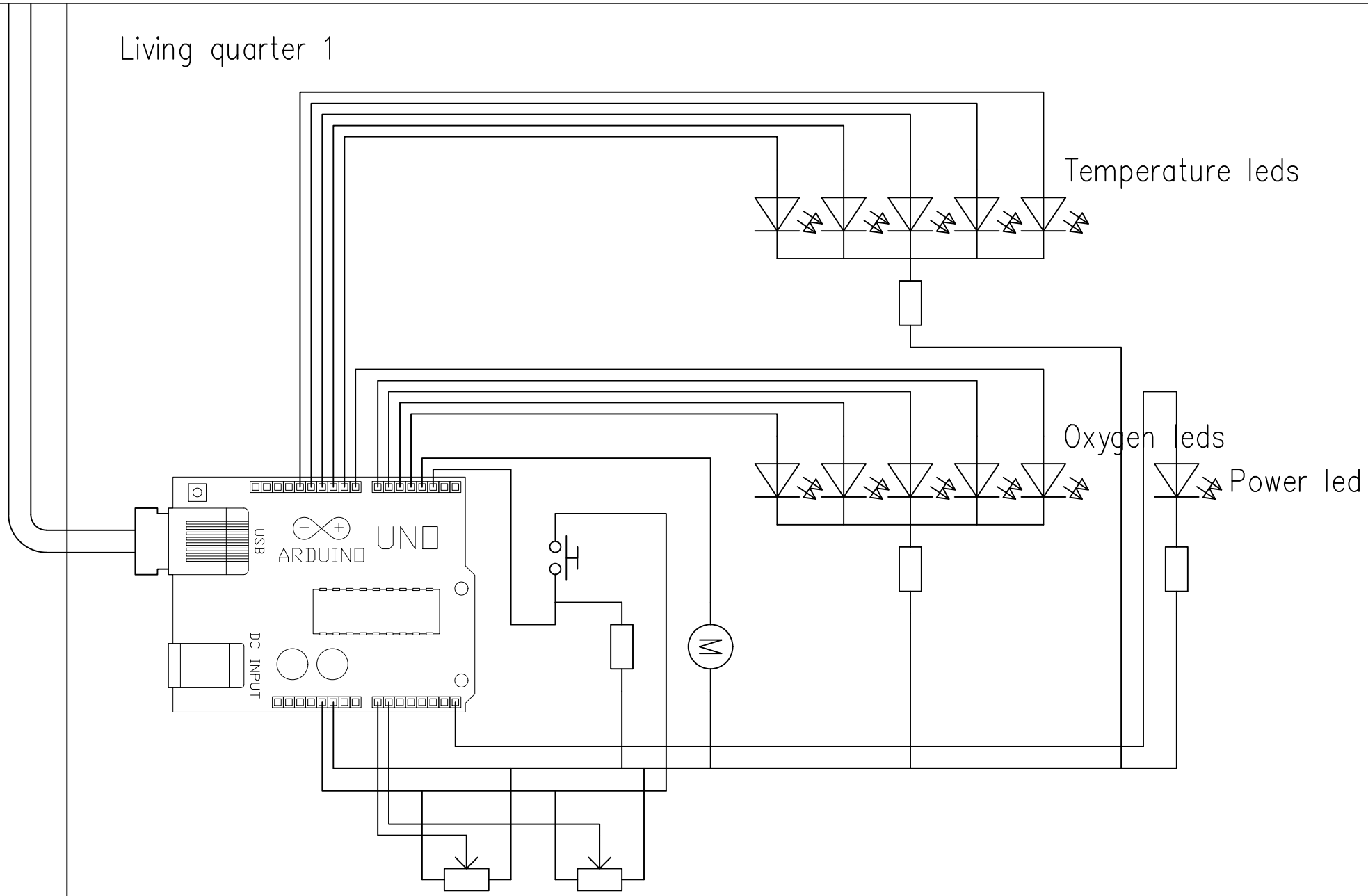
Control room

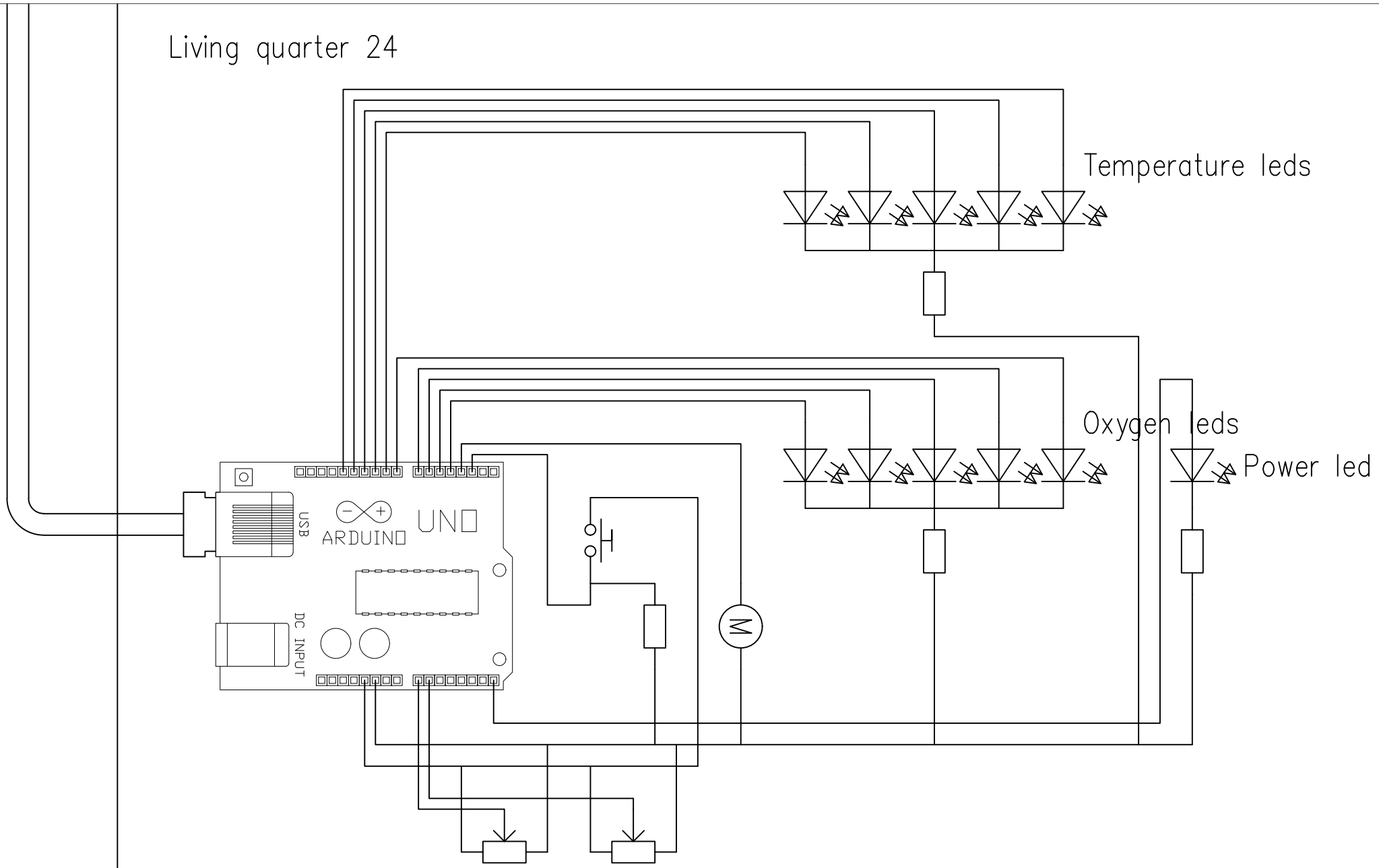
- Receives all messages
- Can turn off the "Panic" feature in the collector
- Can turn off the "Stop" feature in the aggregators
- Update the nominal values in the database / collectors


Client
AdminClient
WebServer
DB
Broker
novia/data
Aggregator
Aggregator
Collector
Collector
Collector
Collector
Nominals
AggregatorID Varchar
DeviceID Int
NomTemp Int
NomOxygen Int
CollectorData
AggregatorID Varchar
DeviceID Int
Time DateTime
Oxygen Int
PowerState Int
Temperature Int

