Temperature and Humidity Monitoring in Switchgear

Johnny Huynh

Degree Thesis for Bachelor of Engineering

Degree Programme in Electrical Engineering and Automation

Vaasa, 2020

EXAMENSARBETE

Författare: Johnny Huynh

Utbildning och ort: El- och automationsteknik, Vasa

Inriktningsalternativ: Automation

Handledare: Ronnie Sundsten, Mats Warg, Tom Kanerva, Thore Björkgren

Titel: Temperatur och fuktövervakning av ställverk

Datum Maj 14, 2020 Sidantal 48 Bilagor 4

Abstrakt

Examensarbetet har gjorts åt företaget VEO Oy som specialiserar sig på lösningar för automation och elektrifiering. Uppgiften gick ut på att undersöka olika temperatur- och fuktgivare och givarsystem som används i ställverken på marknaden för att vidareutveckla VEO:s ställverkprodukt VEDA5000.

Ett modernt tillvägagångsätt för implementering av temperatur och fuktövervakning av ställverk presenteras med design baserade på Zigbees trådlösa nätverk, RFID-passiva givare, Chameleon teknologin, SAW-baserad givare och optisk fiber. Examensarbetet går igenom teknologin som används av olika företag på marknaden och den teori som behövs för att konstruera ett mikrokontrollersystem för plattform, mjukvarutestande och för grundläggande IoT-förståelse. Andra fokus har varit på IoT-domänen, i olika moln, IoT-plattform, mikrokontroller och temperaturgivare.

Praktiska delen av examensarbetet går igenom processen för att ansluta en ESP32 mikrokontroller till temperatur- och fuktgivare med en Raspberry PI och för att sända data till molnet som till exempel IBM cloud genom Node-Red. Data samlas för fortsatt analys och grafisk visualisering.

Språk: engelska Nyckelord: VEO, ställverk, IoT, Zigbee, RFID, SAW, optisk fiber, VEDA5000, Raspberry Pi, ESP32, IBM

BACHELOR'S THESIS

Author: Johnny Huynh
Degree Programme: Electrical Engineering and Automation
Specialization: Automation
Supervisors: Ronnie Sundsten, Mats Warg, Tom Kanerva, Thore Björkgren

Title: Temperature and Humidity Monitoring in Switchgear

Date May 14, 2020 Number of pages 48 Appendices 4

Abstract

This Bachelor's thesis has been done for the company VEO Oy that specializes in providing solutions for automation and electrification, the assignment was to research different temperature and humidity sensors and sensor systems used in switchgear in the market to further improve VEO VEDA5000 switchgear product.

A modern approach to implementation of temperature and humidity monitoring for switchgear is presented on designs based on Zigbee wireless, RFID passive sensors, Chameleon technology, SAW-based sensor, and optic fiber. The thesis goes through the technology used by different companies in the market and the needed theory to construct a microcontroller system for platform, software testing, and basic IoT understanding. Other focuses have been set into the IoT domain, into different clouds, IoT platforms, microcontrollers, and temperature sensors.

The practical part of this thesis goes through the process to connect an ESP32 microcontroller to temperature and humidity sensors with a Raspberry Pi and to send data to the cloud such as IBM cloud through Node-Red, and data collects for further analysis and graphical visualization.

Language: english Key words: VEO, Switchgear, IoT, Zigbee, RFID, SAW, Optic fiber, VEDA5000, Raspberry Pi, ESP32, IBM

Table of Contents

List of figures

1 Introduction

Electrical switchgear is an essential part of an electrical power system and is used to distribute electrical power and to isolate electrical loads. Switchgears are produced in different forms but include a combination of electrical elements such as circuit breakers, disconnectors, fuses and distribution busbars arranged in a lineup of frames. Switchgears distributes electrical current, up to thousands of amperes that build up heat that can have unexpected temperature rises at a particular location, it can be useful to monitor temperature, humidity or partial discharge to know the condition of the switchgear to detect corrosion, some other type of defect or make improvements to the design. The impractical way to monitor the temperature has been to use infrared cameras but the systems available have now developed into more practical systems such as wireless systems and systems connected to the cloud and software for remote data analysis that uses different algorithms and graphical visualization of data.

1.1 Purpose

The main assignment was to research different temperatures and humidity systems and solutions for VEO's Low-voltage-switchgear VEDA5000 product to develop or implement a system for the switchgear in the future and possibly also into other VEO switchgear products like the Medium-voltage switchgear Vector. The systems researched could improve the switchgear design and condition and become more automated from monitoring from infrared cameras. As switchgears are installed, access to their backside content can be difficult to reach and when a huge amount of current is conducted, a strong magnetic field is developed so a non-interference system, non-maintenance, and wireless system would be most practical.

As the IoT domain is becoming more and more active in the industrial sector, implementing a system connecting the switchgear to the internet and taking advantage of the data a switchgear can give, could be beneficial, cost-saving, and efficient for data analysis using different algorithms.

1.2 VEO Oy

Vaasa engineering now VEO Oy, was founded in 1989 by Mauri Holma, Harri Niemelä, Martti Manner, Pekka Haakana, Jan Sandvik, Heikki Ojakoski and Martti Ehrnrooth. Vaasa Engineering's main business idea was to deliver automation and electrical system solutions for energy and power plants in Finland and abroad has now expanded into different business sectors. The different areas VEO offers solutions in are power production, power distribution, and power utilization. [1]

POWER PRODUCTION
+
POWER DISTRIBUTION
+
POWER UTILISATION
EXPERT SERVICES

Figure 1 VEO business sectors [1]

Power Generation applications:

- **Hydropower:** Consultation and turbine and generator turnkey deliveries and modernization of electrification and automation systems for hydropower plants.

- **Thermal Power:** Turnkey deliveries of instrumentation, automation system design and customization of automation and electrification system of power plants.

  Engine and Hybrid power: Hybrid power plant and marine project solutions such as plant commissioning services, spare parts for engines and modernization of power plant equipment and electrification, maintenance, and end-user training.

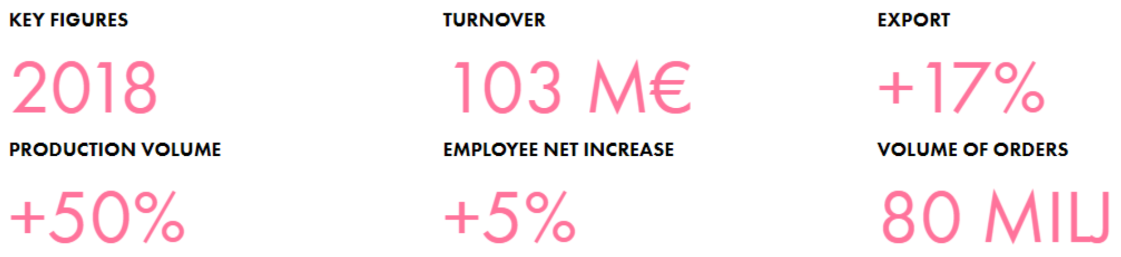
Power Distribution applications:

- **Wind power:** Turnkey deliveries of applications, connecting wind farms to the grid with the process consisting of engineering, civil work, substation building, transformers, wind farm cabling, switchgear, control and protection systems, monitoring systems, installation, and commissioning.

Power Utilization applications:

- **Industry:** Small rebuilds and retrofit projects from greenfield to existing plants for different industries such as pulp and paper, marine. Project deliveries include design, engineering, factory acceptance testing, installation, commissioning, start-up, and customer training.

- **Marine:** Solutions with a product range of main and distribution switchboards, motor control centers, electric drives applications, generators, marine thrusters, transformers, protection equipment. Deliveries can include preliminary design, dimensioning, manufacturing, calculations and engineering, installation supervision, commissioning, and customer training.



KEY FIGURES	TURNOVER	EXPORT
2018	103 M€	+17%
PRODUCTION VOLUME	EMPLOYEE NET INCREASE	VOLUME OF ORDERS
+50%	+5%	80 MILJ

Figure 2 VEO key figures from 2018 [1]

VEO has over 400 employees and offices in Norway, Sweden and the UK with a turnover of over €100M and aiming for over €200M by 2023 and VEO is one of the largest turnkey delivery suppliers of substations in the Nordic countries. [1]

1.3 Switchgear VEDA 5000

Veda 5000 is a low voltage switchgear designed according to customer's needs for specific applications in both land and sea industry.

Standards
- IEC 61439 parts 1-2
- IEC 61439-6
- PSK 1801

Other standards (applicable parts)
- IEC 60092-302
- IEC TR 61641:2008

Rated insulation voltage
- 1000 VAC

Rated operational voltage
- ≤690 VAC, 50/60 Hz

Rated nominal current (max.)
- 5000 A

Rated impulse to withstand the voltage of main circuits
- 12 kV

Short-circuit withstand strength (max.)
- Main bus bars:
  – Thermal limit current Icw (1 sec) 100 kA
  – Dynamic limiting current Ipk 220 kA
- Distribution bus bars:
  – Thermal limit current Icw (1 sec) 80 kA
  – Dynamic limiting current Ipk 176 kA

Arc withstand
- 50 kA, 300 ms

Degree of protection by enclosure
- IP21 –IP55, EN 60529
- Against mechanical impact, EN 62262: IK10

Support and lead-through insulators
- Ball pressure test, EN 61439-3: 185°C
- Glow-wire test, EN 61439-3: 960°C
- Tracking test, IEC 112: > 400 V

Surface finish
- SFS 5225 SP 60/1 ZnFo
- Paint thickness 60 um
- Colour shade RAL 7035
- Shockproof effect surface

Recommended environmental classification for switchgear room
- EN 721-3K3/372/374/3B1/3C1/3S1

Dimensions mm
- Height 2050, 2250
- Depth 640, 840, 1040

Figure 3 Technical data of the VEDA5000 low-voltage switchgear [1]

Main areas:

- VEDA Motor Control Center
- VEDA Drives
- VEDA Marine
- VEDA Hybrid

2 Internet of Things and microcontroller

Chapter 2 gives a short overview of the IoT domain, hardware, sensors, and software used in the practical part to test IoT platforms, different software and to get a basic understanding of IoT.

2.1 The Internet of Things (IoT)

The Internet of Things is a wide technology category that includes connected devices working together as a system to deliver data within an application, the subsets that can be viewed with IoT are for example:

- Machine to Machine communication (M2M)
- Machine to Human communication (M2H)
- Radio Frequency Identification (RFID)
- Location-based services (LBS),
- Lab-on-a-Chip (LOC) sensors
- Augmented Reality (AR)
- Robotics and vehicle telematics

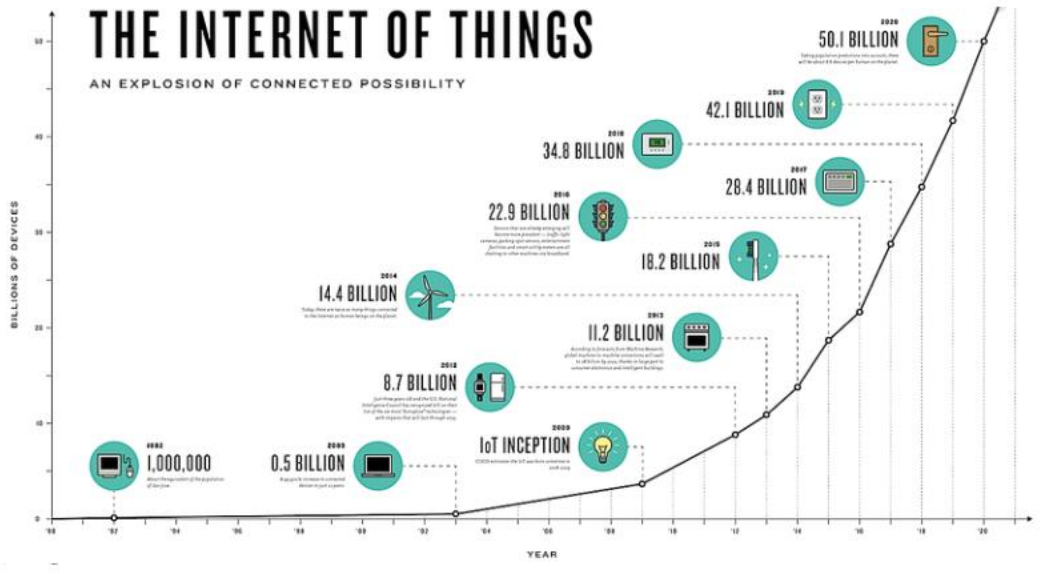


Figure 4 IoT growth graph from 1992 to 2020 and onward [2]

Connected devices by 2020 are estimated to reach 50 billion devices with many of the technologies started from military and industrial supply chain applications. The common features are combined sensory objects with communication intelligence and data running over a mix of wired and wireless networks, the data is associated with an analytics or decision support engine enabling an actionable outcome. The data collected from the system at different points is sent to a platform that enables the application and data transmission to the device endpoint. [3]

2.2 Cloud computing

In Cloud computing there are three main types of cloud computing providing its distinct range of services, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS).

Infrastructure as a Service (IaaS): Offers cloud computing based on on-demand to networking, storage, and server computing resources. According to the storage and processing needs, the hardware resource can scale and run on your platform and applications through the provider's infrastructure. [4]

Platform as a Service (PaaS): Provides access to a cloud environment to develop, manage and host applications with a range of tools to support testing and development, and the provider is responsible for the underlying infrastructure, backups, security, and operating system. [4]

Software as a Service (SaaS): Provides access to cloud-based software that self manages itself instead of installing a software application on your local device. You access the applications using the web or an API, An API (Application Programming Interface) is a standardized, documented interface that allows one computer application to "talk" to another application for the exchange of messages and data between one computer application and another. [4]

Users pay for per-use-basis according to how much of the service is used and can be cost-saving, reducing resources spent on managing hardware and software and gives you the ability to scale your compute and storage requirements according to demand. It offers security solutions to protect your system and data with expert monitoring and provides rapid data recovery/vast data center if an outage occurs. [4]

2.2.1 Cloud platforms

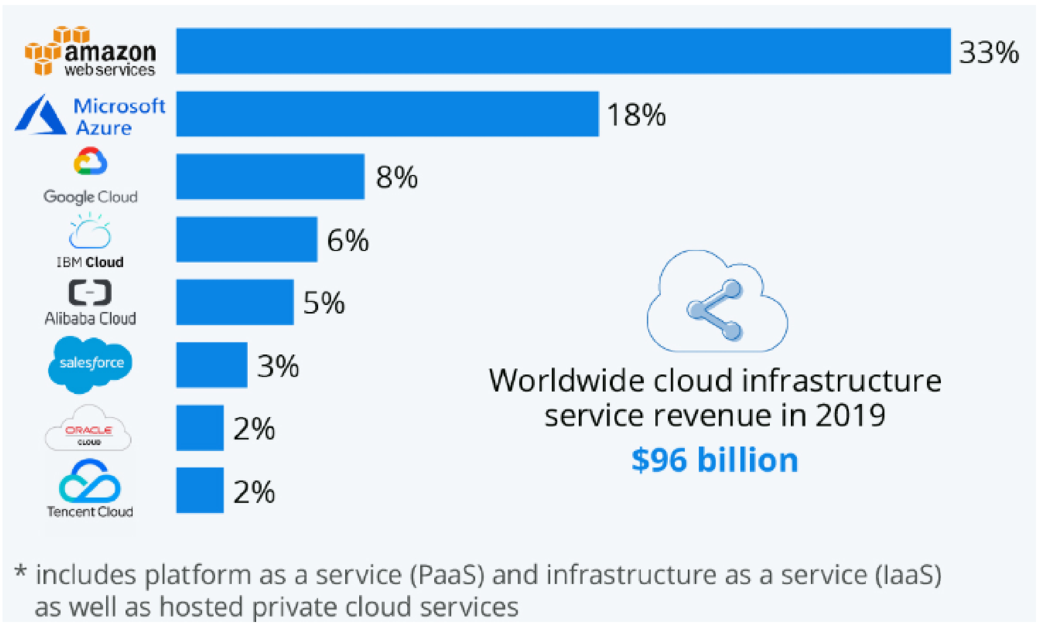
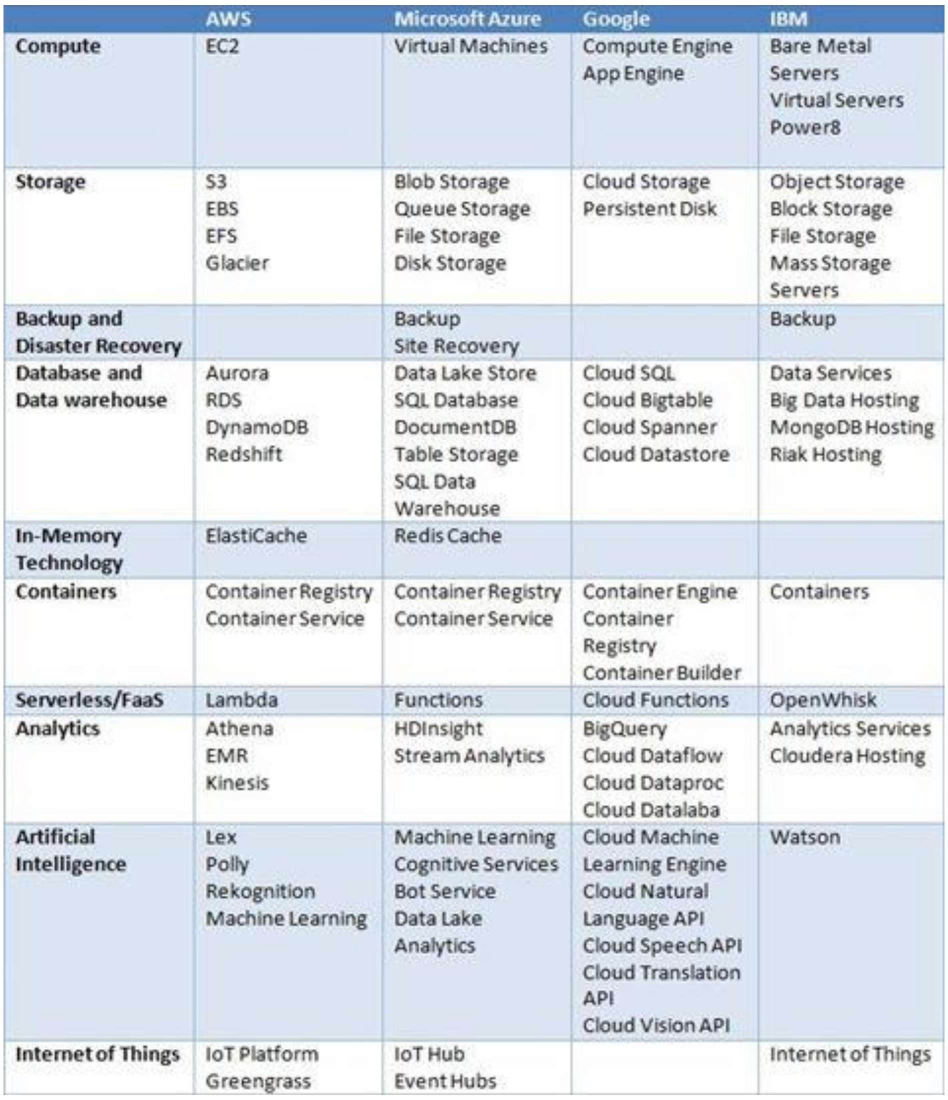


Figure 5 Market leading Cloud platforms from 2019 [5]

Amazon web services is the market leader with 33% of the share for 2019 after entering the market in 2016 followed by Microsoft, Google, and IBM.

Choosing one cloud vendor over the other depends on the needs of each customer and the workloads that are being run. Some organizations choose to use multiple providers within different parts of the operations/use case for a multi-cloud approach. AWS, for example, offers a wider choice of services because of entering the market first and Microsoft is a popular choice because of its combination with other Microsoft applications such as Office 365 and Teams, all of them are strong in machine learning but Google stands out for its deep expertise around open source technologies. Most of the cloud platforms offer mostly the same services but some have better quality, different prices/price models, and different options.



