ARDUINO – THINGWORX COMPOSER SCHULUNG

MÄRZ 2019

VORTRAGENDER: ING. ARMIN FISCHER, M.SC. (TGM WIEN)

E-MAIL: AFISCHER2@TGM.AC.AT

VERSION 1.0

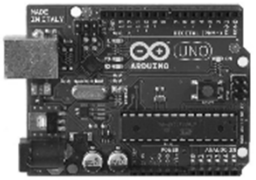
EINFÜHRUNG ARDUINO

WAS IST ARDUINO?

- Elektronische Open Source Plattform (Hardware und Software)
- Mikrocontroller mit In- und Outputs
- Verschiedene Typen erhältlich (Inputs, Outputs, Schnittstellen, Stromverbrauch, …)
- Eigene Arduino Programmiersprache (ähnlich C/C++)
- Einsteigerfreundlich
- Riesige Arduinocommunity mit Projekten erhältlich
- Offizielle Homepage https://www.arduino.cc/

WAS IST ARDUINO?

Arduino Uno



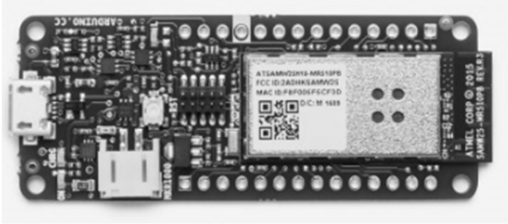
Arduino Mega 2560

Arduino Pro Mini

Arduino Leonardo with Headers

SCHULUNGSMATERIAL

ARDUINO MKR1000

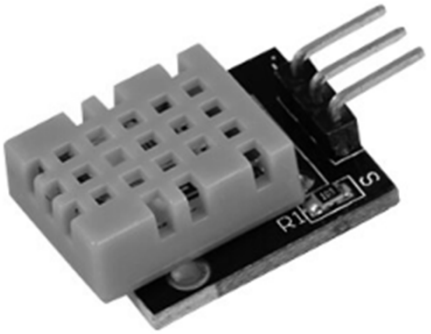


- Stromsparender Mikrocontroller für IOT Anwendungen
- Stromversorgung 5V (USB)
- Wifi Anbindung
- 7 Analoge (0-3.3V) und 14 digitale Pins (LOW-0V & HIGH-3.3V)
- ACHTUNG: Pins dürfen nur bis 3.3V belastet werden
- Weitere Informationen siehe https://store.arduino.cc/arduino-mkr1000

ARDUINO MKR RELAY PROTO SHIELD

- MKR1000 aufsteckbar
- 2 Relays mit digitalem Pin 1 und 2 verbunden (D1 & D2)
- Klemmen um A1-A4 leicht zu verbinden
- Eigener Bereich um Sensoren, Schalter, LED's usw. selbst zu applizieren.
- Weitere Informationen siehe https://store.arduino.cc/mkr-relay-proto-shield

DHT11 TEMPERATUR- UND FEUCHTIGKEITSSENSOR



- Messung von Temperatur 0-50°C
- Messung von Luftfeuchtigkeit 20-90%
- Betriebsspannung 3.3-5V
- Eingebauter 1kOhm Widerstand
- Abtastrate maximal alle 2 Sekunden
- Pinbelegung

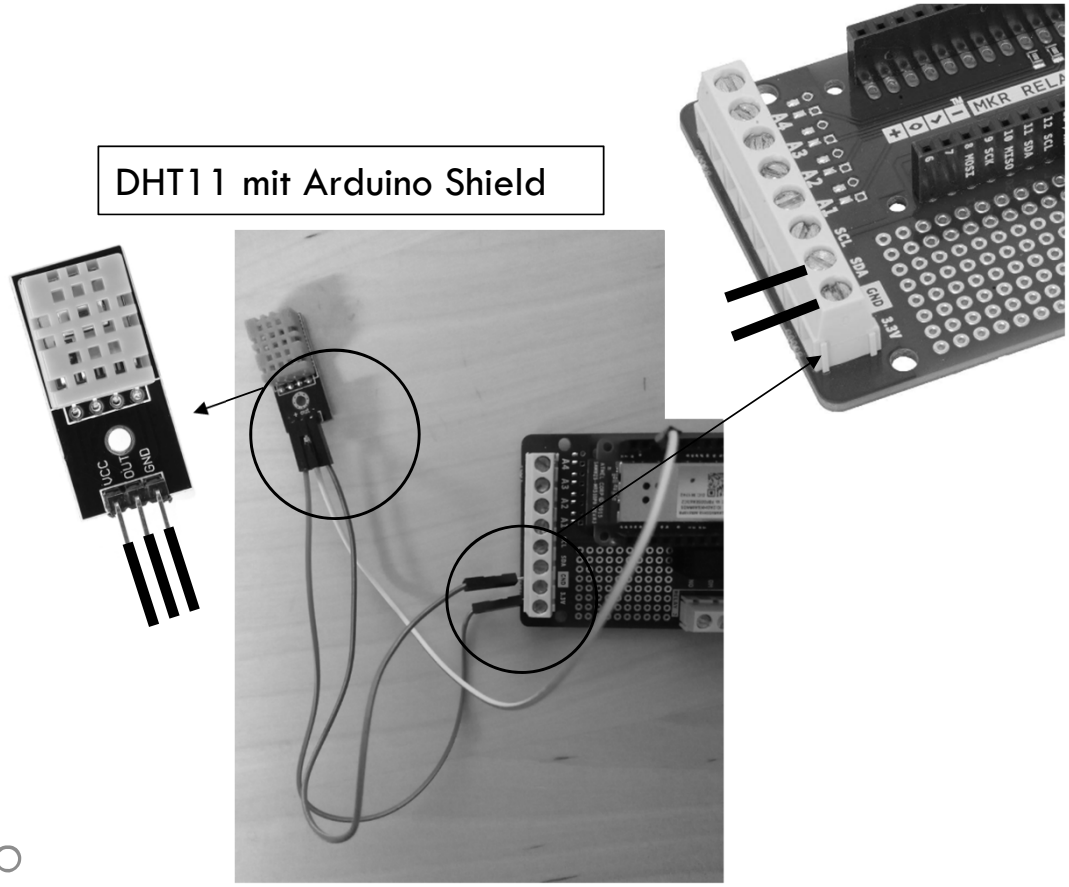
- Weitere Informationen siehe http://sensorkit.en.joy-it.net/index.php?title=KY-015_Combi-Sensor_Temperature%2BHumidity

