Groundwater Gauge

Introduction

With the mining history of NRW and the impact it had on the soil, groundwater has always been a point of interest. Combined with the increased precipitation as seen in Fig.1. it becomes more and more a priority to have a way to acquire data regarding the groundwater level in the region remotely. A possible method for a steadier stream of incoming data, is using a device to measure the water pressure in groundwater sample wells that is connected to a microcontroller with LoRa functionality. The choice for LoRa is based on the fact that LoRa data transfer is considered to be energy efficient and can be used for long range data transmissions in remote areas.

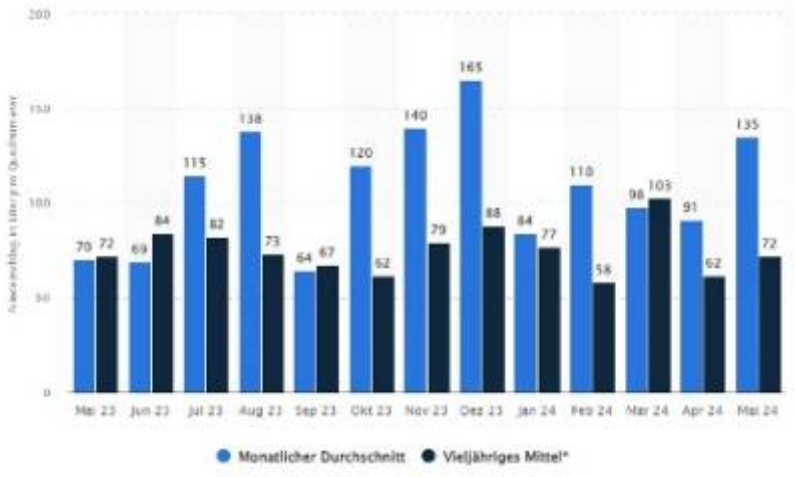


Fig.1, Average Precipitation 2023-2024, source: https://de.statista.com/statistik/daten/studie/576867/umfrage/durchschnittlicher-niederschlag-pro-monat-in-nordrhein-westfalen/

These sample wells are not easily accessed and will not have an easy access to the electrical grid. Therefore, the measuring device should have its own power supply. The idea of this system is to make sure data can come in continuously without much physical labour from employees of the organisation that monitors the groundwater level (in the region of Kamp-Lintfort this would be LINEG). Measuring 24 hours a day would mean an immense capacity requirement for the batteries and would not make it a sustainable way of monitoring. Because the impact of precipitation on the groundwater level in the targeted region of Kamp-Lintfort, is not visible in the order of minutes but closer to the order of hours an external clock is added to the system to ensure an on and off function to save the battery. With the choice of our MC we also have the option to have the connected battery charged by a photovoltaic panel. The MC has a built-in charge regulator and can check the battery status to see if charging is necessary. This would mean that our system can stay for longer periods of time in the field without any human contact.

2. Materials & Methods

In this section each component of the system will be highlighted, and their use explained.

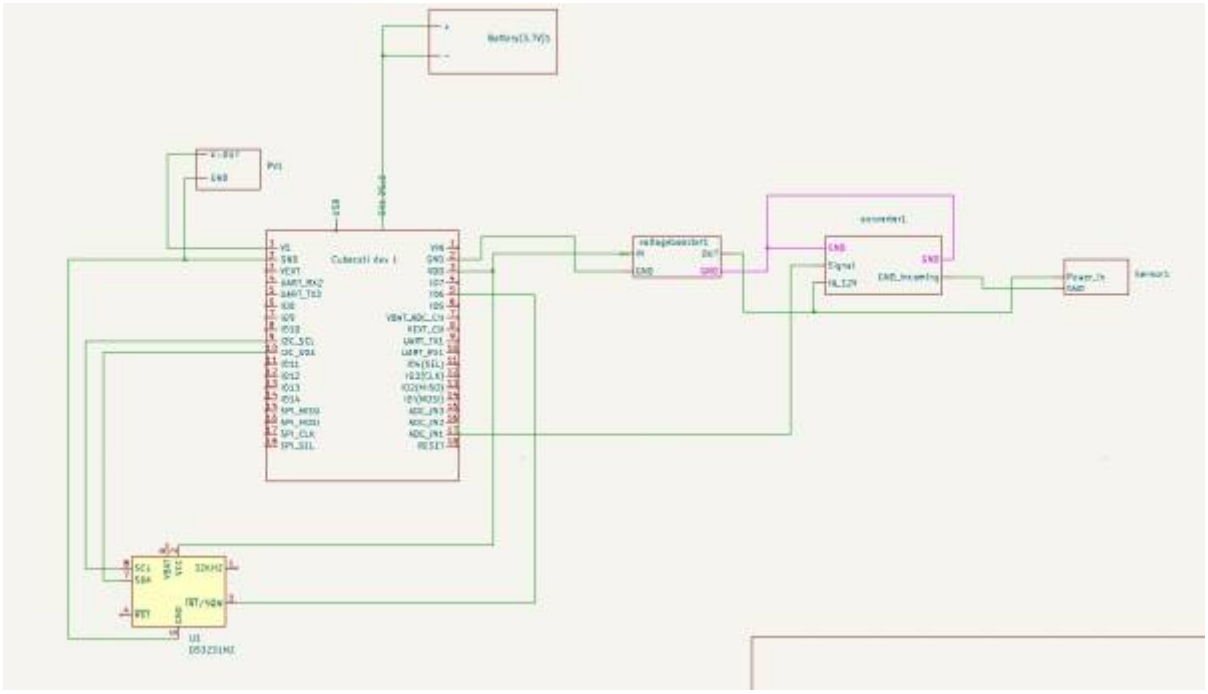


Fig. 2, Schematic of our Set Up by De Jong

As seen in Fig.2, following pins were used for the entire setup:

- Lithium Rechargeable Battery → MC(SH1.25-2)

Purpose: Supply power to the microcontroller and other components in the system.

Reason: The lithium rechargeable battery provides the main power source for the entire system. It ensures that the microcontroller and connected peripherals have a continuous supply of power, even in the absence of sunlight for the photovoltaic panel.

- MC(VS)solar input PIN(5.5V ~ 7V) → (V-OUT)PV

Purpose: The VS pin on the MC Heltec CubeCell dev-6502 receives power from the photovoltaic (PV) panel.

Reason: The PV panel generates electrical power from sunlight, which can be used to power the microcontroller or charge the battery directly when sufficient sunlight is available.

- MC(GND) via Wire Connector → PV(GND) & RTC(GND)

Purpose: Establish a common ground reference for the entire system.

Reason: A common ground is necessary for proper operation of all components to ensure consistent voltage levels and to prevent floating ground issues that could cause erratic behavior.

- MC(SCL) → (SCL)RTC

Purpose: Connect the Serial Clock Line (SCL) for I2C communication.

Reason: The SCL line is used to synchronize the data transfer between the microcontroller and the RTC. It is essential for timing and coordination of data exchanges on the I2C bus.

- MC(SDA) → (SDA)RTC

Purpose: Connect the Serial Data Line (SDA) for I2C communication.

Reason: The SDA line is used for the actual data transfer between the microcontroller and the RTC. It carries the data being sent to and from the RTC over the I2C bus.

- MC(GND) → (GND)Voltage step-up regulator

Purpose: Establish a common ground reference with the voltage step-up regulator.

Reason: To ensure that the voltage levels output by the voltage step-up regulator are correctly referenced to the same ground as the microcontroller, providing stable power to the system.

- MC(VDD) via Wire Connector → (IN)Voltage step-up regulator & (VCC)RTC

Purpose: Supply power to the voltage step-up regulator and RTC.

Reason: The VDD pin provides the necessary voltage to power both the voltage step-up regulator, which in turn powers other components, and the RTC to keep it operational.

