

Name: _____

Class: _____

Date: _____

Task 1

Weather station

In this task, you will begin by building the sensor station. Then, we will convert it into a weather station that will transfer your measured data to a web server via MQTT protocol (Message Queuing Telemetry Transport). It will complete a weather forecast and will regularly transmit a camera image, in order to function as a webcam.

Construction task

Build the IoT station according to the building instructions. Connect the sensors and actuators to the TXT 4.0 according to the wiring diagram.

During the experimental tasks, data will be transferred to a web server using the MQTT protocol, and the transmitted values will be visualised there. To do so, create an account in the fischertechnik cloud at (www.fischertechnik-cloud.com).

Programming tasks

1. Measuring humidity, temperature, and air pressure

The environmental sensor, a Bosch sensor with type designation BME680, includes a temperature, moisture, and pressure sensor, among other features. The sensor values are transmitted to the TXT via the I²C protocol. You can query the sensor values using the relevant Blockly commands.

Write a Blockly program that turns the IoT station into a weather station which outputs the measured humidity (in %), the temperature (in °C), and the air pressure (in hPa) to the display of the TXT, and updates these once per second.

2. Barometer

You can use the measured air pressure to add a forecasting function to the weather station: if the air pressure is dropping, this indicates that precipitation is imminent; if the air pressure is rising, this indicates dry and sunny weather.

At sea level, the air pressure is between around 950 and 1050 hPa; it fluctuates by approx. ± 50 hPa around a mean value of about 1000 hPa. If you divide the entire

measurement range into three segments, you can use the measured air pressure to make a forecast (“rainy”, “variable”, “dry”).

However, the air pressure drops at higher elevations. The values measured by the sensor, therefore, should be interpreted differently at different elevations. You can take this into account by converting the measured air pressure $p(h)$ based on the “barometric formula” into a “theoretical air pressure” p_0 at sea level [1]:

$$p_0 = p(h) \cdot \left(\frac{T_0}{T(h)} \right)^{5.255} \text{ hPa}$$

In this formula,

- $p(h)$ the air pressure measured at the elevation h ,
- $T(h)$ the temperature measured at this elevation (in Kelvins) and
- T_0 the (theoretical) temperature at sea level, which drops approximately 0.0065 K with each meter of elevation:

$$T_0 = T(h) + 0.0065 \cdot h \frac{\text{K}}{\text{m}}$$

Determine the elevation h of your weather station above sea level, and use this to calculate the “air pressure at sea level” based on the measured air pressure $p(h)$ and temperature $T(h)$ in your Blockly program.

Then, add a weather forecast to your weather station to be shown on the display of the TXT.

Experimental tasks

1. Determining temperature using the NTC resistor

In task 1 for the Base Set, you used the NTC resistor (thermistor) to determine the temperature using the Steinhart-Hart equation (see accompanying materials). To do so, you had to first determine three resistance values at different temperatures using the thermistor. The environmental sensor will now provide these to you with a high level of accuracy.

Connect the NTC resistor to I8, and add a measurement of the NTC resistance to your Blockly program. Enter the results of three measurements into the following table:

Resistance value	Temperature

Note: The environmental sensor will react to temperature changes at a long delay, due to the housing. Therefore, only complete the measurement once the temperature value of the sensor has stabilised.

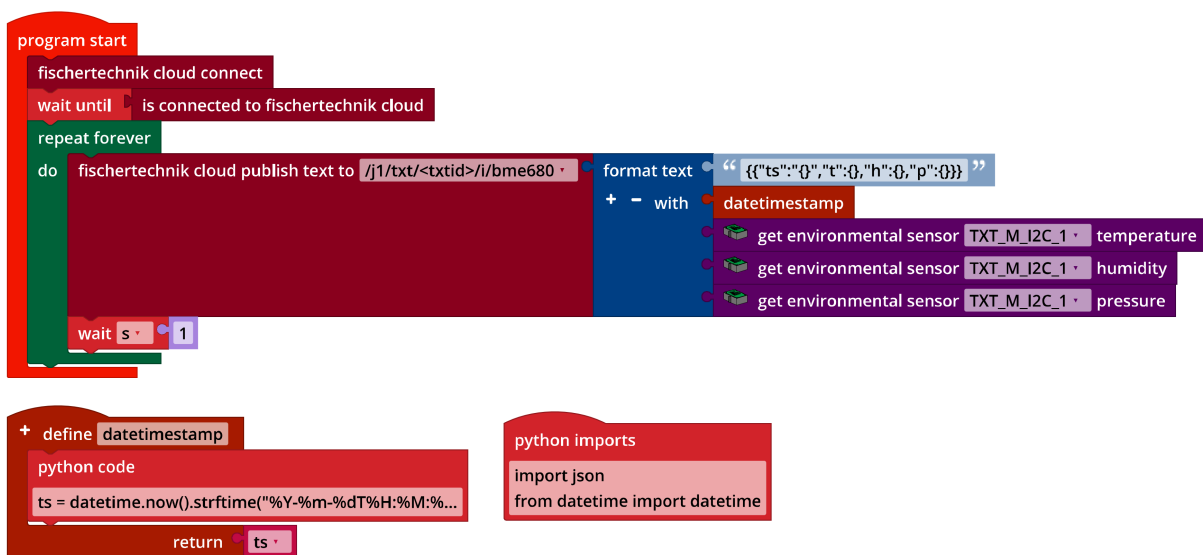
You can now determine the coefficients of the Steinhart-Hart equation using the accompanying materials and the website indicated in the attachment [4], and also output the temperature calculated from the NTC resistance on the display of the TXT.

2. Data visualisation on an IoT server

Now, the measurement data from your weather station should be transmitted to a web server and displayed in a (prepared) dashboard there. To do so, connect your TXT to the account you set up previously in the fischertechnik cloud.

2a. Now, configure the dashboard so that the temperature, humidity, and air pressure are displayed. Click all displays you do not need to deselect them.

2b. The following example program demonstrates how the sensor data is transmitted to the IoT server: First, open the connection using your cloud account (“MQTT connect”); then, the current measured values from the sensor are transmitted at regular intervals in a fixed format (time stamp, temperature, moisture, pressure) to the cloud server (“MQTT publish”).



```

program start
  fischertechnik cloud connect
  wait until is connected to fischertechnik cloud
  repeat forever
    do fischertechnik cloud publish text to /j1/txt/<txtid>/i/bme680
    + - with format text “{{"ts":{:0}, "t":{:0}, "h":{:0}, "p":{:0}}}”
    + - with datetimestamp
    + - with get environmental sensor TXT_M_I2C_1 temperature
    + - with get environmental sensor TXT_M_I2C_1 humidity
    + - with get environmental sensor TXT_M_I2C_1 pressure
    wait 1s
  
```

```

+ define datetimestamp
  python code
  ts = datetime.now().strftime("%Y-%m-%dT%H:%M:%S...")
  return ts

```

```

python imports
import json
from datetime import datetime

```

IoT_MQTT.ft

Add the display on the IoT server to your Blockly program for the barometer.

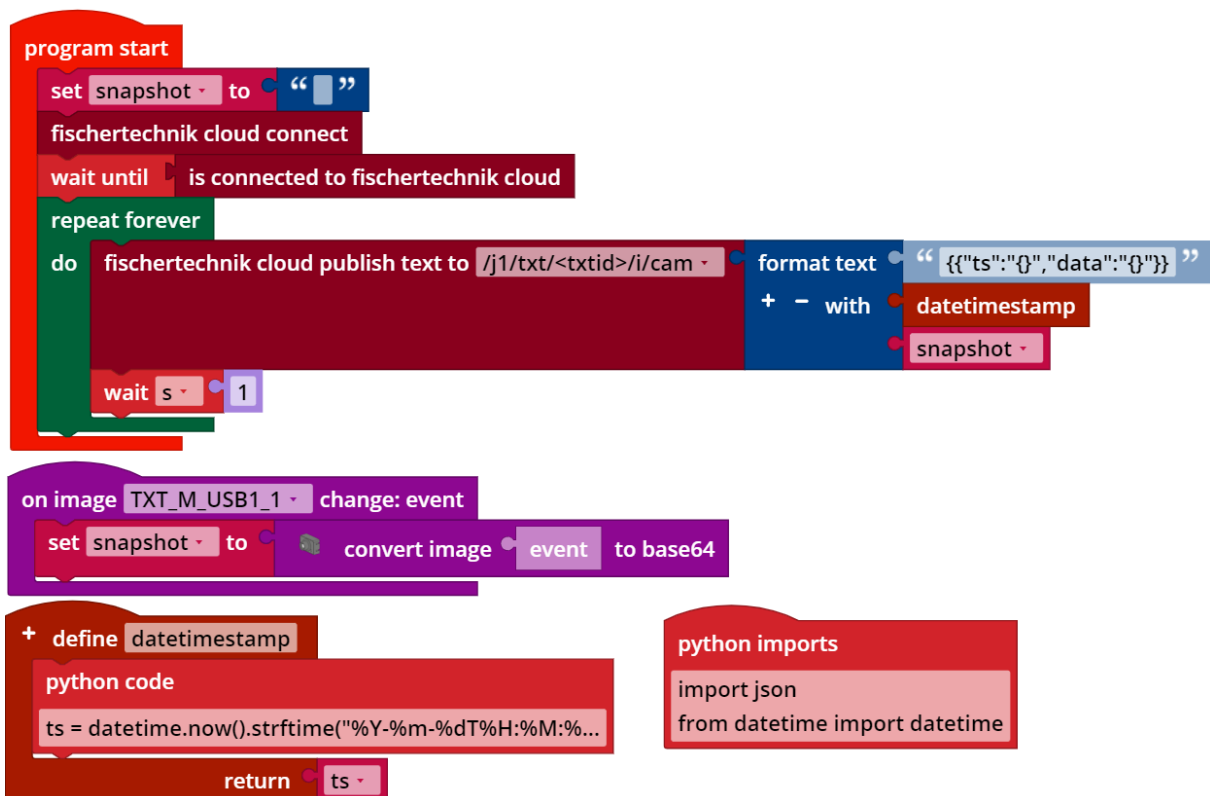
Now, you can generate measurement series, display the measured values, and download them to the dashboard as a csv file. You can analyse and edit the csv file in a spreadsheet (such as in Excel).

3. Webcam

What would a weather station be without a webcam? Expand your program so that it transmits the camera image to the dashboard of the IoT server.

3a. Now, add a display of the camera image to the dashboard.

3b. The following example program shows you how transmission of the camera image works: Each image the camera “takes” is encoded in Base64 format. Then, the current image is transmitted to the IoT server once per second (“MQTT publish”).



```

program start
  set snapshot to ""
  fischertechnik cloud connect
  wait until is connected to fischertechnik cloud
  repeat forever
    do fischertechnik cloud publish text to /j1/txt/<txid>/i/cam
      format text [{"ts": "0", "data": "0"}]
      + - with datetimestamp
      snapshot
    wait s 1

on image TXT_M_USB1_1 change: event
  set snapshot to convert image event to base64

+ define datetimestamp
  python code
  ts = datetime.now().strftime("%Y-%m-%dT%H:%M:%S...")
  return ts

python imports
import json
from datetime import datetime
  
```

IoT_Webcam.ft

Expand your program so that it transmits the camera image to the IoT server.

You can save snapshots in a gallery and download the images individually using the dashboard.

Annex

Task 1: Weather station

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- Example programs “IoT_MQTT.ft” and “IoT_Webcam.ft”
- Account in the fischertechnik cloud

Further information

- [1] Wikipedia: [Barometric formula](#).
- [2] Online diagram editor for creating state transition diagrams (drawio format): <https://www.diagrammeditor.de/>
- [3] fischertechnik: [NTC resistance](#). Data sheet, Art. no. 36437.
- [4] Stanford Research Systems (SRS): [Thermistor Calculator](#). V1.1
- [5] Dirk Fox: [“Einmessen” eines digitalen Messgeräts \(“Calibrating” a digital measurement device\)](#). ft:pedia 1/2013, p. 39-48.

Name: _____

Class: _____

Date: _____

Solution sheet task 1:

Weather station

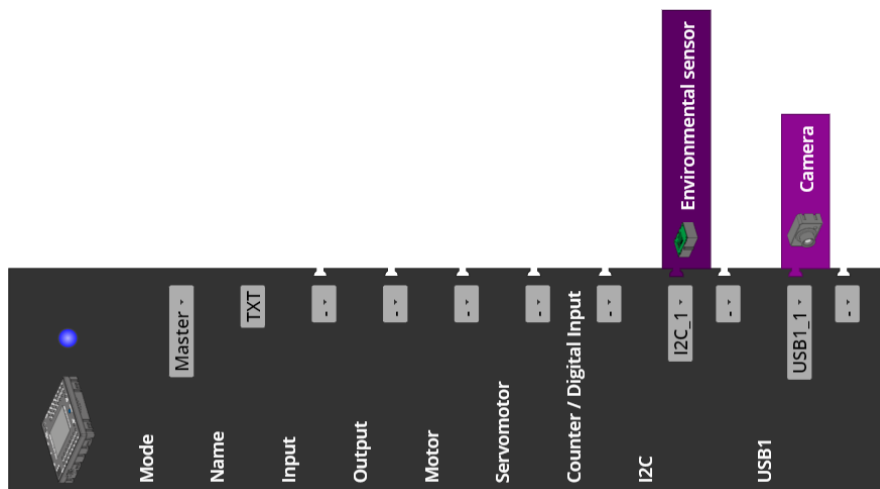
Tasks 1 and 2 turn the sensor station into a local weather station. In the experimental tasks, the weather station is connected to a web server via MQTT that displays the measured data and transmitted images in a dashboard.