VENTILATOR

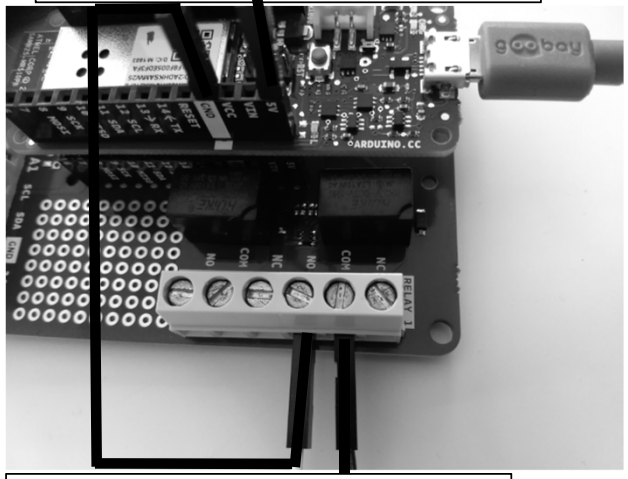
- Betriebsspannung 5VDC
- Weitere Informationen siehe www.reichelt.at/ Prduktnummer: RPI FAN 30X30

VERBINDEN DER EINZELTEILE

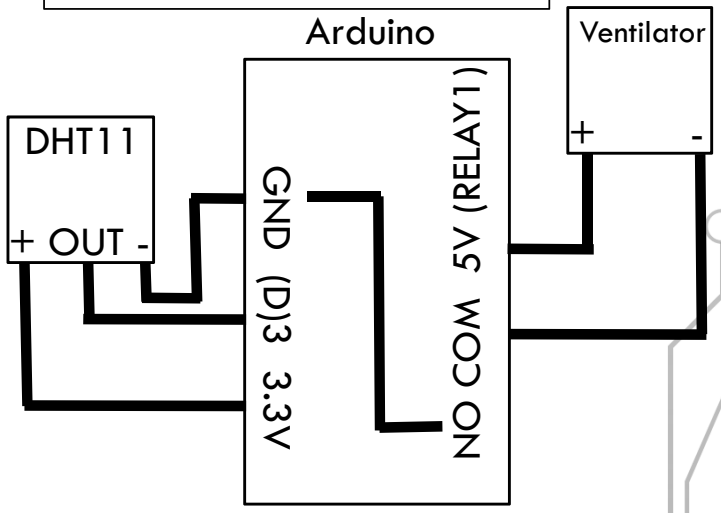
DHT11 mit Arduino Shield



Ventilator



Verbindungsplan



AUFGABENSTELLUNG

AUFGABENSTELLUNG

- Messen der Raumtemperaur und -feuchtigkeit mittels DHT11 Sensor
- Übertragen der Daten in die Cloud
- In der Cloud befindet sich die Software Thingworx Composer
- Zuordnen der übertragenen Daten in Thingworx Composer
- Einschalten des Ventilators über das Internet
- Einbindung der Sensordaten in Vuforia Studio

DATENÜBERTRAGUNG

WIE WERDEN DATEN IN DIE CLOUD ÜBERTRAGEN?

- Thing 1
  - Property 1
  - Property 2
  - ...

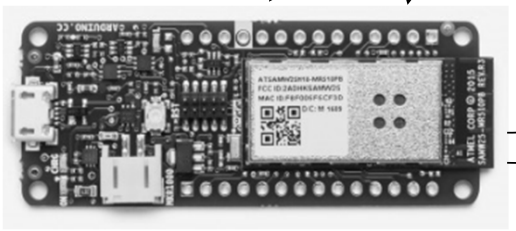
- Thing 2
  - Property 1
  - Property 2
  - ...