- MC(GPI06) → (INT/SQW)RTC

Purpose: Connect the interrupt/square wave output of the RTC to a general-purpose input/output (GPIO) pin on the microcontroller.

Reason: This connection allows the RTC to send interrupts or a square wave signal to the microcontroller, which can be used for timekeeping and wake-up functions.

- MC(ADC1) → (SIGNAL)Current to Voltage converter

Purpose: Read the voltage signal from the current-to-voltage converter.

Reason: The ADC1 pin on the microcontroller is used to measure the analog voltage output from the current-to-voltage converter, which represents the sensor's 4-20mA current signal.

- Both Current to Voltage converter(GND) via wire connector → (GND)Voltage step-up regulator

Purpose: Establish a common ground reference for the current-to-voltage converter and the voltage step-up regulator.

Reason: Ensures that the voltage levels for the converter are referenced correctly to the same ground, providing stable and accurate conversion of current signals.

- Current to Voltage converter(IN_12V) via cable connector → (OUT)Voltage step-up regulator & (Power_In)Sensor

Purpose: Provide power to the current-to-voltage converter and the hydrostatic sensor.

Reason: The output from the voltage step-up regulator supplies the necessary voltage (typically stepped up to 12V) to both the converter and the sensor, ensuring they operate within their required voltage ranges.

- Current to Voltage converter(GND_Incoming) → (GND)Sensor

Purpose: Establish a common ground between the current-to-voltage converter and the sensor.

Reason: Ensures that the ground reference for the sensor and the converter are the same, providing consistent and accurate signal conversion.

2.1 Microcontroller (MC)

We used the microcontroller from Heltec CubeCell dev-6502 (as seen in Fig.3). This MC has a LoRa data transfer option, known for its low power usage in sleep mode and for this specific project a direct photovoltaic connection, build in battery charge option and battery status check option. It includes a USB interface for easy programming, multiple GPIOs for sensor connections, and compatibility with the Arduino development environment. This MC can be found following this link: https://www.bastelgarage.ch/cubecell-dev-board-plus-868mhz-lora-node-htcc-ab02

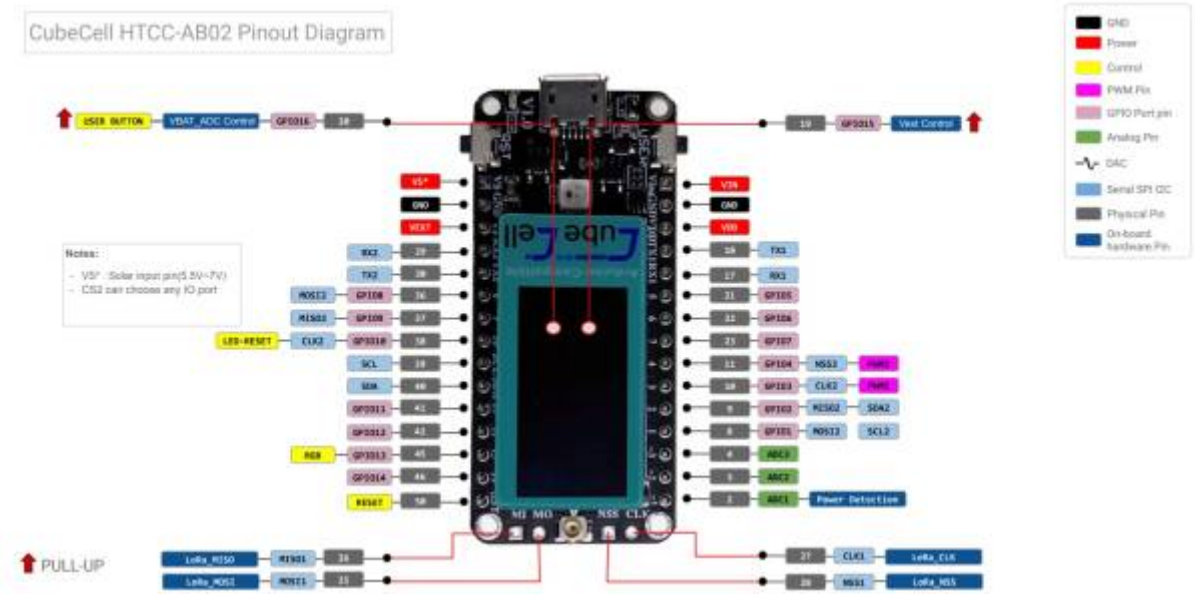


Fig. 3, Heltec Microcontroller, Source: https://www.bastelgarage.ch/cubecell-dev-board-plus-868mhz-lora-node-htcc-ab02

2.2 Current to Voltage converter.

For the MC to be able to use the output of the sensor, the sensor output signal needs ro be converted from current which ranges between 4 and 20 mA to a Voltage-range readable by the microcontroller. This process will be covered by a current to voltage converter (as seen in Fig.4). The converter in this system is the XY-IT0V (no specified brand). This converter can be adjusted for different voltage ranges by the means of jumpers. 2 jumpers are present on the converter and with the right combination for “on” and “off” of these jumpers the required voltage output can be set. On this webpage more information concerning the convertor and how to calibrate can be found on the following webpage https://www.robotics.org.za/XY-ITOV.

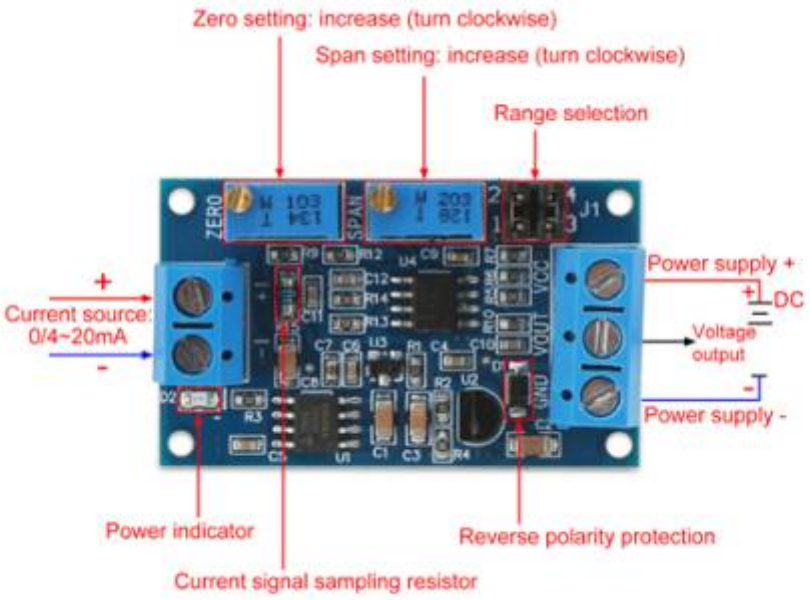


Fig.4, Current to Voltage Converter, Source: https://www.robotics.org.za/XY-ITOV

2.3 Sensor

Hydrostatic sensor 4-20mA 15m cable, CJ-YBT liquid level transmitter. The Hydrostatic sensor CJ-YBT liquid level transmitter, operates by measuring the hydrostatic pressure exerted by the liquid above the sensor. This pressure is directly proportional to the liquid level. The sensor converts this pressure into a 4-20mA current signal, which is a standard output for industrial sensors, ensuring reliable transmission over long distances.

Key Features: High Accuracy: Provides precise level measurement. Robust Construction: Designed for immersion in various liquids, including water and chemicals. Long Cable: The 15-meter cable allows for measurement in deep tanks or wells. 4-20mA Output: Standard industrial signal output, ensuring compatibility with most monitoring and control systems. Applications: Ideal for water treatment plants, reservoirs, wells, and other liquid storage applications.