Construction task

See building instructions.

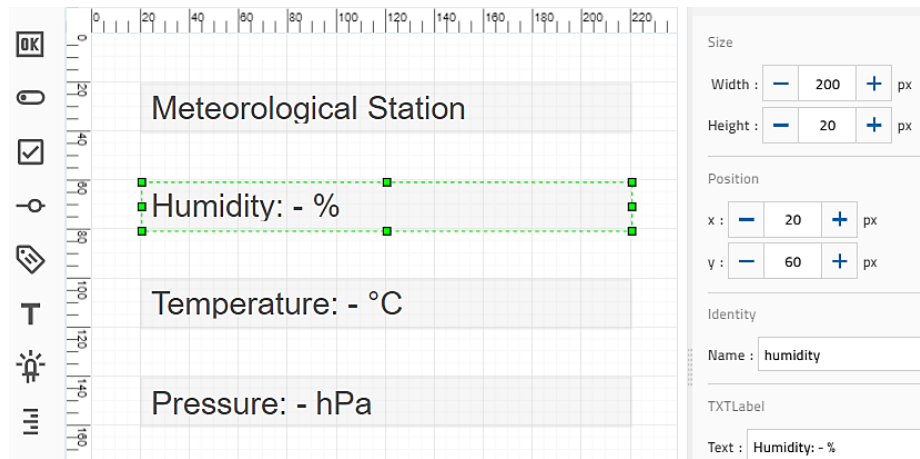
Programming tasks

Configuring the sensors and actuators:



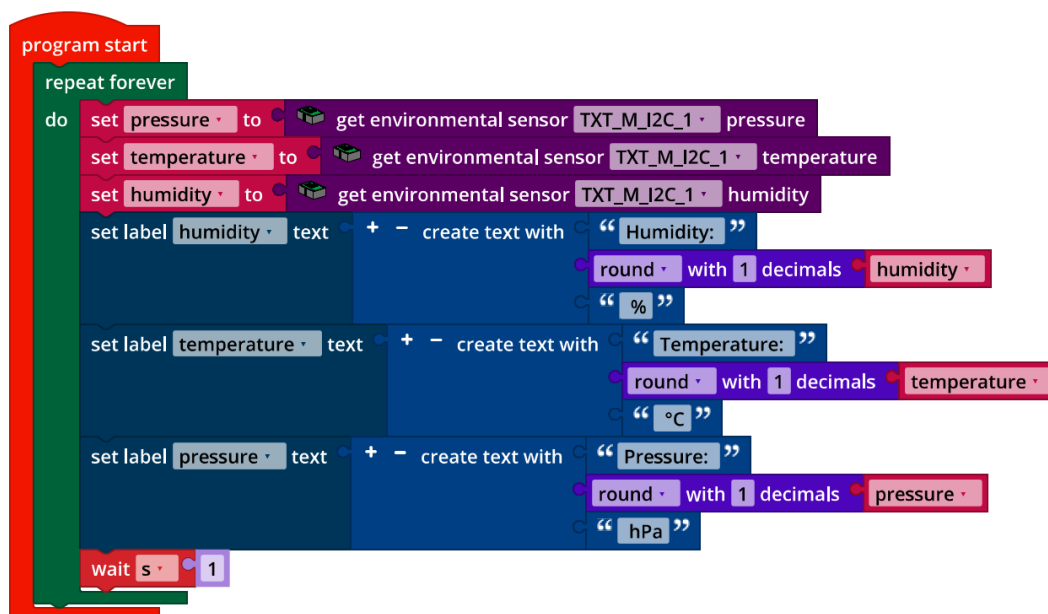
1. Measuring humidity, temperature, and air pressure

1a. Display configuration (example):



Configuring the display output

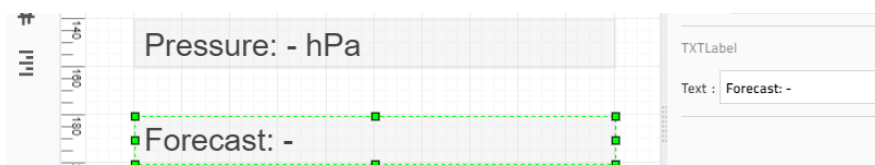
1b. Program (example):



IoT_Meteorological_Station.ft

2. Barometer

2a. Expanding the display configuration (example):



Expanding the display output

2b: Program (example):

The temperature values must be converted into Kelvin before the calculation: $0^{\circ}\text{C} = 273.15\text{ K}$.

```

program start
set h to 125
repeat forever
do
set pressure to get environmental sensor TXT_M_I2C_1 pressure
set temperature to get environmental sensor TXT_M_I2C_1 temperature
set humidity to get environmental sensor TXT_M_I2C_1 humidity
set label humidity text + - create text with " Humidity: "
round with 1 decimals humidity
" %"
set label temperature text + - create text with " Temperature: "
round with 1 decimals temperature
" °C "
set p0 to
temperature + 273.15
+ 0.0065 x h
÷
temperature + 273.15
^ 5.255
x pressure
set label pressure text + - create text with " Pressure: "
round with 1 decimals pressure
" hPa ( "
round with 1 decimals p0
" hPa to NN) "
+ if p0 < 980
do set label forecast text " Forecast: Rain "
else if p0 > 1020
do set label forecast text " Forecast: Fair "
else set label forecast text " Forecast: Change "
wait s 1
    
```

IoT_Barometer.ft

Experimental tasks

1. Determining temperature using the NTC resistor

1a. The resistance of the thermistor and the associated measured temperature values for the environmental sensor can easily be determined via a Log output on the console (see example program below).

Three measured values can be used to determine the coefficients a, b, and c of the Steinhart-Hart equation [4, 5] (T: Temperature in Kelvins, R: NTC resistance in Ohms):

$$\frac{1}{T} = a + b \cdot \ln(R) + c \cdot \ln(R)^3$$

Resistance value	Temperature
1500 Ω	25.0 °C
2362 Ω	13.7 °C
2530 Ω	11.8 °C

Example measurement

The measured values are dependent on the NTC resistance and may deviate.

Coefficients: a = 0.004535418128, b = -0.0003767491105, c = 0.000004023802790

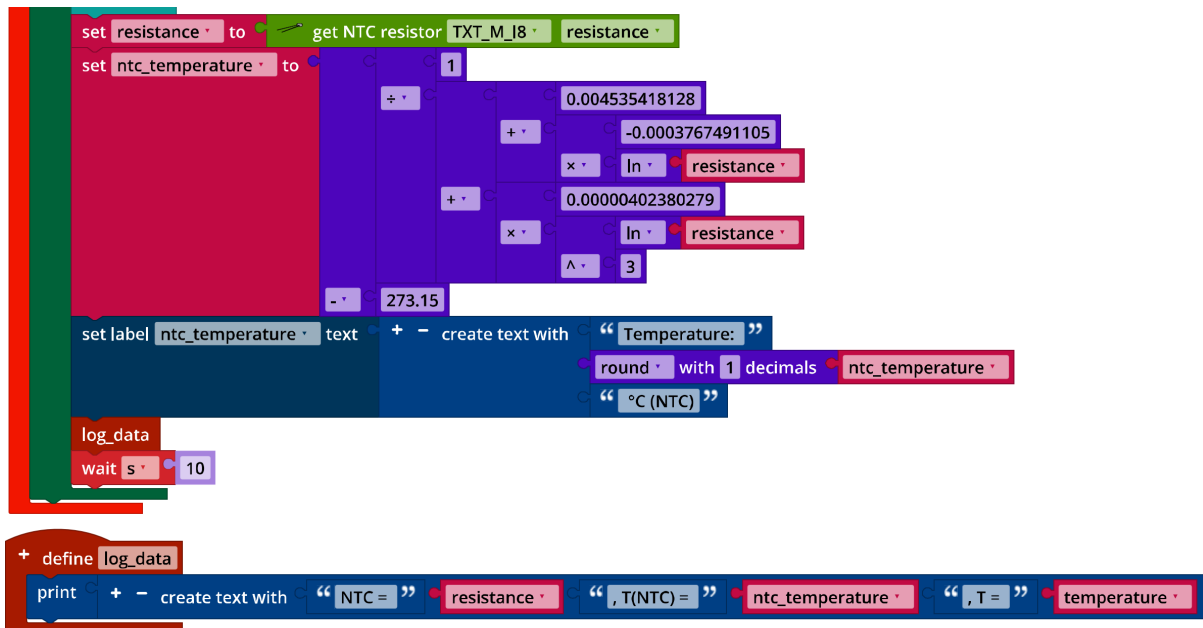
1b. Expanding the display:

Configuring the display output of NTC temperature measurement

1c. Expanding the program (example):

Calculating the temperature value based on the NTC resistance and output of the measured values for the NTC resistor and temperature sensor on the console.

Here as well, the temperature value calculated from the thermistor must then be converted into Kelvins: $0^{\circ}\text{C} = 273.15\text{ K}$.



```

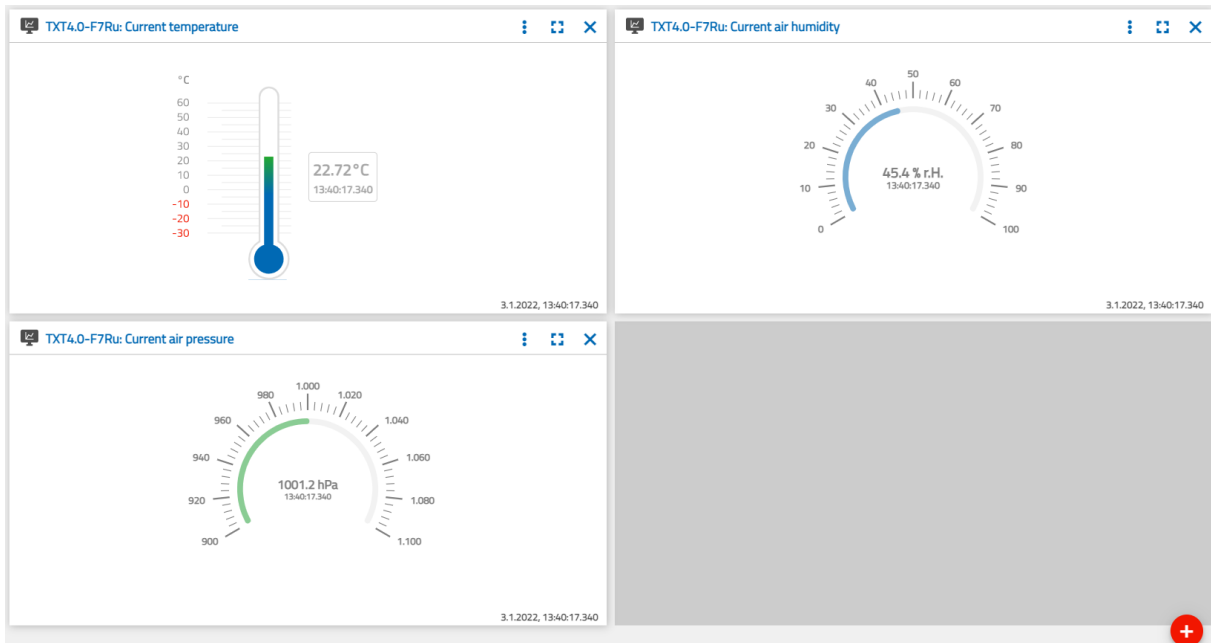
set resistance to get NTC resistor TXT_M_I8 resistance
set ntc_temperature to 1
÷ 0.004535418128
+ -0.0003767491105
× ln resistance
+ 0.00000402380279
× ln resistance
^ 3
- 273.15
set label ntc_temperature text + - create text with "Temperature:"
round with 1 decimals ntc_temperature
"C (NTC)"
log_data
wait s 10

+ define log_data
print + - create text with "NTC =" resistance ", T(NTC) =" ntc_temperature ", T =" temperature
    
```

IoT_Barometer_with_NTC_resistor.ft

2. Data visualisation on an IoT server

2a. Configuring the dashboard on the IoT server in the fischertechnik cloud:



Configuring the display in the dashboard

2b. Expanding the program (example):

```

fischertechnik cloud publish text to /j1/txt/<txtid>/i/bme680
+ - with format text [{"ts":0,"t":0,"h":0,"p":0}]
+ - with timestamp
+ - with get environmental sensor TXT_M_I2C_1 temperature
+ - with get environmental sensor TXT_M_I2C_1 humidity
+ - with get environmental sensor TXT_M_I2C_1 pressure

wait s 1

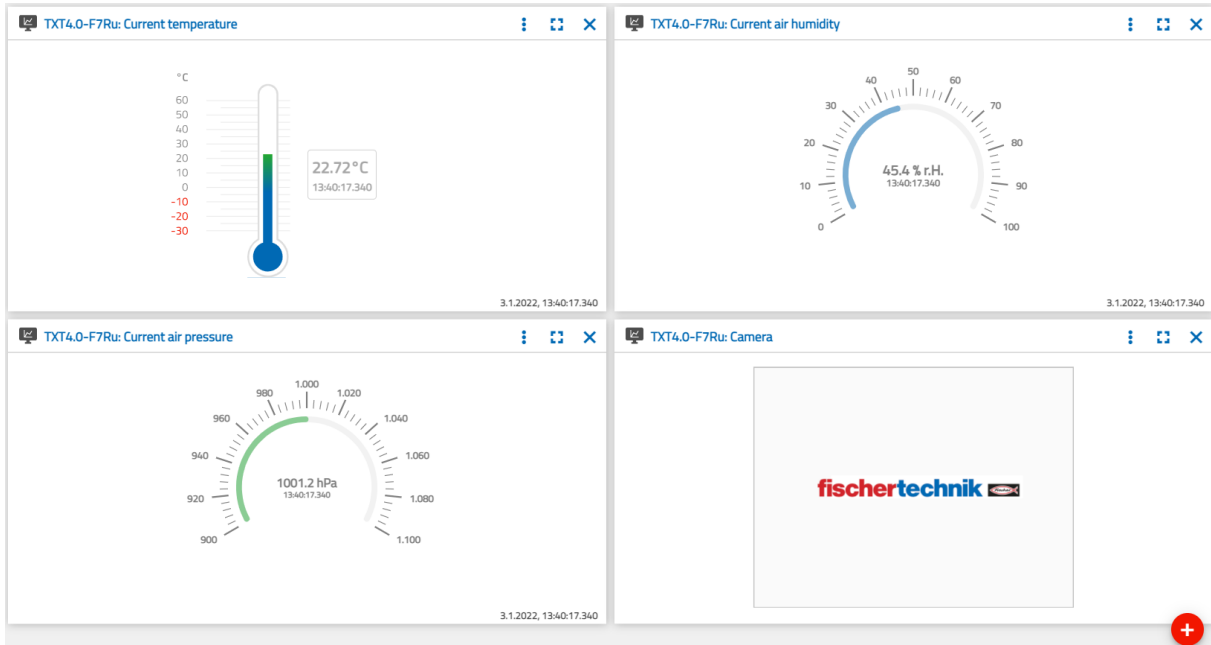
+ define timestamp
python code
ts = datetime.now().strftime("%Y-%m-%dT%H:%M:%S...")
return ts

python imports
import json
from datetime import datetime
  
```

IoT_MQTT_Barometer.ft

3. Webcam

3a. Expanding the dashboard in the fischertechnik cloud:



Dashboard with webcam

3b. Expanding the program (example):

```

    fischertechnik cloud publish text to /j1/txt/<txtid>/i/cam
    format text with " {{ts}}:{{data}} "
    + with
    + timestamp
    + snapshot
    wait s 1

    + define timestamp
    python code
    ts = datetime.now().strftime("%Y-%m-%dT%H:%M:%S...")
    return ts

    python imports
    import json
    from datetime import datetime

    on image TXT_M_USB1_1 change: event
    set snapshot to
    convert image event to base64
    
```

IoT_MQTT_Barometer_with_Webcam.ft

Annex

Task 1: Weather station

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- Example programs “IoT_MQTT.ft” and “IoT_Webcam.ft”
- Account in the fischertechnik cloud

Further information

- [1] Wikipedia: [Barometric formula](#).
- [2] Online diagram editor for creating state transition diagrams (drawio format): <https://www.diagrameditor.de/>
- [3] fischertechnik: [NTC resistance](#). Data sheet, Art. no. 36437.
- [4] Stanford Research Systems (SRS): [Thermistor Calculator](#). V1.1
- [5] Dirk Fox: [“Einmessen” eines digitalen Messgeräts \(“Calibrating” a digital measurement device\)](#). ft:pedia 1/2013, p. 39-48.

Task 1: Weather station

You will begin by building a sensor station that will serve as a model for all subsequent tasks. Then, you will use the environmental sensor (a Bosch Sensortec BME680) to collect different weather data (humidity, temperature, and air pressure), calculate a weather forecast based on this data, and transmit the data for visualisation to a dashboard on an IoT server in the fischertechnik cloud.

Topics

- Analysing sensor data to display the weather
- Weather forecast using measured air pressure and barometric formula
- Visualising the sensor data on a web server
- Adding a webcam to the weather station

Learning objectives

- Recording measured values, meaningful visualisation and analysis
- Using the barometric formula to create a weather forecast
- NTC resistance and temperature calculation (Steinhart-Hart equation)

Time required

Assuming the student has some familiarity with fischertechnik, building the sensor station according to the building instructions will take around 60-90 minutes.

Students will need around 45 to 60 minutes to solve the programming tasks (assuming previous programming experience). The experimental tasks build on task 1 for the Base Set and the programming tasks; it should take a maximum of 60 minutes to complete them when using the example programs. Because the sensor can take a long time to adjust, determining the measured values necessary to solve experimental task 1 may require a 30 minute wait time.

Annex

Task 1: Weather station

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- Example programs “IoT_MQTT.ft” and “IoT_Webcam.ft”
- Account in the fischertechnik cloud

Further information

- [1] Wikipedia: [Barometric formula](#).
- [2] Online diagram editor for creating state transition diagrams (drawio format): <https://www.diagrammeditor.de/>
- [3] fischertechnik: [NTC resistance](#). Data sheet, Art. no. 36437.
- [4] Stanford Research Systems (SRS): [Thermistor Calculator](#). V1.1
- [5] Dirk Fox: [“Einmessen” eines digitalen Messgeräts \(“Calibrating” a digital measurement device\)](#). ft:pedia 1/2013, p. 39-48.

Name: _____

Class: _____

Date: _____

Task 2

Indoor climate

In this task, our sensor station will monitor the indoor climate and warn us if it is too hot, too moist, too dark, or too loud – or if the air quality drops (for instance due to excessive CO₂).

Construction task

You can use the sensor station you built in task 1 without modification.

Programming tasks

1. Measuring indoor climate values

You can use the measured values for “humidity” and “temperature” collected in task 1 to determine not only the weather, but also the indoor climate.

Humidity and temperature are important factors for a good indoor climate. The (absolute) humidity indicates the quantity of water (in g) which is present in a cubic meter (m³) of air as water vapour.

The sensor measures the relative humidity, or the quotients of (absolute) humidity and the maximum possible humidity in %. The maximum humidity changes with the temperature: At 0°C, air can absorb a maximum of 5g of water vapour, at 30°C, however, it can absorb 30g – therefore, cold air is “drier” than warm air.

A relative humidity of 45-55% at a temperature of 20°C is considered the optimal indoor climate for living and working areas.

1a. Adjust the display configuration for the weather station from task 1 so that it displays relative humidity and temperature.

1b. Modify the Blockly program for the weather station to make it into an indoor climate station.

Note: The sensor will need around 10 seconds after the program is started for the measured values to stabilise [1].

2. Measuring air quality

The environmental sensor can detect a variety of gases in the ambient air, such as formaldehyde, alcohol, solvents, and evaporating varnishes, stains, cleaning agents, or adhesives. It can use these to determine an Index for Air Quality, IAQ in a range of 0-500.

An IAQ value below 50 indicates good air quality, while a value over 200 indicates significant contamination in the air (see table).

IAQ Index	Air Quality	Impact (long-term exposure)	Suggested action
0 – 50	Excellent	Pure air; best for well-being	No measures needed
51 – 100	Good	No irritation or impact on well-being	No measures needed
101 – 150	Lightly polluted	Reduction of well-being possible	Ventilation suggested
151 – 200	Moderately polluted	More significant irritation possible	Increase ventilation with clean air
201 – 250 ⁹	Heavily polluted	Exposition might lead to effects like headache depending on type of VOCs	optimize ventilation
251 – 350	Severely polluted	More severe health issue possible if harmful VOC present	Contamination should be identified if level is reached even w/o presence of people; maximize ventilation & reduce attendance
> 351	Extremely polluted	Headaches, additional neurotoxic effects possible	Contamination needs to be identified; avoid presence in room and maximize ventilation

Index of Air Quality [2]

2a. Adjust the display configuration of the indoor climate station so that it outputs the air quality.

2b. Expand the Blockly program for the indoor climate station accordingly.

Note: When the program is started, the air quality sensor is calibrated automatically. The calibration process will take five minutes, and the progress is displayed on the console. During this time, -1 will be indicated as the IAQ value.

Experimental tasks

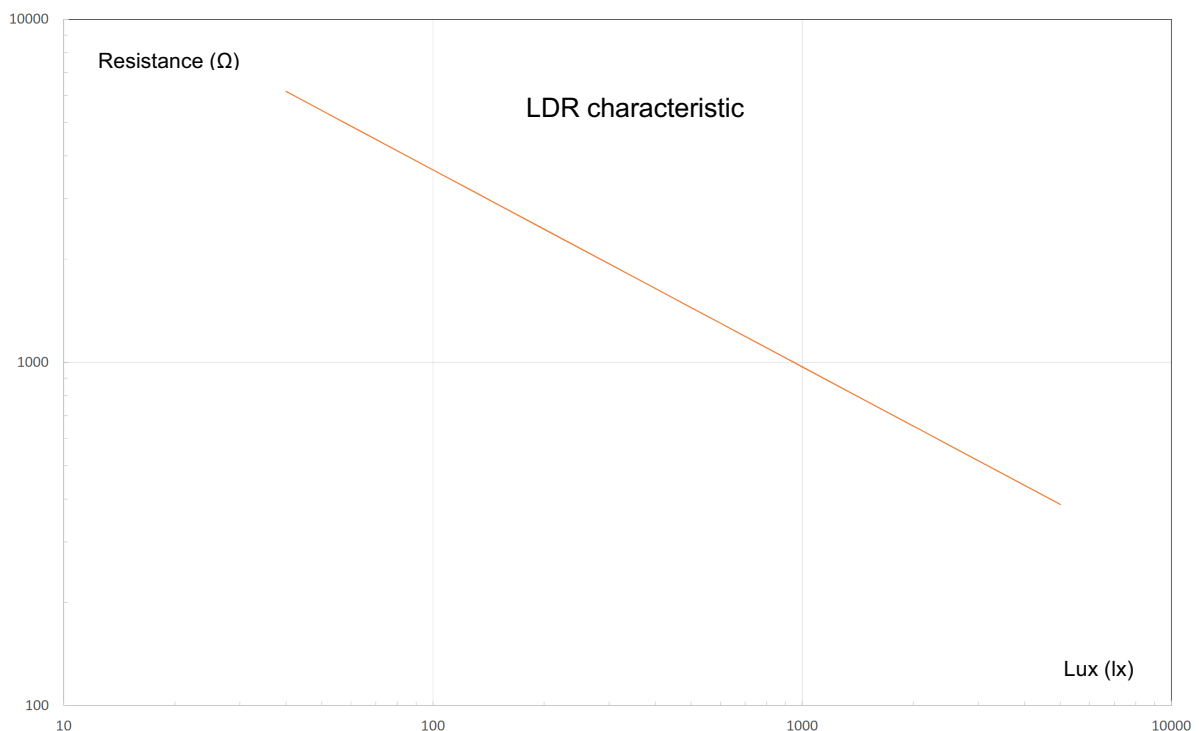
1. Brightness

In contrast to the phototransistor, which was used in task 2 of the Robotics Base Set as a sensor in a light barrier, a photoresistor (LDR – *Light Dependent Resistor*) is not a digital sensor, but rather an analogue one. We can use it to determine the ambient brightness.

Lux (lx) is the unit for illumination level – the luminous flux (measured in lumens) that strikes a certain surface. The following formula applies: $1 \text{ Lux} = 1 \text{ Lumen/m}^2$. A lux corresponds to around the illumination level that can be measured at one meter's distance from a candle flame.

For comparison: On a cloudless summer day, the illumination level under direct sunlight can reach 100,000 lux, but on a cloudy winter day it will be only around 2,000 lux.

The fischertechnik photoresistor delivers resistance values up to more than 60 kOhm; the brighter it is, the lower the resistance. A resistance value of 1 kOhm corresponds to around 1000 lux, the so-called “resistance under illumination”). The characteristic curve for resistance indicates the relationship between resistance value and illumination level. It is logarithmic (note the scales):



1a. Now, use a light meter (an app, for example) to complete multiple measurements, and enter the resistance value and measured lux value in a table. You can use a spreadsheet program to obtain a simple formula for your measured values in the form of $y = a \cdot x^b$, which you can use to calculate the approximate illumination level in lux based on the resistance value.

1b. Add a display of illumination level in lux to the Blockly program from programming task 2.

1c. Define a suitable threshold value for the minimum illumination level in the classroom. What do the workplace regulations require, for instance [4]? Connect the second LED to O7 and O8 (ensure correct polarity). It should be switched on by your Blockly program if it gets too dark in the room.

2. CO₂ traffic light

The environmental sensor cannot determine the level of CO₂ in the air directly, but it can detect the air breathed out by people using the other gases contained in it. Therefore, the air quality measured by the sensor is directly connected to the CO₂ content in the air as a result of breathing out.

