Internet of Cows

09.12.2023.

Students:

Hieu Huynh Thai

Ahnaf Tahmid Ul Huq

Aleksa Stanivuk

Mentor:

Prof Davide Di Ruscio

Table of Contents

Introduction

Internet of Cows is a complete SaaS solution that is enabling farm owners to manage their cattle more easily by having a better insight into the cattle health status. It enables tracking of the cow data while significantly decreasing necessary manual work and the associated costs.

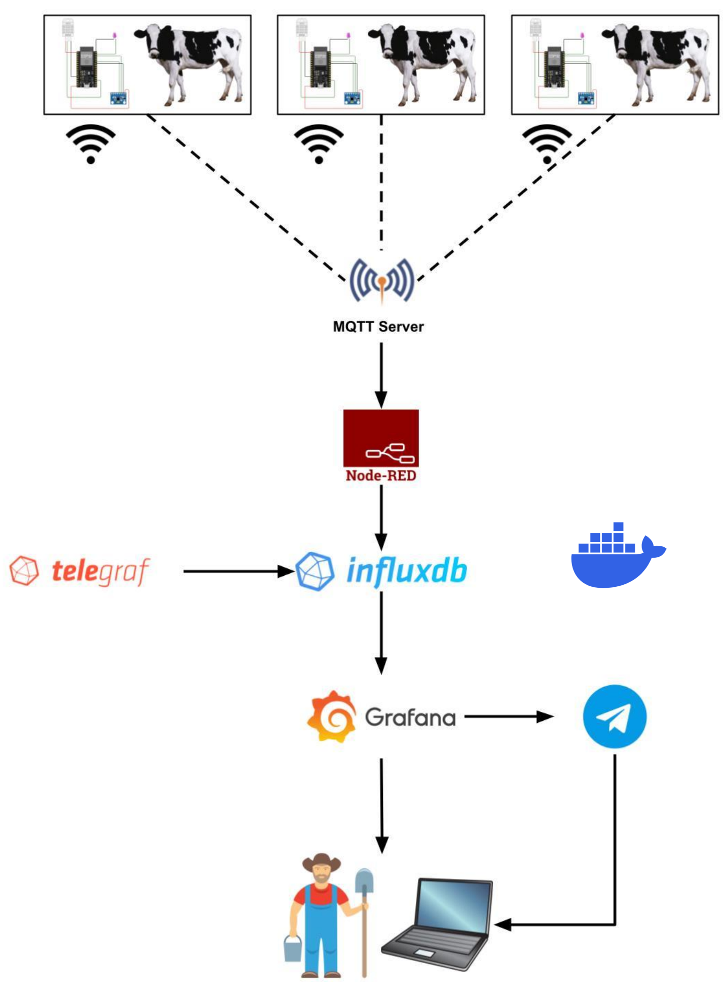
The data are collected by the customly designed collar attached to the animal's neck and transferred over the network to be stored in the database and visualized on the farmer's dashboard. Information collected about the cattle are their body temperature, pulse (heart beats) and the speed of their movement.

The system is capable of real-time detection of alarming states and immediately informs the farmer through a specialized Telegram channel.

System Architecture

The implementation of the system is based on combining and setting up existing solutions specialized for solving certain parts of the whole problem, such as Node-RED, Telegraf, InfluxDb and Grafana. Besides, the system also introduces the customized simulated cow device consisting of ESP32 microcontroller and associated sensors. All the services, except the simulated device, as dockerized.

Starting point of the system are sensors inside the custom device which gather information about the cow's status. The device sends the collected data through Wifi to an MQTT server where they are accessed by Node-RED and forwarded to the InfluxDb to be stored and to Grafana to be visualized. Grafana forwards information about alarming states through the Telegram channel. Telegraf is used for configuring the InfluxDb.



Cowlar - Device and sensors

We opted for designing a device that would be directly attached to an individual animal - cow. Focusing on the already existing devices - this device should be fixed to the cow's neck in the form of a collar. Therefore, the name of our product would be Cowlar (cow + collar).