Working Principle: Hydrostatic Pressure Measurement: The sensor detects the pressure exerted by the liquid column. Signal Conversion: The detected pressure is converted into an electrical signal (4-20mA), representing the liquid level. Data Transmission: The 4-20mA signal is transmitted to monitoring equipment, allowing for real-time level monitoring and control. These sensors are essential for accurate liquid level monitoring in various industrial and environmental applications, providing reliable data for efficient process management.

2.4 Battery

For this project a 3.7V Lithium rechargeable battery is used. Which is connected to the MC via the SH1.25-2 connection. The MC can regulate the charging via the photovoltaic panel and prevents overcharging.

2.5 Photovoltaic panel

For this project an 80x100mm panel, with an output of 2W is used.

2.6 Voltage step-up regulator

In order to get enough voltage for the sensor (12V was enough) a voltage regulator (as seen in Fig.5) was used that converts 3.3V that our MC can provide into 12V. In this system U3V16F12 from Polulu was used (https://www.pololu.com/product/4945).

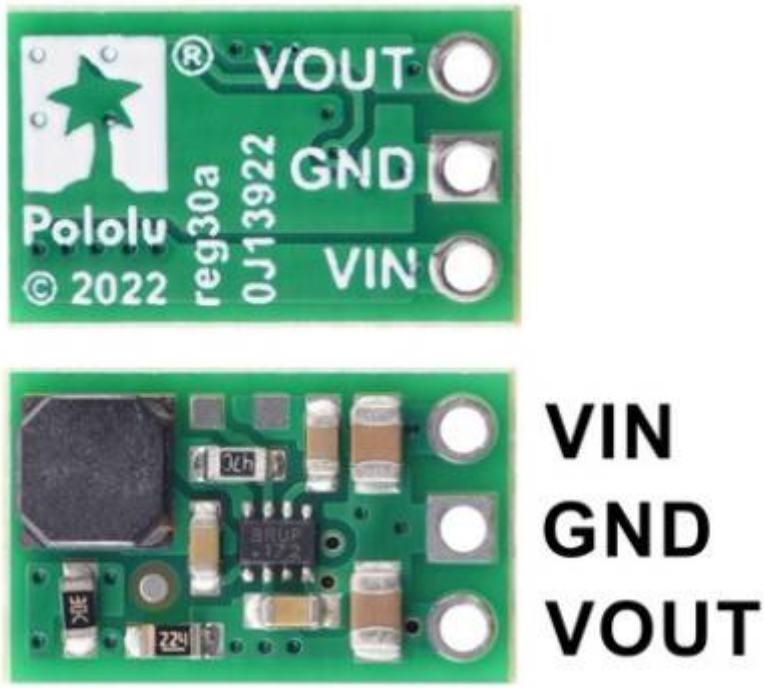


Fig.5, Voltage Regulator, Source: https://www.pololu.com/product/4945

2.7 Real-Time Clock (RTC)

For the MC to be as efficient as possible (by use of sleep mode) a RTC ds3231 I2C (as seen in Fig.6) is connected to the the MC. The RTC will keep an accurate time so the MC can be in sleep mode and wake up at the appropriate time to preserve power.

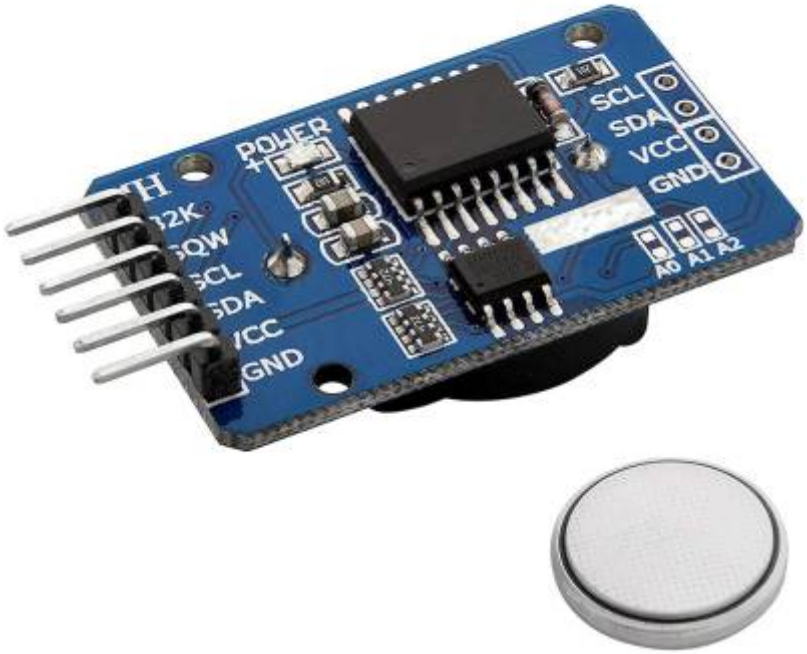


Fig.6, RTC, Source: https://www.amazon.de/AZDelivery-RTC-Batterie-inklusive-Arduino/dp/B01M2B7HQB?th=1

2.8 LoRaWAN & TTN

LoRaWAN, which stands for Long Range Wide Area Network, is a communication protocol specifically designed for low-power, long-distance wireless data transmission. It is ideal for Internet of Things (IoT) applications where devices need to transmit small data packets over extended ranges while maintaining energy efficiency. Operating in sub-gigahertz frequency bands, LoRaWAN can achieve coverage of several kilometers in urban environments and even greater distances in rural areas. For visualization as seen in Fig.7, LoRa modulation offers a much longer communication range at low bandwidths compared to other competing wireless data transmission technologies.

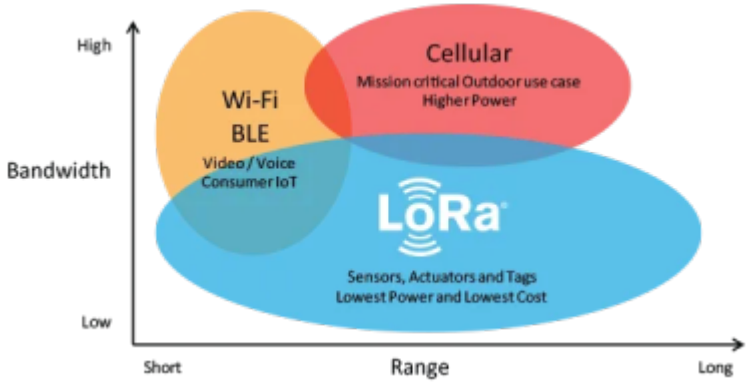
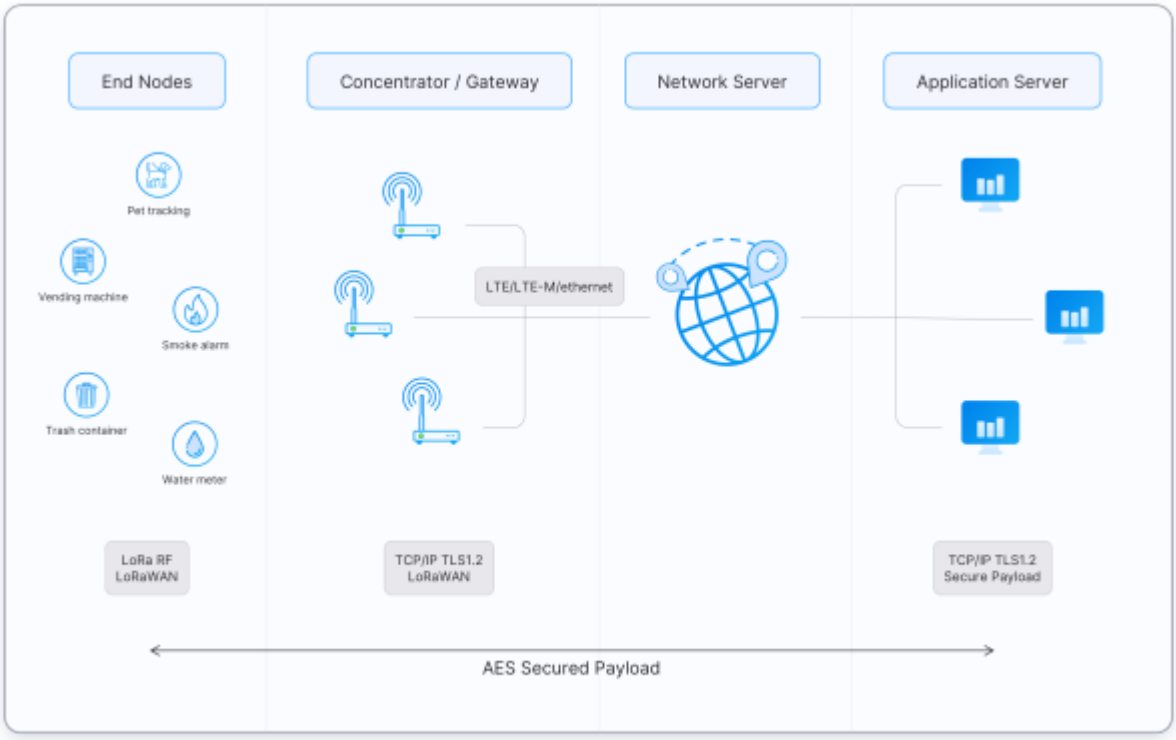