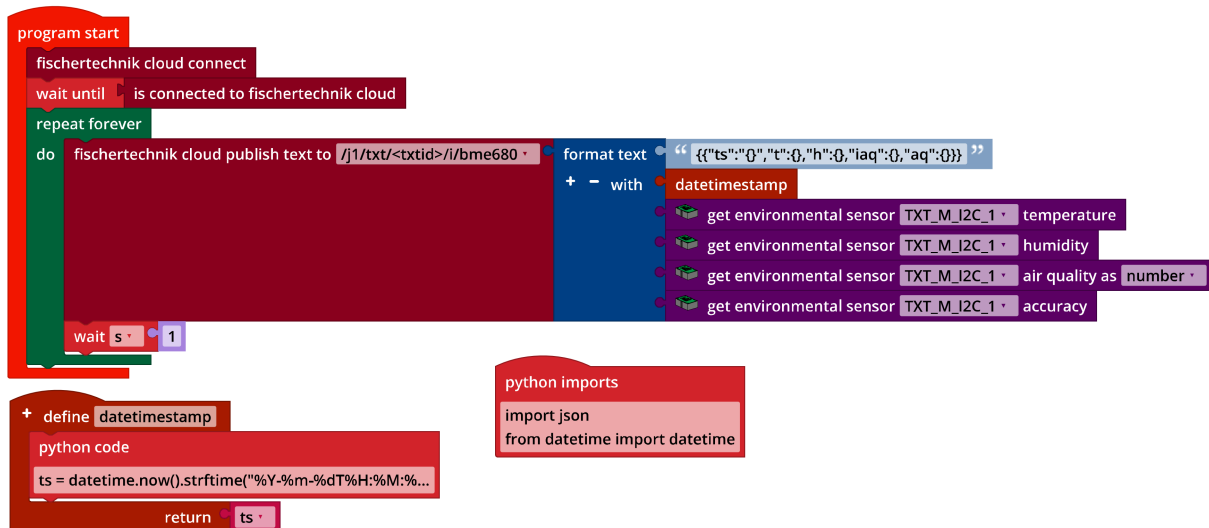
The CO₂ content in the air is around 400 ppm (*parts per million*). Since people emit around 40,000 ppm (= 4%) CO₂ in the air they breathe out (8-10 litres per minute), the percentage of CO₂ in the air increases continuously in an enclosed room where multiple people are present. If it reaches 1,200 ppm, 2% of the air in the room would have already been breathed out once – therefore, every 50th breath would consist of air that had been breathed out. A CO₂ level of 1,000-2,000 ppm, therefore, is considered concerning by the Federal Environmental Agency; in schoolrooms in particular, an average of 1,000 ppm should not be exceeded [5].

Small droplets (aerosols) are also contained in the air we breathe out; these may contain viruses that can stay in the air for a long time (dropping at a speed of 10 cm/h at a virus infectious activity half-life period of 2.7 hours). Therefore, increasing humidity in the air is a risk indicator that viruses may be spreading in the room.

Briefly ventilating the area to allow air to pass through can quickly and effectively reduce the CO₂ level and humidity. Ventilating every 20 minutes is recommended in classrooms to protect against infection [6].

2a. First, determine threshold values for humidity and air quality that should be used for ventilating the classroom. To do so, you can determine the measured values in the classroom after 20 minutes, for example. Target values are the humidity and air quality levels you can achieve through ventilating.

You can display the curve for measured values, relative humidity, and air quality using the following auxiliary program in the dashboard of the IoT server:



IoT_MQTT_Indoor_Air_Quality.ft

Use the threshold values to derive a “ventilation index” (0-100 with: 0 = ideal state after ventilation, 100 = ventilation urgently required), including the relative humidity and air quality in the calculation with a weighting of 1:3.

2b. Program a sub-program that causes the red LED to flash every 0.5 seconds. The red flashing light should be activated at a ventilation index of 100.

2c. Connect the second LED (with the green cap) to O7 and O8 (note the polarity) and activate it when your ventilation index reaches “0”. Compare the ventilation duration when only one window, two windows, ... all windows and door(s) are opened.

3. Noise level

The noise level in our environment can also influence our well-being. Volume is measured as the sound pressure level, and is stated using the unit decibels (dB). 0 dB corresponds to the human auditory threshold (pressure fluctuations around 20 micropascal); 120 dB is usually the pain threshold for human listeners.

The (perceived) volume increases logarithmically with the decibel level: It is roughly doubled every time the sound pressure level increases by 10 dB. The sound pressure level, in turn, depends on the distance from the source of the noise: Tripling the distance will cut the volume in half (reduction in sound pressure level by 10 dB).

Comparisons can help us get a feeling for the volumes of certain decibel levels: A ticking alarm clock (1 m distance) is around 30 dB, a normal discussion is around 60

dB, a bus driving by is around 80 dB, a pneumatic drill is 100 dB, and a jet aircraft is 120 dB.

High volume levels can have significant negative impacts:

- People can have trouble concentrating at just 40 dB
- Volumes above 60 dB can cause hearing damage following long-term exposure
- The risk of cardiovascular illness increases at continuous levels over 65+ dB
- Over 120 dB, hearing damage may occur after just a short time

3a. Add a scale to display volume (in dB) to the display of your program from programming task 2 (or experimental task 1).

3b. The microphone in the camera will indicate the volume received in dB. Add the volume measurement and display to your Blockly program.

3c. Define a threshold value above which the red LED should flash. Expand your program accordingly.

Annex

Task 2: Indoor climate

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- Program “IoT_MQTT_Indoor_Air_Quality.ft”

Further information

- [1] Bosch Sensortec: *BME680 – Application Note*. Rev. 1.6, 20.09.2020.
- [2] Bosch Sensortec: *BME680 – Data sheet*. Rev. 1.7, 20.12.2021.
- [3] fischertechnik: [Photoresistor LDR03 \(32698\)](#). Data sheet, 17.10.2018.
- [4] German Federal Ministry of Labour and Social Affairs: [Beleuchtung](#). Technische Regeln für Arbeitsstätten (Lighting: technical work station regulations), ASR A3.4, April 2011.
- [5] Federal Environmental Agency: [Gesundheitliche Bewertung von Kohlendioxid in der Innenraumluft \(Health evaluation of carbon dioxide in indoor air\)](#). Mitteilungen der Ad-hoc-Arbeitsgruppe Innenraumrichtwerte der Innenraumluftthygiene-Kommission des Umweltbundesamtes und der Obersten Landesgesundheitsbehörden, Bundesgesundheitsblatt – Gesundheitsforschung – Gesundheitsschutz (Reports of the ad hoc working group on indoor reference values of the indoor air hygiene commission of the Federal Environmental Agency and Higher State Health Agencies, Federal health gazette - health research - health protection) 2008, 51, p. 1358–1369.
- [6] Deutsche Gesetzliche Unfallversicherung (DGUV - German Social Accident Insurance): [Coronavirus SARS-CoV-2 – Ergänzende Empfehlungen der gesetzlichen Unfallversicherung für die Gefährdungsbeurteilung in Schulen \(Supplementary recommendations of the social accident insurance for risk assessment in schools\)](#). 03/12/2021

Task 2: Indoor climate

The sensor station will be used to analyse the indoor climate, and will issue a warning if the specified limit values are exceeded or not met. The measurement series can be collected using the IoT server and then analysed.

Topics

- Measuring factors that impact indoor climate (relative humidity, temperature, air quality, brightness, volume)
- Defining and measuring a “ventilation indicator” for classrooms (“CO₂ traffic light”)
- Determining illumination level

Learning objectives

- Understanding the term relative humidity
- Understanding how to measure noise level (decibels)
- Understanding the term illumination level (lux) and determining this value based on the values from a photoresistor
- Learning typical limit values for relative humidity, temperature, volume, and illumination level
- Deriving an index value from multiple measured values based on a specified weighting

Time required

If the sensor station was constructed previously in task 1, then no time is required for construction.

60 minutes is the suggested time to solve the programming tasks (building on the weather station from task 1).

Solving experimental tasks 1 and 2 requires the student to create a mathematical mode; students may require support to do so. The suggested time required to develop the program (depending on programming experience) is 60 - 120 minutes.

Annex

Task 2: Indoor climate

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- Program “IoT_MQTT_Indoor_Air_Quality.ft”

Further information

- [1] Bosch Sensortec: *BME680 – Application Note*. Rev. 1.6, 20.09.2020.
- [2] Bosch Sensortec: *BME680 – Data sheet*. Rev. 1.7, 20.12.2021.
- [3] fischertechnik: [Photoresistor LDR03 \(32698\)](#). Data sheet, 17.10.2018.
- [4] German Federal Ministry of Labour and Social Affairs: [Beleuchtung](#). Technische Regeln für Arbeitsstätten (Lighting: technical work station regulations), ASR A3.4, April 2011.
- [5] Federal Environmental Agency: [Gesundheitliche Bewertung von Kohlendioxid in der Innenraumluft \(Health evaluation of carbon dioxide in indoor air\)](#). Mitteilungen der Ad-hoc-Arbeitsgruppe Innenraumrichtwerte der Innenraumluftthygiene-Kommission des Umweltbundesamtes und der Obersten Landesgesundheitsbehörden, Bundesgesundheitsblatt – Gesundheitsforschung – Gesundheitsschutz (Reports of the ad hoc working group on indoor reference values of the indoor air hygiene commission of the Federal Environmental Agency and Higher State Health Agencies, Federal health gazette - health research - health protection) 2008, 51, p. 1358–1369.
- [6] Deutsche Gesetzliche Unfallversicherung (DGUV - German Social Accident Insurance): [Coronavirus SARS-CoV-2 – Ergänzende Empfehlungen der gesetzlichen Unfallversicherung für die Gefährdungsbeurteilung in Schulen \(Supplementary recommendations of the social accident insurance for risk assessment in schools\)](#). 03/12/2021

Name: _____

Class: _____

Date: _____

Solution sheet task 2:

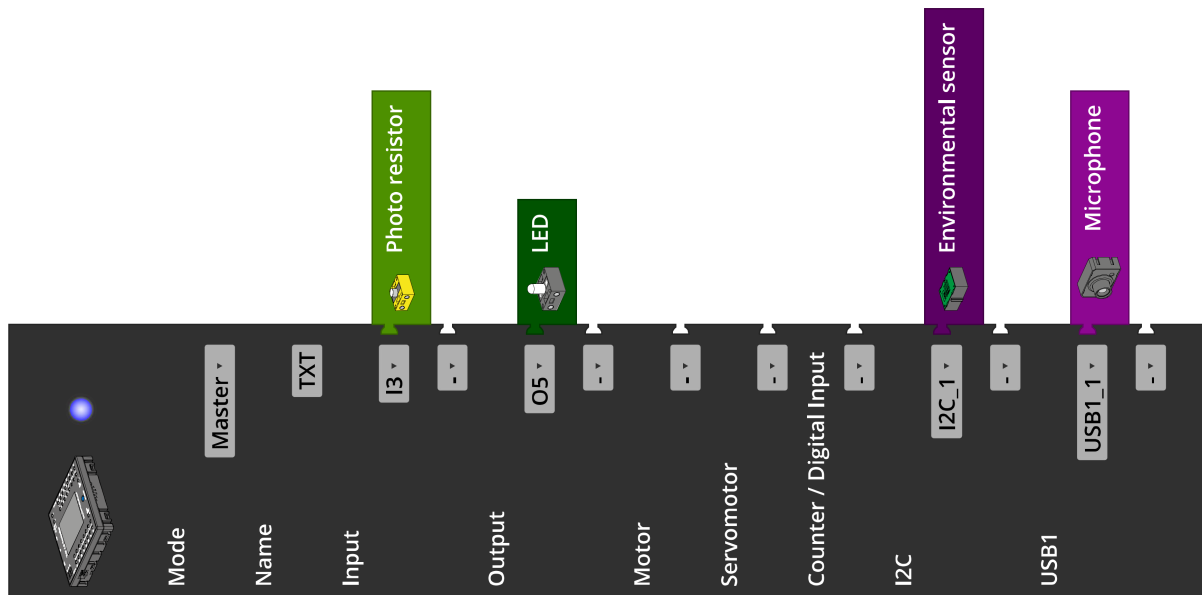
Indoor climate

Construction task

See building instructions.