	AWS	Microsoft Azure	Google	IBM
Compute	EC2	Virtual Machines	Compute Engine App Engine	Bare Metal Servers Virtual Servers Power8
Storage	S3 EBS EFS Glacier	Blob Storage Queue Storage File Storage Disk Storage	Cloud Storage Persistent Disk	Object Storage Block Storage File Storage Mass Storage Servers
Backup and Disaster Recovery		Backup Site Recovery		Backup
Database and Data warehouse	Aurora RDS DynamoDB Redshift	Data Lake Store SQL Database DocumentDB Table Storage SQL Data Warehouse	Cloud SQL Cloud Bigtable Cloud Spanner Cloud Datastore	Data Services Big Data Hosting MongoDB Hosting Riak Hosting
In-Memory Technology	ElastiCache	Redis Cache		
Containers	Container Registry Container Service	Container Registry Container Service	Container Engine Container Registry Container Builder	Containers
Serverless/FaaS	Lambda	Functions	Cloud Functions	OpenWhisk
Analytics	Athena EMR Kinesis	HDInsight Stream Analytics	BigQuery Cloud Dataflow Cloud Dataproc Cloud Datalaba	Analytics Services Cloudera Hosting
Artificial Intelligence	Lex Polly Rekognition Machine Learning	Machine Learning Cognitive Services Bot Service Data Lake Analytics	Cloud Machine Learning Engine Cloud Natural Language API Cloud Speech API Cloud Translation API Cloud Vision API	Watson
Internet of Things	IoT Platform Greengrass	IoT Hub Event Hubs		Internet of Things

Figure 6 Services different cloud providers offers from 2018 [6]

2.2.2 IBM Cloud services

IBM cloud offers over 170 services such as:

Compute

Compute resources, bare metal servers, virtual servers, serverless computing and containers. [7]

Network

Cloud networking services, virtual private network (VPN) tunnels and firewalls. [7]

Storage

Object, block and file storage for cloud data. [7]

Integration

Services to integrate on-premises systems or different kind of applications. [7]

IoT Watson platform

Cloud-hosted service with the capabilities to register, control and connect devices for visualization and data storage. [7]

IBM Cloudant

Open-source Apache CouchDB based database service running as a service in the IBM cloud which stores JSON documents.

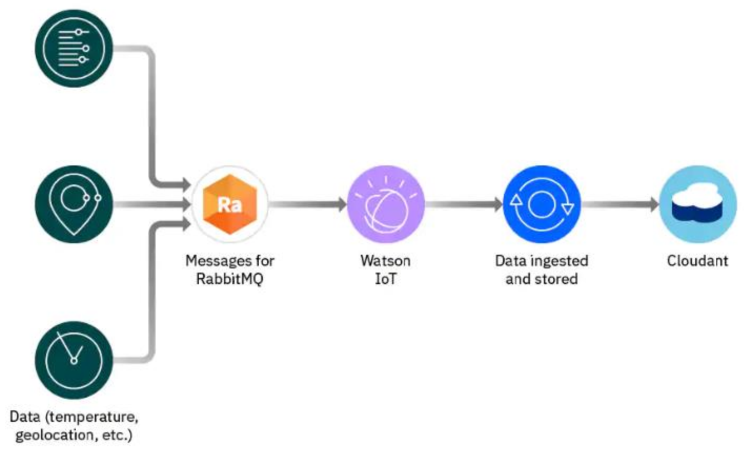


Figure 7 Example system configuration with Cloudant service [8]

Artificial Intelligence (AI)

Machine learning, natural language processing, and visual recognition. [7]

2.3 Node-red

Invented by J, Paul Morrison, and developed by IBM's Emerging Technology (a collaboration between experts and customer), Node-red is a programming open-source tool for wiring together "nodes" (code boxes), hardware devices and APIs through a user-friendly browser-based editor. A wide range of nodes containing data with specific codes can be wired together so that tasks pass on to the next node which creates a flow until the end of the node flow. Without the need to program complex code and understand the individual lines of code, node-red simplifies the complex tasks by a visual representation to make it more accessible to a wider range of users. [9]

Node red allows users to start doing complex tasks quite quickly without the need to spend countless hours writing code.

Tasks Node-Red can do are for example:

- MQTT connection between development boards (Arduino, Raspberry, ESP32)
- Retrieve data from the web (stock prices, weather forecast)
- Create time-triggered events
- Store and retrieve data from different database services (MySQL, Cloudant, Db2)
- Communicate with third-party services (IFTTT.com, adafruit.io)

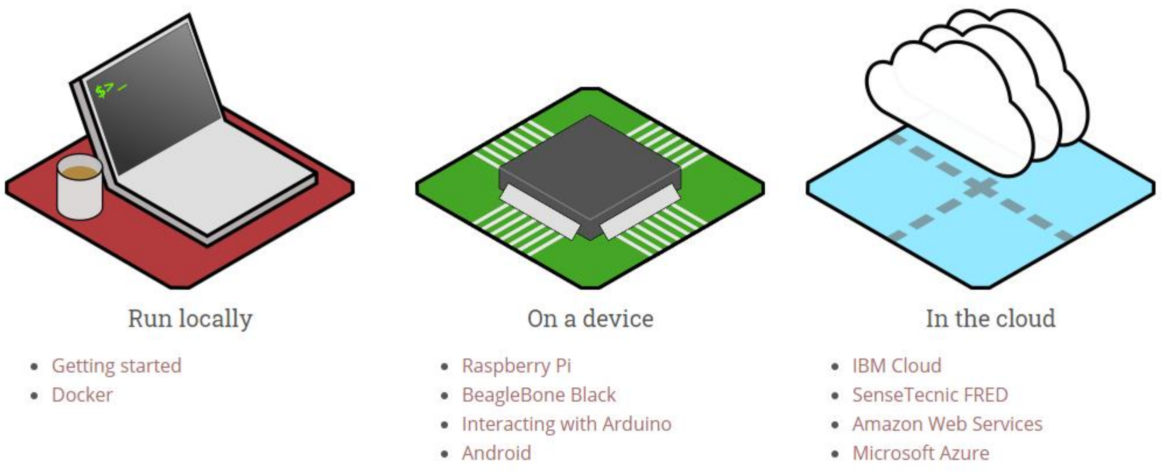
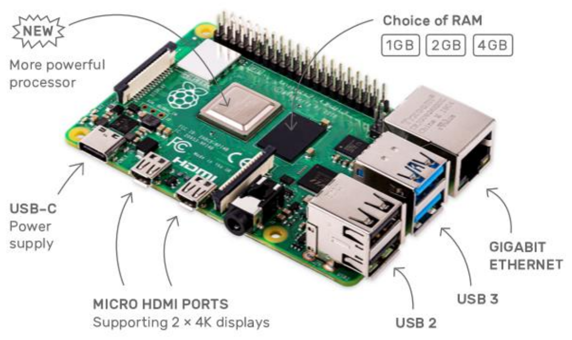


Figure 8 Different options to run Node-Red on [9]

Node-Red is built on Node.js making it possible to run it on different low-cost hardware, cloud, and virtual computer.

2.4 Raspberry Pi

Raspberry pi is a low-cost single-board computer (SBC) built on a single circuit board with microprocessor(s), memory, input/output (I/O) as the usual features. The device is made by the Raspberry Pi Foundation, a charity from the UK aiming at educating people in computing and making computing education more accessible. The first Raspberry Pi was launched 2012 with a single-core 700 MHz CPU and 256 MB RAM but has now developed to the 4th version that has quad-core 1.5 GHz CPU and 4 GB RAM. Raspberry Pi runs on the Linux operating system and is a perfect cost-effective device for hardware projects, home automation, and industrial applications. [10]



- **SoC:** Broadcom BCM2711B0 quad-core A72 (ARMv8-A) 64-bit @ 1.5GHz
- **GPU:** Broadcom VideoCore VI
- **Networking:** 2.4 GHz and 5 GHz 802.11b/g/n/ac wireless LAN
- **RAM:** 1GB, 2GB, or 4GB LPDDR4 SDRAM
- **Bluetooth:** Bluetooth 5.0, Bluetooth Low Energy (BLE)
- **GPIO:** 40-pin GPIO header, populated
- **Storage:** microSD
- **Ports:** 2 × micro-HDMI 2.0, 3.5 mm analogue audio-video jack, 2 × USB 2.0, 2 × USB 3.0, Gigabit Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
- **Dimensions:** 88 mm × 58 mm × 19.5 mm, 46 g

Figure 9 Different feature and specification of the latest Raspberry Pi version 4 [11]

Raspberry Pi operates in the open-source ecosystem with Linux as its operating system. The Raspberry Pi Foundation releases many of its own software and schematic as open-source but the board is not open hardware. [10]

2.5 ESP32

Upgraded by Espressif from the previous version ESP8266 now supports WiFi and Bluetooth 4.0. The ESP32 contains Tensilica Xtensa® Dual-Core 32-bit LX6 microprocessor with 240MHz clock frequency and has a 4 MB flash memory for program and data storage and 448 KB of ROM. The ESP32 has 802.11b/g/n HT40 WiFi transceiver integrated which allows it to work as its network and WiFi Direct for peer to peer connection without the need for an access point.

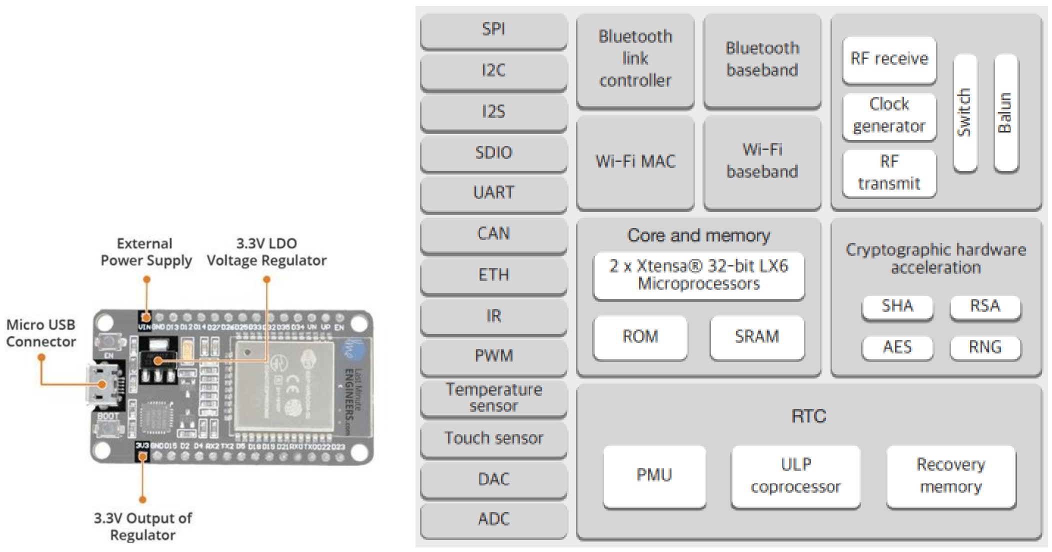


Figure 10 ESP32 main functions [12]

The microcontroller has both analog to digital and digital to analog channels and Pulse Width Modulation pins, it supports both SPI, I^{2}C interfaces, and UART for RS232 or RS485 communication. Development platforms ESP32 can be used with are, for example, ESPRRUINO (Javascript SDK), Mongoose OS (OS for IoT), and Arduino IDE. Arduino IDE (Integrated Development Environment) is open-source software for wiring, compiling, and uploading code to Arduino compatible devices for operating systems mac, Windows, Linux. The environment supports C and C++ programming languages. [13]

2.6 Modbus and MAX485 Module

Modbus is a serial communication protocol developed by Modicon in 1979 who is now owned by Schneider Electric for programmable logic controllers (PLC). RS-485 uses half-duplex where only one transmitter can be active while transmitting to one or more receiver (slave) and allows transmission distances of up to 1.2 km and can be extended with a repeater. Its task is to transmit information over serial lines between electronic devices from the Modbus Master to Modbus Slaves who function as devices supplying information. [14]

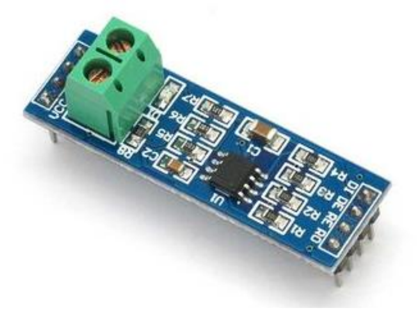


Figure 11 MAX485 Module [15]

There can be 247 Slaves in a standard Modbus network and each with a unique address from 1-247 to tell the Slaves which message to ignore and to receive from the master, the second byte sent by the master is the function code that tells the slave what to write or read and checks if the right command was initiated, the last bytes is for error detection to tell if the message content is correct. The Slave then confirms the data received and sends the data requested and lastly error checks. There are 3 different Modbus protocols, Modbus ASCII, Modbus RTU, and Modbus TCP/IP. Modbus ASCII uses readable ASCII to transmit bytes, RTU uses binary and Modbus TCP/IP standard TCP/IP network communication over port 502. [14]

The MAX485 module is used to connect microcontrollers like raspberry pi to RS-485 communication. The module works with +3.3 V/+5 V power supply with the rated current 300 µA and uses half-duplex communication when converting the TTL level (Transistor-Transistor logic) into RS-485 level with the maximum transmission rate of 2.5 Mbps. [15]

2.7 Open Platform Communications United Architecture (OPC UA)

Open Platform Communications United Architecture (OPC UA) is a data exchange standard for industrial communication M2M, PC2M. makes it possible to turn different protocols like Profibus, Modbus, from different manufacturers like Siemens, Beckhoff into one single line of communication. [16]

2.8 I²C and SPI chip communication interface

Inter-inter grated circuit (I²C) and serial peripheral interconnect (SPI) are both serial interfaces where I²C is more software complex and SPI less and requires more hardware resources from pins.

2.8.1 SPI

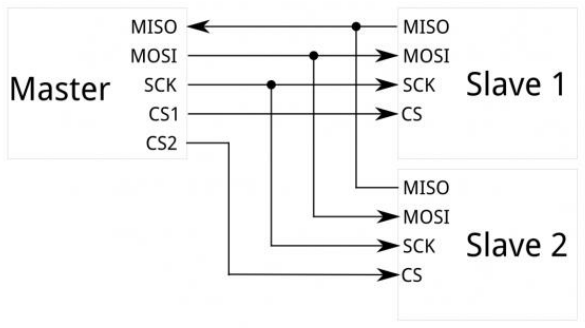


Figure 12 SPI communication interface connection example [17]

SPI uses separate lines for data and clock to keep both sides in sync, the clock oscillates a signal and tells the receiver when to sample the bits on the data line. When the receiver detects low to high or high to the low edge, it will look at the data line to read the next bit. SCK (serial Clock) or CLK (Clock signal) is the only side that generates the clock signal, the side is called "master" and the other side slave and there is always only one master but there can be multiple slaves. [18]

The line where data is sent from master to slave is called MOSI (Master Out/Slave in) and when the slave sends back a response, the master will generate a prearranged number of clock cycles meaning that the master always generates the clock cycle and needs to know in advance how much data a slave returns data and when the slave will put data onto a third data line called MISO (Master In / Slave Out). The SS (Slave Select) / CS (Chip Select) tells a Slave when to wake up to receive or send data so the line can choose a slave to talk to. [18]

2.8.2 I²C

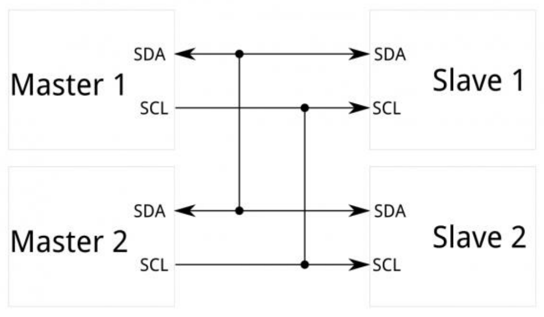


Figure 13 I²C communication interface connection example [17]

The Inter-inter grated circuit (I²C) only requires two signal wires to exchange information and can allow multiple slaves to communicate with one or more master. The signals are SCL (Serial Clock) for the clock signal and SDA (Serial Data) for the data signal. The clock tells the device when to read a bit when the data signal is either low (0) or high (1), I²C is an open-drain configuration where you need a pull-up resistor to make the signal high by default. [19]

2.9 Temperature and humidity sensors

Different types of sensors are used in different environments and applications, its function can either make it suited or not suited for the specific need. Some of the temperature sensors in the market today are semiconductor sensors, thermocouples, resistance temperature detectors, thermistors and they all sense a change in a physical characteristic.

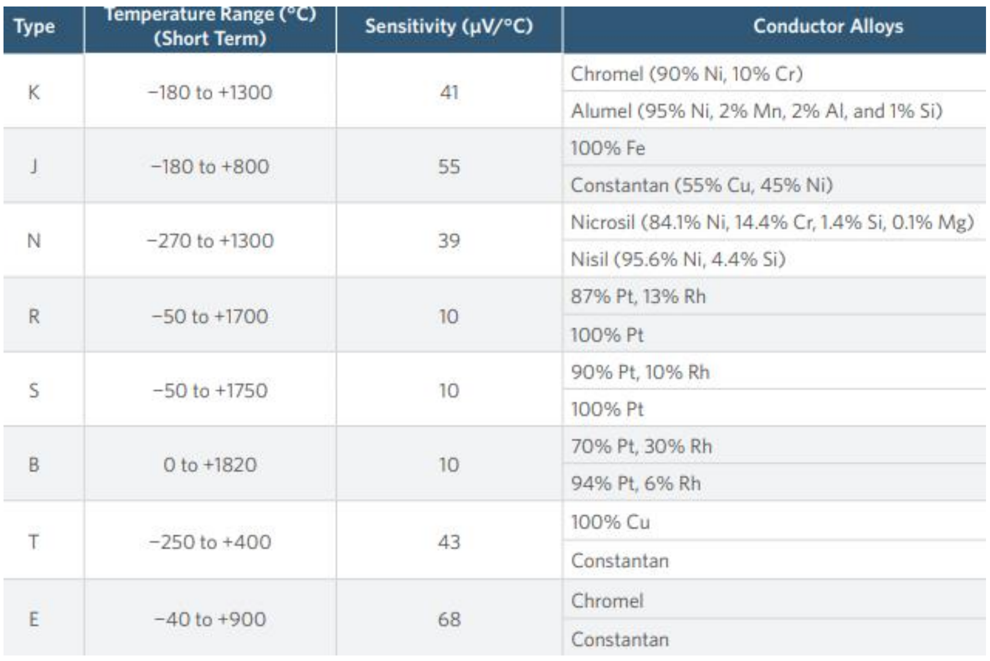
2.9.1 Semiconductor based integrated circuit

Two different types of semiconductor-based temperature sensors are local temperature and remote digital temperature sensors, local measures their die temperature by using the physical properties of a transistor. The remote digital measures using an external transistor away from the sensor chip. Local temperature sensors use analog (voltage/current) or digital output (I²C, SPI). [20]

2.9.2 Thermocouple

Thermocouples use Seebeck effect to measure temperature from the temperature differential between junctions of two wires of dissimilar metals wired together, the voltage change is measured from the change of temperature between the heated and cooler area.