CLOUD https://twx.htl.schule
(Server Mödling mit
Thingworx Composer Software)

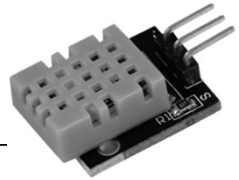
- Arduino (Thing)
  - Temperatur (Property)
  - Feuchtigkeit (Property)
  - Ventilator (Property)

Kommunikation Arduino und Cloud Http Request

Sicherheitsschlüssel zur Anmeldung



Arduino MKR1000 (Thing)



DHT11 (Property)

Ventilator (Property)

HTTP REQUEST

- Protokoll zum Übertragen von Daten (genauso wie im Browser)
- 1 HTTP Request um 1 Sensorwert (Property) vom Arduino (Thing) auf die Cloud zu übertragen
- HTTP Request enthält Sicherheitsschlüssel um eine automatische Authentifizierung/Anmeldung zur Cloud zu erhalten
- Beispiel

Thingworx Server Adresse | Thing | Property

https://twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Properties/Sensorwert_1

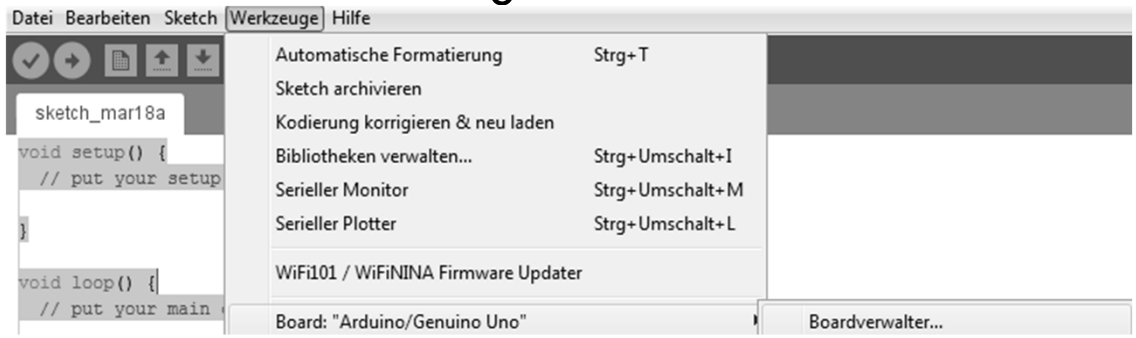
?appKey=de5d03ab-6cac-4e13-a4a9-9b5bfe9f7eda

Sicherheitsschlüssel

REALISIERUNG DER DATENÜBERTRAGUNG MIT ARDUINO

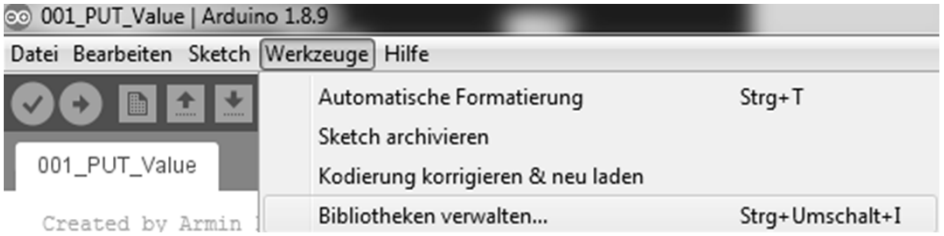
ARDUINO IDE

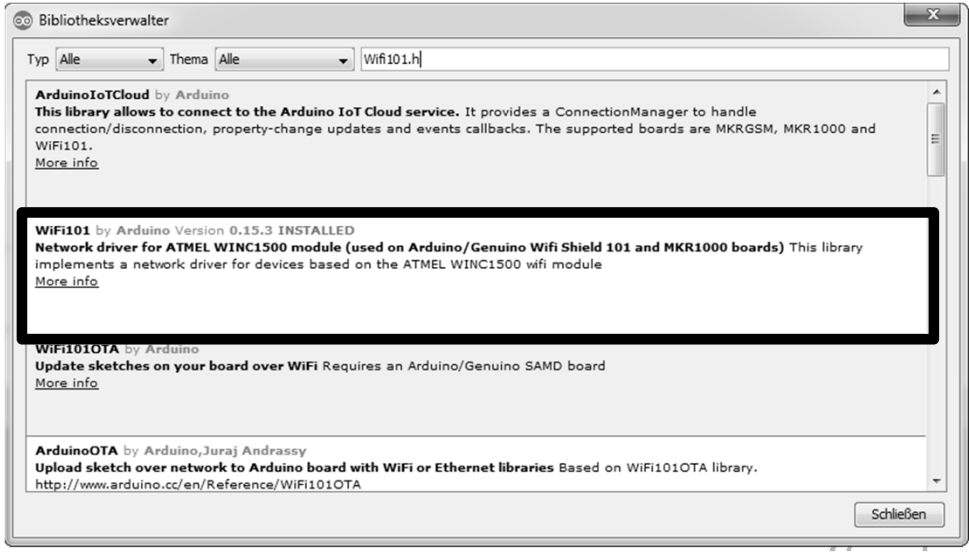
- Software zum Programmieren von Arduino Mikrocontroller
- Download unter https://www.arduino.cc/en/main/software
- Um das Arduino MKR1000 Board verwenden zu können muss dies installiert werden: Werkzeuge > Board > Boardverwalter > Arduino SAMD Boards