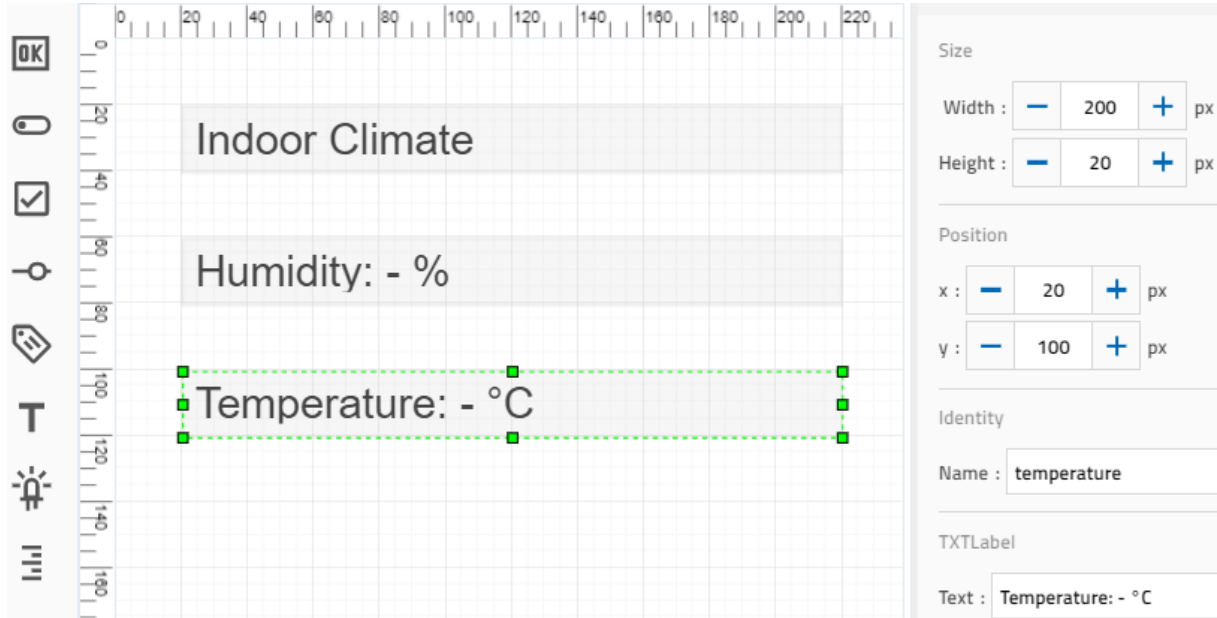
Programming tasks

Configuring the sensors and actuators:



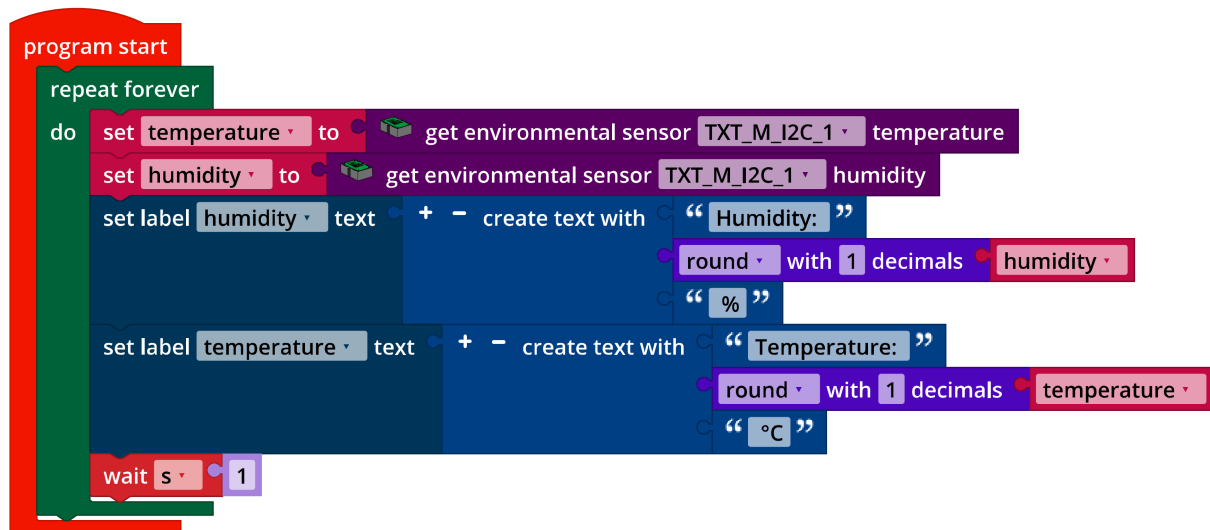
1. Measuring indoor climate values

1a. Configuring the TXT display:



Configuring the display on the TXT

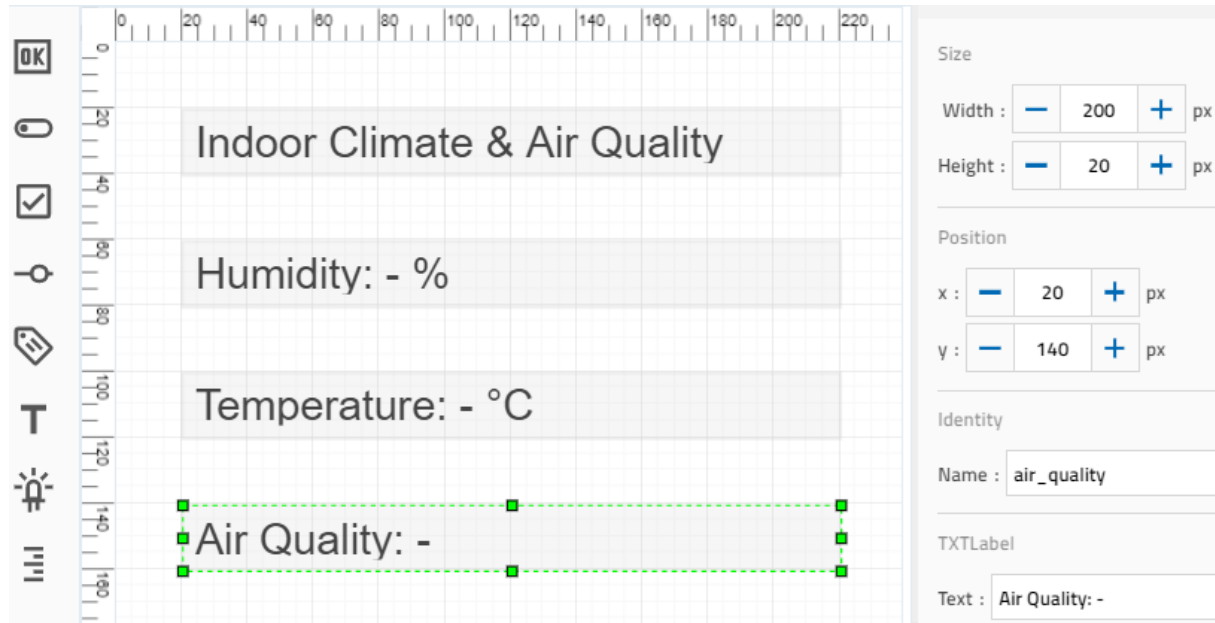
1b. Program (example):



IoT_Indoor_Climate.ft

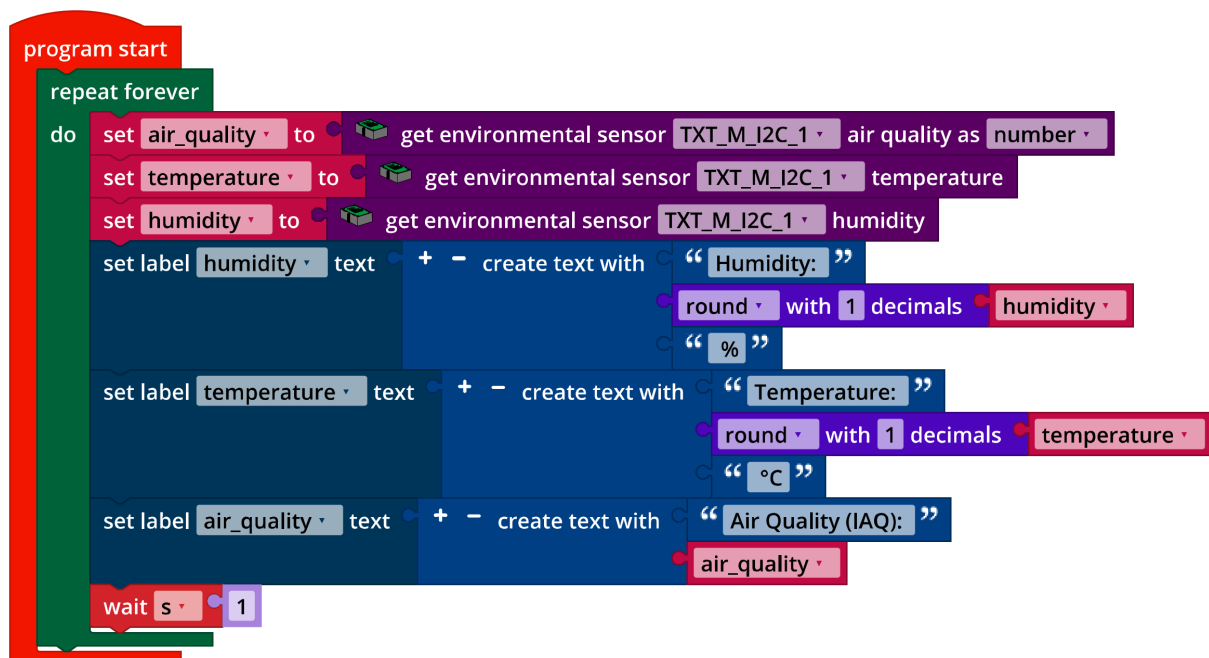
2. Measuring air quality

2a. Configuring the TXT display:



Expanding the display on the TXT

2b. Program (example):



IoT_Indoor_Air_Quality.ft

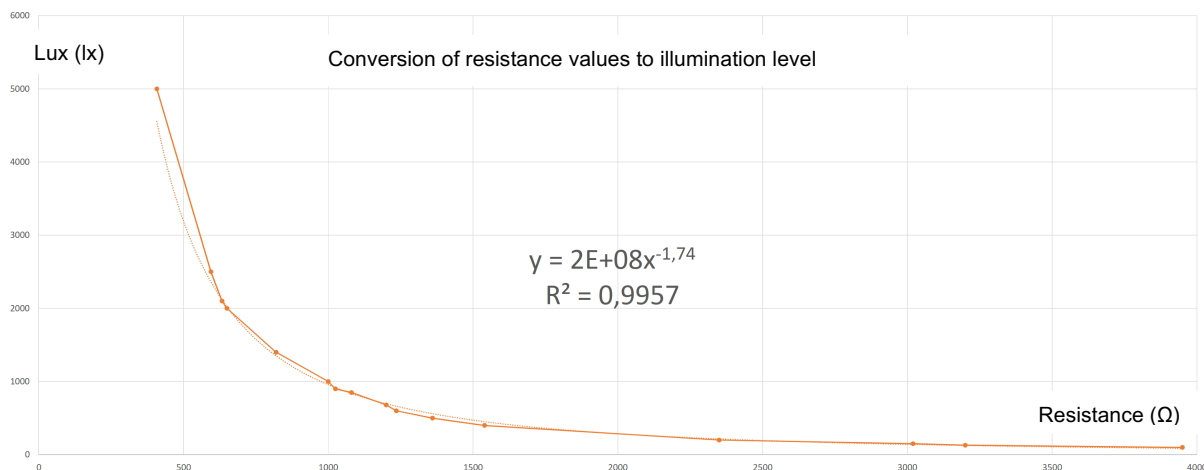
Experimental tasks

1. Brightness

The measured values (resistance/measured illumination level) should be displayed in a graph using a spreadsheet program (x: resistance, y: illumination level).

The resistance value r can be converted approximately into illumination level i using an equation derived from the characteristic curve. The equation displayed, for instance in Excel for an added trend line could be as follows (based on the daylight measurements):

$$i = 2 \cdot 10^8 \cdot r^{-1.74} \text{ lx}$$



Based on the work station regulations, a work station should have an illumination level of at least 500 lx; this is also a good minimum requirement for a work station in a classroom.

Configuring the TXT display:

Expanding the display output on the TXT

Program (example):

```

program start
set illuminance_threshold to 500
repeat forever
do
set resistance to get photo resistor TXT_M_I3 value
set illuminance to (2 x 10^8) / (resistance ^ -1.74)
set air_quality to get environmental sensor TXT_M_I2C_1 air quality as number
set temperature to get environmental sensor TXT_M_I2C_1 temperature
set humidity to get environmental sensor TXT_M_I2C_1 humidity
+ if illuminance < illuminance_threshold
do
set LED TXT_M_O7 on
else
set LED TXT_M_O7 off
set label humidity text + - create text with " Humidity: "
round with 1 decimals humidity
" %"
set label temperature text + - create text with " Temperature: "
round with 1 decimals temperature
" °C "
set label air_quality text + - create text with " Air Quality: "
air_quality
set label illuminance text + - create text with " Illuminance: "
round with 0 decimals illuminance
" lx "
wait s 1

```

IoT_Illuminance.ft

2. CO₂ traffic light

2a. The “ventilation index” L should have a value up to 100, and should equal zero when the humidity and IAQ each reach the desired target value, and be 100 when the humidity and IAQ reach the “Ventilate now!” level.