For the purpose of designing the device, which consists of a microcontroller board and attached sensors, we used Wokwi platform. Wokwi is a platform that enables simulation of Arduino, ESP32 and STM32 microcontrollers. Simulation is done through an iterative process of adding and visually connecting different sensors to the chosen motherboard, and configuring the functionalities using C++ through an IDE available on the other half of the screen.

Device

For the implementation of our device, we chose to use an ESP32 microcontroller instead of an Arduino because of its significant advantage of already built-in Bluetooth and Wifi capabilities. In addition, ESP32 have low-power modes, which would be very beneficial since Cowlar would be powered by batteries — it's important to preserve as much energy as possible, so to further ease the necessary manual work.

We connected three items to our microcontroller:

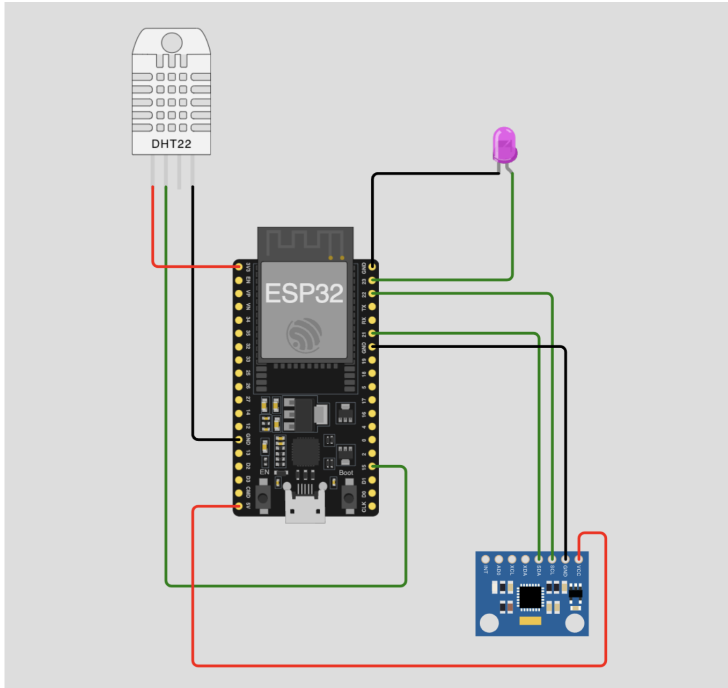
1. Temperature sensor,
2. Accelerometer,
3. LED

For the temperature sensor we chose DHT22 which is in essence capable of tracking information about humidity as well, however we were using only the data about temperature.

For the purpose of tracking an animal's speed, it was not possible to have the sensor that we really needed, because of the platform's capabilities. Therefore we used the MPU6050 accelerometer and gyrometer, which is capable of tracking speed in all three dimensions and even the angle of rotation. Only the speed on X margin was used in our implementation.

The platform does not contain any sensor capable of tracking or simulating pulse. However, we found a library that does this through code - <HeartBeat.h>. That being the case, as a visual representation of the ongoing process, a light emitting diode was used to give visual information about the pulse, but also as a space-taker for the real heartbeat sensor in future.

The design of Cowlar's microcontroller and sensors can be seen below:



Communication with the system

Most of the current solutions on the market require manual collection of the gathered data. Meaning that a physical worker would have to check each cow's collar, connect a device to it and download the data.