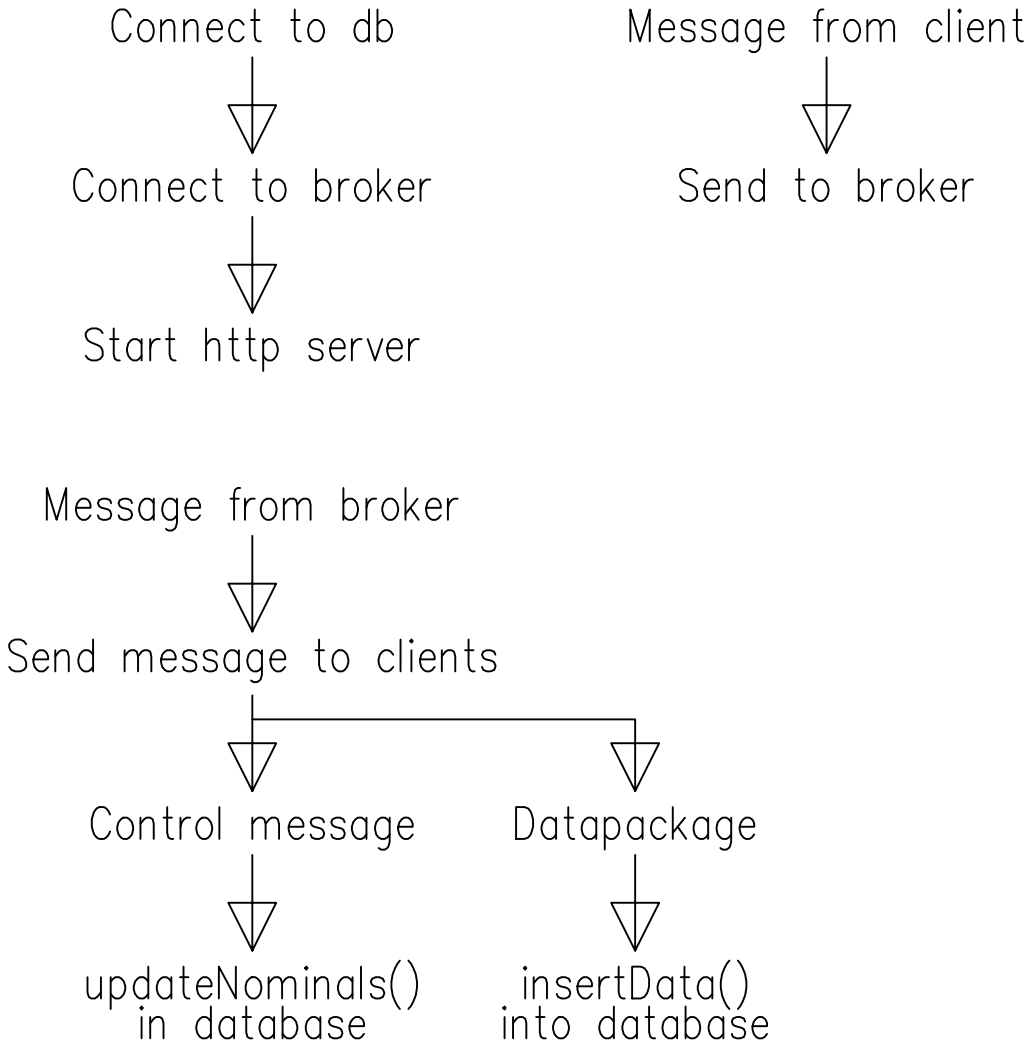

Control room

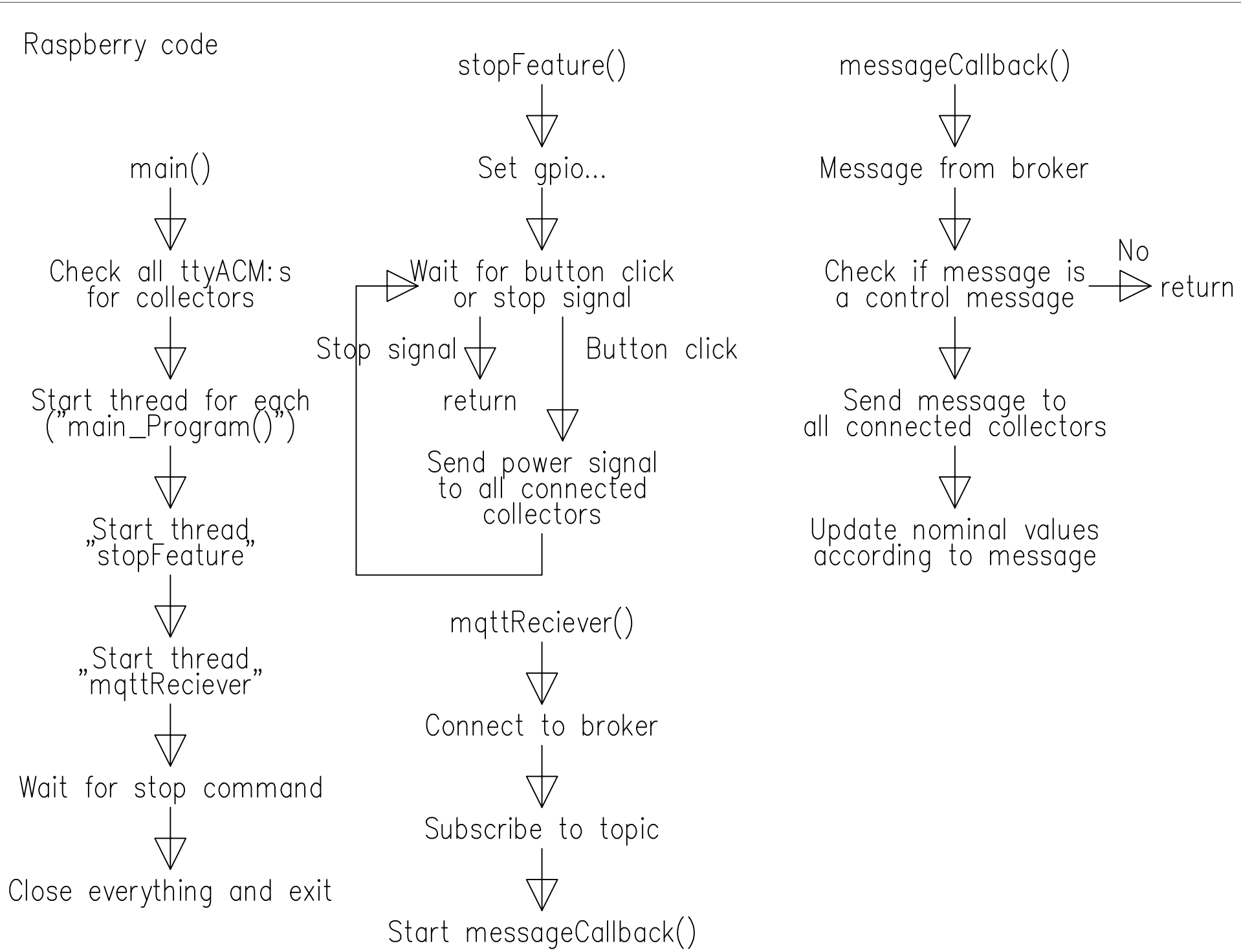

Aggregators
25
26
48
1
2
24

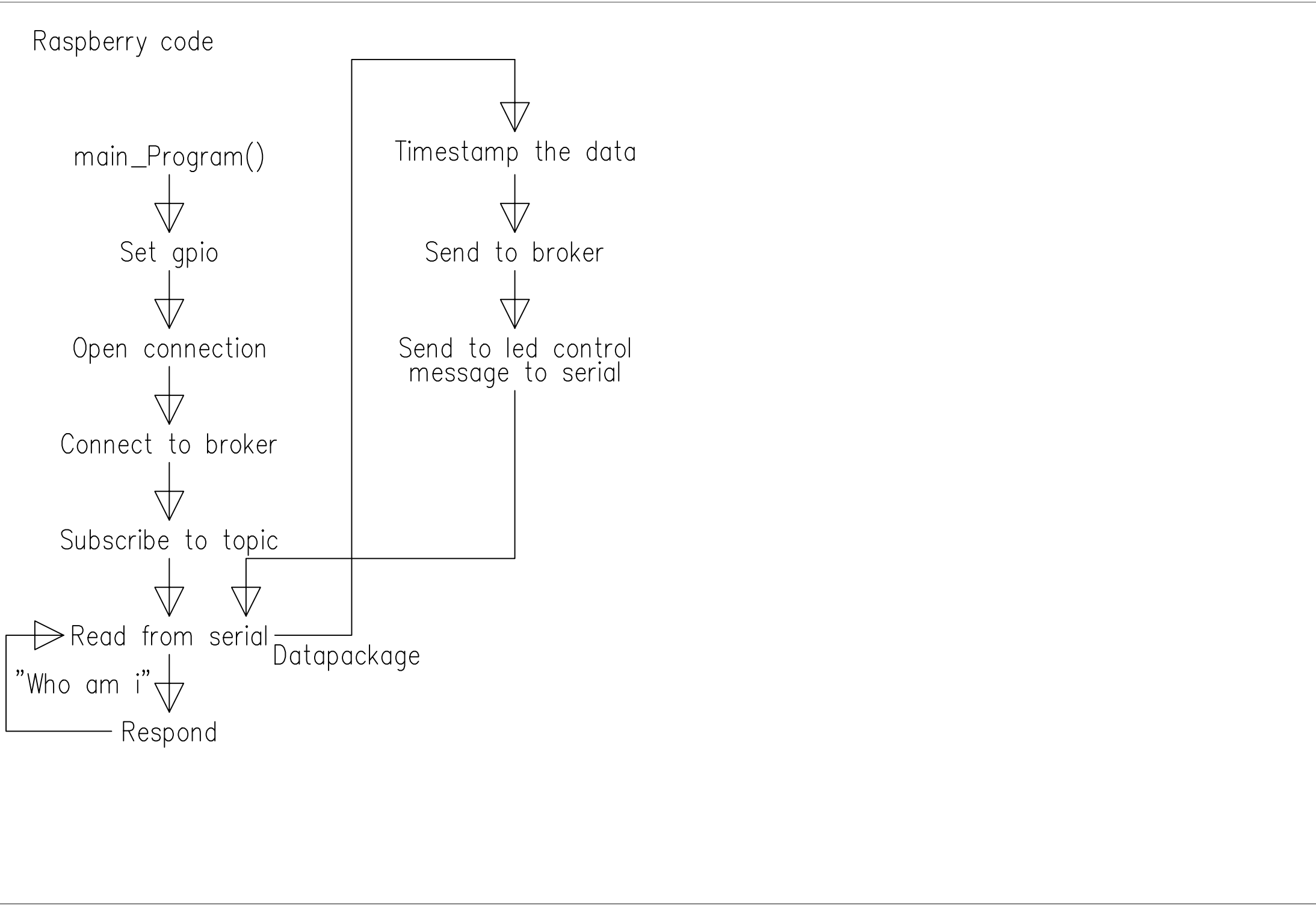

Living quarter 1
Temperature leds
Oxygen leds
Power led
USB
ARDUINO
UNO
DC INPUT
M


Living quarter 24
Temperature leds
Oxygen leds
Power led
USB
ARDUINO
UNO
DC INPUT
M

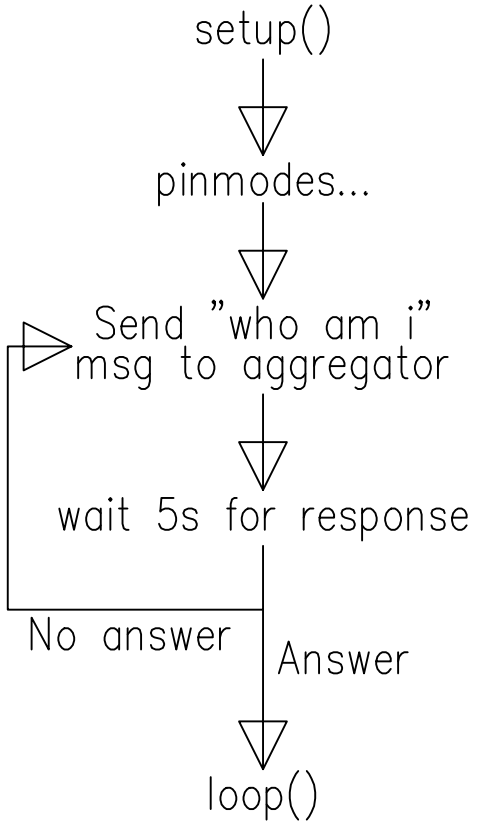
Server code


Connect to db
Connect to broker
Start http server
Message from client
Send to broker
Message from broker
Send message to clients
Control message
Datapackage
updateNominals()
in database
insertData()
into database


Raspberry code
main()
Check all ttyACM:s for collectors
Start thread for each ("main_Program()")
„Start thread„ "stopFeature"
„Start thread„ "mqttReciever"
Wait for stop command
Close everything and exit
stopFeature()
Set gpio...
Wait for button click or stop signal
Stop signal
Button click
return
Send power signal to all connected collectors
mqttReciever()
Connect to broker
Subscribe to topic
Start messageCallback()
messageCallback()
Message from broker
Check if message is a control message
No
return
Send message to all connected collectors
Update nominal values according to message


Raspberry code
main_Program()
Set gpio
Open connection
Connect to broker
Subscribe to topic
Read from serial
"Who am i"
Respond
Datapackage
Timestamp the data
Send to broker
Send to led control
message to serial