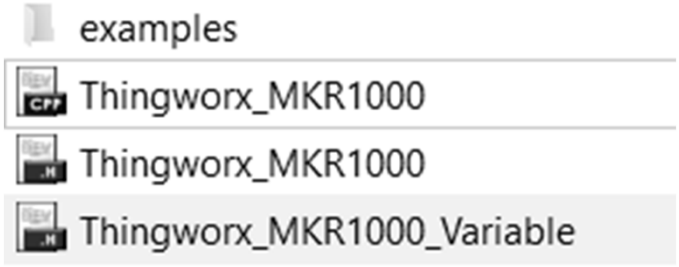
INSTALLATION NÖTIGER LIBRARIES

- Libraries = Vorprogrammierte Codes, welche mit Befehlen aufgerufen werden
- Benötige Libraries für den HTTP Request
  - Thingworx MKR1000 (von USB Stick)
  - Wifi101.h (Werkzeuge > Bibliotheken verwalten > Wifi101.h
  - Dht.h (von USB Stick)





THINGWORX MKR1000 LIBRARY



- Library von Armin Fischer geschrieben
- Auf Schulungs USB Stick gespeichert (Thingworx_MKR1000)
- Muss in folgenden Ordner kopiert werden \Documents\Arduino\libraries
- Mit dieser Library sind folgende Operationen möglich
  - Verbinden mit WLAN
  - Abfragen eines Wertes am Thingworx Server
  - Schicken eines Wertes zum Thingworx Server

THINGWORX MKR1000 LIBRARY - EINSTELLUNGEN

- Die folgenden Informationen müssen in der Datei Thingworx_MKR1000_Variable.h eingestellt werden

```
#ifndef Thingworx_MKR1000_Variable_H
#define Thingworx_MKR1000_Variable_H


const unsigned long TPOST = 5000;  //Time between requests to TWX server (every 5 secs)

//Wifi - Variables
char* ssid = ""; //WiFi SSID
char* password = ""; //WiFi Pass

//Host Thingworx
char* host = "twx.htl.schule";  //TWX Host for HTL Austria twx.htl.schule (without http at beginning)
unsigned int port = 8080; //TWX host port

//Thingworx Variables
char appKey[] = "4db65fd0-58bb-49f1-8670-8837b4f24bc0"; //API Key from TWX
char thingName[] = "";  //Thing name from TWX
//->Timing Vars
unsigned long lastConnectionTime = 0; //Last connection ms time between server requests
```

Benutzername und Passwort von WLAN

Sicherheitsschlüssel und Thingname

THINGWORX MKR1000 LIBRARY - EINSTELLUNGEN

- Für die ersten Beispiele werden folgende Einstellungen verwendet
  - SSID und Passwort von Schulnetzwerk (oder HOTSPOT von Handy)
  - Voreingestellten Sicherheitsschlüssel (kann sich durch Serverumstellung verändern)
  - Thingname 920417_18FISA_00_Dummy

THINGWORX MKR1000 LIBRARY – BEISPIELE

- Um die Benutzung der Library besser zu verstehen werden vorgefertigte Programme verwendet:

  - 001_PUT_Value
  - 002_PUT_DHT11_Value
  - 003_GET_Value
  - 004_GET_Relay

THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"
```

Einfügen der gebrauchten Libraries

```
// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

//Variable for Sensor Values
float sensor_1;

void setup() {
  Serial.begin(9600);                          //Serial communications with computer at 9600 bauds for debug purposes
  myThing.Wifi(ssid, password);                //Start the Wifi Connection

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)      //Send request to server every TPOST seconds
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");     //Get data with GET Request from Thingworx

    lastConnectionTime = millis();                //Refresh last connection time for if
  }
}
```

THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

//Variable for Sensor Values
float sensor_1;

void setup() {
  Serial.begin(9600);                          //Serial
  myThing.Wifi(ssid, password);                //Start t

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)    //Send req
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");    //Get data with GET Request from Thingworx

    lastConnectionTime = millis();               //Refresh last connection time for if
  }
}
```

for debug purposes

Erstellen einer Thingworx Klasse:

- Die Klasse verhält sich wie eine Variable
- In der Klasse sind die verschiedenen Library Funktionen enthalten (Verbinden mit WLAN, schicken und abfragen eines Wertes)
- Die Klasse heißt hier „myThing“ und kann beliebig verändert werden
- Die Klasse benötigt die Variablen host, port, appKey und thingname von der Datei Thingworx_MKR1000_Variable

THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

//Variable for Sensor Values
float sensor_1;

void setup() {
  Serial.begin(9600);
  myThing.Wifi(ssid, password);