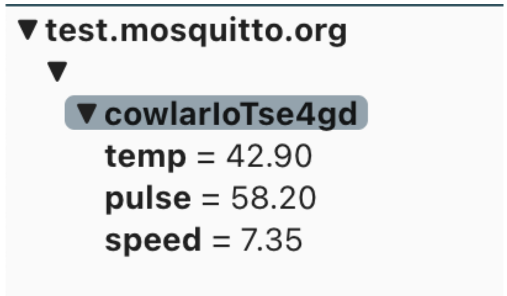
In our implementation, for the previously mentioned ESP32's in-built Wifi capabilities this problem is solved by connecting the device to the internet and therefore having real-time information about each individual animal. For the simulation environment, Wokwi enables connecting to a simulated network called *Wokwi-GUEST*, however for real-life purposes, it is possible to code in the name and password of the existing Wifi network on the farm.

```
const char* ssid = "Wokwi-GUEST";
const char* password = "";
```

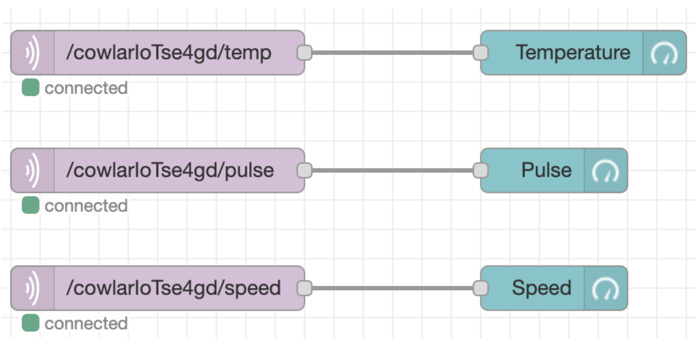
```
Serial.print("Connecting to WiFi: ");
Serial.println(ssid);

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
  delay(500); // try to reconnect in 0.5s
  Serial.print(".");
}
```

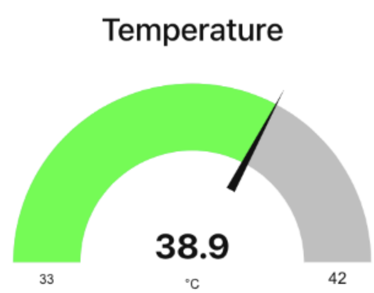
Upon being successfully connected to the internet, the device is set to try to connect to an MQTT server. For this purpose the official and publicly available [MQTT test server] was used. In a real production environment, the farm owner (or the SaaS provider) would have to invest in acquiring their private and well protected server. By connecting the device to an MQTT server it is possible for it to not only publish the data to topics, but also to subscribe to certain.

In Cowlar's implementation, the microcontroller connected to following topics, visible through the MQTT Explorer's interface:



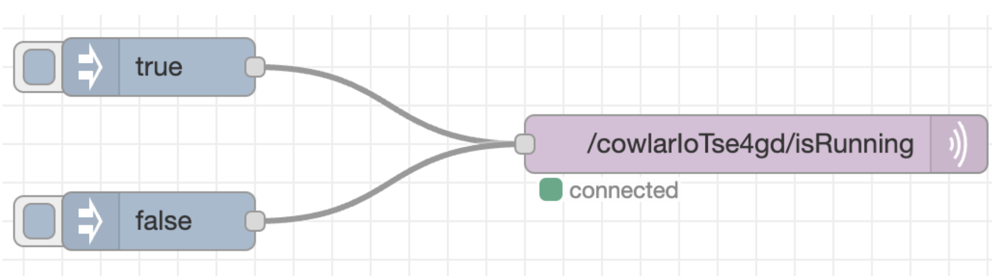
This way it was also possible to automate the process by accessing the gathered data through the Node-RED platform. The following images present the isolated Node-RED's use for the communication with the device.