There are various types of thermocouples made of different materials for different temperature ranges and sensitivity. Assigned by letters, there are for example E, J, N types and the most commonly used are K, J, and T type. Their small output voltage requires precise amplification and external noise can occur with long wires but thermocouples have a fast response. [20]



Type	Temperature Range (°C) (Short Term)	Sensitivity (µV/°C)	Conductor Alloys
K	−180 to +1300	41	Chromel (90% Ni, 10% Cr)
			Alumel (95% Ni, 2% Mn, 2% Al, and 1% Si)
J	−180 to +800	55	100% Fe
			Constantan (55% Cu, 45% Ni)
N	−270 to +1300	39	Nicrosil (84.1% Ni, 14.4% Cr, 1.4% Si, 0.1% Mg)
			Nisil (95.6% Ni, 4.4% Si)
R	−50 to +1700	10	87% Pt, 13% Rh
			100% Pt
S	−50 to +1750	10	90% Pt, 10% Rh
			100% Pt
B	0 to +1820	10	70% Pt, 30% Rh
			94% Pt, 6% Rh
T	−250 to +400	43	100% Cu
			Constantan
E	−40 to +900	68	Chromel
			Constantan

Figure 14 Different types of thermocouples and their configurations [21]

2.9.3 Resistance temperature detector (RTD)

Resistance temperature detector (RTD) sensors are based on measuring the resistance of a metal that changes with temperature change. The most common and accurate material used to make RTDs are platinum with 100 Ω and 1000 Ω resistance at 0 °C referred to PT100 and PT1000, other materials can be nickel or copper. Near-linear response to temperature changes makes platinum RTDs a stable and accurate sensor but responds slower to temperature changes than thermocouples because of its high thermal mass, to be reliable, RTDs requires external current and signal conditioning. Different types of techniques exist to measure as the self-heat from a current flow can affect the measurement, the methods are two wired, three wired and four wired with the three wired almost compensating and four wired method completely compensates. [20]

2.9.4 Thermistor

Thermistors are small resistance measuring based sensors similar to RTDs but use material such as polymer, ceramic, and have a non-linear temperature to resistance which makes them a bit unstable and in need of correction. Positive Temperature Calibrated (PTC) thermistor is where the resistance increases with temperature and in Negative Temperature Calibration (NTC) resistance decreases. [20]

2.9.5 BME280

BME280 is a digital temperature, humidity, and pressure sensor manufactured by Bosch, Its measures temperature from -40 °C to 85 °C with ± 1.0 °C accuracy, barometric pressure from 300 Pa to 1100 hPa with ± 1.0 hPa accuracy and humidity from 0-100% with ±3% accuracy.

The module only requires 3.3 V or 5 V and uses an onboard 3.3 V regulator with the possibility to either use I²C communication interface or SPI. The BME280 consumes less than 1 mA during active measurements and 5 µA during idle.

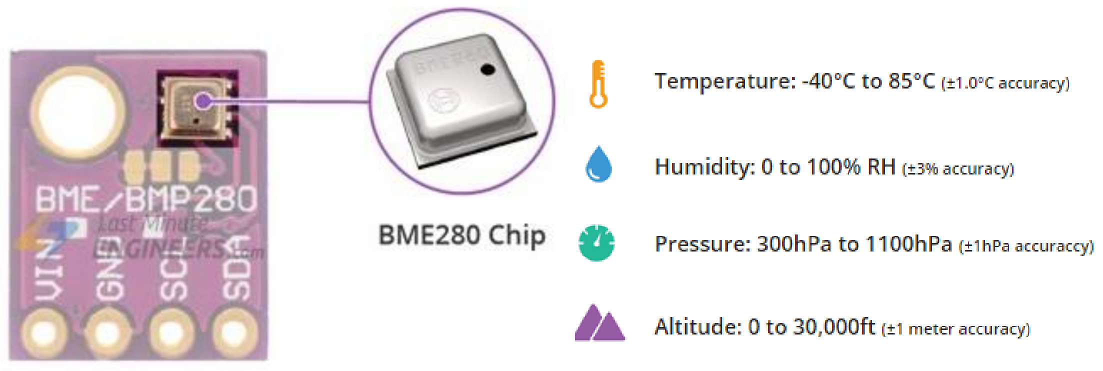

BME/BMP280
VIN
GND
SCL
SDA
BME280 Chip
Temperature: -40°C to 85°C (±1.0°C accuracy)
Humidity: 0 to 100% RH (±3% accuracy)
Pressure: 300hPa to 1100hPa (±1hPa accuraccy)
Altitude: 0 to 30,000ft (±1 meter accuracy)

Figure 15 BME280 measurement capabilities [22]

2.9.6 Thermocouple K type/MAX6675

The MAX6675 performs cold-junction compensation and digitizes the signal from a type-K thermocouple from 0 °C to 1024 °C with ± 3 °C accuracy. The MAX6675 has been discontinued and replaced with MAX31855 that can output 14 bits instead of 12 bits and measure in a wider temperature measurement range from -250 °C to 1600 °C. [23]

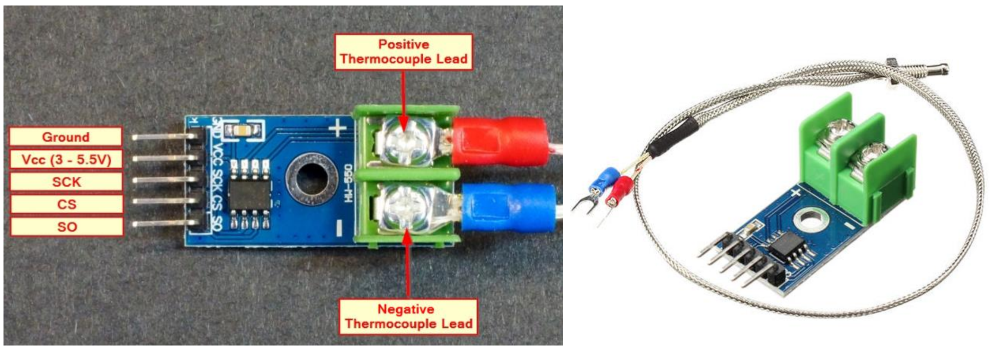


Figure 16 MAX6675 [24]

2.10 Message Queuing Telemetry Transport (MQTT)

In an IoT system, you usually want different kind of devices exchanging data between each other, the devices will request data from other devices and process the data received from the devices that responded with the demanded data. The devices can be for example Raspberry Pi, ESP32, a mobile phone running either on iOS or Android, and devices with a different programming language. What is often wanted in the IoT system is real-time data, low power consumption, and low bandwidth usage, especially in an area where the wireless network is somewhat unreliable. The MQTT protocol is a Machine to Machine (M2M) connectivity protocol, it's a lightweight messaging protocol with a broker-based publish-subscribe mechanism and runs on top of TCP/IP (Transmission Control Protocol/ Internet Protocol). [25]

The domains which MQTT can be used in are for example

- Asset tracking and management
- Home automation
- Automotive telematics
- SCADA

The protocol enables transmitting a high volume of data in small packets and from one client to many, it provides reliable delivery of data in an unreliable network and with its responsiveness makes it possible to send near real-time information.

2.10.1 Publish-Subscribe (pub-sub)

A client works as an individual device without knowing the existence of other clients, its main purpose is to publish a message and then there are the clients who are known as the subscribers who receive the message. Between the publisher and subscriber client, there is a broker (server) who establishes the connection between the publisher and subscriber. The broker filters the data received from the publishers and sends a specific type of message to the subscribers according to what is in interest. This pattern makes it possible to send filtered messages to hundreds of clients from a single publisher through the broker, which is the one responsible for sending the published data. [25]

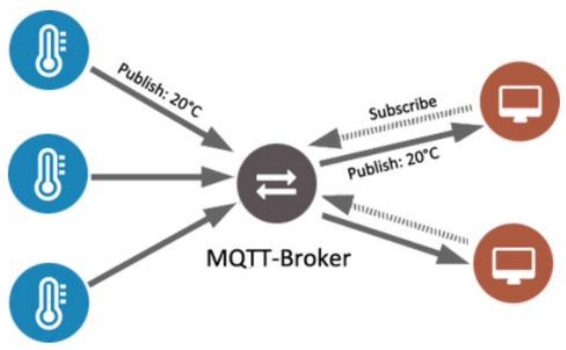


Figure 17 MQTT communication model [26]

2.10.2 Quality of service

QoS level is an agreement between a sender and receiver on the guarantee of delivering the message, it is possible to have a different level in publishing and subscribing and there are three levels, 0,1,2.

O, At most once delivery

The sender sends the message without a reply once

1, At least once delivery

Sends a confirmation that the message has been delivered at least once to the subscriber.

2, Exactly once delivery

Guarantees that the message has been delivered

3 Switchgear sensor systems

Chapter 3 goes through different sensor systems used in the market for switchgear and its technology and different brands, the focus is in Zigbee wireless network system, Radio frequency identification (RFID) system, Chameleon technology, SAW and optical fiber system.

3.1 Zigbee

Zigbee is a low-cost, low-power wireless mesh-network standard, standardized as IEEE 802.15.4, and using the 2.4 GHz frequency band. A mesh-network has each node connected and operates in a wireless personal area network (WPAN) like Bluetooth, the network adapts to the change in the network and changes path if one node were to fail as all the devices serve as routers. The two other major wireless standards are WiFi and Bluetooth and the Zigbee technology named after the dance of honeybees to communicate with other honeybees is designed to carry small amounts of data over a short distance while consuming very little power, its low transfer speed of max 250 kbps makes it most suitable for low data rate applications that needs long battery life. Zigbee has several subsets like Zigbee RF4CE (Radio Frequency for Consumer Electronics) for simpler/low power remote controlling and Zigbee Green Power for ultra-low-power wireless standard supporting energy harvesting devices.

A Zigbee network can consist of:

Zigbee Coordinator: Works as the heart of the network and has the task to approve routers and end devices attempting to join the network and there can only be one coordinator per network.

Zigbee Router: Links the network and routes the packet between nodes to extend the network and routes devices acting as the parent for end devices.

Zigbee End Device: Only achieves its task and communicates with the parent node.

3.1.1 Zigbee Green Power

The green power technology is several technologies combined [27]:

Energy harvesting: The energy of a single led flash or a solar cell can be enough to transmit radio messages every minute, the ability to harvest energy from local sources like heat or light can allow sensors and actuators for automation systems to work without costly installation with wires and batteries. [27]

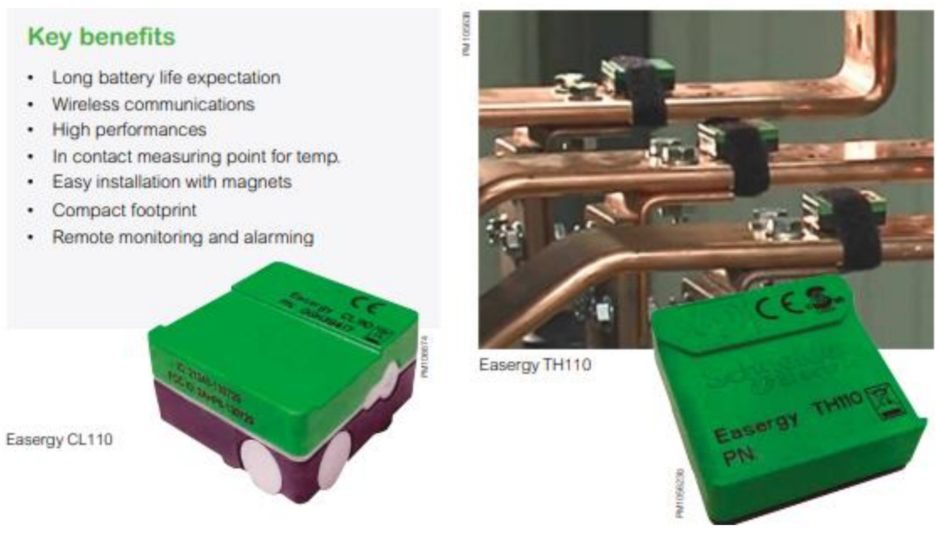
Ultra-low-power RF silicon: A high power-optimized silicon designed to run on mA with the ability to shut unnecessary functions, components, data transmission and optimizes power from a cold start. [27]

Open global standard network technology: Energy saving by reducing packet length, round trips, connection rediscovery, and on network time for devices that can be offline for some time. [27]

Open global standard application layer protocol: Uses compressed message and optimized message structure. Zigbee Green Power uses a universal language for smart objects called Dotdot. [27]

3.2 Zigbee Green Power system

TH110 temperature sensor is self-powered through energy harvesting and CL110 humidity sensor is battery-powered with both using Zigbee Green Power communication protocol. The TH110 measures from -25 °C to 115 °C (max 150 °C) with ± 1 °C accuracy and CL110 humidity from 10% to 98% and temperature of the surface in contact with from -25 °C to 90 °C with ±1 °C accuracy. With Zigbee Green Power, CL110 battery is expected to last at least 15 years.


Key benefits

- Long battery life expectation
- Wireless communications
- High performances
- In contact measuring point for temp.
- Easy installation with magnets
- Compact footprint
- Remote monitoring and alarming

Figure 18 CL110 humidity sensor and TH110 temperature sensor from Schneider [28]

3.2.1 Concentrator

The sensor devices connect with a concentrator and the concentrator can send the data to an HMI or software, platform, Schneider uses their AVEVA software on the EcoStruxure platform.

Figure 19 Harmony Hub ZBRN2, an example of a concentrator [29]

3.2.2 Mounting locations

The TH110 sensor can be installed directly on the conductive metal part or the shielded insulated part and uses a ferromagnetic ribbon around inside of TH110 for energy harvesting. CL110 sensor uses magnets to mount on the switchgear. The sensors are paired with a concentrator through holding a button on the sensors or inputting their ID number in the concentrator.

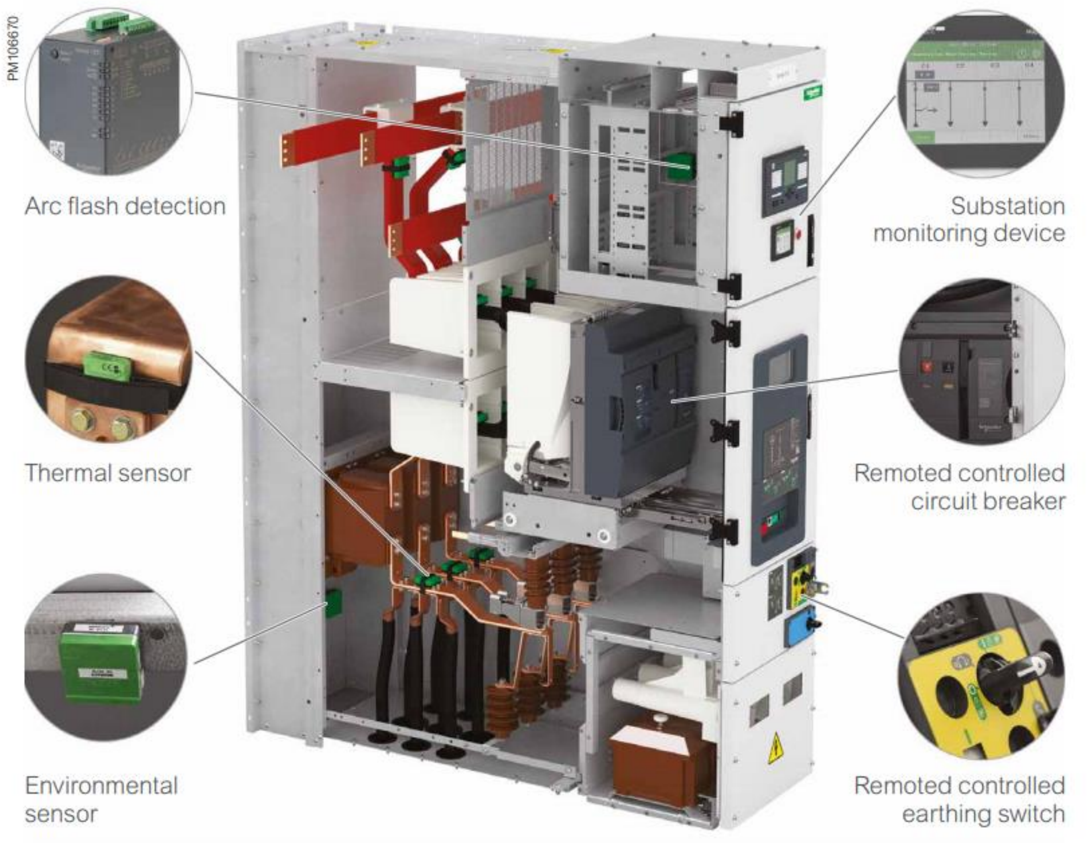

PM106670
Arc flash detection
Thermal sensor
Environmental sensor
Substation monitoring device
Remoted controlled circuit breaker
Remoted controlled earthing switch

Figure 20 Example of possible mounting locations [28]

3.3 Radio Frequency Identification (RFID)

Passive RFID systems consist of a reader (interrogator), passive tag, and a host computer. The tag's main components are an antenna coil and a microchip or integrated circuit (IC) with basic modulation circuitry and non-volatile memory. The tag harvest energy through electromagnetic radio frequency (RF) wave transmitted by the reader and the signal is called the carrier signal. RF signal passes through the antenna coil and generates AC voltage rectified to enough DC voltage for the device to become functional. The tag uses "backscattering" to reflect the information stored in the tag to the reader. Due to its low power requirement, limits the distance the tag can be read but depends on the design parameters such as frequency, RF power level, size of the antenna, data rate, current consumptions of IC, communication protocol, and readers receiving sensitivity. [30]

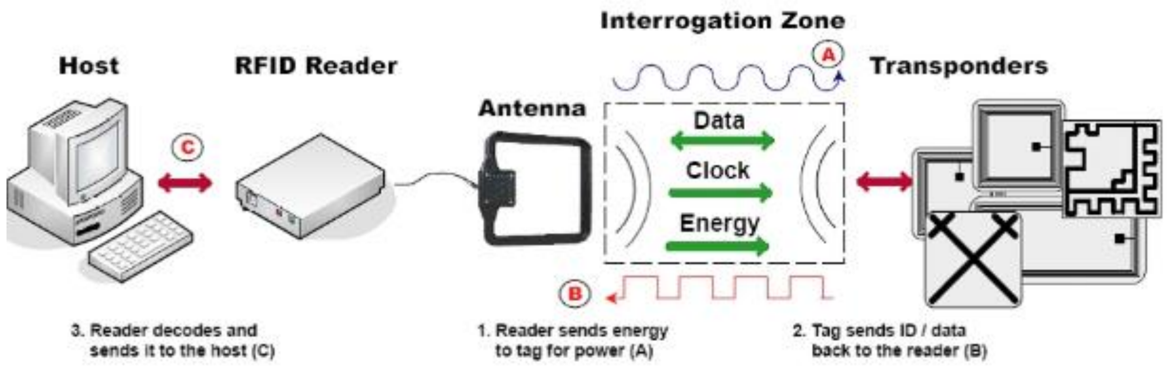

Interrogation Zone
Host
RFID Reader
Antenna
Transponders
Data
Clock
Energy
A
B
C
3. Reader decodes and sends it to the host (C)
1. Reader sends energy to tag for power (A)
2. Tag sends ID / data back to the reader (B)

Figure 21 RFID system overview [31]

There are three types of RFID tags:

Passive tags: Has no on-board power source and relies on harvesting electromagnetic energy from an RFID reader.

Active tags: Has its power source and transmitter, often larger and more expensive than passive tags.

Semi-Passive tags: Utilizes passive tag configuration and has its power source.

3.3.1 Frequencies

The frequency ranges of RFID tags can operate in are Low Frequency (LF), High Frequency (HF) and Ultra-High Frequency (UHF). RFID tags can be affixed to a variety of surfaces in different sizes and designs and in different forms like for example, labels, wristbands, stickers, and the most common frequency range used for temperature sensors are in the UHF range.