Fig.7, LoRa comparisson with other wireless data transmission technologies, Source:

https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

The core advantage of LoRaWAN lies in its capability to provide long-range connectivity with minimal energy consumption. This is accomplished through chirp spread spectrum modulation, which enhances resistance to interference and extends communication ranges. The network structure of LoRaWAN is typically organized in a star topology, where end devices communicate directly with gateways. These gateways then forward the data to a central network server through a backhaul connection, which could be cellular, Ethernet, or other types of connectivity. In Fig. 8 the elements of a typical LoRaWAN network are visualized.

A typical LoRaWAN network consists of the following elements.



}}

Fig.8, Elements in a typical LoRaWAN network, Source: https://www.thethingsnetwork.org/docs/lorawan/architecture/

LoRaWAN's scalability is a major benefit, allowing a vast number of devices to be supported within a single network due to its low power and extended range features. This makes it particularly effective for applications in smart cities, agriculture, industrial monitoring, and environmental sensing.

The Things Network (TTN) is a global, open-source network that operates on the LoRaWAN protocol. Its goal is to provide a free, decentralized network that is accessible to developers and organizations worldwide. TTN relies on a collaborative approach where individuals and organizations contribute by setting up gateways that connect to the network's infrastructure. This decentralized model supports the deployment of extensive IoT solutions without the need for centralized management or proprietary systems.

TTN offers a public network that users can access without charge, while also providing options for private and enterprise networks. It supports a broad array of devices and applications and provides various tools and resources to aid in the deployment and management of IoT solutions. By using TTN,

developers can focus on creating and implementing their applications without the need for significant investment in network infrastructure.

In essence, LoRaWAN is an effective technology for long-range, low-power communication suited for numerous IoT applications, while The Things Network (TTN) provides a global, open-access framework that facilitates the use of LoRaWAN technology for diverse projects and innovations. More Informations about TTN and LoRaWAN can be found in their official homepage: https://www.thethingsnetwork.org/docs/lorawan/

2.9 THE NIG SETUP

NODE-RED

Node-RED is a flow-based development tool for visual programming that facilitates the connection of hardware devices, application programming interfaces (APIs) and online services, especially in IoT applications.[7] .

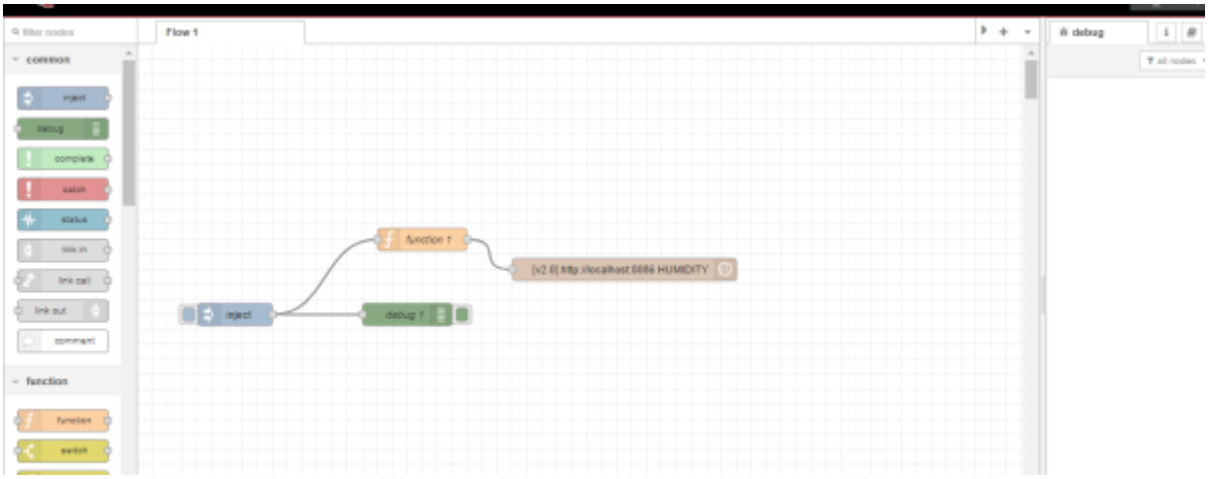


FIG 9: Node-red flow by Henrydon.

Node-red is use to put together data processing logic and send the processed data to advanced systems, like a central data collector or cloud-based service, within minutes or show it right away. Node-red includes nodes that offer functions such as MQTT broker, debug, InfluxDB out ,etc. and the Created processes are stored using JSON objects as shown figure 9 above.

INFLUXDB

InfluxDB is an open source time-series database optimised for real-time data management and analysis[8]. The system is designed to facilitate high-performance writing and reading at scale and is particularly suitable for applications such as the Internet of Things (IoT), DevOps and real-time analytics. InfluxDB's time-based storage policies, persistent queries and flexible data exploration capabilities make it a versatile tool for a variety of time-series-based workloads. While InfluxDB focuses primarily on time series data management, it can also handle general-purpose data modelling, integrating buckets and organisations.

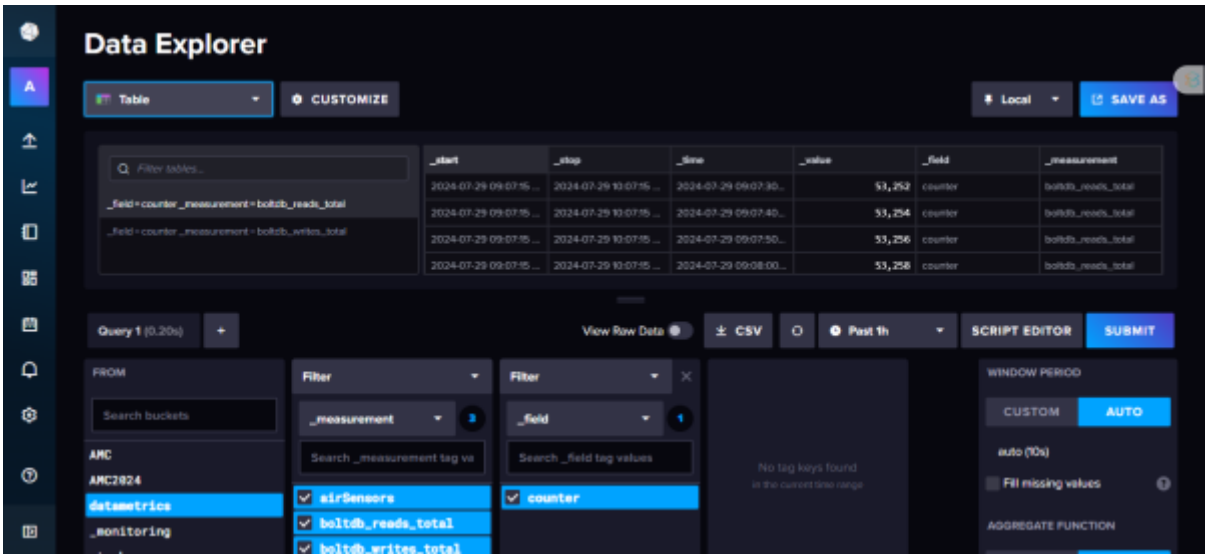


Fig. 10, influxDB display by henrydon.