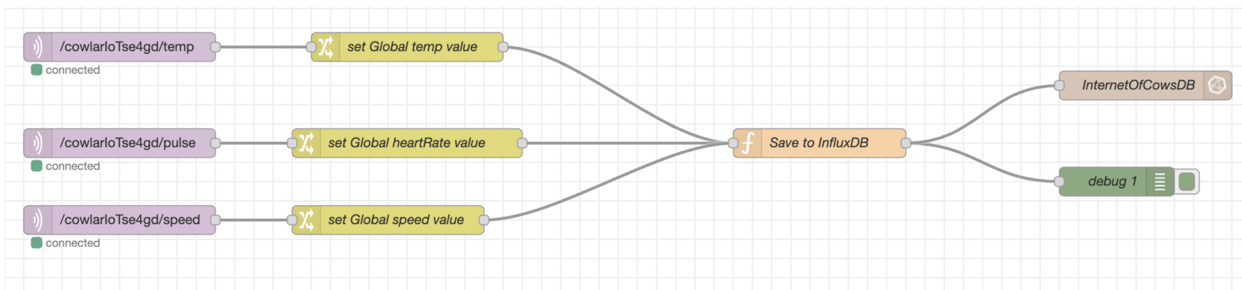
As mentioned, it is possible to subscribe the device to a topic, so that it could get feedback information. Upon receiving a message from the subscribed topic, a callback function is called. This possibility we utilized for the sake of simulating the cow's will. Even though not a real-life example, it still proves the concept and enables a better showcase of the completed work.

The device subscribed to the */cowlarIoTse4gd/isRunning* topic and through Node-RED we can choose if the value is *True* or *False*. This turns On or Off the *Cow-Is-Running* mode which affects the pulse rate in return.

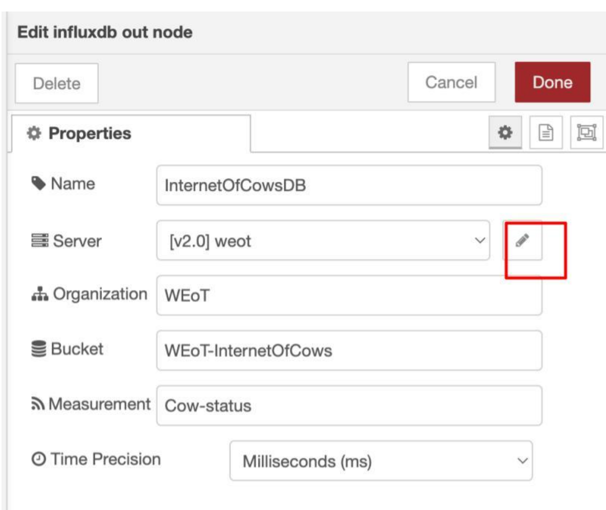


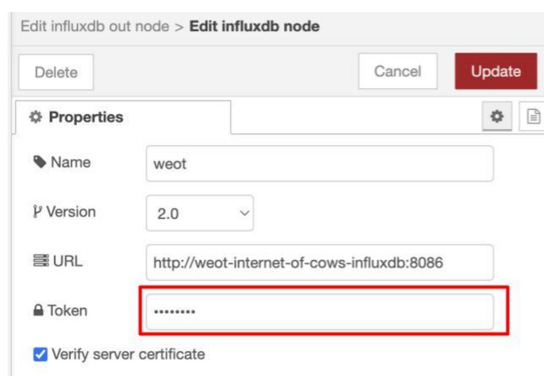
Data storage - InfluxDB

For real-time storing of the gathered data our system is using InfluxDB. As seen in the Node-RED flow, data is coming from three different topics and is being congested into a single data instance when being saved in the InfluxDB.



InfluxDb configuration comes included in the docker package, just like the rest of the configurations. However, every time the image gets rebuilt, the token that connects Node-Red to the InfluxDb gets removed. To solve that, users need to add the token manually into the InfluxDb node. The steps are show below:





Telegraf is used for setting up the InfluxDb with the parameters such as port, credentials, organization, token and data buckets.

Data visualization - Grafana Dashboard

Data stored in InfluxDb is being forwarded to Grafana for the purposes of user-friendly visualization on the Dashboard. Our solution shows to the user the parameters collected from the cattle's coller - Body Temperature, Movement Speed and Heart Rate.

Speed is presented with the Gauge, similar to the ones used in presenting car speed and is put in the upper right corner.