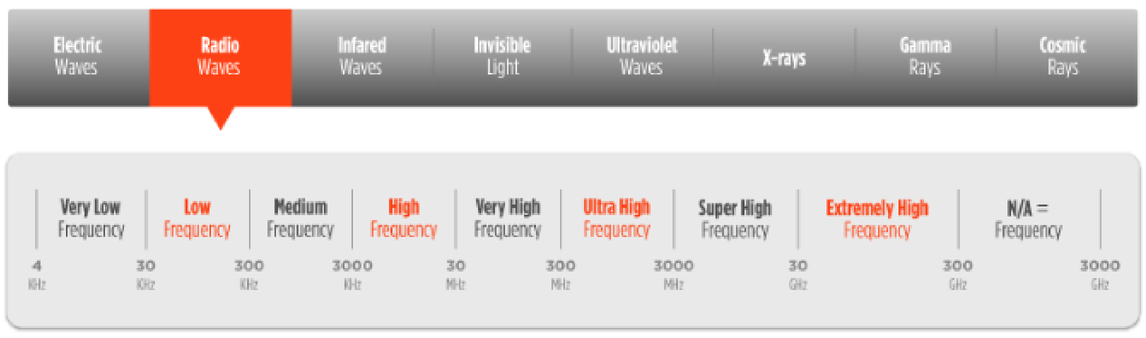


Figure 22 Frequencies in the radio frequency domain [32]

Each country has its own frequency range for UHF RFID transmission according to regulations:

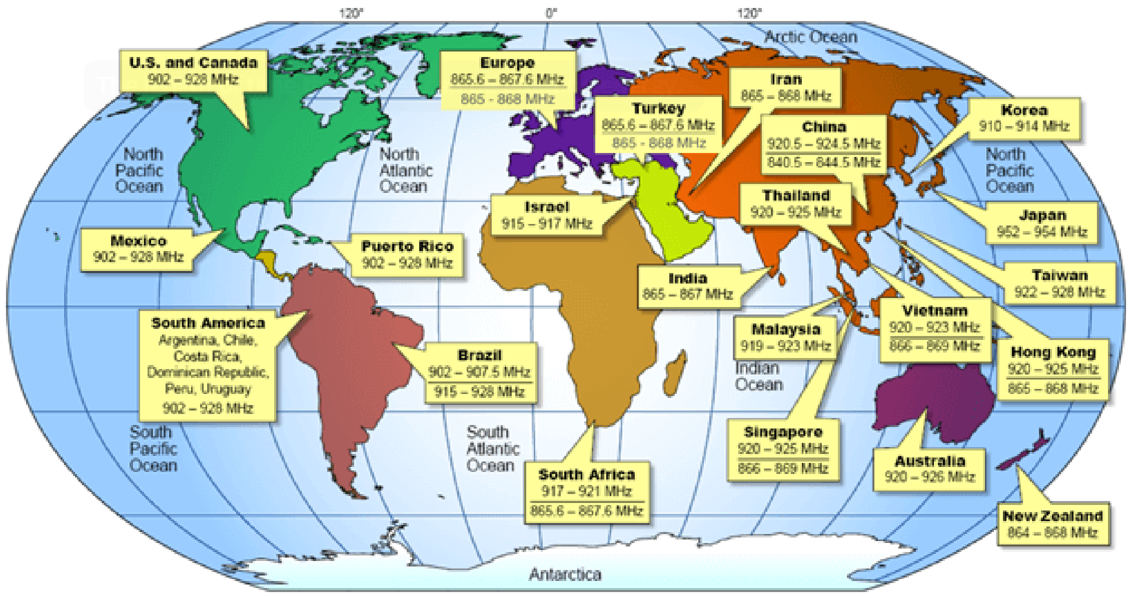

120°
0°
120°
Arctic Ocean
U.S. and Canada 902 – 928 MHz
Europe 865.6 – 867.6 MHz 865 - 868 MHz
Iran 865 – 868 MHz
Turkey 865.6 – 867.6 MHz 865 - 868 MHz
China 920.5 – 924.5 MHz 840.5 – 844.5 MHz
Korea 910 – 914 MHz
North Pacific Ocean
North Atlantic Ocean
North Pacific Ocean
Thailand 920 – 925 MHz
Japan 952 – 954 MHz
Israel 915 – 917 MHz
Mexico 902 – 928 MHz
Puerto Rico 902 – 928 MHz
Taiwan 922 – 928 MHz
India 865 – 867 MHz
Malaysia 919 – 923 MHz
Vietnam 920 – 923 MHz 866 – 869 MHz
South America Argentina, Chile, Costa Rica, Dominican Republic, Peru, Uruguay 902 – 928 MHz
Brazil 902 – 907.5 MHz 915 – 928 MHz
Hong Kong 920 – 925 MHz 865 – 868 MHz
Indian Ocean
South Pacific Ocean
South Atlantic Ocean
Singapore 920 – 925 MHz 866 – 869 MHz
Australia 920 – 926 MHz
South Africa 917 – 921 MHz 865.6 – 867.6 MHz
New Zealand 864 – 868 MHz
Antarctica

Figure 23 UHF ranges used around the world [33]

- US/FCC 902-928 MHz
- EU/ETSI 865-868 MHz
- Global 860-960 MHz

3.4 Axzon (formerly RFMicron) Chameleon technology

RFMicrons Magnus (IC) uses Chameleon technology to adjust and tune the internal variable capacitance of the sensor to match the impedance of the antenna to the reader signal that can detune from environment changes, fixed impedance reduces the performance of the tag. The battery-free wireless sensor works as an RFID tag that consists of an antenna and a Magnus-S sensor die, the Magnus-s die has a bank of capacitors of 32 capacitance states represented by a 5 -bit sensor code for the tuning setting. With the sensor code enables it to become a wireless passive sensor, from receiving a signal from the reader starts the data collection process that uses some of the energy to power the Chameleon engine to align itself to the RF readers same frequency, by then the sensor decodes the command embedded in the RF reader signal to execute the command. [34]

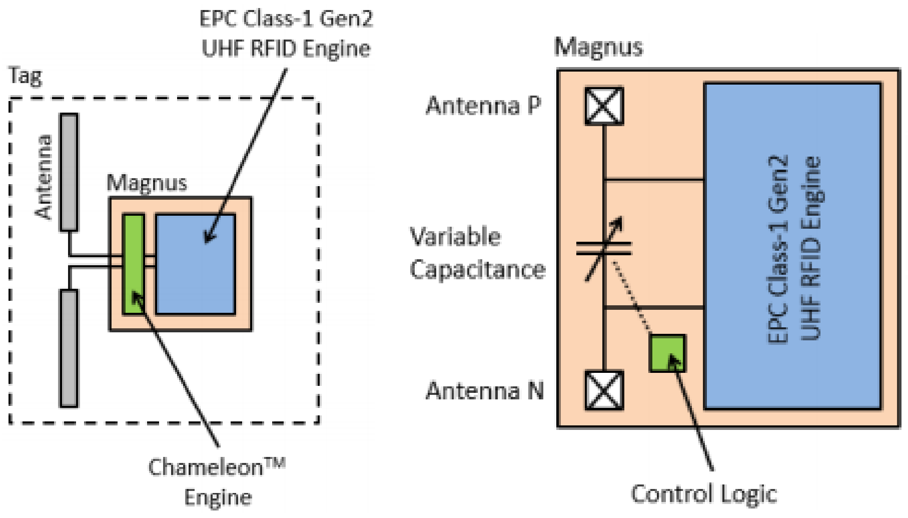
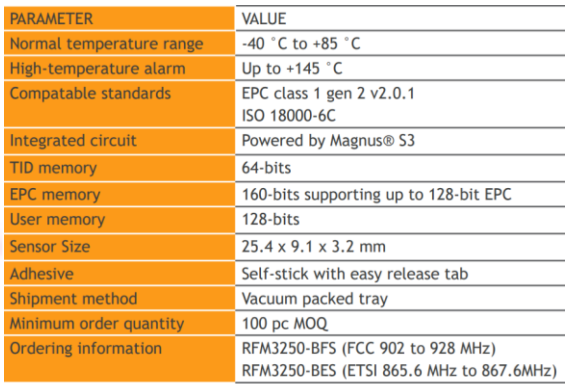


Figure 24 Magnus S sensor design overview [35]

3.4.1 Axzon sensor system

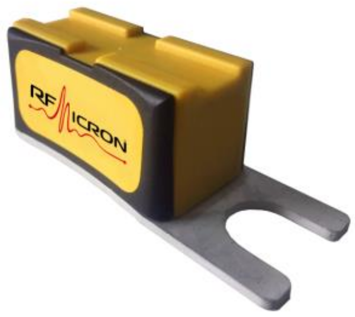
The biggest difference between the sensors is their reading range and form which can make it more suitable for different locations, accuracy for all the sensors starts to decrease when the temperature value increase over the normal temperature range.

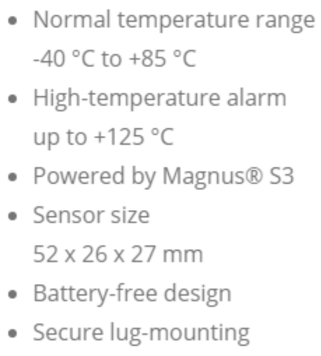


PARAMETER	VALUE
Normal temperature range	-40 °C to +85 °C
High-temperature alarm	Up to +145 °C
Compatable standards	EPC class 1 gen 2 v2.0.1 ISO 18000-6C
Integrated circuit	Powered by Magnus® S3
TID memory	64-bits
EPC memory	160-bits supporting up to 128-bit EPC
User memory	128-bits
Sensor Size	25.4 x 9.1 x 3.2 mm
Adhesive	Self-stick with easy release tab
Shipment method	Vacuum packed tray
Minimum order quantity	100 pc MOQ
Ordering information	RFM3250-BFS (FCC 902 to 928 MHz) RFM3250-BES (ETSI 865.6 MHz to 867.6MHz)

Figure 25 Axzon RFM3250 sensor device [36]

The RFM3250 has the measurement range of -40 °C to 145 °C with an accuracy of ± 2, it harvests energy through integrated RF antenna and is suitable on metal surfaces. The device is a rugged ceramic sensor with an adhesive backing to mount it with the max reading range of 6 m.

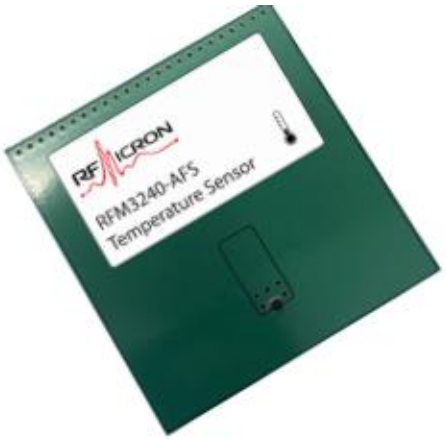


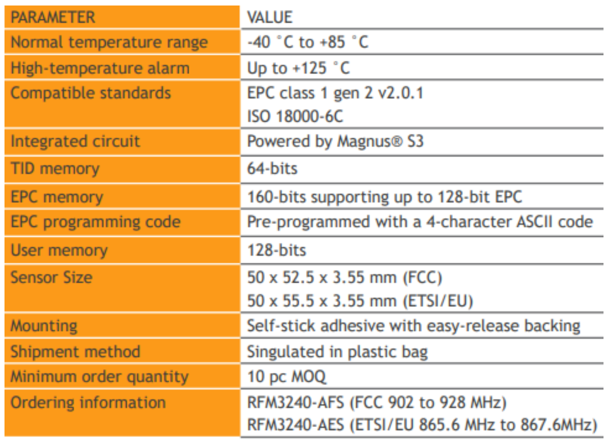


- Normal temperature range
  -40 °C to +85 °C
- High-temperature alarm
  up to +125 °C
- Powered by Magnus® S3
- Sensor size
  52 x 26 x 27 mm
- Battery-free design
- Secure lug-mounting

Figure 26 Axzon RFM3260 lug-mounted sensor device [36]

The RFM3260 has the measurement range of -40 ℃ to 125 ℃ with an accuracy of ± 2 mounted using a nut and bolt or strap with the max reading range of 9 m.

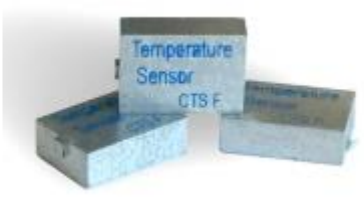


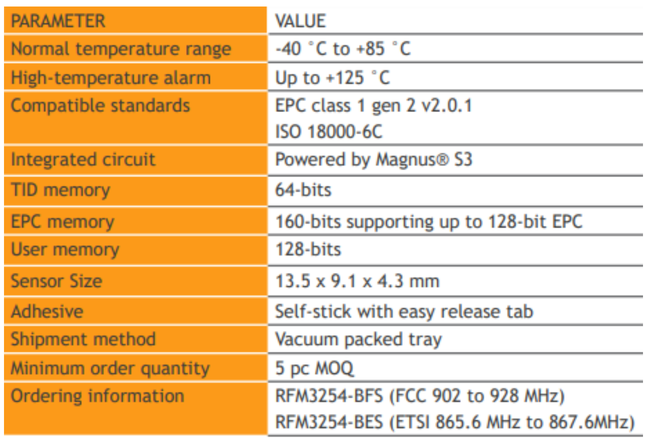


PARAMETER	VALUE
Normal temperature range	-40 °C to +85 °C
High-temperature alarm	Up to +125 °C
Compatible standards	EPC class 1 gen 2 v2.0.1 ISO 18000-6C
Integrated circuit	Powered by Magnus® S3
TID memory	64-bits
EPC memory	160-bits supporting up to 128-bit EPC
EPC programming code	Pre-programmed with a 4-character ASCII code
User memory	128-bits
Sensor Size	50 x 52.5 x 3.55 mm (FCC) 50 x 55.5 x 3.55 mm (ETSI/EU)
Mounting	Self-stick adhesive with easy-release backing
Shipment method	Singulated in plastic bag
Minimum order quantity	10 pc MOQ
Ordering information	RFM3240-AFS (FCC 902 to 928 MHz) RFM3240-AES (ETSI/EU 865.6 MHz to 867.6MHz)

Figure 27 Axzon RFM3240 sensor device [36]

The RFM3240 has the measurement range of -40 ℃ to 125 ℃ with an accuracy of ± 2 and uses adhesive to mount with the max reading range of 18 m.





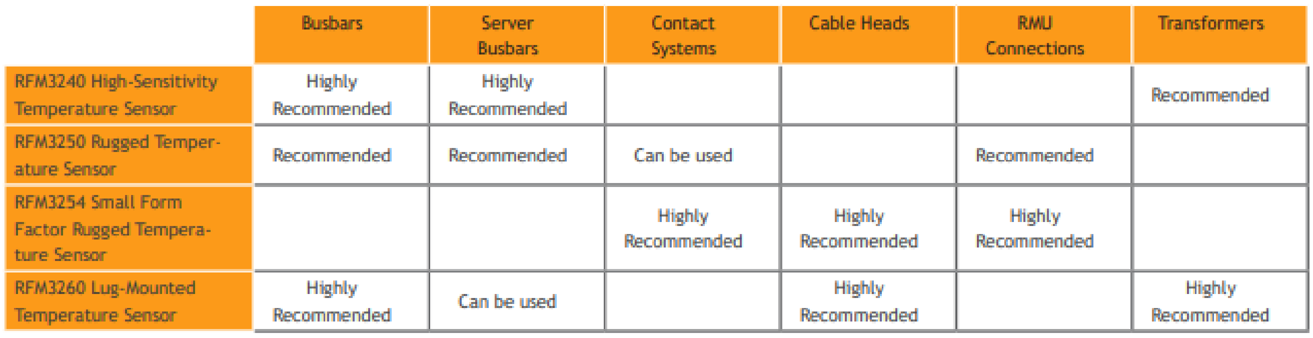
PARAMETER	VALUE
Normal temperature range	-40 °C to +85 °C
High-temperature alarm	Up to +125 °C
Compatible standards	EPC class 1 gen 2 v2.0.1 ISO 18000-6C
Integrated circuit	Powered by Magnus® S3
TID memory	64-bits
EPC memory	160-bits supporting up to 128-bit EPC
User memory	128-bits
Sensor Size	13.5 x 9.1 x 4.3 mm
Adhesive	Self-stick with easy release tab
Shipment method	Vacuum packed tray
Minimum order quantity	5 pc MOQ
Ordering information	RFM3254-BFS (FCC 902 to 928 MHz) RFM3254-BES (ETSI 865.6 MHz to 867.6MHz)

Figure 28 Axzon RFM3254 [36]

The RFM3254 has the measurement range of -40 ℃ to 125 ℃ with an accuracy of ± 2 mounted using adhesive backing with the max reading range of 5 m.

3.4.2 Mounting locations

The sensors are recommended for different locations (see figure 29).



	Busbars	Server Busbars	Contact Systems	Cable Heads	RMU Connections	Transformers
RFM3240 High-Sensitivity Temperature Sensor	Highly Recommended	Highly Recommended				Recommended
RFM3250 Rugged Temperature Sensor	Recommended	Recommended	Can be used		Recommended	
RFM3254 Small Form Factor Rugged Temperature Sensor			Highly Recommended	Highly Recommended	Highly Recommended	
RFM3260 Lug-Mounted Temperature Sensor	Highly Recommended	Can be used		Highly Recommended		Highly Recommended

Figure 29 Table on recommended locations to mount the sensors [36]

Possible to insert the RFM3250 sensor into a flexible band for mounting.

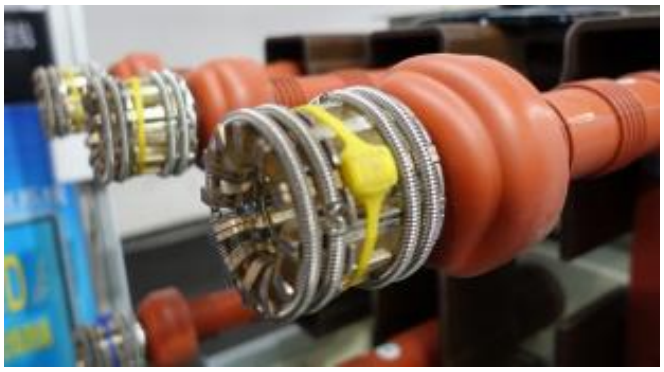


Figure 30 RFM3250 mounted using a flexible band [36]

The antenna should be placed roughly on the level of the sensors and the reader in a suitable place such as the upper chamber of the switchgear depending on the cables needed to route for the antenna.

Figure 31 Example of an antenna and reader [36]

3.5 Surface-acoustic waves (SAW) RFID

Instead of using semiconductor physics, this operation uses micro-acoustics of piezoelectric crystals instead.