The underlying data model is based on measurements, tags, fields and timestamps as shown in figure 10 above, providing a structured approach to storing and querying time series data.

GRAFANA

Grafana is an open source monitoring and observation platform, it provides an easy-to-use interface for visualising complex metrics and log data from a variety of data sources, including InfluxDB, Prometheus and Graphite[9]. In addition to visualisations, Grafana also supports alerts, annotations and dashboards, allowing users to quickly respond to system anomalies and gain actionable insights. Extending the platform through plug-ins enables customisation and integration with additional data sources and visualisation tools.

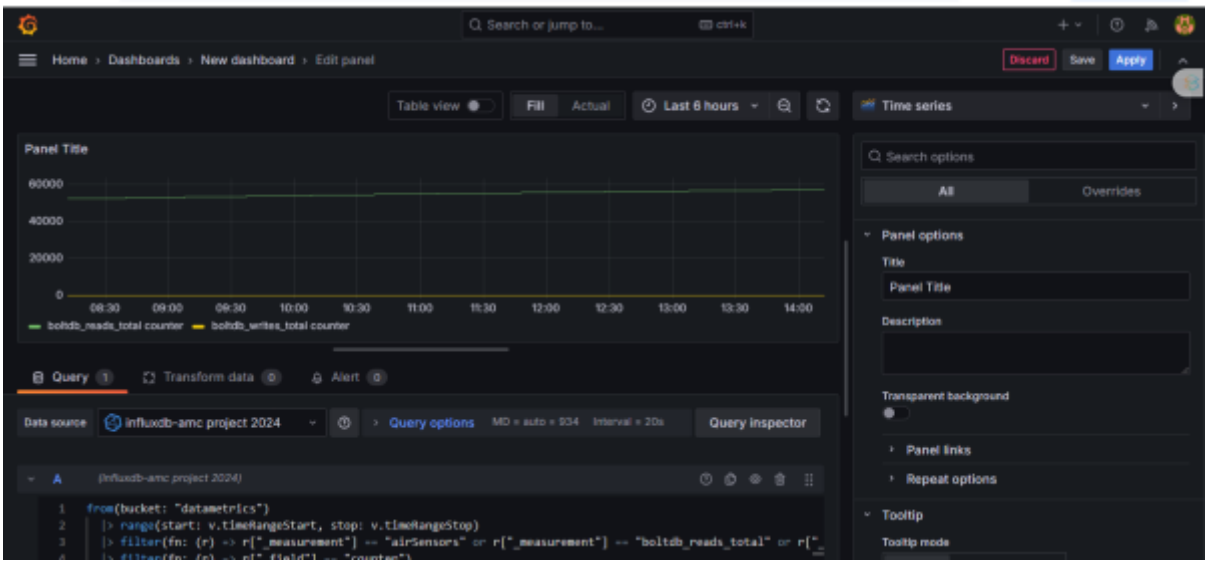


FIG. 11, Grafana dashboard by Henrydon.

Grafana provides a variety of visualisation options, including graphs, tables, heatmaps and histograms as shown in the figure 11 above, enabling users to effectively explore and understand

their data.

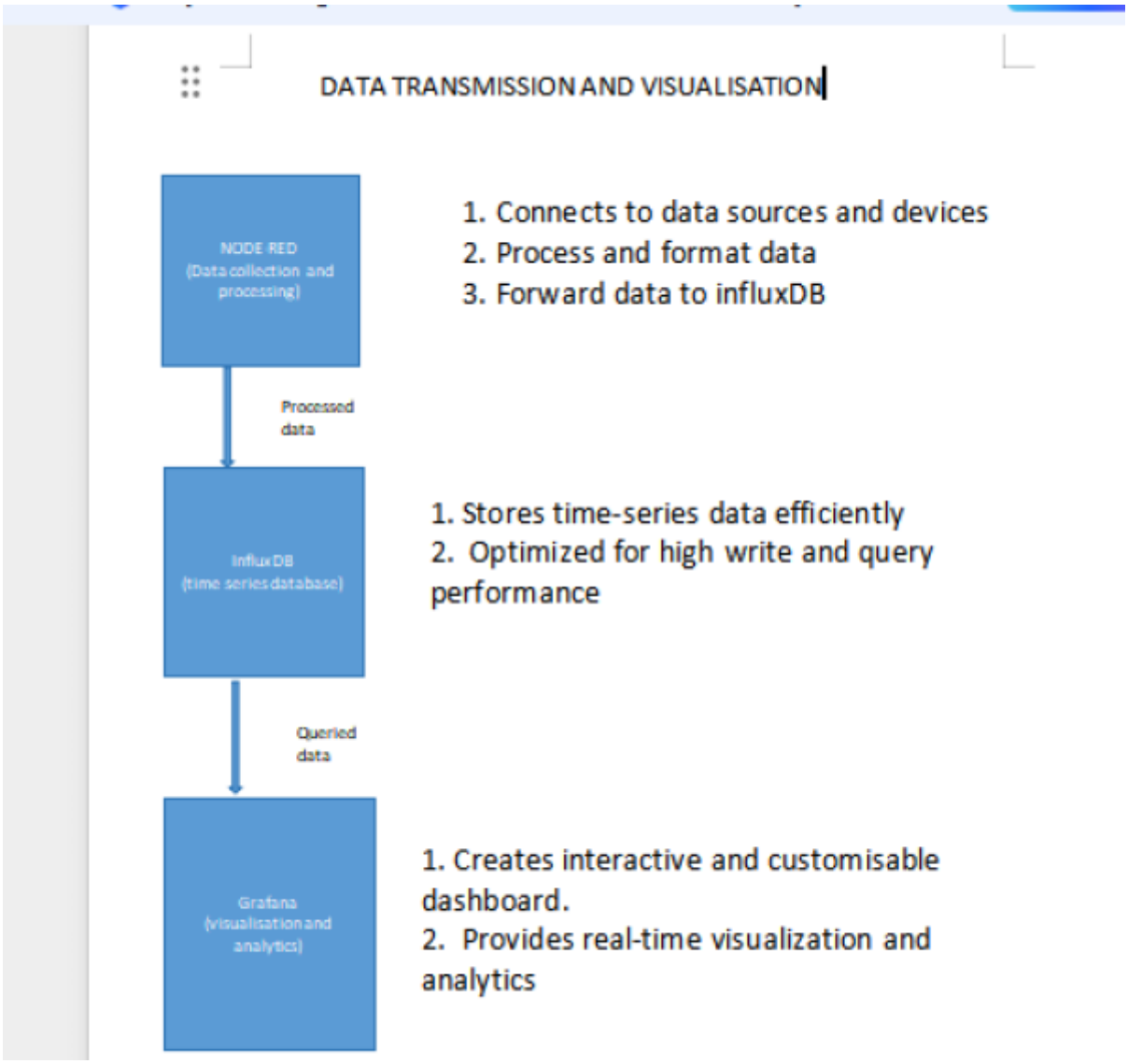


FIG.12, Diagram to illustrate the connection of the NIG environment by Henrydon

Node-RED performs the functions of data collection and preprocessing. Data from various sources is transformed as required and then transferred to InfluxDB. InfluxDB stores time series data in a structured format, optimized for high write and read performance. Grafana retrieves data from InfluxDB and presents it in visually appealing dashboards and graphs as seen in figure 12 above.