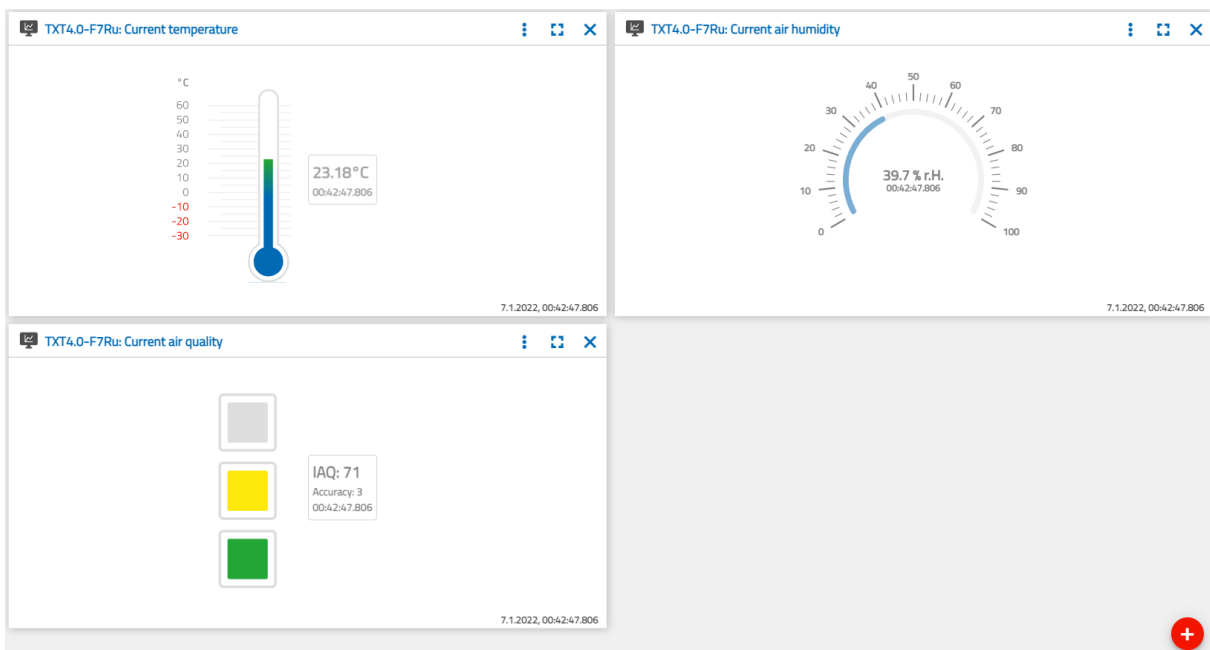
If H_0 is the target value and H_1 is the threshold value (upper limit) for relative humidity and H is the measured value; and furthermore if Q_0 is the target value, Q_1 is the threshold value (upper limit) for air quality and Q is the measured IAQ value. Then, we can define the “ventilation index” L as follows:

$$L = \left(\frac{H - H_0}{H_1 - H_0} + \frac{Q - Q_0}{Q_1 - Q_0} \cdot 3 \right) \cdot 25$$

Example calculation: If the relative humidity after ventilation should be 30% and the IAQ should be 30 (target values), and if the threshold values (such as the values after 20 minutes in the classroom without ventilation) are defined at 55% and 100, then:

$$L = \left(\frac{H - 30}{25} + \frac{Q - 30}{70} \cdot 3 \right) \cdot 25$$

Configuration of the IoT dashboard to display the measured values (temperature, relative humidity and air quality):



Configuration of the IoT dashboard

2b/c. Program excerpts (example):

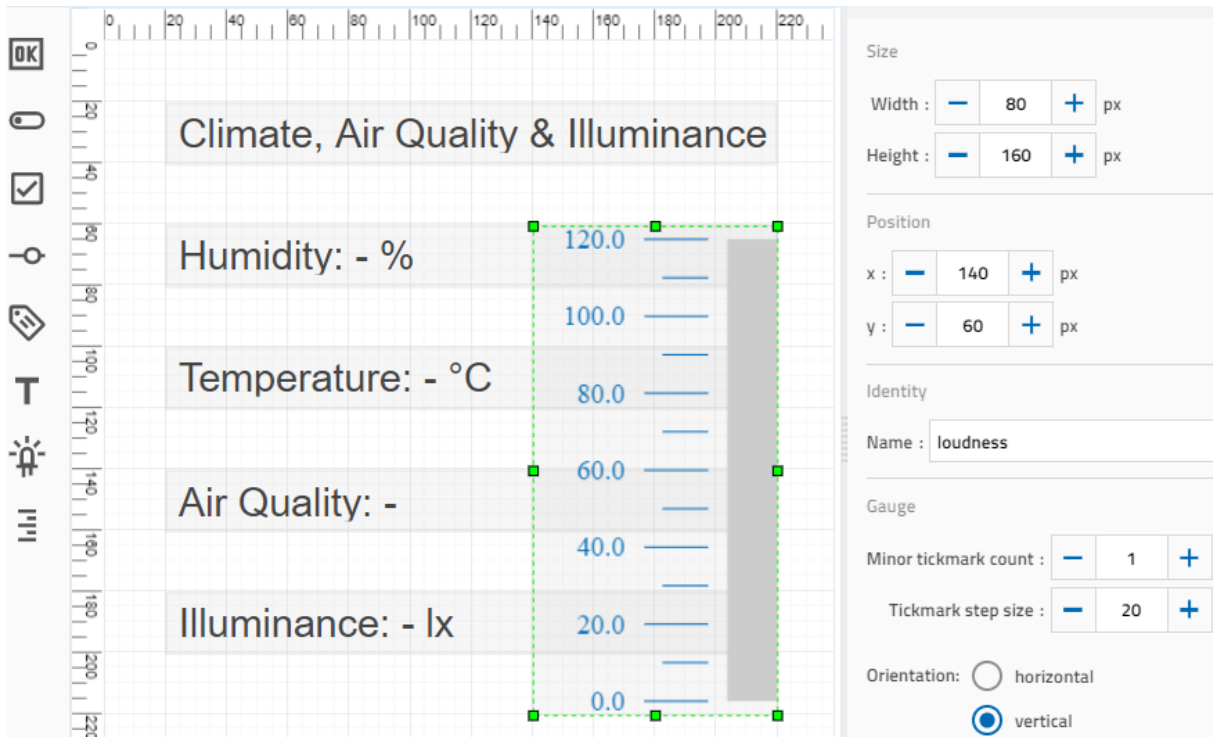
```

program start
  set air_quality_target_value to 30
  set humidity_target_value to 30
  set air_quality_range to 70
  set humidity_range to 25
  set ventilation_warning to 0
  execute function blink in a thread
  ...
  set ventilation_index to
    (humidity - humidity_target_value) / humidity_range +
    (air_quality - air_quality_target_value) / air_quality_range * 3
  ...
  if ventilation_index > 100
  do set ventilation_warning to 1
  else set ventilation_warning to 0
  if ventilation_index ≤ 0
  do set LED TXT_M_O7 on
  else set LED TXT_M_O7 off
  ...
  define blink
  repeat forever
  do if ventilation_warning = 1
  do set LED TXT_M_O5 on
  wait s 0.5
  set LED TXT_M_O5 off
  wait s 0.5
  
```

IoT_CO2_Signal_Light

3. Noise level

3a. Configuring the TXT display:



Expanding the TXT display by adding a volume scale

3c. 70 dB can be selected, for instance, as the limit value for the volume level.

Program (example):

```

program start
  set illuminance_threshold to 500
  set loudness_threshold to 70
  set loudness_warning to 0
  execute function blink in a thread

  repeat forever
  do
    set loudness to microphone TXT_M_USB1_1 volume
    set resistance to get photo resistor TXT_M_I3 value
    set illuminance to (2 * 10^8) * resistance ^ -1.74
    set air_quality to get environmental sensor TXT_M_I2C_1 air quality as number
    set temperature to get environmental sensor TXT_M_I2C_1 temperature
    set humidity to get environmental sensor TXT_M_I2C_1 humidity

    + if illuminance < illuminance_threshold
    do
      set LED TXT_M_O7 on
    else
      set LED TXT_M_O7 off

    + if loudness > loudness_threshold
    do
      set loudness_warning to 1
    else
      set loudness_warning to 0

    set label humidity text + - create text with " Humidity: "
    + round with 1 decimals humidity
    + "%"

    set label temperature text + - create text with " Temperature: "
    + round with 1 decimals temperature
    + " °C"

    set label air_quality text + - create text with " Air Quality: "
    + air_quality

    set label illuminance text + - create text with " Illuminance: "
    + round with 0 decimals illuminance
    + " lx"

    set gauge loudness value loudness
  wait s 1

+ define blink
  repeat forever
  do
    + if loudness_warning = 1
    do
      set LED TXT_M_O5 on
      wait s 0.5
      set LED TXT_M_O5 off
      wait s 0.5
  
```

IoT_Loudness.ft

Annex

Task 2: Indoor climate

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- Program “IoT_MQTT_Indoor_Air_Quality.ft”

Further information

- [1] Bosch Sensortec: *BME680 – Application Note*. Rev. 1.6, 20.09.2020.
- [2] Bosch Sensortec: *BME680 – Data sheet*. Rev. 1.7, 20.12.2021.
- [3] fischertechnik: [Photoresistor LDR03 \(32698\)](#). Data sheet, 17.10.2018.
- [4] German Federal Ministry of Labour and Social Affairs: [Beleuchtung](#). Technische Regeln für Arbeitsstätten (Lighting: technical work station regulations), ASR A3.4, April 2011.
- [5] Federal Environmental Agency: [Gesundheitliche Bewertung von Kohlendioxid in der Innenraumluft \(Health evaluation of carbon dioxide in indoor air\)](#). Mitteilungen der Ad-hoc-Arbeitsgruppe Innenraumrichtwerte der Innenraumluftthygiene-Kommission des Umweltbundesamtes und der Obersten Landesgesundheitsbehörden, Bundesgesundheitsblatt – Gesundheitsforschung – Gesundheitsschutz (Reports of the ad hoc working group on indoor reference values of the indoor air hygiene commission of the Federal Environmental Agency and Higher State Health Agencies, Federal health gazette - health research - health protection) 2008, 51, p. 1358–1369.
- [6] Deutsche Gesetzliche Unfallversicherung (DGUV - German Social Accident Insurance): [Coronavirus SARS-CoV-2 – Ergänzende Empfehlungen der gesetzlichen Unfallversicherung für die Gefährdungsbeurteilung in Schulen \(Supplementary recommendations of the social accident insurance for risk assessment in schools\)](#). 03/12/2021

Name: _____

Class: _____

Date: _____

Task 3

Alarm system

Now, you will turn your sensor station into an alarm and surveillance station: It should react to noises and movements, and allow you to monitor the room using the IoT dashboard.

Construction task

You can use the sensor station you built in task 1 without modification.

Before testing the program, you should check to ensure the wiring provides enough play to turn the camera to the left and right at a wide angle.

Programming tasks

1. Home position

The camera in your alarm system must first be moved to home position. To do so, you must first turn the camera horizontally and vertically – based on the respective drive motor – anti-clockwise until the associated stop position sensors are activated, so that you know the exact position of the camera. From there, you can turn it to the desired home position.

1a. By how many pulses of the encoder motor must the camera be turned upward vertically from the end position to align it horizontally to the front? Determine the value first using the gear ratio.

Then, check the result experimentally using an appropriate Blockly program that first turns the camera to the end position, and then aligns it horizontally.

Note: As a reminder: The encoder motor outputs 63.9 pulses per axis rotation.

1b. By how many pulses of the encoder motor must the camera be turned clockwise horizontally from the end position to align it “straight ahead”? Determine the value first using the gear ratio. Then, check the result experimentally using an appropriate Blockly program that first turns the camera to the end position, and then aligns it to the front.

Note: The slewing ring has 58 teeth.

1c. Combine both programs into a function that aligns the camera in both dimensions. How can you conduct the two motions in parallel?

2. Camera surveillance

In task 1, you programmed the weather station so that it sent an image from the webcam to the dashboard every second. You can now monitor the room with the camera in the same way.

Expand your program from programming task 1 by adding image transmission to the dashboard. Determine the maximum number of images you can transmit each second that will be displayed on the dashboard.

3. Noise activation

Security personnel can become fatigued if they have to continuously watch screens where nothing is happening. Therefore, image transmission should begin only when the microphone registers an unusual noise.

3a. First, test the “normal” noise level in the room using a simple Blockly program. Display the values on the TXT. Then, set a suitable threshold value above which image transmission should begin.

Note: The Blockly command for the microphone will indicate the measured volume to you in decibels (dB).

3b. Write a Blockly program that begins transmitting the camera image each second once the threshold value you have defined for noise level has been exceeded. If things stay quiet for more than one minute, image transmission should stop.

4. Motion detection

You can also set the camera surveillance so that it is activated by a motion in the room. Configure the camera motion detection for this purpose.

4a. Expand your Blockly program from programming task 3 so that image transmission is also activated when a motion is detected.

4b. The red LED should also flash while the camera is activated.

Experimental tasks

1. Voice control

The camera should now be controlled using voice commands, so that it can monitor a larger portion of the room. To do so, download the “Voice Control” app from the Apple app store (for iOS) or the Google Play store (for Android), and connect it to the TXT 4.0.

- Connection via WiFi: The TXT 4.0 Controller and device (smartphone or tablet) must be connected to the same WiFi router. The router must also permit communication between the devices. The IP address of the TXT 4.0 with which the app must be connected can then be queried via the touchscreen menu under “Info” / “WiFi”.
- Connection via WiFi AP: Instead of “WiFi”, the “Access Point” option can be activated on the TXT 4.0 under “Settings” / “Network”. Then, the smartphone can be connected directly to the controller. The WPA2 key required for the WiFi connection is available in the TXT menu under “Access Point” (it can also be changed and deactivated there).

Once you have connected the app to the Controller, the voice commands are transmitted to the Controller in text form, and you can analyse them using the following event function:



1a. Write functions for rotation to the right, to the left, up, and down. Since voice recognition reacts at a slight delay, it makes sense to turn the camera by an angle transmitted as a parameter each time it is accessed.

Consider the following: How can you prevent the camera from turning too far?

1b. Select appropriate voice commands for the camera control. Add the control functions from sub-task 1a to your program from programming task 4.