Surface acoustic wave (SAW) temperature sensor rely on signal modulation taking advantage of the piezoelectric effect. The sensor consists of a piezoelectric substrate which is a material that changes electrical charges due to mechanical stresses and commonly uses quartz or lithium niobite for temperature sensing, a high-temperature coefficient is more effective for its sensitivity to temperature change. The sensor can have two interdigitated transducer resonator (IDT) and antennas and in the IDT resonator are interlocking comb-like metallic electrodes deposited on the surface of the substrate and space between these electrodes determines the frequency of the acoustic wave, the physical temperature change on the crystal alters the IDT electrodes spacing that changes the frequency of the surface acoustic wave. [37] [38]

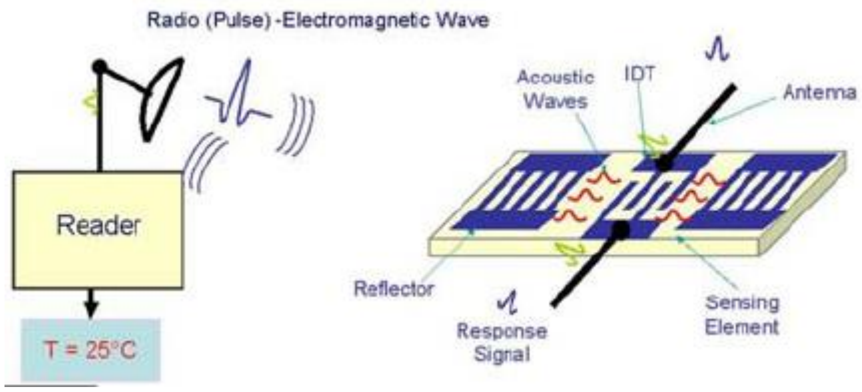
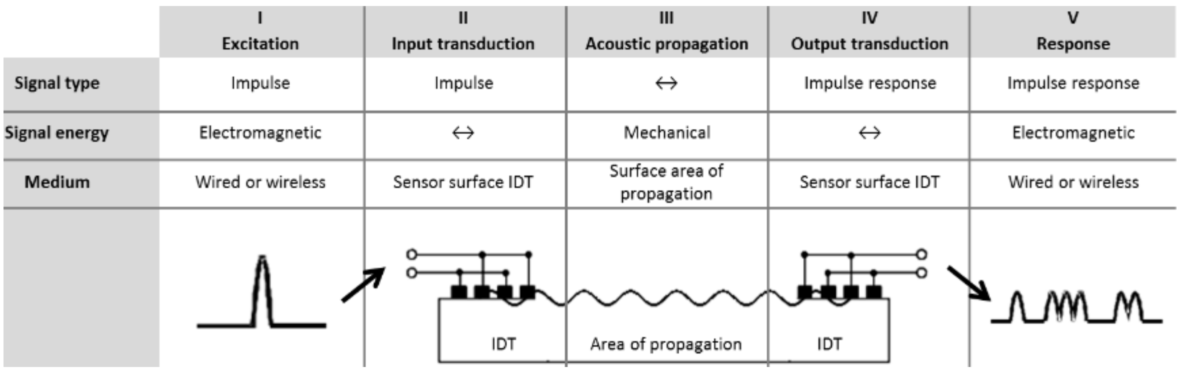


Figure 32 Surface Acoustic Wave technology overview [39]

The operating principle starts with an electromagnetic signal as the input from an interrogation unit that transduces the signal into a surface acoustic signal by the IDT resonator, a mechanical wave on the surface of the substrate is generated and the acoustic surface wave is reverted to an electrical signal by the IDT that is sent back to the interrogation unit for temperature measurement. The wave senses the environment for changes in the phase, amplitude, and/or frequency to a reference. [40]



	I Excitation	II Input transduction	III Acoustic propagation	IV Output transduction	V Response
Signal type	Impulse	Impulse	↔	Impulse response	Impulse response
Signal energy	Electromagnetic	↔	Mechanical	↔	Electromagnetic
Medium	Wired or wireless	Sensor surface IDT	Surface area of propagation	Sensor surface IDT	Wired or wireless

IDT
Area of propagation
IDT

Figure 33 Saw technology signal overview [40]

3.6 ABB SAW system

The sensor receives an electromagnetic signal from a transceiver and converts the signal through IDT resonator into a surface acoustic wave to thereafter echo the modified signal back to the transceiver, the transceiver computes and translate it into exploitable data. The ABB SAW sensor can measure from -25 ℃ to 150 ℃

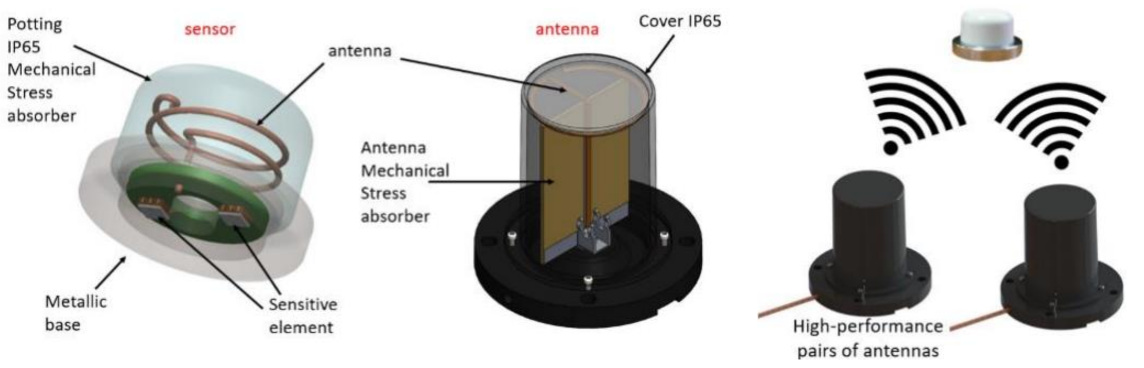


Figure 34 ABB SAW sensor design overview [41]

3.6.1 Mounting locations

The antennas have to be mounted on a suitable place to be able to read the sensors. ABB uses MyRemoteCare platform that enables remote monitoring and different analytics such as algorithms calculation the probability of failure and remaining useful life, circuit breaker and switchgear health condition, and dashboard for visualization.

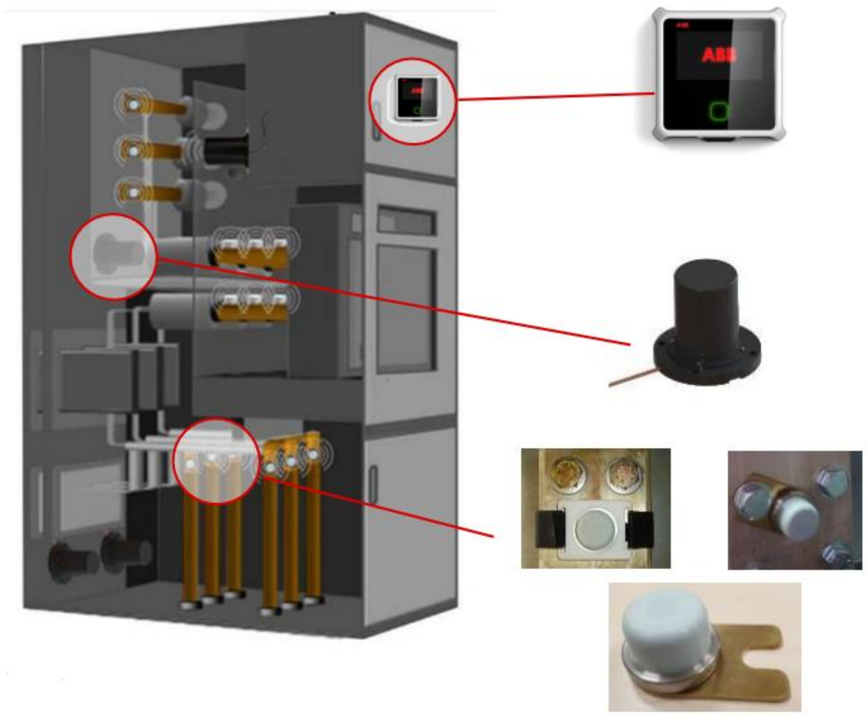


Figure 35 Example of mounting areas and positions [41]

3.7 Intellisaw SAW system

Communicates with a transceiver and can measure from -25°C to 125°C with ± 2°C accuracy at 0°C to 80°C and ± 4°C accuracy at full range and is mounted with bolts or alternative heat resistant tie wraps or heat resistant tape. The cover material is made of Polycarbonate, UL94-HB with brass tin-plated base plate. With 15KV dielectric cover strength.

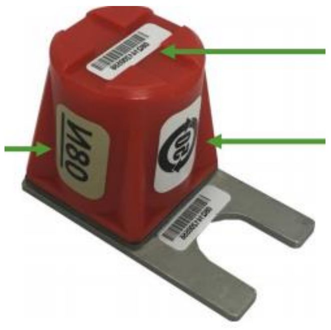


Figure 36 Intellisaw temperature sensor [42]

3.7.1 Mounting locations

There are different possible mounting positions according to switchgear type IEC and ANSI, the antennas have to be 45 degrees mounted from the sensor or straight toward the sensor.

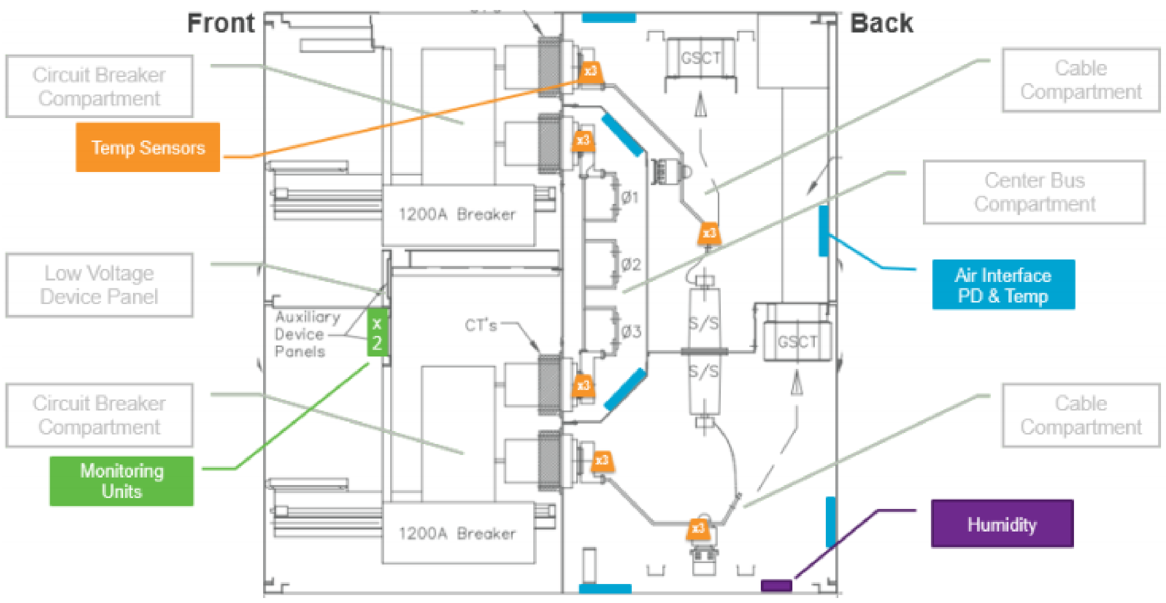


Figure 37 IEC type switchgear mounting positions [42]

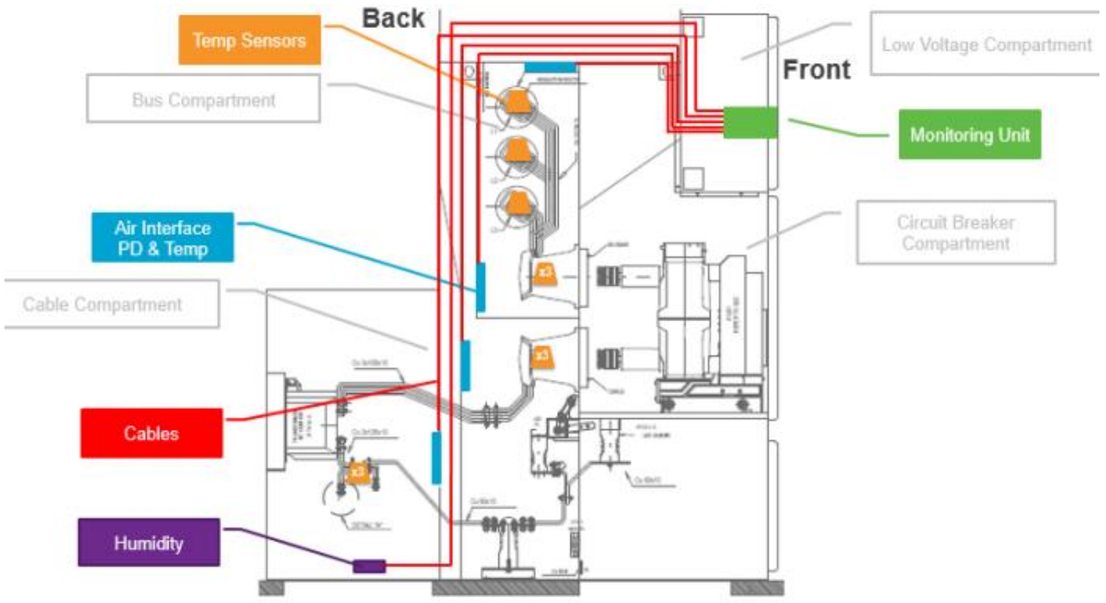


Figure 38 ANSI type switchgear mounting positions [42]

3.8 Optical Fiber

Optical fibers are immune to electromagnetic interference, require no electrical power at the remote location, small in size, low transmission loss, large bandwidth, and low cost make it suitable not only for telecommunication applications but also for other various applications such as laser, sensing and medical sciences.

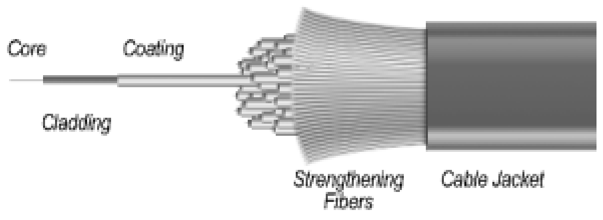


Figure 39 Fiber-optic cable overview [43]

A fiber-optic cable is composed of three parts, the core, the cladding, and the coating as a cylindrical rod. The light propagates along with the core and the cladding has a lower refractive index to confine the light guiding in the core decreasing the scattering loss and absorption of environmental contaminants. For resistance, a coating layer is used to protect the optical fiber from physical damage and abrasions. [43]

Optical fibers can be categorized as either intrinsic where the sensing mechanism operates within an element of the optical fiber or extrinsic where the optical fiber is only used as an information carrier, coupling the optical signal to and from to be monitored. In operating principle or modulation/demodulation, a fiber-optic sensor can be classified as an intensity, a phase, a frequency or a polarization sensor. In application, the fiber-optic sensor is classified as a physical sensor, a chemical sensor, or a biomedical sensor. [44]

3.8.1 Fluorescence sensor

Fluorescence-based fiber sensors are developed by doping fibers with materials of luminescent properties and is an example of a wavelength modulated sensor with black body sensors and the Bragg grating sensor in the same category. There exists a wide range of luminescent doped optical fiber that has been developed and elements such as rare earth is used as dopant mostly into silica-based fibers. There are different fluorescent sensor configurations with the fiber end tip sensor as one of the most common configurations.

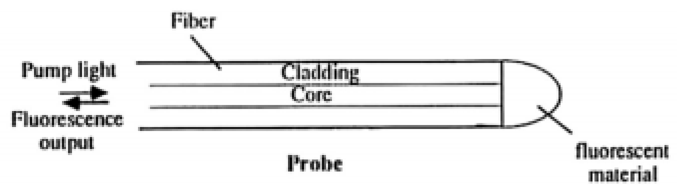


Figure 40 Fluorescent sensor overview [45]

The fiber end tip configuration propagates the optical signal in the optical fiber into a probe of fluorescent material, the fluorescent signal is then coupled into the same fiber and guided to an optical signal analyzer. [46]

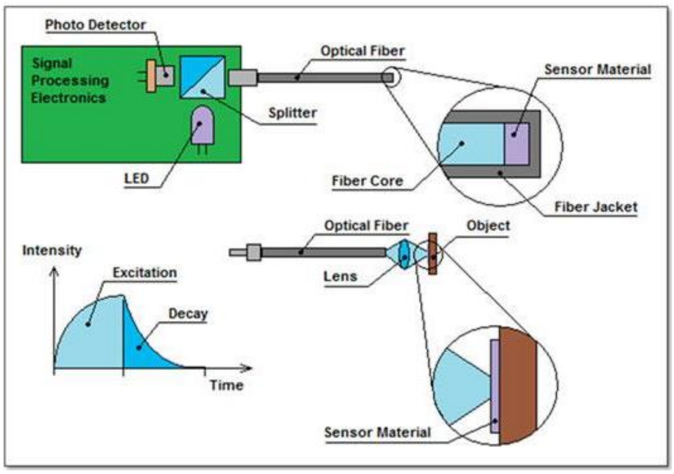


Figure 41 Schematic overview over the use of a fluorescence-based sensor by Osensa [47]

3.9 Osensa optical fiber system

FTX-910-PWR+R is a fiber-optic temperature transmitter device mountable on 35 mm DIN-rail. It is powered by 12-24 V DC and has 9 optical fiber sensor inputs, two programmable 2 A relays for over-temperature alarms and an isolated RS-485 connectivity to Modbus RTU protocol. It is possible to connect multiple transmitters in series supplied by a five-pin T-bus connector. There are also other versions such as FTX-910-PWR+R, FTX-610-PWR+, FTX-310-PWR+, and HTX-110-PWR+.

Figure 42 FTX-910-PWR+R transmitter produced by OSENSA innovations [48]

3.9.1 Temperature probes

The PRB-110 probe can measure from -40 to 120 ℃ with ± 1 accuracy and the PRB-910 from -40 to 200 ℃ with ± 1 accuracy with the probe length of 10 m that can be cut shorter. They can be installed on switchgear contacts, busbars, cast resin transformers, motors, and generators. Both probes are constructed from durable and high dielectric strength materials and can operate on equipment rated up to 38 kV 3 phase with different optional probe tip attachments according to application. The PRB-110 probe uses nylon as cable material and the PRB-910 also nylon but in the last 1 meter of the cable FEP (Fluorinated ethylene propylene) material.

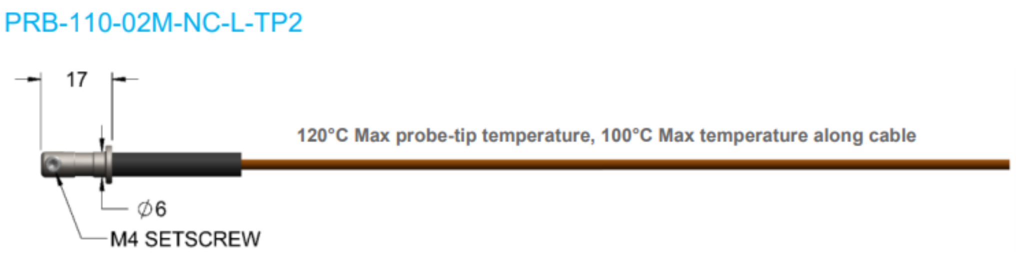

PRB-110-02M-NC-L-TP2
17
120°C Max probe-tip temperature, 100°C Max temperature along cable
∅6
M4 SETSCREW

Figure 43 PRB-110 fiber-optic temperature probe [48]

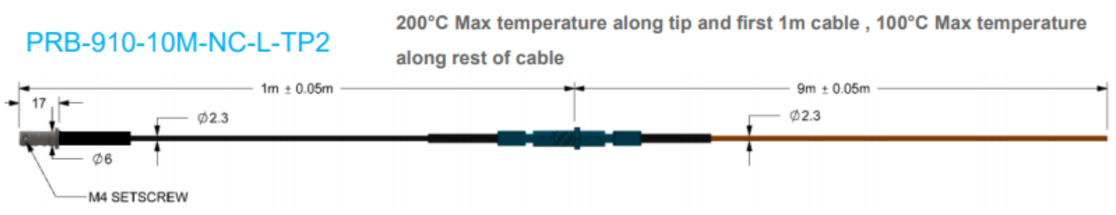

PRB-910-10M-NC-L-TP2
200°C Max temperature along tip and first 1m cable , 100°C Max temperature along rest of cable
17
1m ± 0.05m
9m ± 0.05m
Ø2.3
Ø2.3
Ø6
M4 SETSCREW

Figure 44 PRB-910 fiber-optic temperature probe [48]

It is possible to connect different ring lugs sizes to the probes.