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)      //Send request to server every TPOST seconds
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");     //Get data with GET Request from Thingworx

    lastConnectionTime = millis();                //Refresh last connection time for if
  }
}
```

Die void setup() Schleife ist Arduino spezifisch. Die darin enthaltenen Befehle werden einmalig beim Starten des Arduinos aufgerufen.

Serial.begin(9600): Es wird der serielle Monitor (Ausgabefenster am PC) aktiviert.

myThing.Wifi: Es wird eine Kommunikation mit dem WLAN aufgebaut.

THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

//Variable for Sensor Values
float sensor_1;

void setup() {
  Serial.begin(9600);                          //Serial                    for debug purposes
  myThing.Wifi(ssid, password);                //Start th

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)    //Send req
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");   //Get data with GET Request from Thingworx

    lastConnectionTime = millis();              //Refresh last connection time for if
  }
}
```

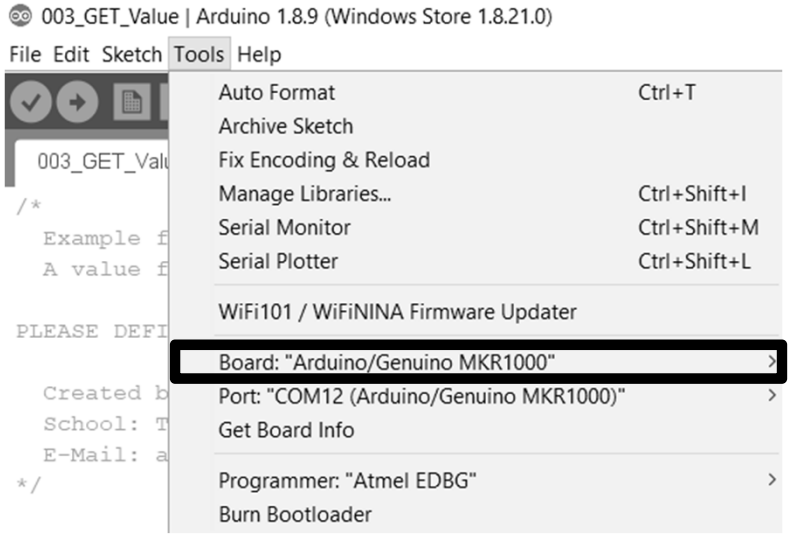
Die void loop() Schleife wird endlos ausgeführt.

If-Abfrage: Es wird alle TPOST (Variable in der Datei Thingworx_MKR1000_Variable.h) Sekunden ein Wert abgefrage.

Sensor_1=myThing.getjson():
Es wird in die Variable sensor_1 mit dem Befehl myThing.getjson (in myThing ist schon der Thingname enthalten!) die Property „Sensorwert_1“ abgerufen

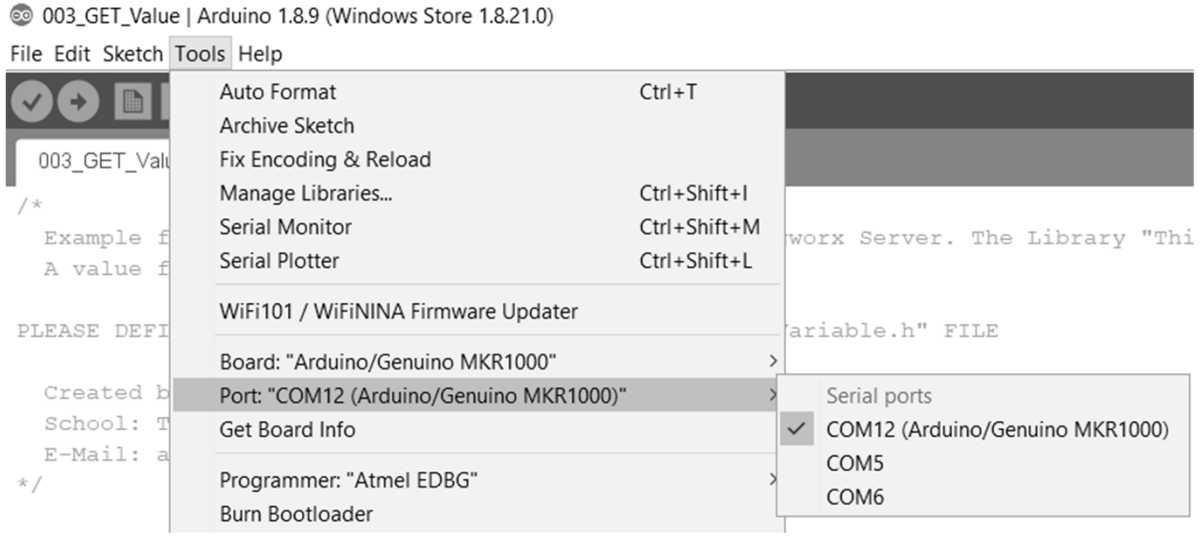
THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