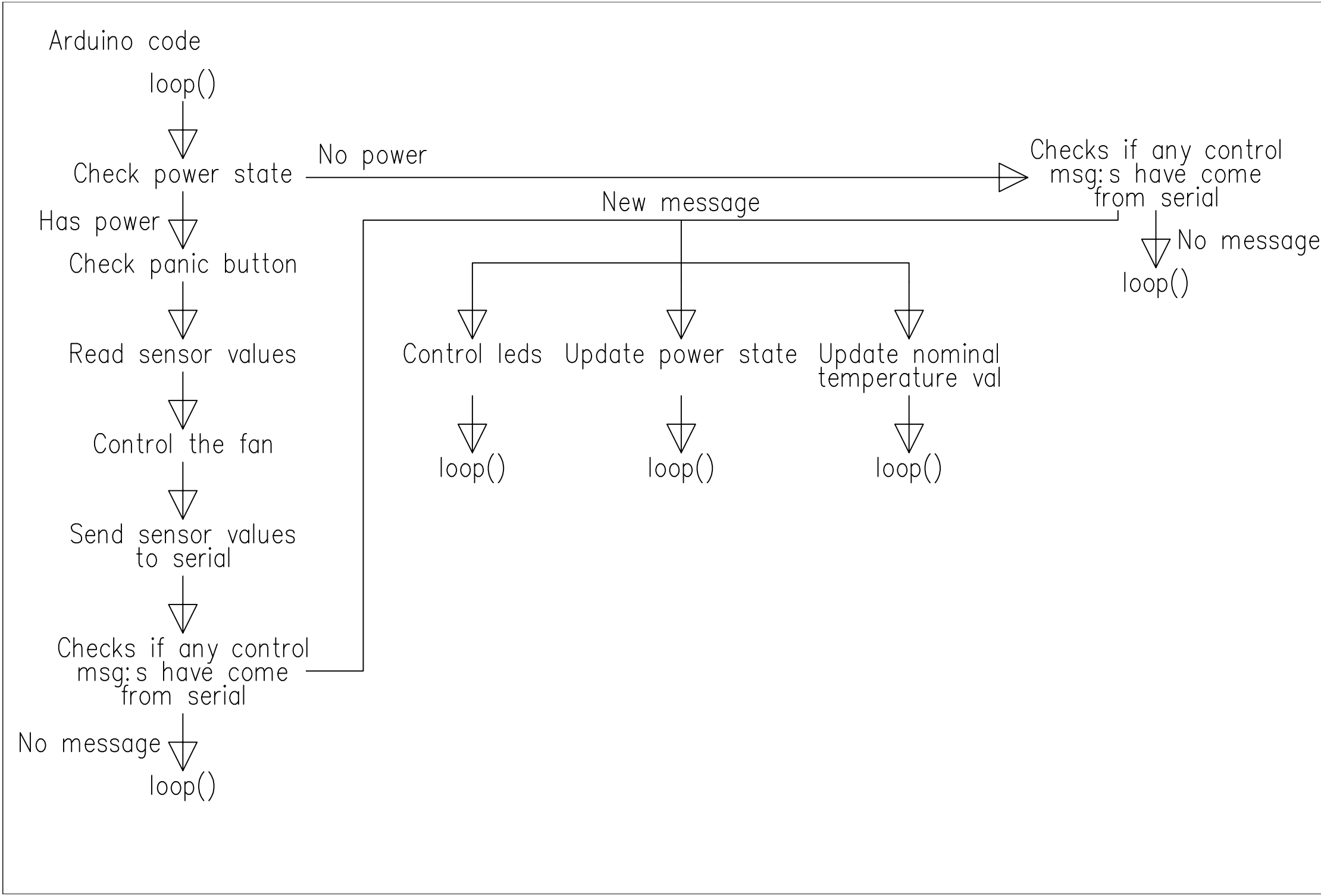
Arduino code



JSON data

- "Who am i" message expects an answer containing:
  - Unique id
  - Nominal temperature
  - Power state*
- "Control message" can contain
  - Power settings*
  - Updated nominal values*
  - Led settings*
- Message from collector contains
  - Unique id
  - Power state
  - Temperature value (0–1023)
  - Oxygen value (0–1023)

*Not necessary


Arduino code
loop()
Check power state
No power
Checks if any control msg:s have come from serial
No message
loop()
Has power
New message
Check panic button
Read sensor values
Control leds
Update power state
Update nominal temperature val
loop()
loop()
loop()
Control the fan
Send sensor values to serial
Checks if any control msg:s have come from serial
No message
loop()