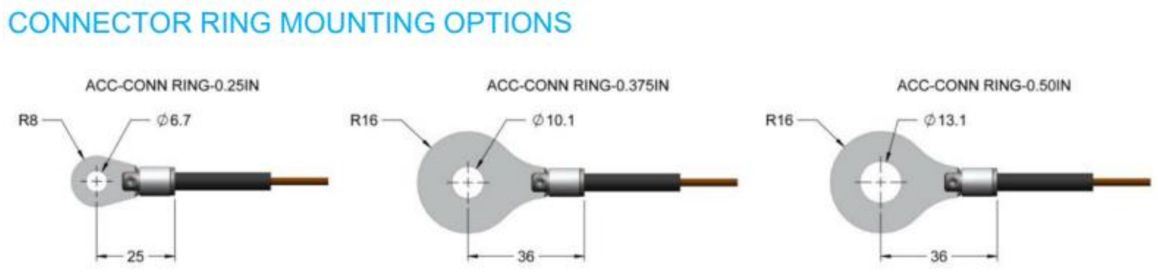


Figure 45 Rings lugs in 1/4, 3/8, and 1/2 sizes can be fastened to the end of the probes [48]

The probes can be bolted to the desired location using a nut and bolt.

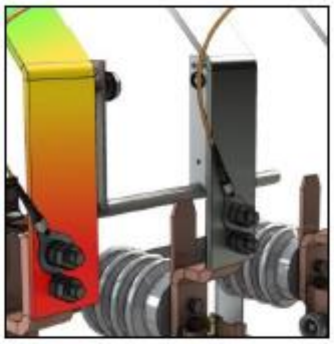


Figure 46 Mounting position example [48]

4 ESP32 and Raspberry Pi prototype system

Chapter 4 is about the practical process to get the microcontrollers (Raspberry Pi, ESP32) working and connected to the cloud using I²C and SPI communication interface, IBM cloud services, Node-Red, MQTT communication protocol and the sensors BME280/MAX6675. BME280 sensor was chosen for its temperature reliability and MAX6675 thermocouple module for its fast response and high temperature measurement range, IBM Cloud was the most suitable with the best compatibility and had a more user-friendly interface for the task I tried to do than Microsoft azure which was first tested with.

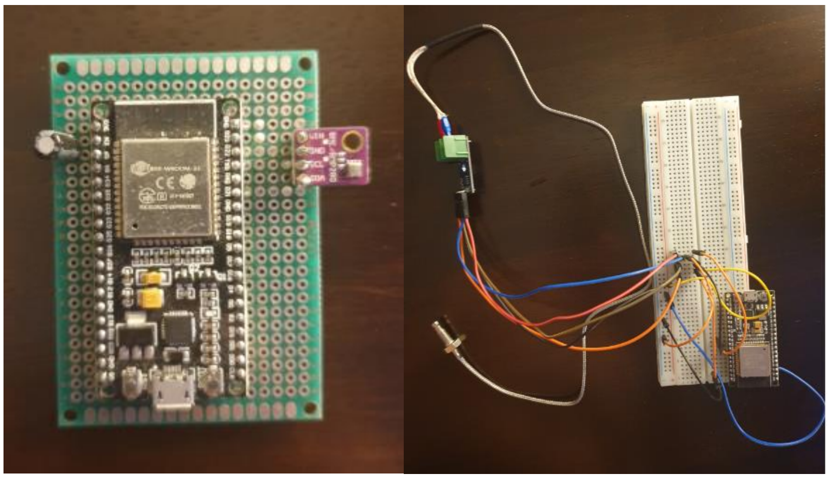


Figure 47 Prototype ESP32-1 BME280 and ESP32-2 MAX6675

4.1 Hardware and software

Components:

- Two ESP32 WROOM-32 model
- Raspberry pi 3+
- Bosch BME-280 temperature/humidity sensor
- MAX 66750 thermocouple K-type temperature sensor
- Two 10 uF capacitor

Software:

- Arduino IDE
- Node-Red
- IBM Cloud, Cloudant, IBM IoT Platform
- Putty

4.1.1 ESP32 Code overview

Code functions:

- Access to the WiFi
- Access to MQTT server
- Over the Air Programming (OTA)
- Publish to Node-Red as client

The code written and uploaded into ESP32 consists of Over the Air (OTA), MQTT connection to the Raspberry Pi, and connection to the local WiFi. OTA enables the upload of a new update to the hardware over the local WiFi. Connecting the device to MQTT server IP given by the Raspberry Pi enables the ESP32 to work as a client. As the client, the ESP32 will be publishing sensor data to Node-Red hosted by Raspberry Pi which can be accessed from the host local IP address. ESP32-1 reads the temperature and humidity value from the BME280 sensor connected through I^2C and ESP32-2 reads MAX6675 thermocouple through SPI communication interface.

The pins for the communication interface is defined in the code which varies according to ESP32 version and both ESP32 publishes to different topics, in this project, the temperature value received from the thermocouple publishes to esp32/MAX6675 and temperature from BME280 to esp32/BME280Temp and humidity to esp32/BME280Humid, there can be additional topic levels by adding / after the topic.

The libraries required for ESP32-1 are:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_BME280.h>
#include <Adafruit_Sensor.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
```

and ESP32- 2:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <max6675.h>
```

To be able to bypass the need to press EN button while uploading code to ESP32, a 10uF capacitator is connected, negative to GND and positive to the EN pin. The whole code for ESP32-1 and ESP32-2 can be found in appendix 3 and 4.

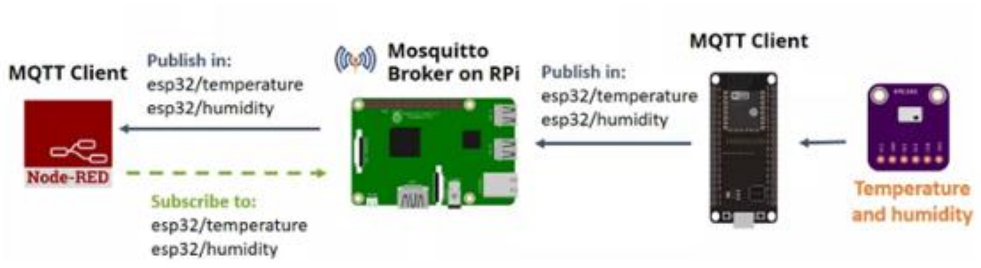


Figure 48 Overview over the whole process [49]

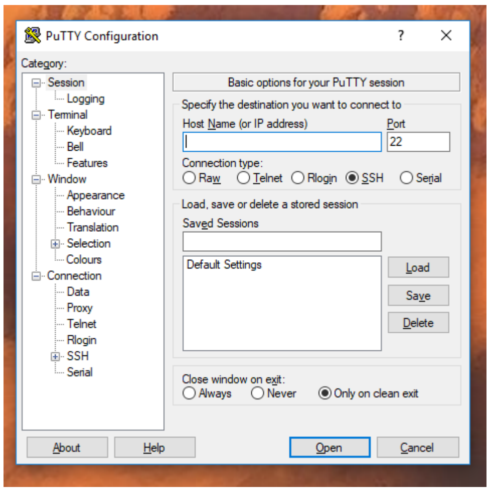
4.1.2 Raspberry Pi

For the raspberry to work as the broker, Mosquitto is installed to process the MQTT lightweight messaging protocol.

```
pi@raspberry:~ $ sudo apt update
pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-clients
pi@raspberry:~ $ sudo systemctl enable mosquito.service
```

First command updates, second row installs mosquitto, and third row auto starts the program on boot

Node-Red should already exist in the library in the latest Raspberry Pi if not, it will have to be downloaded. The local IP can be checked by entering **hostname – I** in the terminal window which is in this project **192.168.1.143**.



Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Figure 49 Putty

For remote access, Secure Shell (SSH) software can be used such as Putty by enabling the SSH interface on the raspberry and connecting with the local IP address.

4.1.3 Pin connections

Temperature sensor BME280 which can communicate to either SPI or I^{2}C communication interface is connected through I^{2}C by connecting the pins:

- GND to GND
- D22 to SCK
- D21 to SDI
- 3V3 to VCC

The temperature sensor MAX 6675 pins are connected to SPI connection by connecting the pins:

- D19 to SO
- D23 to CS

- D5 to CLK
- GND to GND
- 3V3 to VCCS

4.1.4 Raspberry Node-Red connection

To access Node-Red on the raspberry, type in your local IP address and:1880 which is in my case **192.168.1.143:1880**. For a "flow" to work, it always needs an input and output node, in between the nodes could have functions to check the data for resolving errors or change the data format, there are many nodes that can be used.

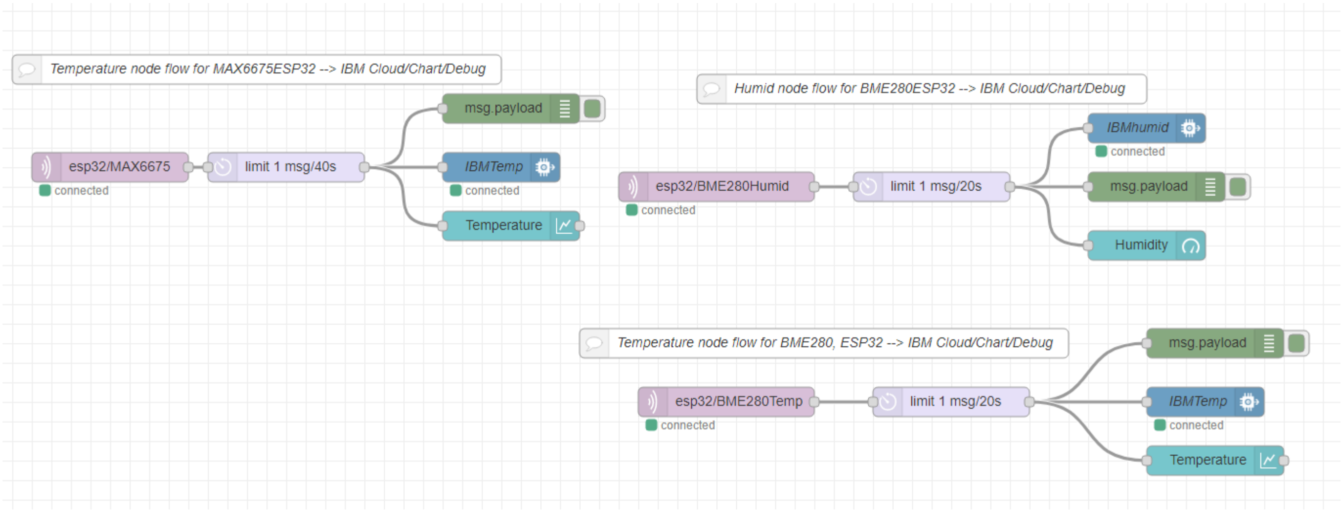
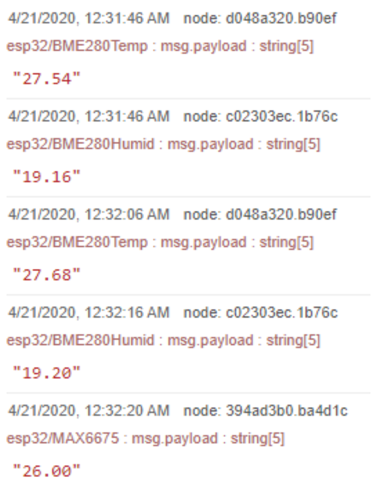


Figure 50 Raspberry Pi Node-Red configuration

ESP32-2 code for the MAX6675 module is set to publish its value to the esp32/MAX6675 topic and ESP32-1 to esp32/BME280Humid, esp32/BME280Temp, from there the message is set to be delayed by 40s per message and deletes all the messages occurred in-between. The message is lastly sent to the IBM Cloud platform, a graph for graphical visualization on the local network and to a debug node to see its content. The IBM cloud node needs the right credentials to send to which is created in the IBM IoT service on IBM Cloud, the information that is needed is the device type, device id, and Authentication token.



```
4/21/2020, 12:31:46 AM   node: d048a320.b90ef
esp32/BME280Temp : msg.payload : string[5]
 "27.54"
4/21/2020, 12:31:46 AM   node: c02303ec.1b76c
esp32/BME280Humid : msg.payload : string[5]
 "19.16"
4/21/2020, 12:32:06 AM   node: d048a320.b90ef
esp32/BME280Temp : msg.payload : string[5]
 "27.68"
4/21/2020, 12:32:16 AM   node: c02303ec.1b76c
esp32/BME280Humid : msg.payload : string[5]
 "19.20"
4/21/2020, 12:32:20 AM   node: 394ad3b0.ba4d1c
esp32/MAX6675 : msg.payload : string[5]
 "26.00"
```

Figure 51 Message received visible from using the debug node

To use the nodes that do not come with the core library on Node-Red, the nodes IBM Watson IoT and dashboard node has to be downloaded from the Node-Red library to be able to send data to IBM IoT platform and to visualize the data received in a local network dashboard from the ESP32 that can be accessed by typing **http//your IP address:1880/ui**

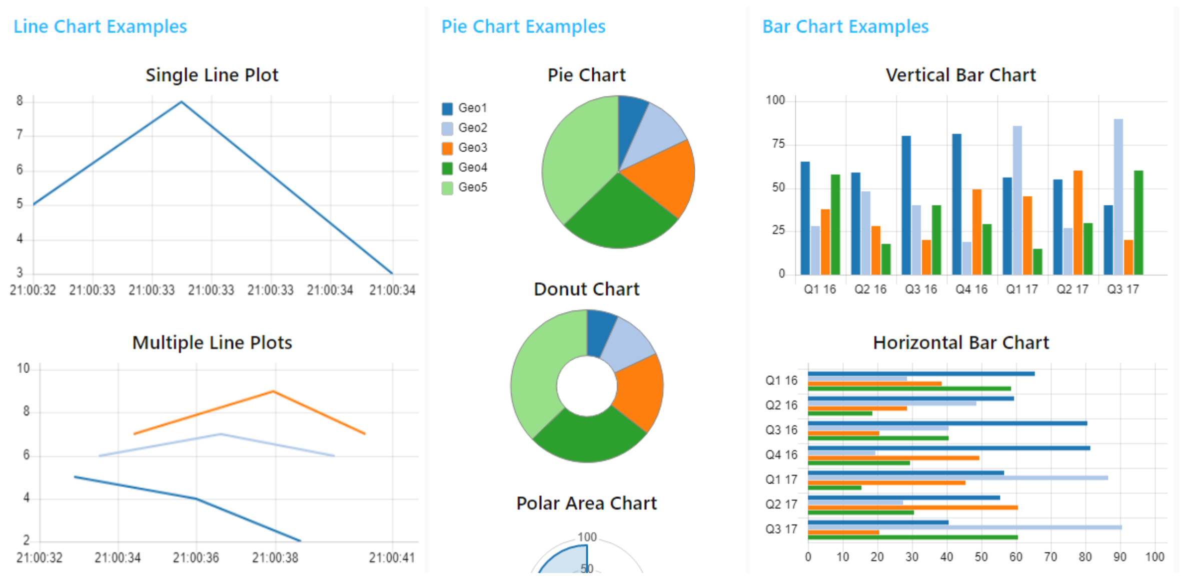
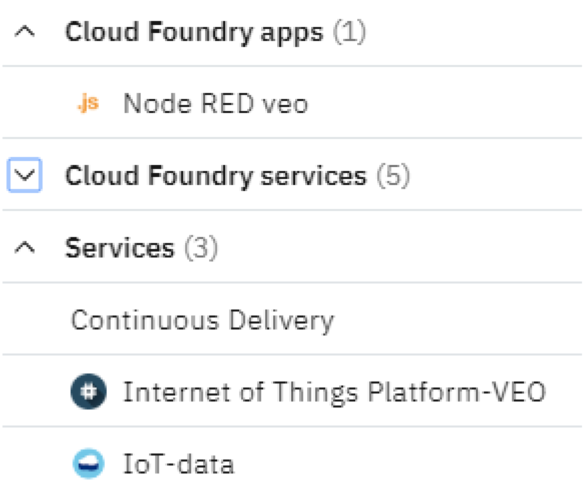


Figure 52 Different chart options on the dashboard

4.1.5 IBM cloud services

In IBM Cloud, three services are created, Node-Red to connect the data from the Raspberry Pi broker to the IBM cloud Node-Red, Internet of Things Platform to connect the devices to IBM Cloud and Cloudant for data storage.



Cloud Foundry apps (1)

Node RED veo

Cloud Foundry services (5)

Services (3)

Continuous Delivery

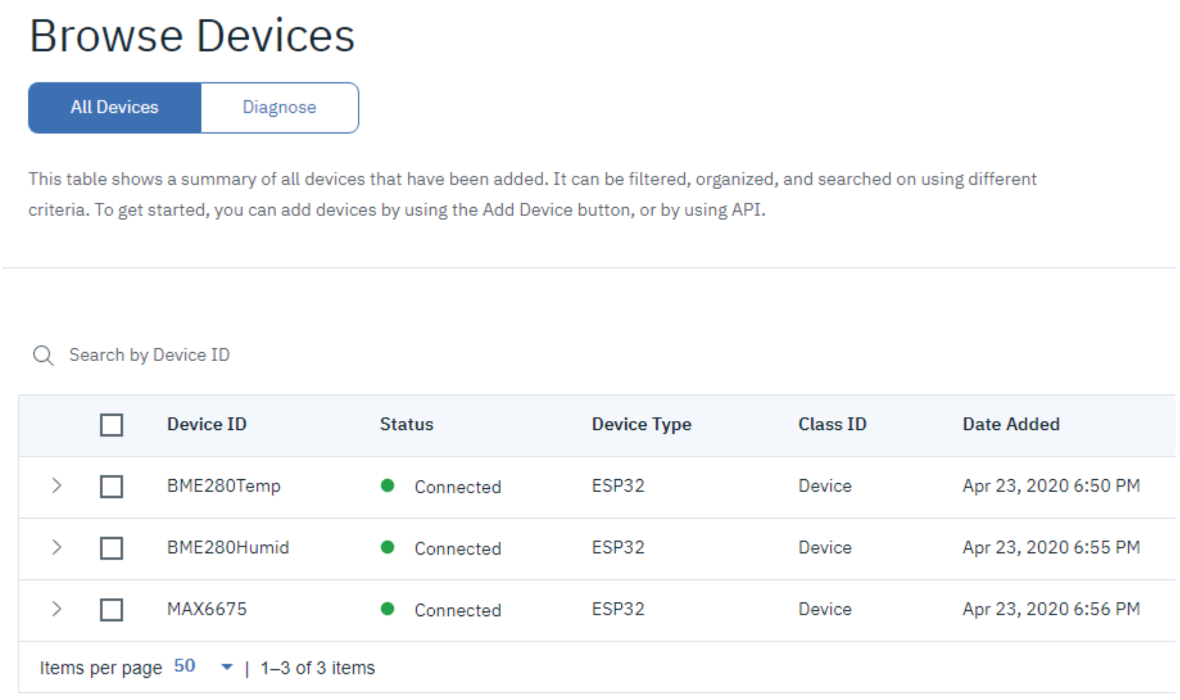
Internet of Things Platform-VEO

IoT-data

Figure 53 Services created in IBM cloud

4.1.6 IBM Watson IoT Platform

Three devices are added to the IoT platform with different values by giving each device a Device ID and an authentication token is provided which can be self-provided or auto-generated that consists of 18 characters of mixed alphanumeric, characters and symbols or between 8-36 self-generated. Two devices were added for ESP-1 for the separate humidity and temperature value.