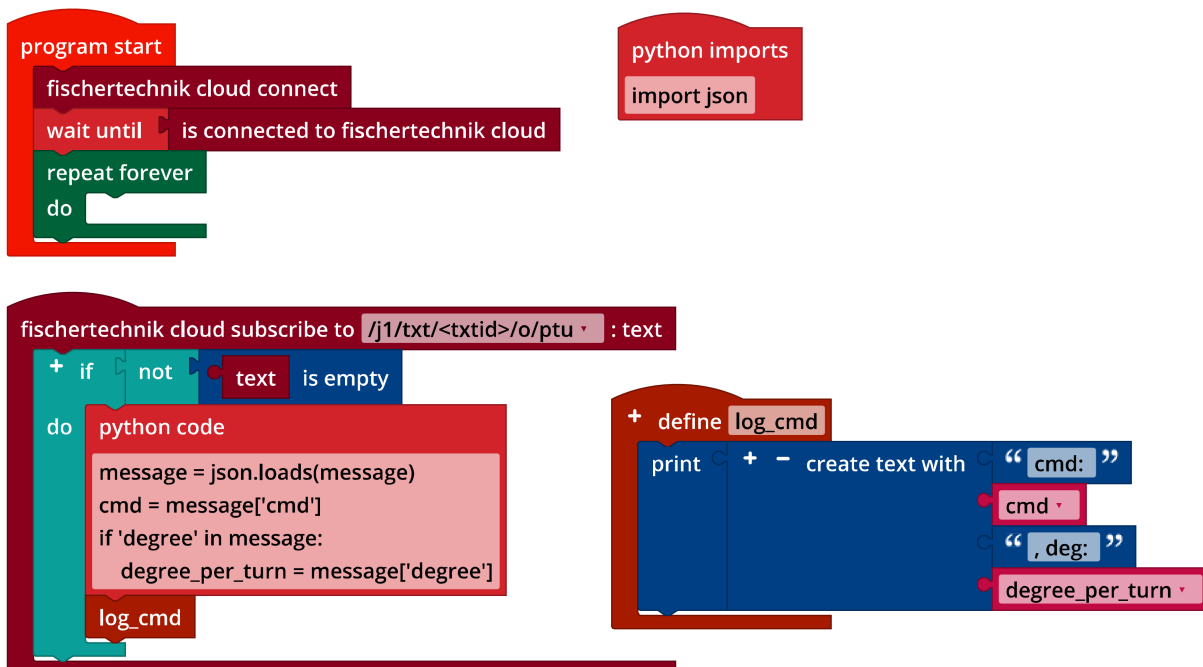
Now, control the camera using your voice commands. Ensure the camera does not turn too far.

2. Controlling the camera using the cloud dashboard

You can also control the camera using the IoT dashboard in the fischertechnik cloud. To do so, after connecting to the IoT server, you must register the TXT as an “MQTT subscriber” using the following function:

fischertechnik cloud subscribe to `/j1/txt/<txtid>/o/ptu` : text

2a. Then, you can read out the command you clicked using the mouse with the following test program:



The code is organized into several sections:

- program start:**
 - fischertechnik cloud connect
 - wait until is connected to fischertechnik cloud
 - repeat forever loop:
 - do:
 - fischertechnik cloud subscribe to `/j1/txt/<txtid>/o/ptu` : text
- python imports:**
 - import json
- if not text is empty:**
 - do:
 - python code:


```
message = json.loads(message)
cmd = message['cmd']
if 'degree' in message:
    degree_per_turn = message['degree']
```
 - log_cmd
- define log_cmd:**
 - print:
 - create text with:
 - “cmd: ”
 - cmd
 - “, deg: ”
 - degree_per_turn

IoT_Test_Dashboard_Control.ft

2b. In addition to the control commands from experimental task 1, there is also a command here designed to move the camera to home position, and one that stops the motors immediately. Add the control commands accordingly.

2c. Now, replace the camera voice control with control via the dashboard.

Annex

Task 3: Alarm system

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- fischertechnik “Voice Control“ app
- Auxiliary program “IoT_Test_Dashboard_Control.ft”
- Account in the fischertechnik cloud (www.fischertechnik-cloud.com)

Task 3: Alarm system

The sensor station is converted into an alarm system with surveillance camera that reacts to different triggers (noise, movement) and transmits a camera image to a web server. The camera can be controlled from the IoT dashboard.

Topic

- Voice and remote control of a camera (“pan-tilt control”)
- Motion and noise activation
- Use of a web server for remote control of an IoT system

Learning objectives

- Precise control via encoder motors
- Suitable structuring of a program using functions
- Understanding the MQTT protocol

Time required

If the sensor station was constructed previously in task 1, then no time is required for construction.

The suggested time to solve the four programming tasks is 90-120 minutes. Likewise, it should be possible to solve the experimental tasks (depending on the students’ programming experience) within 90-120 minutes.

Annex

Task 3: Alarm system

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- fischertechnik “Voice Control“ app
- Auxiliary program “IoT_Test_Dashboard_Control.ft”
- Account in the fischertechnik cloud (www.fischertechnik-cloud.com)

Name: _____

Class: _____

Date: _____

Solution sheet task 3:

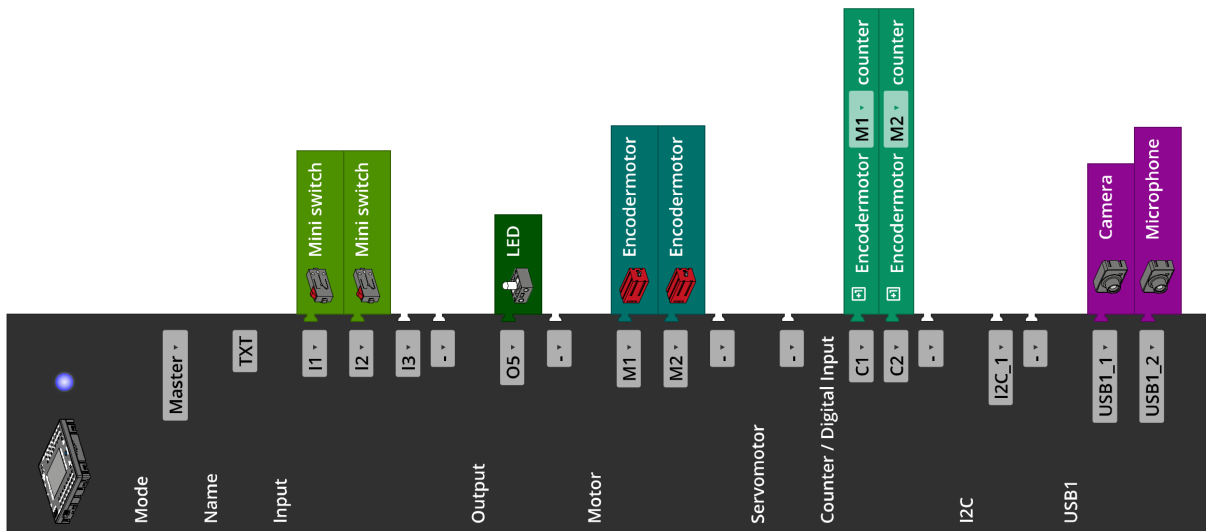
Alarm system

Construction task

See building instructions.

Programming tasks

Configuring the sensors and actuators:



The “Voice Control” app (for iOS or Android) is required to solve experimental task 2. The app must be connected to the internet for voice recognition, and connected to the Controller (via Bluetooth or WiFi).

1. Home position

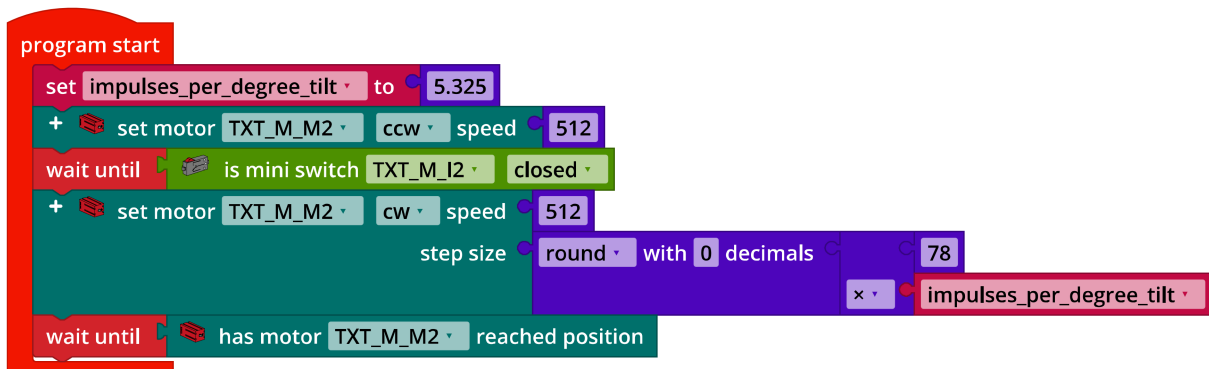
1a. Each rotation of the motor axis corresponds to one rotation of the worm, and each rotation of the worm turns the Z30 by one tooth. Therefore, 30/4 rotations of the motor axis are required to turn the Z30 by 90°. This corresponds (at 63.9 encoder pulses per axial rotation)

$$30 \cdot \frac{63.9}{4} = 479.25$$

Encoder pulses (or 5.325 pulses per angle degree).

In reality, the end position switch is not triggered exactly after one quarter turn; the camera must be turned only around 78°. This corresponds to $78/90 \cdot 479.25 = 415.35$ encoder pulses.

Program (example):



```

program start
set impulses_per_degree_tilt to 5.325
+ set motor TXT_M_M2 ccw speed 512
wait until is mini switch TXT_M_I2 closed
+ set motor TXT_M_M2 cw speed 512
step size round with 0 decimals 78
x impulses_per_degree_tilt
wait until has motor TXT_M_M2 reached position
    
```

IoT_Init_Camera_Tilt.ft

1b. The slewing ring has 58 teeth; a 90° turn, therefore, requires 58/4 rotations of the motor axis. This corresponds to

$$58 \cdot \frac{63.9}{4} = 926.55$$

Encoder pulses (or 10.295 pulses per angle degree).

Here as well, the end position switch is not triggered only after a 90° turn, it is triggered already at approx. 3°; the camera must be turned by only around 87°. This corresponds to $87/90 \cdot 926.55 = 895.67$ encoder pulses.

Program (example):

```

program start
  set impulses_per_degree_pan to 10.295
  + set motor TXT_M_M1 ccw speed 512
  wait until is mini switch TXT_M_I1 closed
  + set motor TXT_M_M1 cw speed 512
    step size round with 0 decimals 87
    × impulses_per_degree_pan
  wait until has motor TXT_M_M1 reached position
  
```

IoT_Init_Camera_Pan.ft

1c. When combining both programs, the two movements can be conducted in parallel, thereby significantly shortening the initialisation time. There are two potential solutions for this: Execute the two functions as parallel threads, or integrating them into one function.

Program solution method A – one function (example):

```

+ define Init_Camera
  + set motor TXT_M_M1 ccw speed 512
  + set motor TXT_M_M2 ccw speed 512
  repeat while
    is mini switch TXT_M_I1 open
    or
    is mini switch TXT_M_I2 open
  do
    + if is mini switch TXT_M_I1 closed
      do
        + stop motor TXT_M_M1
      else if is mini switch TXT_M_I2 closed
        do
          + stop motor TXT_M_M2
    + set motor TXT_M_M1 cw speed 512
      step size round with 0 decimals 87
      × impulses_per_degree_pan
    + set motor TXT_M_M2 cw speed 512
      step size round with 0 decimals 78
      × impulses_per_degree_tilt
  wait until
    has motor TXT_M_M1 reached position
    and
    has motor TXT_M_M2 reached position
  
```

IoT_Init_Camera_A.ft

Program solution method B – parallel threads (example):

```

program start
  set impulses_per_degree_pan to 10.295
  set impulses_per_degree_tilt to 5.325
  execute function Init_Pan in a thread
  execute function Init_Tilt in a thread
  repeat forever
  do

+ define Init_Pan
  + set motor TXT_M_M1 ccw speed 512
  wait until is mini switch TXT_M_I1 closed
  + set motor TXT_M_M1 cw speed 512
  step size round with 0 decimals 87
  x impulses_per_degree_pan
  wait until has motor TXT_M_M1 reached position

+ define Init_Tilt
  + set motor TXT_M_M2 ccw speed 512
  wait until is mini switch TXT_M_I2 closed
  + set motor TXT_M_M2 cw speed 512
  step size round with 0 decimals 78
  x impulses_per_degree_tilt
  wait until has motor TXT_M_M2 reached position
  