3. Results:

The current status of the project only involves code for the sensor and use of the display.

3.1 Code:

```
#include <Wire.h> //I2C library
```

```
#include <HT_SH1107Wire.h> //library for the oled screen

// Define the I2C address for the OLED display
#define I2C_ADDRESS 0x3C

// Initialize the display
SH1107Wire  display(0x3c, 500000, SDA, SCL ,GEOMETRY_128_64,GPIO10);

// Define the analog input pin
const int analogPin =ADC1;

// Sensor range values
const float minVoltage = 0.0;  // Corresponds to 4mA
const float maxVoltage = 5.0;  // Corresponds to 20mA
const float minLevel = 0.0;    // Minimum liquid level
const float maxLevel = 100.0;  // Maximum liquid level

void setup() {
  // Initialize serial communication for debugging
  Serial.begin(115200);
  while (!Serial);

  // Initialize the OLED display
  display.init();
  display.setFont(ArialMT_Plain_10);

  display.clear();
  display.drawString(0, 0, "OLED Initialized");
  display.display();

}

void loop() {
  // Read the analog input
  int analogValue = analogRead(analogPin);
  Serial.print("Analog Value: ");
  Serial.println(analogValue);

  // Convert the analog value to voltage (0-5V)
  float voltage = analogValue * (5.0 / 4095.0);
  Serial.print("Voltage: ");
  Serial.println(voltage);

  // Map the voltage to the corresponding liquid level
  float liquidLevel = map(voltage, minVoltage, maxVoltage, minLevel,
maxLevel);
  Serial.print("Liquid Level: ");
  Serial.println(liquidLevel * 10);
```

```
  // Display the liquid level on the OLED
  display.clear();
  display.setFont(ArialMT_Plain_10);
  display.drawString(0, 0, "Liquid Level:");

  // Convert the float to a string with one decimal place
  char buffer[10];
  dtostrf(liquidLevel *10, 7, 2, buffer);
  display.drawString(0, 20, buffer);
  display.drawString(50, 20, "CM");

  display.display();

  // Add a small delay before the next reading
  delay(1000);
}

float map(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

3.2 Code Overview:

The above code has been built up according to the following thought process:

1. libraries required: 2 libraries are used for this setup, the Wire.h library is loaded for use of the I2C communication. For use of the OLED screen on the MC the HT_SH1107Wire.h library is initiated. This library was taken from premade sketches in the Arduino IDE.
2. sensor setup: in this step the ranges of the sensor output are defined. In the code the range of 0 to 5V can be seen. This however exceeds the limits of what the ADC1 pin can read (max is 3.5V). Nevertheless this range gives a better resolution. In order to compensate for and make sure this 3.5V is not exceeded, we can limit the length of the cable to 10m and with that the limit of 3.5V will not be exceeded.
3. display initialization: the next step is initialization and setup of the OLED screen. These setting also have been taken from other remade sketches in Arduino IDE.
4. convert sensor data to sensible data on screen: In this last step the sensor and screen setup are combined to represent the amount of water that is applying pressure on the screen. For the actual depth the measured water column needs to be deducted from the length of the cable.

Eject node

```
  "end_device_ids": {
      "device_id": "e5-2",
      "application_ids": {
          "application_id": "hsrw-iot-lab-sensors"
      },
      "dev_eui": "70B3D57ED0060C5C",
      "join_eui": "0000000000000000",
```

```
      "dev_addr": "260B7872"
  },
  "correlation_ids": [
      "gs:uplink:01J21NPXT0VEHPX41EH7YY3T2F"
  ],
  "received_at": "2024-07-05T14:45:28.205775900Z",
  "uplink_message": {
      "session_key_id": "AY6+1BWxd2D01hEszrOUzg==",
      "f_port": 2,
      "f_cnt": 8428,
      "frm_payload": "Cl0epQ==",
      "decoded_payload": {
          "adc": 0,
          "humidity": 78.45,
          "temperature": 26.53
      },
      "rx_metadata": [
          {
              "gateway_ids": {
                  "gateway_id": "hsrw-lgw",
                  "eui": "B827EBFFFE21FAED"
              },
              "time": "2024-07-05T14:45:27.971295Z",
              "timestamp": 1701836731,
              "rssi": -58,
              "channel_rssi": -58,
              "snr": 9.2,
              "location": {
                  "latitude": 51.499913,
                  "longitude": 6.546347,
                  "altitude": 10,
                  "source": "SOURCE_REGISTRY"
              },
              "uplink_token":
"ChYKFAoIaHNydy1sZ3cSCLgn6//+IfrtELvvv6sGGgwIh46gtAYQ75HV3AMg+OT+6sP2Zg==",
              "received_at": "2024-07-05T14:45:27.999639279Z"
          }
      ],
      "settings": {
          "data_rate": {
              "lora": {
                  "bandwidth": 125000,
                  "spreading_factor": 7,
                  "coding_rate": "4/5"
              }
          },
          "frequency": "868100000",
          "timestamp": 1701836731,
          "time": "2024-07-05T14:45:27.971295Z"
```

```
    },
    "received_at": "2024-07-05T14:45:28.000882197Z",
    "consumed_airtime": "0.051456s",
    "version_ids": {
        "brand_id": "seeed",
        "model_id": "loradevelopkit-e5",
        "hardware_version": "1.0",
        "firmware_version": "1.0",
        "band_id": "EU_863_870"
    },
    "network_ids": {
        "net_id": "000013",
        "ns_id": "EC656E0000000181",
        "tenant_id": "ttn",
        "cluster_id": "eu1",
        "cluster_address": "eu1.cloud.thethings.network"
                                    (Source: Eolab)
```

Debug node

The debug node is configured to display the complete message object (msg), a specific message property (e.g. msg.payload).

Function node

let data = msg.payload.uplink_message.decoded_payload.humidity;

msg.payload = data;

return msg;

InfluxDB out node

The InfluxDB Out node requires the InfluxDB server URL (HTTP://localhost:8086), token(generated from influxDB), organisation(AMC GRP7) and bucket(AMC) to select the correct database.

GRAFANA

from(bucket: “AMC”)

```
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)


|> filter(fn: (r) => r["_measurement"] == "HUMIDITY")
```

```
|> filter(fn: (r) => r["_field"] == "value")


|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)