Browse Devices

All Devices | Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

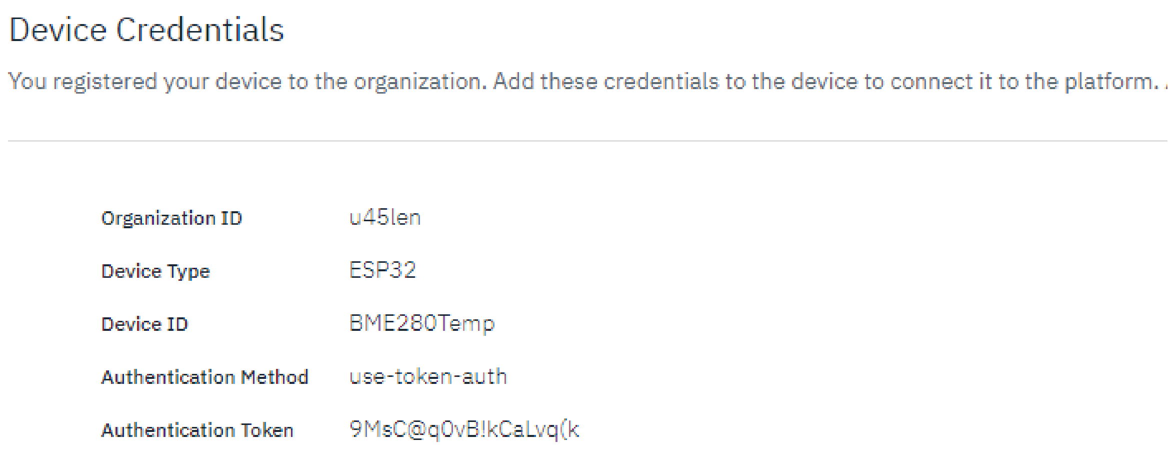
Search by Device ID

Device ID	Status	Device Type	Class ID	Date Added
BME280Temp	● Connected	ESP32	Device	Apr 23, 2020 6:50 PM
BME280Humid	● Connected	ESP32	Device	Apr 23, 2020 6:55 PM
MAX6675	● Connected	ESP32	Device	Apr 23, 2020 6:56 PM

Items per page 50 | 1–3 of 3 items

Figure 54 Devices added to IBM Watson IoT

Creation of a new device is required if the authentication token is lost so it should be stored somewhere safe.



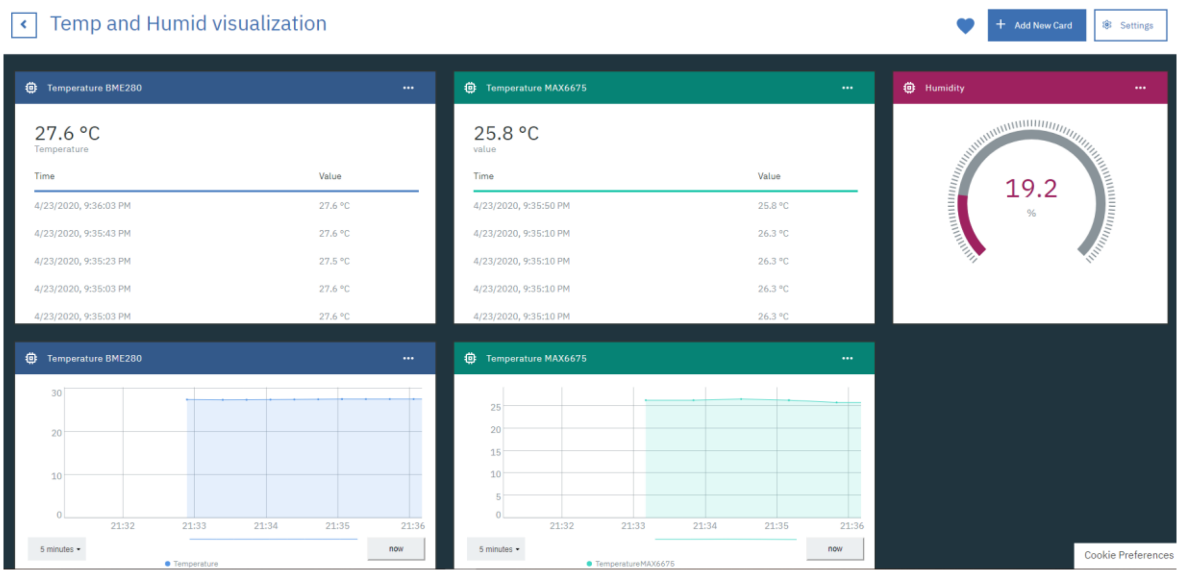
Device Credentials

You registered your device to the organization. Add these credentials to the device to connect it to the platform.

Organization ID	u45len
Device Type	ESP32
Device ID	BME280Temp
Authentication Method	use-token-auth
Authentication Token	9MsC@q0vB!kCaLvq(k

Figure 55 Device Credentials produced from adding a device

Each device gets its unique authentication token and with the authentication token and Device ID inserted into IBM IoT nodes, the device is connected to Node-Red service on IBM Cloud for graphical visualization and status report on the IBM IoT.



Temp and Humid visualization

Temperature BME280

27.6 °C
Temperature

Time	Value
4/23/2020, 9:36:03 PM	27.6 °C
4/23/2020, 9:35:43 PM	27.6 °C
4/23/2020, 9:35:23 PM	27.5 °C
4/23/2020, 9:35:03 PM	27.6 °C
4/23/2020, 9:35:03 PM	27.6 °C

Temperature MAX6675

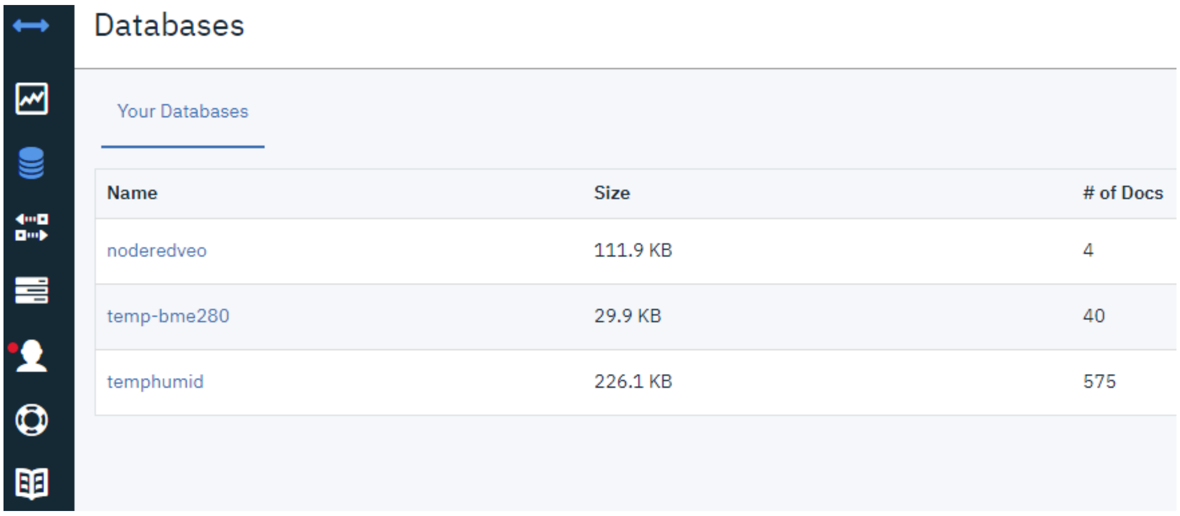
25.8 °C
value

Time	Value
4/23/2020, 9:35:50 PM	25.8 °C
4/23/2020, 9:35:10 PM	26.3 °C
4/23/2020, 9:35:10 PM	26.3 °C
4/23/2020, 9:35:10 PM	26.3 °C
4/23/2020, 9:35:10 PM	26.3 °C

Figure 56 Graphical visualization of the data received

4.1.7 Cloudant

After creating a Cloudant service, a database is created to store the value from ESP32-1 and ESP32-2

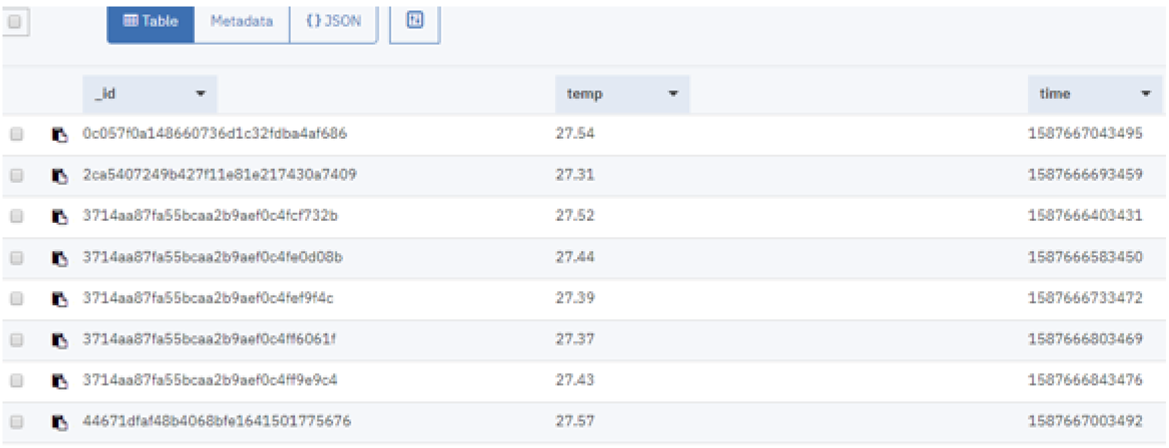

Databases

Your Databases

Name	Size	# of Docs
noderedveo	111.9 KB	4
temp-bme280	29.9 KB	40
temphumid	226.1 KB	575

Figure 57 Creation of database in Cloudant service

The values are stored in JSON format after connecting the flow to the Cloudant node in Node-Red service on the IBM Cloud.



_id	temp	time
0c057f0a148660736d1c32fdba4af686	27.54	1587667043495
2ca5407249b427f11e81e217430a7409	27.31	1587666693459
3714aa87fa55bcaa2b9aef0c4fcf732b	27.52	1587666403431
3714aa87fa55bcaa2b9aef0c4fe0d08b	27.44	1587666583450
3714aa87fa55bcaa2b9aef0c4fef9f4c	27.39	1587666733472
3714aa87fa55bcaa2b9aef0c4ff6061f	27.37	1587666803469
3714aa87fa55bcaa2b9aef0c4ff9e9c4	27.43	1587666843476
44671dfaf48b4068bfe1641501775676	27.57	1587667003492

Figure 58 Data stored as JSON format in the database created in Cloudant

4.1.8 IBM cloud Node-Red service

IBM IoT and dashboard node has to be downloaded from the library to be able to receive the data from the devices in the IBM IoT platform and send it to Cloudant database. In the IBM IoT node settings, the device ID is filled and the type, the function node in between reconstructs the message sent and sends from there to a debug node and the Cloudant node, the Cloudant node needs only the database name which you want to send the data to.

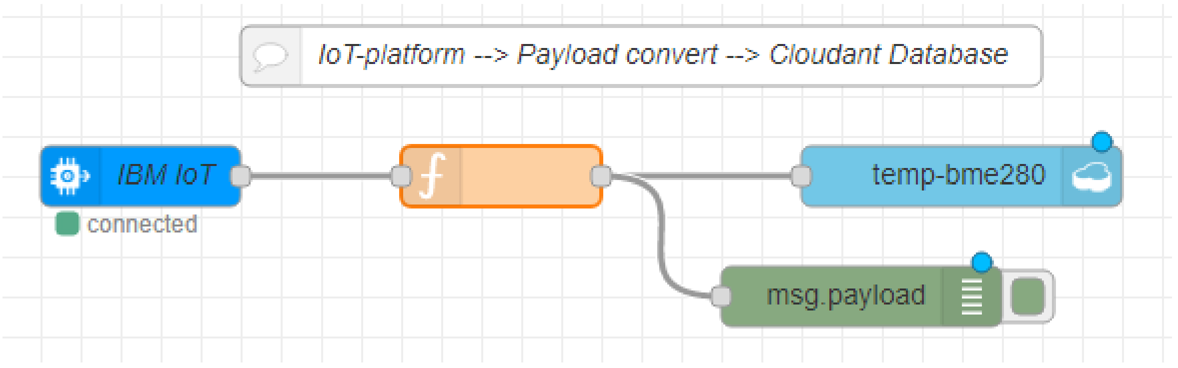
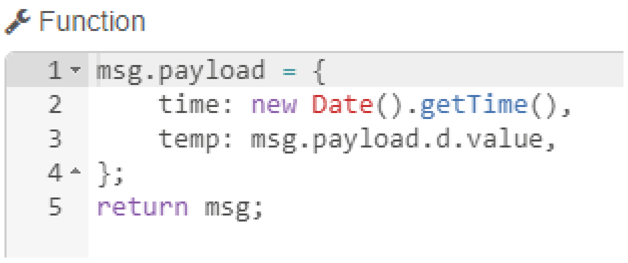


Figure 59 IBM cloud Node-Red configuration to Cloudant

The function code implements a time code and the value to time and temp variable.



Function

```
msg.payload = {
    time: new Date().getTime(),
    temp: msg.payload.d.value,
};
return msg;
```

Figure 60 Function node code for the data reconstruction

4.1.9 Data historian Chart

The first flow starts with a node for a button and access to a Cloudant database according to the database name set, the Cloudant runs through a function node (code found in appendix 1) that converts the data and from the function node to a chart and debug node. The flow operating principle for date range under the first flow (see figure 61) is to load the data from the data input from the start date and end date.

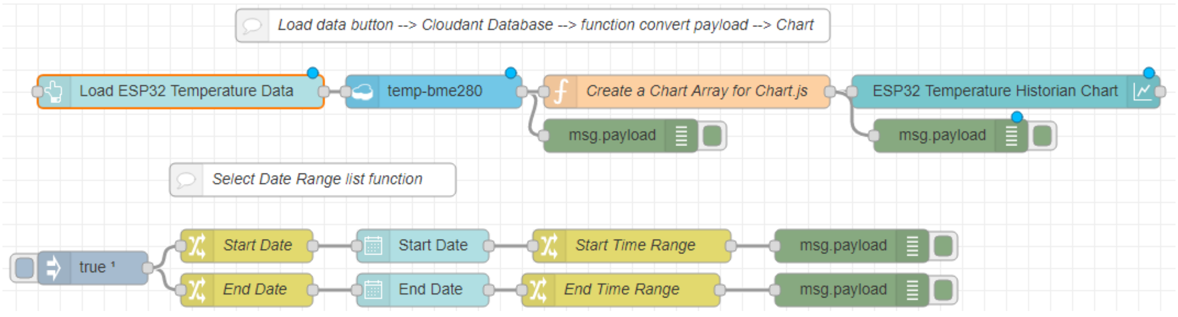

Load data button --> Cloudant Database --> function convert payload --> Chart
Load ESP32 Temperature Data
temp-bme280
Create a Chart Array for Chart.js
ESP32 Temperature Historian Chart
msg.payload
msg.payload
Select Date Range list function
true
Start Date
End Date
Start Date
End Date
Start Time Range
End Time Range
msg.payload
msg.payload

Figure 61 Node-Red configuration on IBM cloud for historian chart

The historian chart can be accessed by typing /ui at the end of the IBM Cloud Node-red address.

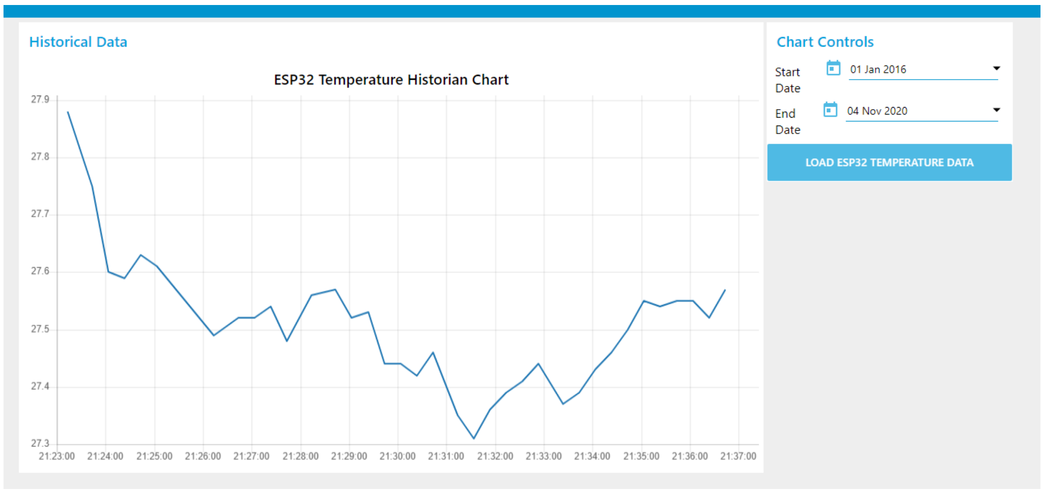


Figure 62 Historian chart

5 Conclusion

Different cloud services have their advantage and disadvantages and there is various software for specific uses, the software or cloud service should be used according to the application as some perform and offers better functions and services than other providers.

Some of the presented sensor systems for switchgear can be impractical with cables running around but such a system as the optic fiber can give a more reliable temperature value than others. The Zigbee and RFID system enables a much more flexible system on the selection of measurement locations but needs more planning on where to place the RFID readers and antennas to best read the sensors. The systems offer self-powered devices that solve the need to not have to put it on service and a web-based user interface makes the system easier to access for personnel and enables remote monitoring and data analysis.

The thesis can possibly give ideas to design a sensor system for switchgear and could give an overview of the modern sensor system used today. The microcontroller part can give a practical cost-effective IoT basic understanding as the device can also be used for many different applications.

The system I would opt for is an RFID/SAW system as it seems to be the most matured and popular sensor system used today, it has long term reliability and is a simple and cheap solution and cheaper and higher temperature sensing capabilities is developing with better integrated circuit. Some cons can be having to carefully consider the placement of the antenna and sensor for readability and possible limits the amount of sensor because of frequency band regulations. Hence, I would also consider using a Zigbee system which is still quite a new system but its simplicity to put it somewhere in interest and connect it to a concentrator through a button or code makes it a very optimal sensor system. Zigbee worldwide unified frequency band of 2.4 GHz to 2.5 GHz and its power source of directly from the temperature rise in conductor makes it less complex than the RFID system. The cons could be loss in data packets and connectivity loss from interference and its need to have enough energy to start instead of energy received from an antenna, its need to harvest energy from electromagnetic field and that the humidity sensor is using battery that is said to last at least 15 years might not be very reliable, the technology seems to be quite new still but has huge potential with its fast development.

Optic fiber can offer very reliable measurement and a very high measurement range to replace thermocouples but I would not consider it because of cables needed to route the signal and the ability to only be able to measure some components. An optic fiber system would be suitable for switchgears where higher than 140 °C needs to be measured or more accurate in very high voltage applications where the inherent noise and interference is really high. The accuracy that RFID and Zigbee gives after going over 100 °C can go over ± 4 but less in optic fiber systems.

6 References

[1] VEO, "VEO," 2020. [Online]. Available: https://www.veo.fi/.

[2] N. Kane, "Caring For Relatives By Robot," 21 October 2014. [Online]. Available: http://www.telepresenceoptions.com/2014/10/caring_for_relatives_by_robot/.

[3] Cisco, "Securing the Internet of Things: A Proposed Framework," 2020. [Online]. Available: https://tools.cisco.com/security/center/resources/secure_iot_proposed_framework.

[4] E. Jones, "Cloud Market Share – a Look at the Cloud Ecosystem in 2020," 10 April 2020. [Online]. Available: https://kinsta.com/blog/cloud-market-share/. [Accessed 2020].

[5] F. Richter, "Amazon Leads $100 Billion Cloud Market," 11 February 2020. [Online]. Available: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/.

[6] Nodericks, "AWS VS AZURE VS GOOGLE VS IBM CLOUD, WHICH IS THE BEST FOR ME?," 21 february 2018. [Online]. Available: http://www.nodericks.com/aws-vs-azure-vs-google-vs-ibm-cloud-best/.