- Einstellung des MKR1000 Mikrocontrollers



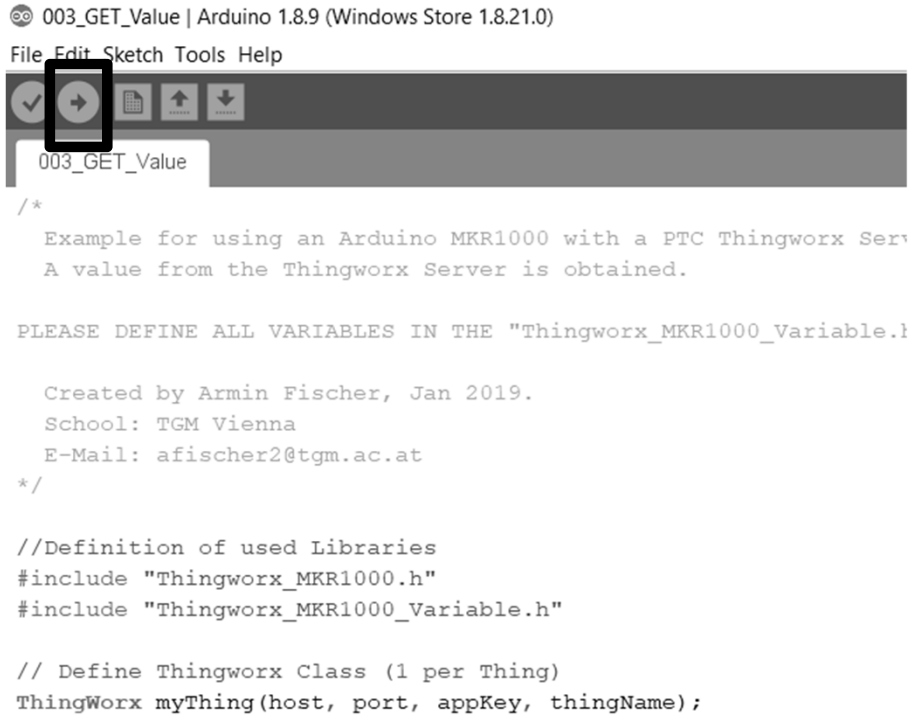
THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

- Einstellung des richtigen Port (Darstellung kann abweichen!)



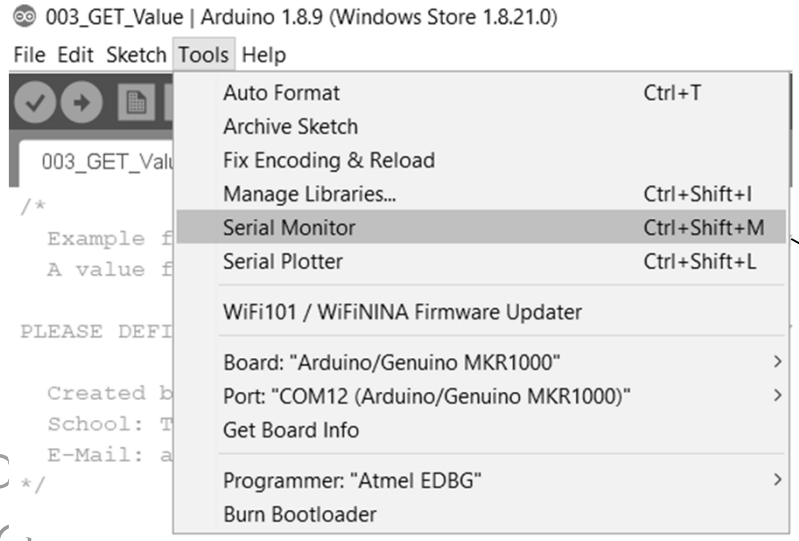
THINGWORX MKR1000 LIBRARY 003_GET_VALUE – ABFRAGEN EINES WERTES

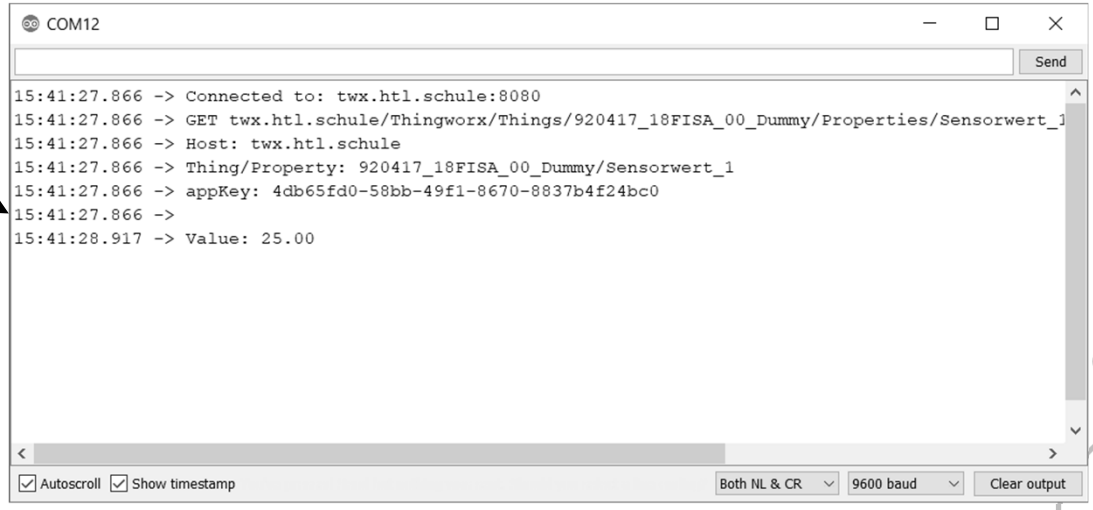
- Kompilierung (Übersetzung des Programmes in die Sprache des Mikrocontrollers) und Upload des Programmes



THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

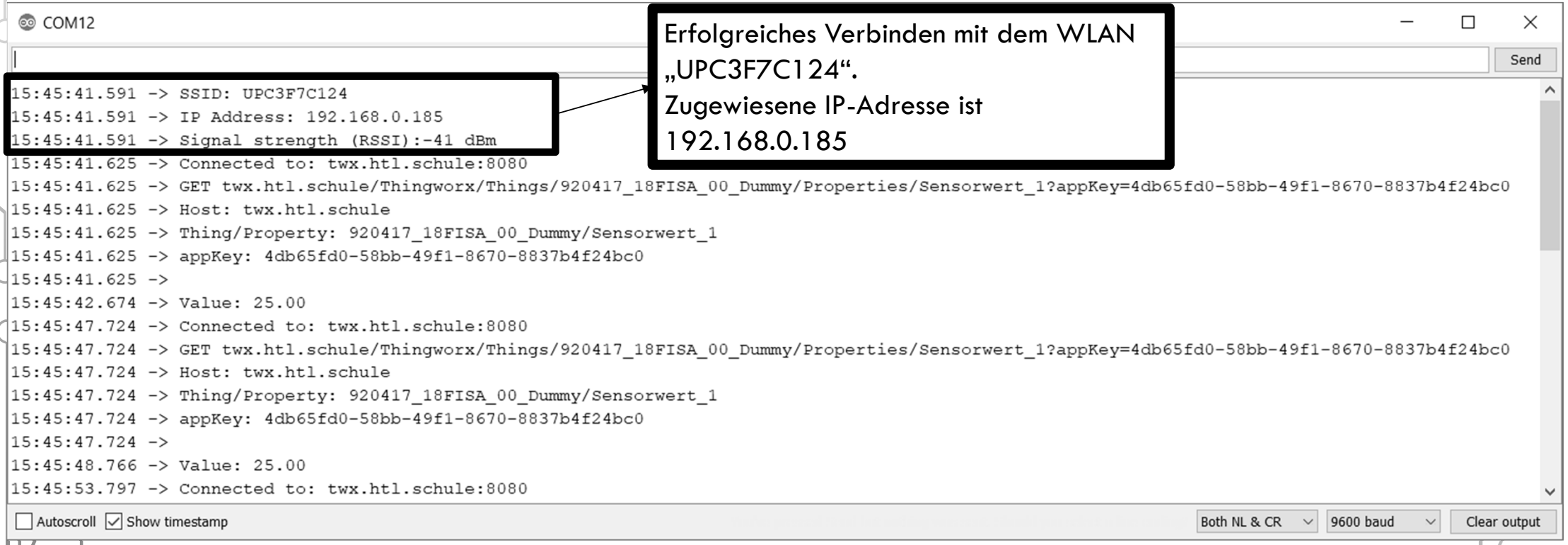
- Einschalten des Serial Monitor um den Ablauf des Programmes nachzuvollziehen





THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

- Beim Empfangen der Werte werden folgende Zeilen ausgegeben. Die Ausgabe kann zum Finden von Fehlern verwendet werden