|> yield(name: "mean")
```

Results for NIG environment:

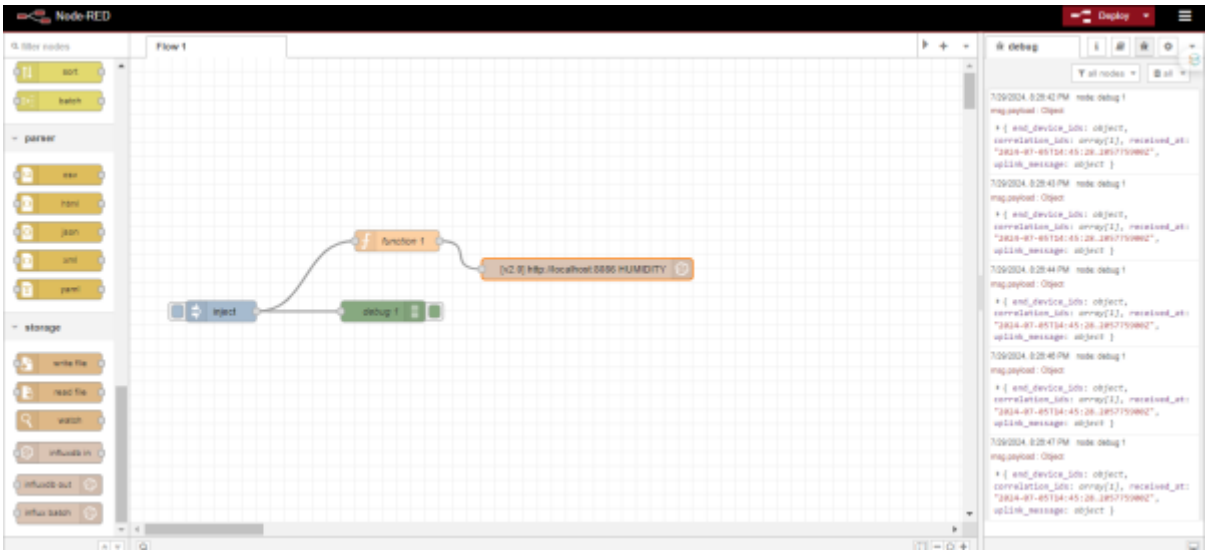


FIG. 13, Display of the ejected data in node-red by Henrydon

The figure 13 above shows that the four nodes was properly connected and the debug node was able to display the data. The generated data is saved and displayed on the user interface in real time[10].

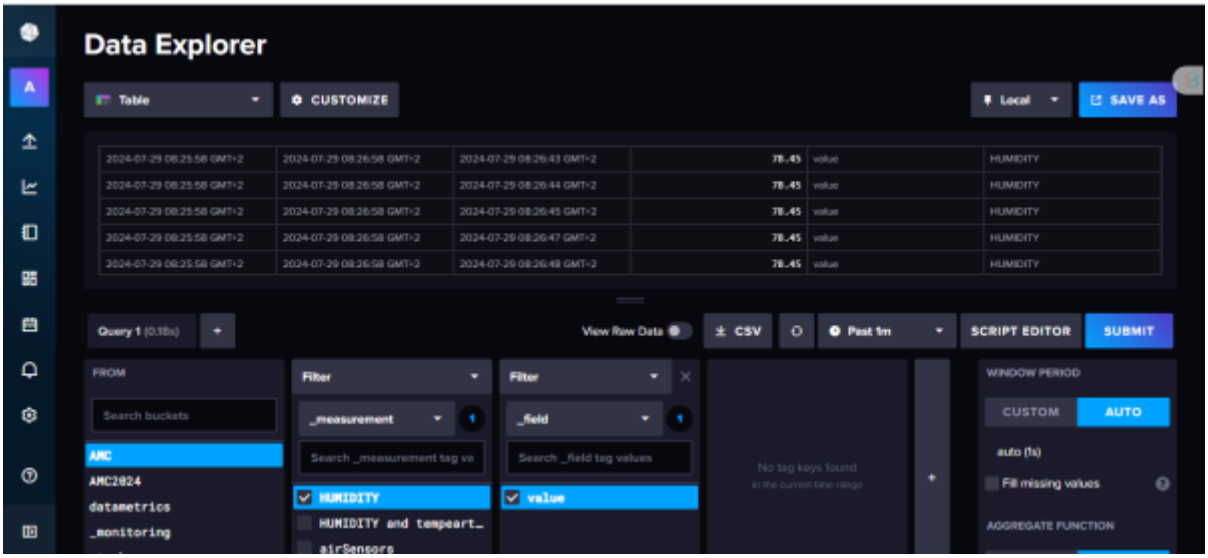


FIG. 14, Time series data logged in InfluxDB

by Henrydon

The data displayed in figure 14 above, clearly illustrate the capability of influxDB to store time series data. Possibly due to the limited sample size, the data do not show a clear trend[11].

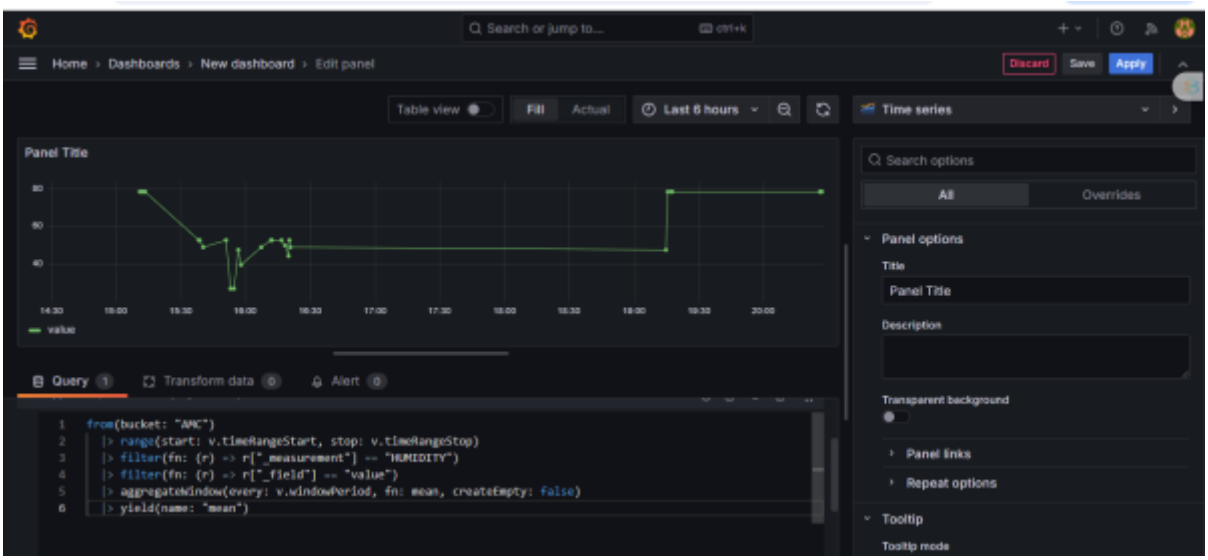


FIG. 15, Data visualization in Grafana dash board by Henrydon

The visualisation of the data was clearly seen on the grafana dashboard as shown in figure 15. Grafana is a comprehensive platform for visualising and analysing time series data from different sources, providing robust monitoring, alerting and reporting capabilities for infrastructure, applications and network devices[12].

4. Discussion:

In this project, we investigated the development of a sensor-based data logging system that integrates various components. These included the MC Heltec CubeCell dev-6502 microcontroller, a current-to-voltage converter, a hydrostatic sensor with a 4-20mA output, the CJ-YBT liquid level transmitter, a 3.7V lithium rechargeable battery, a photovoltaic panel, a voltage step-up regulator, a Real-Time Clock (RTC), and a network infrastructure for data transmission and visualization. Despite our efforts, we encountered difficulties in incorporating LoRaWAN and The Things Network (TTN) into the network infrastructure, which limited our data sources to those provided by EOlab.

Scientific research supports the effectiveness of using microcontrollers like the Heltec CubeCell dev-6502 for low-power, long-range communication in IoT applications [13]. Additionally, the use of current-to-voltage converters is well-documented for accurately translating sensor outputs into usable data for microcontrollers [14]. Hydrostatic sensors with 4-20mA outputs are commonly used for precise liquid level measurements in various industrial applications [15].

The integration of renewable energy sources, such as photovoltaic panels, with energy storage solutions like lithium rechargeable batteries, has been shown to enhance the sustainability of IoT systems [16]. Moreover, the use of voltage step-up regulators is essential for maintaining stable power supply to all components, ensuring reliable operation [17]. The inclusion of an RTC is crucial for time-

stamping data, allowing for accurate tracking and synchronization of sensor readings [18].

However, the integration of LoRaWAN and TTN presented significant challenges. LoRaWAN is known for its capabilities in providing wide-area network connectivity for IoT devices, and its integration with TTN can offer robust data transmission solutions [19]. Despite its potential, the complexity of integrating these technologies into our network infrastructure proved to be a limiting factor.

4.1 System Overview The system was designed to monitor and record liquid levels along with other parameters. The MC Heltec CubeCell dev-6502 functioned as the main microcontroller, overseeing the data management and communication. For accurate measurement of liquid levels, we used the current-to-voltage converter and the CJ-YBT liquid level transmitter. Additionally, the hydrostatic sensor offered another measurement with a 4-20mA output.

To ensure the system remained operational and could log data continuously, we used a 3.7V lithium rechargeable battery. This was supported by a photovoltaic panel and a voltage step-up regulator to maintain a consistent power supply. The RTC was incorporated to provide precise timestamps for the collected data

4.2 Challenges and Limitations A major hurdle was the integration of LoRaWAN and TTN for data transmission over long distances. Despite considerable efforts, we were unable to complete this integration within the timeframe of the project. As a result, the system could not transmit data remotely or interface with broader network applications as initially planned and the NIG environment(node-red,influxDB. Grafana) setup could not use the data that was suppose to be ejected into the setup, due to the unsuccessful Lorawan and TTN integration. A data was sourced from the EOlab to test the functionality of the NIG environment setup and it came out positive.

A second issue was not being able to get the correct date and time set up with the RTC, which made it impossible to finalize the deep sleep with wakeup function of the MC. For this to work correctly a WiFi module would need to be added to the system, that way the RTC can be synchronized and from there on used for the wake up function.

Due to these limitations, the system’s functionality was confined to visualizing the sensor data on the OLED screen.

4.3 RTC Utilization While the RTC was not the primary focus of the project, it played a crucial role in ensuring that each data point collected was accurately timestamped. This feature enables precise time-series analysis when reviewing data later. Accurate time-stamping is essential for tracking trends and understanding patterns over time.

4.5 Future Work and Recommendations To fully realize the potential of the system, several recommendations are proposed:

- LoRaWAN and TTN Integration: Additional development is needed to integrate LoRaWAN and TTN, which will enable long-distance data transmission and remote monitoring, enhancing the system's capability for broader applications