[7] M. Rouse, "IBM Cloud (formerly IBM Bluemix and IBM SoftLayer)," 0. [Online]. Available: https://searchcloudcomputing.techtarget.com/definition/IBM-Bluemix.

[8] IBM, "IBM Cloudant," 2020. [Online]. Available: https://www.ibm.com/cloud/cloudant.

[9] Node-Red, "Node-Red," 2020. [Online]. Available: https://nodered.org/.

[10] Source, Open, "What is a Raspberry Pi?," 2020. [Online]. Available: https://opensource.com/resources/raspberry-pi.

[11] Raspberry Pi, "Raspberry Pi 4," 2020. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[12] Espressif Systems, "ESP32 A Different IoT Power and Performance," 2015. [Online]. Available: https://www.espressif.com/en/products/socs/esp32/overview.

[13] Last Minute Engineers, "Insight Into ESP32 Features & Using It With Arduino IDE," 2020. [Online]. Available: https://lastminuteengineers.com/esp32-arduino-ide-tutorial/.

[14] Schneider Electric, "What is Modbus and How does it work?," 25 July 2019. [Online]. Available: https://www.se.com/nz/en/faqs/FA168406/. [Accessed 2020].

[15] itead, "MAX485 MODULE," 11 June 2014. [Online]. Available: https://www.itead.cc/wiki/MAX485_MODULE. [Accessed 2020].

[16] S. Hoppe, "OPC Unifi ed Architecture," November 2019. [Online]. Available: https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf. [Accessed 2020].

[17] SFUPTOWNMAKER, "I2C," 0. [Online]. Available: https://learn.sparkfun.com/tutorials/i2c/all.

[18] Sparkfun, "Serial Peripheral Interface (SPI)," 2020. [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all.

[19] Amlendra, "I2C Protocol,bus and Interface: A Brief Introduction," 2016. [Online]. Available: https://aticleworld.com/i2c-bus-protocol-and-interface/.

[20] J. Gums, "Types of Temperature Sensors," 26 January 2018. [Online]. Available: https://www.digikey.com/en/blog/types-of-temperature-sensors.

[21] Maxim Integrated, "Temperature Sensor," September 2017. [Online]. Available: https://www.digikey.com/en/pdf/m/maxim/temperature-sensor-tutorial.

[22] Last Minute Engineers, "Interface BME280 Temperature, Humidity & Pressure Sensor with Arduino," 0. [Online]. Available: https://lastminuteengineers.com/bme280-arduino-tutorial/.

[23] Maxim integrated, "MAX6675," 2020. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX6675.pdf.

[24] Protosupplies, "MAX6675 Thermocouple Temperature Module," 0. [Online]. Available: https://protosupplies.com/product/max6675-thermocouple-temperature-module-2/.

[25] G. C. Hillar, MQTT Essentials - A Lightweight IoT Protocol, Packt Publishing Ltd, 2017.

[26] R. Pujar, "MQTT Protocol tutorial using Mosquitto and CloudMQTT," 0. [Online]. Available: http://www.raviyp.com/mqtt-protocol-tutorial-using-mosquitto-and-cloudmqtt/.

[27] Zigbee alliance, "Zigbee Green Power White Paper," 2016. [Online]. Available: https://zigbeealliance.org/wp-content/uploads/2019/11/Green-Power-White-Paper.pdf.

[28] Schneider Electric, "MCset," 2018. [Online]. Available: https://download.schneider-

electric.com/files?p_enDocType=Catalog&p_File_Name=McSet_NRJED312404EN_0319.pdf&p_Doc_Ref=NRJED312404EN.

[29] Schneider Electric, "ZBRN2," 0. [Online]. Available: https://www.se.com/ie/en/product/ZBRN2/harmony-hub%2C-wireless-to-modbus%2C-serial-line-gateway%2C-24...240-v-ac-dc/.

[30] Microchip, "Passive RFID Basics," 2004. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/21299e.pdf.

[31] M. B. H. H. a. M. E. Mehdia Ajana El Khaddar, "RFID Middleware Design and Architecture," 15 June 2011. [Online]. Available: https://www.intechopen.com/books/designing-and-deploying-rfid-applications/rfid-middleware-design-and-architecture.

[32] atlasRFIDstore, "What is RFID? | The Beginner's Guide to RFID Systems," 0. [Online]. Available: https://www.atlasrfidstore.com/rfid-beginners-guide/.

[33] RFID 4u, "Basics – RFID Regulations," 0. [Online]. Available: https://rfid4u.com/rfid-basics-resources/basics-rfid-regulations/.

[34] RFMicron, "Chameleon™ Technology Enables Low-Cost Sensors," RFMicron, 0.

[35] Azxon, "Chameleon™ Sensor Engine," 0. [Online]. Available: http://rfmicron.com/technology/chameleon/.

[36] Axzon, "Sensors," 2020. [Online]. Available: https://axzon.com/sensors/.

[37] J. A. Jonathan Murray, "They Eye On Medium-Voltage Switchgear," 2016. [Online]. Available: https://www.intellisaw.com/technology/case-studies-and-white-papers2/60-publication-eye-on-mv-switchgear-electricity-today-2016-pdf/file.html.

[38] C. Turcu, Development and Implementation of RFID Technology, 2009.

[39] S. S. D. S. S. S. J. J. M. B. B. W. W. T. M. R. G. Jeffrey C. Andle, "Temperature Monitoring System Using Passive Wireless Sensors for Switchgear and Power Grid Asset Managemen," 9 December 2010. [Online]. Available: https://www.researchgate.net/publication/228863244_Temperature_Monitoring_System_Using_Passive_Wireless_Sensors_for_Switchgear_and_Power_Grid_Asset_Management.

[40] Jonathm, "Surface acoustic wave (SAW) sensors," 1 May 2014. [Online]. Available: https://wiki.metropolia.fi/display/sensor/Surface+acoustic+wave+%28SAW%29+sensors.

[41] ABB, "ABB Ability™ Condition Monitoring for switchgear," June 2019. [Online]. Available: https://library.e.abb.com/public/6c851fbb8eeb45f68630a6f6f219d087/ABB

%20Ability%20Condition%20Monitoring%20for%20switchgear%20-%20SWICOM%20-%20June%202019.pdf.

[42] IntelliSAW, "IntelliSAW CAM™ Platform Sensors," 19 September 2016. [Online]. Available: https://www.abmicro.pl/pdf/IntelliSAW-Sensor-Installation-Manual.

[43] H. K. Hisham, "Optical Fiber Sensing Technology: Basics, Classifications," 2018. [Online]. Available: https://www.researchgate.net/publication/322935904_Optical_Fiber_Sensing_Technology_Basics_Classifications_and_Applications.

[44] P. B. R. F. T. S. Y. SHizhuo Yin, Fiber optic Sensors Second edition, 2008.

[45] T. M. A. M. David Krohn, Fiber optic Sensors, Fundamentals and Applications, 2014.

[46] M. F. F. R. A. Domingues, Optical Fiber Sensors for IoT and Smart Devices, 2017.

[47] Osensa Innovations, "OSENSA’s Optical Fluorescent Sensor Technology," 0. [Online]. Available: https://www.osensa.com/technology.

[48] Osensa Innovations, "Products," 2020. [Online]. Available: https://www.osensa.com/products.

[49] Random Nerd Tutorials, "ESP32 MQTT – Publish and Subscribe with Arduino IDE," 0. [Online]. Available: https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/.

[50] B. Innes, "esp8266Workshop," 10 January 2019. [Online]. Available: https://github.com/binnes/esp8266Workshop/blob/master/en/part3/HISTORY.md.

6.1 Appendix-1 Function code for message conversion and utilization for the historian chart

```
// This function reformats the timeseries data into an array that
var starttime = flow.get("start-time");
var endtime = flow.get("end-time");
if( msg.payload.length === 0 ) {
    // The historical database does not contain values for this date
range
    // Reset the chart with a blank array
    msg.payload = [] ;
    return msg;
}
// The Array has this structure, inserting the first element as a
template.
```

```
var ChartData = [ {"series":["Data"],
                   "data":[[{"x":msg.payload[0].time,"y":msg.payload[0].t
emp}]],
                   "labels":["Data"]}];
// Start at 1 because we've already added element 0 in the initial array
definition
for( var i=1; i < msg.payload.length; i++ ) {
    if( (msg.payload[i].time >= starttime) && (msg.payload[i].time <=
endtime) ) {
        // This temperature reading is within the Date Range
        ChartData[0].data[0].push( {
"x":msg.payload[i].time,"y":msg.payload[i].temp } );
    }
}
// Now we need to sort on the time so that we give the Chart node an
array in time order
ChartData[0].data[0].sort(function sortNumber(a,b) { return a.x - b.x;
});
msg.payload = ChartData;
return msg;
```

[50]

6.2 Appendix-2 Exported Node-Red configuration on Raspberry Pi

[{"id":"b6698309.ca1d","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"873ec389.c5265","type":"mqtt
in","z":"b6698309.ca1d","name":"","topic":"esp32/MAX6675","qos":"2","datatype":"auto","broker":"d2bb4b5a.1802b8","x":120,"y":18
0,"wires":[["3c351fa.68404e"]]},{"id":"394ad3b0.ba4d1c","type":"debug","z":"b6698309.ca1d","name":"","active":true,"tosidebar":true
,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":530,"y":120,"wires":[]},{"id":"8d290548.087328","type":
"ui_chart","z":"b6698309.ca1d","name":"","group":"9eb59948.650c78","order":0,"width":0,"height":0,"label":"Temperature","chartTy
pe":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":false,"ymin":"","ymax":"","removeOlder":1,"
removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"colors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c
","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"useOldStyle":false,"outputs":1,"x":530,"y":240,"wires":[[]]},{"id":"f78d98b.c
a4f668","type":"mqtt
in","z":"b6698309.ca1d","name":"","topic":"esp32/BME280Temp","qos":"2","datatype":"auto","broker":"d2bb4b5a.1802b8","x":750,"y
":420,"wires":[["13815ed3.c5c5f1"]]},{"id":"d048a320.b90ef","type":"debug","z":"b6698309.ca1d","name":"","active":true,"tosidebar":t
rue,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":1250,"y":360,"wires":[]},{"id":"e58676f5.8f4f98","typ
e":"mqtt
in","z":"b6698309.ca1d","name":"","topic":"esp32/BME280Humid","qos":"2","datatype":"auto","broker":"d2bb4b5a.1802b8","x":740,"
y":200,"wires":[["f33fbfc4.bc3f6"]]},{"id":"c02303ec.1b76c","type":"debug","z":"b6698309.ca1d","name":"","active":true,"tosidebar":tr
ue,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":1190,"y":200,"wires":[]},{"id":"93325f6d.9b3ec","type
":"ui_chart","z":"b6698309.ca1d","name":"","group":"700734d4.e0c7ac","order":0,"width":0,"height":0,"label":"Temperature","chartT
ype":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":false,"ymin":"","ymax":"","removeOlder":1,
"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"colors":["#ff0000","#ff0000","#ff7f0e","#2ca02c
","#98df8a","#ff0000","#ff9896","#9467bd","#c5b0d5"],"useOldStyle":false,"outputs":1,"x":1250,"y":480,"wires":[[]]},{"id":"6a115f12.
49aa1","type":"wiotp
out","z":"b6698309.ca1d","authType":"d","qs":"false","qsDeviceId":"","deviceKey":"a3da4257.b2675","deviceType":"","deviceId":"","ev
ent":"event","format":"json","qos":"","name":"IBMTemp","x":1240,"y":420,"wires":[]},{"id":"60e10fcc.943dc","type":"ui_gauge","z":"b
6698309.ca1d","name":"","group":"700734d4.e0c7ac","order":0,"width":0,"height":0,"gtype":"gage","title":"Humidity","label":"%","for
mat":"{{value}}","min":0,"max":"100","colors":["#00b3d9","#0073e6","#001bd7"],"seg1":"33","seg2":"66","x":1180,"y":260,"wires":[]},{
"id":"c9daaad0.994618","type":"wiotp
out","z":"b6698309.ca1d","authType":"d","qs":"false","qsDeviceId":"","deviceKey":"15bcfec5.62efd1","deviceType":"","deviceId":"","e
vent":"event","format":"json","qos":"","name":"IBMhumid","x":1180,"y":140,"wires":[]},{"id":"7399a6cf.9f81b8","type":"wiotp
out","z":"b6698309.ca1d","authType":"d","qs":"false","qsDeviceId":"","deviceKey":"83ed13a.db70ef","deviceType":"","deviceId":"","ev
ent":"event","format":"json","qos":"","name":"IBMTemp","x":520,"y":180,"wires":[]},{"id":"13815ed3.c5c5f1","type":"delay","z":"b6698
309.ca1d","name":"","pauseType":"rate","timeout":"5","timeoutUnits":"seconds","rate":"1","nbRateUnits":"20","rateUnits":"second","
randomFirst":"1","randomLast":"5","randomUnits":"seconds","drop":true,"x":980,"y":420,"wires":[["d048a320.b90ef","6a115f12.49aa
1","93325f6d.9b3ec"]]},{"id":"f33fbfc4.bc3f6","type":"delay","z":"b6698309.ca1d","name":"","pauseType":"rate","timeout":"5","timeou
tUnits":"seconds","rate":"1","nbRateUnits":"20","rateUnits":"second","randomFirst":"1","randomLast":"5","randomUnits":"seconds","
drop":true,"x":960,"y":200,"wires":[["c9daaad0.994618","c02303ec.1b76c","60e10fcc.943dc"]]},{"id":"3c351fa.68404e","type":"delay","
z":"b6698309.ca1d","name":"","pauseType":"rate","timeout":"5","timeoutUnits":"seconds","rate":"1","nbRateUnits":"40","rateUnits":"
second","randomFirst":"1","randomLast":"5","randomUnits":"seconds","drop":true,"x":300,"y":180,"wires":[["394ad3b0.ba4d1c","7399
a6cf.9f81b8","8d290548.087328"]]},{"id":"4a72d12c.78db2","type":"comment","z":"b6698309.ca1d","name":"Temperature node flow
for MAX6675ESP32 --> IBM
Cloud/Chart/Debug","info":"ssda","x":270,"y":80,"wires":[]},{"id":"f325e515.132e38","type":"comment","z":"b6698309.ca1d","name":"
Humid node flow for BME280ESP32 --> IBM
Cloud/Chart/Debug","info":"","x":950,"y":100,"wires":[]},{"id":"5abf0113.bfa9a","type":"comment","z":"b6698309.ca1d","name":"Tem
perature node flow for BME280, ESP32 --> IBM
Cloud/Chart/Debug","info":"","x":850,"y":360,"wires":[]},{"id":"d2bb4b5a.1802b8","type":"mqtt-
broker","z":"","name":"","broker":"localhost","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansessi
on":true,"birthTopic":"","birthQos":"0","birthPayload":"","closeTopic":"","closeQos":"0","closePayload":"","willTopic":"","willQos":"0",
"willPayload":""},{"id":"9eb59948.650c78","type":"ui_group","z":"","name":"Main","tab":"e2a4f55d.715dd8","disp":true,"width":"6","co

llapse":false},{"id":"700734d4.e0c7ac","type":"ui_group","z":"","name":"main
2","tab":"e2a4f55d.715dd8","disp":true,"width":"6","collapse":false},{"id":"a3da4257.b2675","type":"wiotp-
credentials","z":"","name":"TempBME280","org":"u45len","serverName":"","devType":"ESP32","devId":"BME280Temp","keepalive":"
60","cleansession":true,"tls":"","usetls":false},{"id":"15bcfec5.62efd1","type":"wiotp-
credentials","z":"","name":"HumidBME280","org":"u45len","serverName":"","devType":"ESP32","devId":"BME280Humid","keepalive"
:"60","cleansession":true,"tls":"","usetls":false},{"id":"83ed13a.db70ef","type":"wiotp-
credentials","z":"","name":"D1Temp","org":"u45len","serverName":"","devType":"ESP32","devId":"MAX6675","keepalive":"60","clean
session":true,"tls":"","usetls":false},{"id":"e2a4f55d.715dd8","type":"ui_tab","z":"","name":"test","icon":"test","disabled":false,"hidden"
:false}]

6.3 Appendix-2 Exported Node-Red configuration on IBM cloud

[{"id":"206926d6.abaa5a","type":"ui_button","z":"7e42a0c6.b8b84","name":"","group":"3cd36509.dfe61a","order":3,"width":0,"height":0,"passthru":false,"label":"Load ESP32 Temperature Data","tooltip":"","color":"","bgcolor":"","icon":"","payload":"","payloadType":"str","topic":"","x":310,"y":620,"wires":[["9e581429.6de8a8"]]},{"id":"3cd36509.dfe61a","type":"ui_group","z":"","name":"Chart Controls","tab":"f5bf89a9.d50d78","order":2,"disp":true,"width":"6"},{"id":"f5bf89a9.d50d78","type":"ui_tab","z":"","name":"Historical Data","icon":"fa-area-chart","order":5}]

6.4 Appendix-3 Arduino code for ESP32-2 MAX6675

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <max6675.h>

int ktcSO = 19;
int ktcCS = 23;
int ktcCLK = 5;

MAX6675 ktc(ktcCLK, ktcCS, ktcSO);

const char* ssid = "ASUS";
const char* password = "mashedpotatoes";
const char *ID = "MAX6675";
const char* mqtt_server = "192.168.1.143";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

float temperature = 0;

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


  client.setServer(mqtt_server, 1883);

  setup_wifi();
  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else
        type = "filesystem";
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
    .onError([](ota_error_t error) {
      Serial.printf("Error[%u]: ", error);
      if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
      else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
      else if (error == OTA_CONNECT_ERROR) Serial.println("Connect
Failed");
      else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive
Failed");
      else if (error == OTA_END_ERROR) Serial.println("End Failed");
    });

  ArduinoOTA.begin();
  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("MAX6675")) {
      Serial.println("connected");

    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 10000) {
    lastMsg = now;
    temperature = ktc.readCelsius();

    // Convert the value to a char array
    char tempString[8];
    dtostrf(temperature, 1, 2, tempString);
    Serial.print("Temperature: ");
    Serial.println(tempString);
    client.publish("esp32/MAX6675", tempString);

  }
  ArduinoOTA.handle();
}
```

6.5 Appendix-4 Arduino code for ESP32-1 BME280 sensor

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_BME280.h>
#include <Adafruit_Sensor.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
const char* ssid = "ASUS";
const char* password = "mashedpotatoes";
const char* ID = "BME280";
const char* mqtt_server = "192.168.1.143";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

Adafruit_BME280 bme; // I2C
float temperature = 0;
float humidity = 0;

void setup() {
  Serial.begin(115200);
  if (!bme.begin(0x76)) {
    Serial.println("Could not find a valid BME280 sensor, check
wiring!");
    while (1);
  }
  client.setServer(mqtt_server, 1883);

  setup_wifi();
  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else
        type = "filesystem";
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
    .onError([](ota_error_t error) {
      Serial.printf("Error[%u]: ", error);
      if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
      else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
      else if (error == OTA_CONNECT_ERROR) Serial.println("Connect
        Failed");
      else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive
        Failed");
      else if (error == OTA_END_ERROR) Serial.println("End Failed");
    });

  ArduinoOTA.begin();

  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("BME280")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 10000) {
    lastMsg = now;
    temperature = bme.readTemperature();
    // Convert the value to a char array
    char tempString[8];
    dtostrf(temperature, 1, 2, tempString);
    Serial.print("Temperature: ");
    Serial.println(tempString);
    client.publish("esp32/BME280Temp", tempString);
    humidity = bme.readHumidity();
    // Convert the value to a char array
    char humString[8];
    dtostrf(humidity, 1, 2, humString);
    Serial.print("Humidity: ");
    Serial.println(humString);
    client.publish("esp32/BME280Humid", humString);
  }
  ArduinoOTA.handle();
}
```