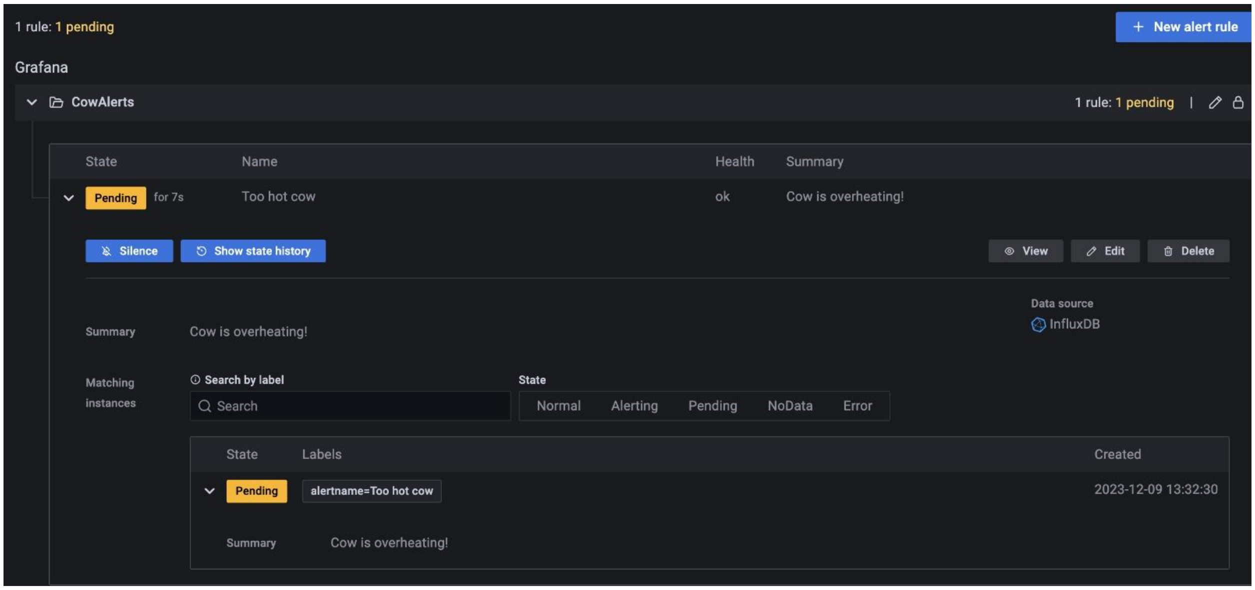
The biggest part, half of the dashboard, is taken by the Heart Rate visualizer, which is the most important for observing the cow's health status.

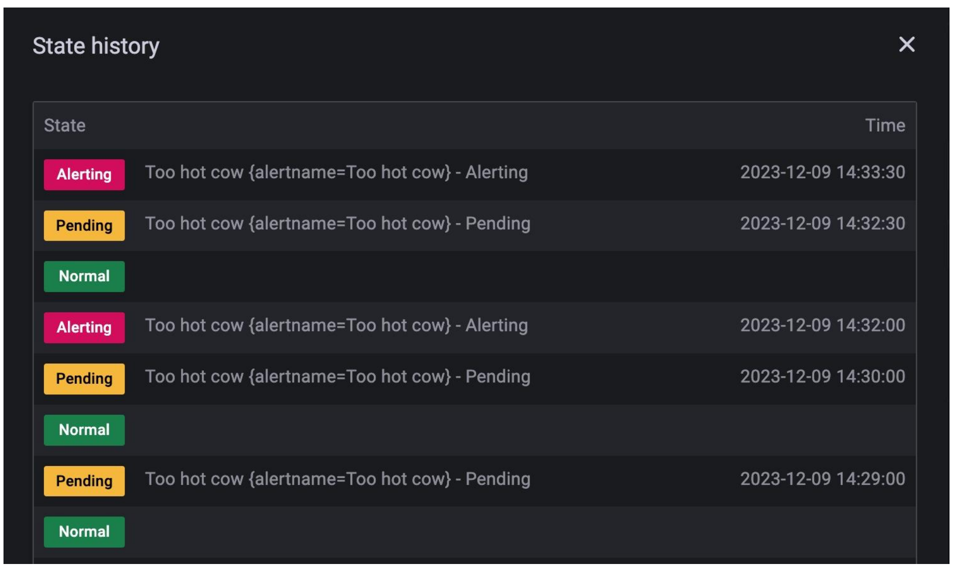
Alarming

Two Alarm thresholds were implemented:

1. TooHotCow - activates if cow's temperature reaches over 41°C
2. DeadCow - activates if cow's temperature is below 35°C and heart rate is below 40

Images below show how the dashboard looks when showing the alarm states - Normal, Pending and Alerting. Pending state is if the value goes above (or below) the threshold, but waits some more time before checking again, and eventually turning into the Alerting state.

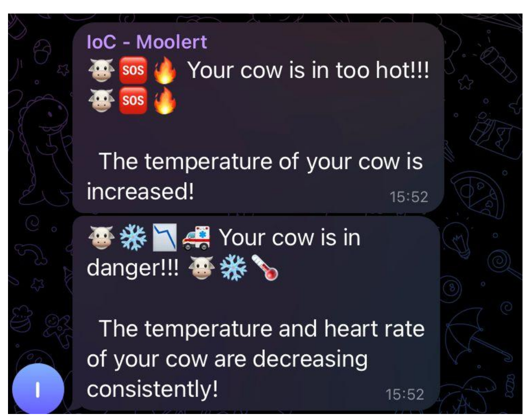




Telegram bot

In case of any of the two alarms being activated, the system sends a message through Telegram bot to the specialized group created for the farm owners.

Messages looks as following:



Docker and setting up

As mentioned, all the components, except the simulated device on the Wokwi platform, are dockerized. This enables portability and easy deployment on other machines. Build docker container looks as following:

weot-internet-of-cows		Running (4/4)	1.2%	
telegraf 5d6ed0788cb2	telegraf:1.1	Running	0.1%	
nodered-1 aee80c20d945	weot-intern	Running	0%	1880:1880
influxdb e0198124d973	influxdb:2.1	Running	0.96%	8086:8086
grafana 0a56f00b5589	grafana/gra	Running	0.14%	3000:3000

Credentials

InfluxDb: - username: weot

- password: internetofcows

Grafana: - username: admin

- password: admin

InfluxDb token for Node-RED:

tfg7A3pr7Y2obvYNsSikeYXtHJk9R3z_yvSRm6os_T_6WaXziGDRiEPY4vk9xBy6eA7ErRX6hjRM49Y8rAjifQ==

Steps for setting up the system

1. Clone the repository: https://github.com/hieuhuynh1752/WEoT-Internet-Of-Cows/tree/main
2. docker compose build
3. docker compose up
4. Open Node-RED on http://localhost:1880
5. Open InfluxDb on http://localhost:8086
    a. Login with the credentials

6. Open Grafana on http://localhost:3000
    a. Login with the credentials
7. Open Wokwi project on https://wokwi.com/projects/383544447455853569
    a. Start the device simulation by pressing the play button ▶️
    b. The device will be simulating only if it's on the active screen (if you have another tab open it will go idle, and continue when you are back at it, you can counter this by splitting your browsers screen and looking at the multiple tabs at the same time)
    c. Wait for the device to connect to Wifi and MQTT server
    d. You can adjust the sensor values by clicking on each of the sensors
        i. Remember that the LED is only for showing the beats (you can influence beats through Node-RED's True/False flow)
8. Watch changes happening in your InfluxDb and Grafana Dashboards
9. Play with True/False flow on Node-RED to make cow start or stop running
    a. **NOTE**: This is *simulation of cow's will,* so the idea is that the cow *decided* to start doing a demanding activity which will increase its heartbeats in return. So don't expect to get an increase in speed measurements, but only in heartbeats/pulse. Speed is collected through sensor, so the values are not simulated but the idea is that they would be collected from the reality.