```

IoT_Init_Camera_B.ft

2. Camera surveillance

Program (example):

```

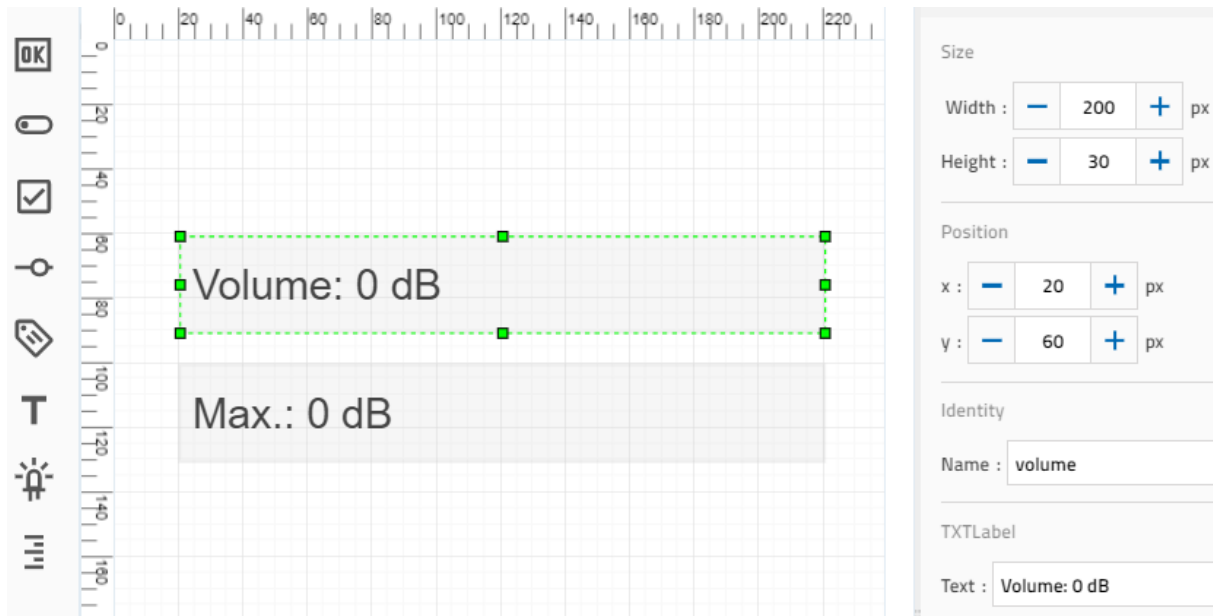
program start
  set impulses_per_degree_pan to 10.295
  set impulses_per_degree_tilt to 5.325
  set snapshot to ""
  execute function Init_Pan in a thread
  execute function Init_Tilt in a thread
  fischertechnik cloud connect
  wait until is connected to fischertechnik cloud
  repeat forever
    do fischertechnik cloud publish text to /j1/txt/<txtid>/i/cam
      format text
      + - with
      timestamp
      snapshot
    wait ms 100

on image TXT_M_USB1_1 change: event
  set snapshot to
  convert image event to base64
  
```

IoT_Surveillance_Camera.ft

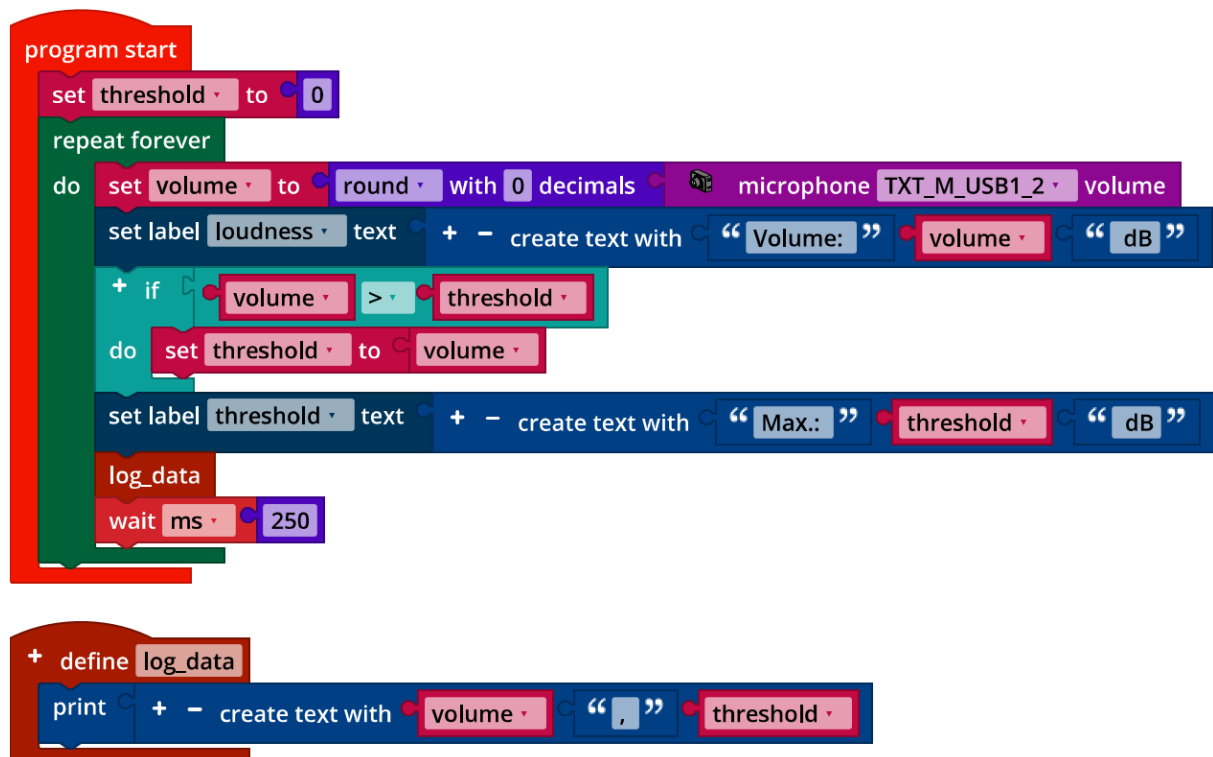
3. Noise activation

3a. Configuration of the TXT display (example):



Display configuration

Program (example):



IoT_Microphone_Level.ft

3b. Program with volume threshold 70 dB (example):

```

program start
  set impulses_per_degree_pan to 10.295
  set impulses_per_degree_tilt to 5.325
  set snapshot to ""
  set threshold to 70
  set surveillance_period to 60
  set surveillance_status to 0
  set start_time to Timestamp s
  execute function Init_Pan in a thread
  execute function Init_Tilt in a thread
  fischertechnik cloud connect
  wait until is connected to fischertechnik cloud
  repeat forever
    do + if microphone TXT_M_USB1_2 volume > threshold
      do set start_time to Timestamp s
        set surveillance_status to 1
      + if surveillance_status = 1
        and Timestamp s - start_time > surveillance_period
      do set surveillance_status to 0
      + if surveillance_status = 1
      do fischertechnik cloud publish text to /j1/txt/<txtid>/i/cam
        format text {"ts":0,"data":0}
        + - with datetimestamp
        snapshot
      log_data
      wait s 1
  end repeat

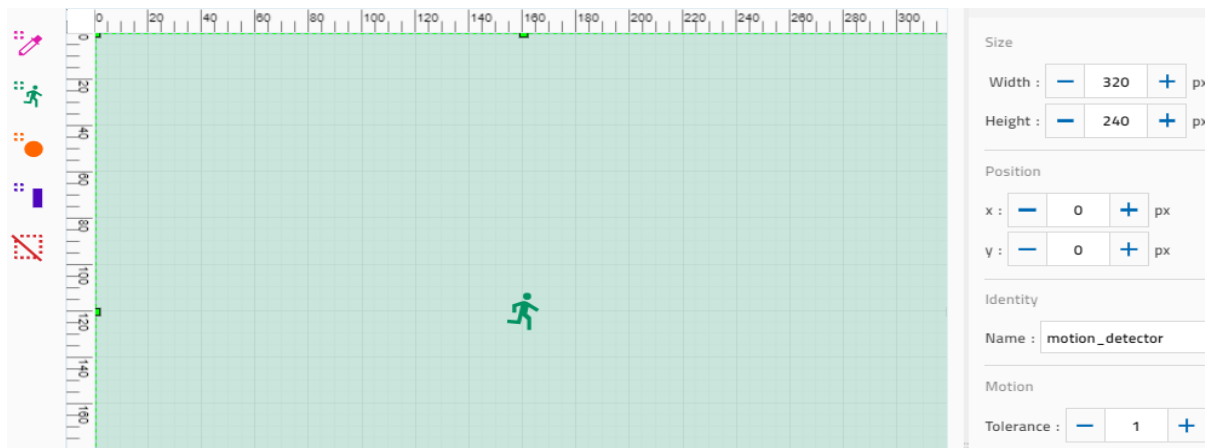
  on image TXT_M_USB1_1 change: event
  set snapshot to convert image event to base64

  + define log_data
  print + - create text with surveillance_status
  "("
  round with 0 decimals Timestamp s - start_time
  "s)"
  
```

IoT_Surveillance_Camera_Noise_Detection.ft

4. Motion detection

Motion detection is defined over the entire camera window. The “tolerance” can be used to set the sensitivity of detection to a value of 0 to 1 in the inspector:



Configuring motion detection

4a. Program excerpt (example):

```

on motion motion_detector detected: event
  set start_time to Timestamp s
  set surveillance_status to 1
  
```

IoT_Surveillance_Camera_Motion_Detection.ft

If a motion is detected, this is saved in a flag (“surveillance_status”), along with the time of detection.

4b. The “flash” function is started as a thread at the beginning of the main program:

```

+ define blink
  repeat forever
    do + if surveillance_status = 1
      do
        set LED TXT_M_05 on
        wait ms 500
        set LED TXT_M_05 off
        wait ms 500
  
```

IoT_Surveillance_Camera_Motion_Detection.ft

Experimental tasks

1. Voice control

1a. Control functions (example):

```

+ define turn_right with:
- variable: degree
+ set motor TXT_M_M1 cw speed 512
step size round with 0 decimals degree impulses_per_degree_pan
wait until has motor TXT_M_M1 reached position
    
```

```

+ define turn_left with:
- variable: degree
+ set motor TXT_M_M1 ccw speed 512
step size round with 0 decimals degree impulses_per_degree_pan
wait until has motor TXT_M_M1 reached position
    
```

```

+ define turn_up with:
- variable: degree
+ set motor TXT_M_M2 cw speed 512
step size round with 0 decimals degree impulses_per_degree_tilt
wait until has motor TXT_M_M2 reached position
    
```

```

+ define turn_down with:
- variable: degree
+ set motor TXT_M_M2 ccw speed 512
step size round with 0 decimals degree impulses_per_degree_tilt
wait until has motor TXT_M_M2 reached position
    
```

IoT_Camera_Motion_Control.ft

The easiest way to prevent the camera from turning too far is by limiting the rotational angle in both the horizontal and vertical direction to a total of $\pm 90^\circ$ each. If you set both deflection angles to 0° after initialising home position, then the program is easy to check.

1b. Program excerpt: Camera control (example):

IoT_Surveillance_Camera_Voice_Control.ft

The following variables are initialised at the beginning of the main program:

IoT_Surveillance_Camera_Voice_Control.ft

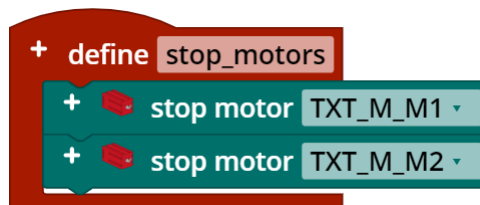
The maximum and minimum deflection angles should be adjusted to the design if necessary (cabling range).

2. Controlling the camera using the cloud dashboard

The control knobs on the dashboard deliver the commands “stop”, “home”, “relmove_left”, “relmove_right”, “relmove_up”, and “relmove_down”. The “relmove” commands correspond to the four control functions from experimental task 1. If a value “degree” (2, 5, 10, or 20°) is transmitted by the IoT server, the rotational angle specified in the program must be adjusted. The “home” command moves the camera to the home position; a function should be added for the “stop” command that stops the two motors.

The query of the control command can be taken from the solution to experimental task 1.

“stop_motors” function (example):



IoT_Surveillance_Camera_Dashboard_Control.ft

Program excerpt: Camera control (example):

```
fischertechnik cloud subscribe to /j1/txt/<txtid>/o/ptu : text
+ if not text is empty
do python code
  message = json.loads(message)
  cmd = message['cmd']
  if 'degree' in message:
    degree_per_turn = message['degree']
camera_control
```

```
+ define camera_control
+ if cmd = "relmove_left" and pan_angle + degree_per_turn ≤ max_angle
do turn_left with:
  degree degree_per_turn
  set pan_angle to pan_angle + degree_per_turn
else if cmd = "relmove_right" and pan_angle - degree_per_turn ≥ min_angle
do turn_right with:
  degree degree_per_turn
  set pan_angle to pan_angle - degree_per_turn
else if cmd = "relmove_up" and tilt_angle + degree_per_turn ≤ max_angle
do turn_up with:
  degree degree_per_turn
  set tilt_angle to tilt_angle + degree_per_turn
else if cmd = "relmove_down" and tilt_angle - degree_per_turn ≥ min_angle
do turn_down with:
  degree degree_per_turn
  set tilt_angle to tilt_angle - degree_per_turn
else if cmd = "home"
do init_Camera
else if cmd = "stop"
do stop_motors
```

IoT_Surveillance_Camera_Dashboard_Control.ft

Annex

Task 3: Alarm system

Required materials

- PC for program development, local or via web interface.
- USB cable or BLE or WiFi connection for transmitting the program to the TXT4.0.
- fischertechnik “Voice Control“ app
- Auxiliary program “IoT_Test_Dashboard_Control.ft”
- Account in the fischertechnik cloud (www.fischertechnik-cloud.com)