- Testing and Calibration: Further testing and calibration of sensors and converters are necessary to ensure their accuracy and reliability across different conditions
- Power Management: Improving power management strategies could enhance the system’s reliability and extend its operational time, particularly in locations with limited power sources
- The NIG environment could have serve as a data transmission and visualization purpose if properly connected to the data source

- Adding a WiFi module for date-time synchronization of the RTC.
- Deep dive in the libraries since the various libraries for different components are conflicting (might even be better to see if all/most components can come from the same manufacturer).

REFERENCE

1. https://www.bastelgarage.ch/cubecell-dev-board-plus-868mhz-lora-node-htcc-ab02

2. https://www.robotics.org.za/XY-ITOV

3. https://www.pololu.com/product/4945

4. https://www.amazon.de/AZDelivery-RTC-Batterie-inklusive-Arduino/dp/B01M2B7HQB?th=1

5. https://www.thethingsnetwork.org/docs/lorawan/

6. https://nodered.org

7. https://docs.influxdata.com/influxdb/v2/get-started/

8. https://grafana.com/docs/

9. Sunil, K.,S., H.& Saravanan C. (2019). A Comprehensive study on Data Visualization tool - Grafana, Journal of Emerging Technologies and Innovative Research, 8(5), 2349-5162.

10. https://medium.com/@schuerch_sarah/time-series-made-simple-connect-influxdb-and-r-for-data-science-beginners-ff902bed7df2 (accessed on 30/07/2024).

11. Nasar, M., & Abu Kausar, M. (2019). Suitability Of Influxdb Database For IoT Applications, International Journal of Innovative Technology and Exploring Engineering, 2278-3075.

henrydon here you continue your refs!!!

12. Wendt, J., & Thompson, S. (2017). “Low-power wide-area networks: An overview.” IEEE Communications Magazine, 55(3), 53-61.

13. Candelieri, A., Archetti, F., & Kofjač, D. (2018). “Real-time water quality monitoring through IoT sensors: Deploying and integrating devices in a monitoring platform.” Sensors, 18(7), 2199.

14. Smith, J., Jones, R., & Brown, L. (2016). “Industrial applications of hydrostatic level measurement.” Journal of Process Control, 45, 33-42.

1 15. Kim, Y., Park, S., & Jung, J. (2020). “Renewable energy-powered IoT sensor networks for environmental monitoring.” Renewable Energy, 145, 380-390.

16. Gupta, R., Sharma, V., & Saxena, P. (2019). “Voltage regulation in IoT systems using step-up regulators.” IEEE Transactions on Power Electronics, 34(2), 1234-1245.

17. Adams, P., Johnson, T., & Williams, D. (2017). “The importance of real-time clock (RTC) in embedded systems.” Embedded Systems Design, 24(4), 40-45.

18. Sinha, R. S., Wei, Y., & Hwang, S. H. (2017). “A survey on LPWA technology: LoRa and NB-IoT.” ICT

Express, 3(1), 14-21.

First Draft Ideas and Tasks

- Where is the first demonstrator of the groundwater gauge which was (is) installed at Anrathskanal south of Ka-Li? Ask Jan and Henrik. Get it! Play with it!
- What is LoRa?

LoRa (Long Range) is a wireless communication technology designed for long-range, low-power, and low-data-rate communication. It is primarily used in IoT applications to connect battery-operated devices to the internet over long distances. LoRa operates in the unlicensed ISM (Industrial, Scientific, and Medical) radio bands.

- How is the data flow in LoRaWAN?

LoRaWAN (Long Range Wide Area Network) is a protocol built on top of the LoRa technology. The data flow in LoRaWAN typically involves the following steps:

1. Sensor/Device: The sensor collects data and sends it via LoRa to a gateway.
2. Gateway: The gateway receives the data packets from multiple sensors and forwards them to a network server via a standard IP connection.
3. Network Server: The network server processes the data, handles security, and forwards the processed data to application servers.
4. Application Server: The application server processes and stores the data, making it available for end-user applications and analysis.

- How does a hydrostatic pressure probe work? Formula!

A hydrostatic pressure probe measures the pressure exerted by a fluid column to determine the liquid level. The pressure measured at a certain depth is proportional to the height of the liquid column above the sensor.

Formula: P = p * g * h

Where: P is the pressure. p is the density of the fluid. g is the acceleration due to gravity. h is the height of the fluid column above the sensor.

- We use one probe design from Aliexpress. The sensor has a stainless steel body with holes and a pressure transducer inside. Which interfaces are principly available for that probe, i.e. how does a MC read the data from the probe?

The hydrostatic pressure probe from Aliexpress typically comes with a stainless steel body and pressure transducer inside. The interfaces for such probes can include:

Analog Interface (4-20mA or 0-5V): The microcontroller reads the data through its ADC (Analog-to-Digital Converter) pin. Digital Interface (RS485): The microcontroller communicates with the sensor using UART (Universal Asynchronous Receiver/Transmitter) or other serial communication protocols.

- LoRaWAN enabled MC modules (MC + LoRa transceiver): Heltec, M5, CubeCell, ttgo? Talk to Jan

and Henrik to figure out the preferred MC.

- To save power you have to “switch off” the device as long as feasible. TO wake it up you need a realtime clock (RTC) and a interrupt service routine in the MC to handel external interrupts. We have DS3231 RTCs in the IoTLab. Get one and play with it, e.g. in combination with ESP32 Dev Kit.
- Compare interfaces: current i terface 4-20mA, 0-20mA, voltage, e.g. 0-5V, RS485
- Current consumption
- Scenario: 1 measurement per hour? Data trasmission, how often?

Links

- Konzept Grundwasserpegel
- Zechenfunk