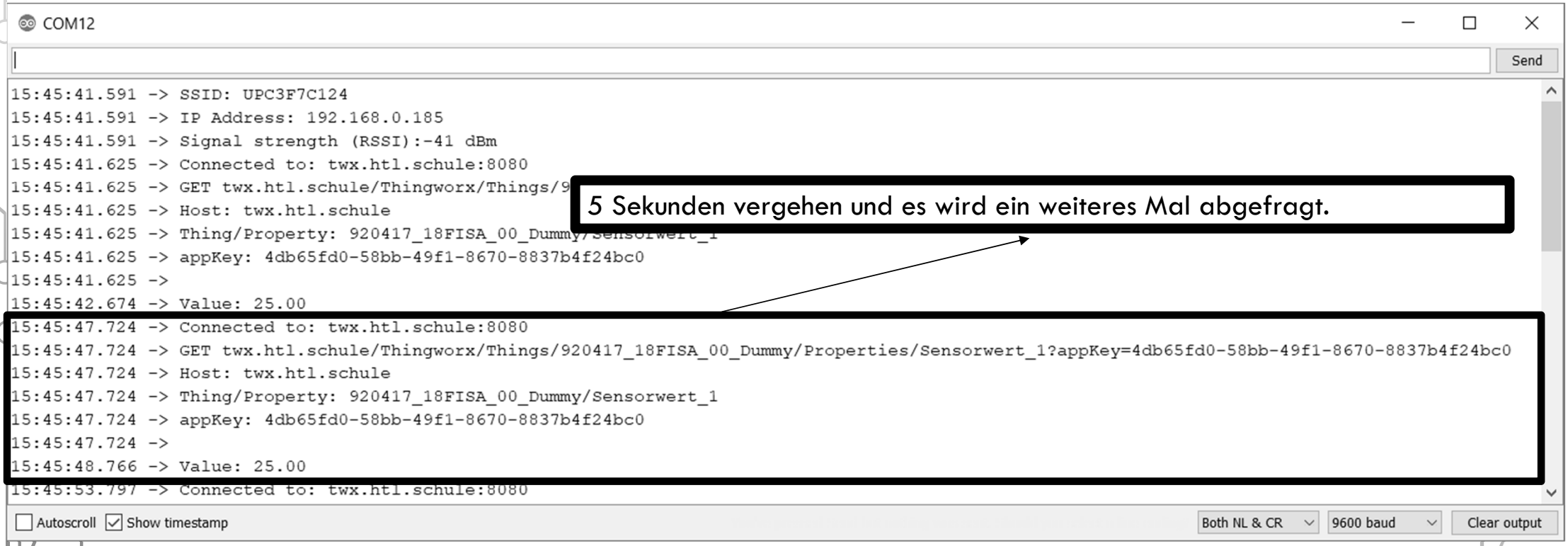
THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

- Beim Empfangen der Werte werden folgende Zeilen ausgegeben. Die Ausgabe kann zum Finden von Fehlern verwendet werden

Mit dem Server twx.htl.schule am Port 8080 verbunden.
Es wird ein GET Request geschickt mit der Adresse twx.htl.schule/Thingworx/… geschickt.
Es wird das Thing 920417_18FISA_00_Dummy angesprochen. Die Property Sensorwert_1 wird abgefragt.
Der Sicherheitsschlüssel ist 4db…
Der abgefragt Wert beträgt 25

THINGWORX MKR1000 LIBRARY
003_GET_VALUE – ABFRAGEN EINES WERTES

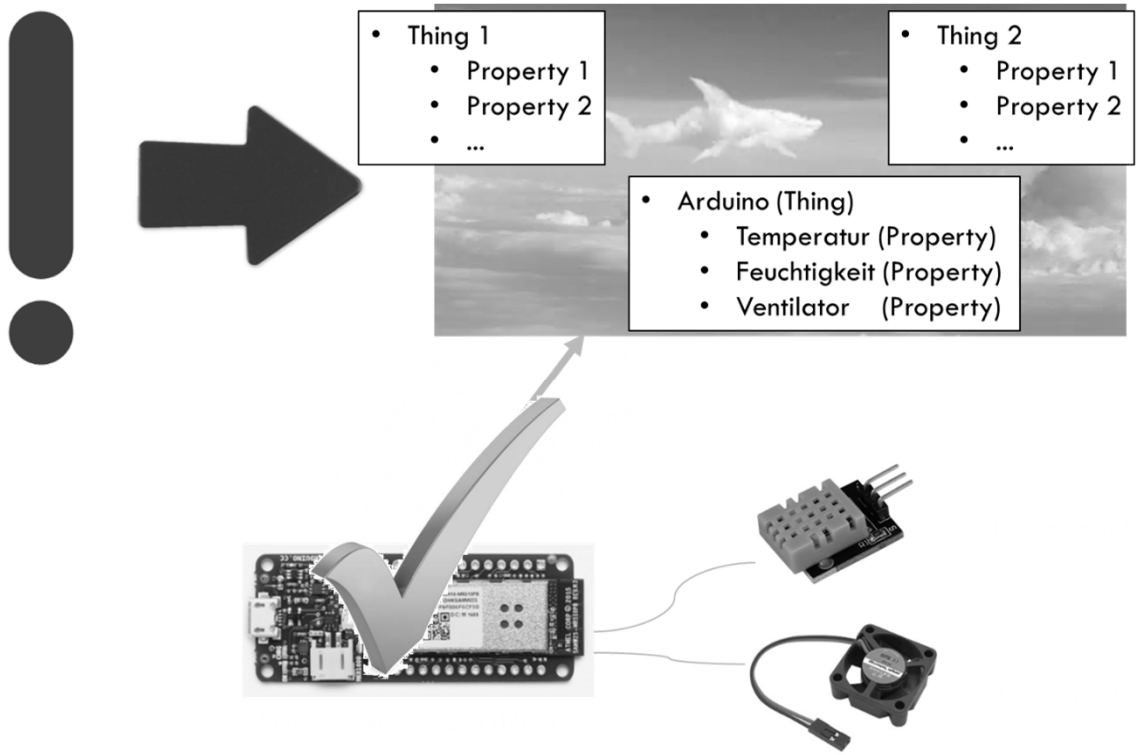
- Beim Empfangen der Werte werden folgende Zeilen ausgegeben. Die Ausgabe kann zum Finden von Fehlern verwendet werden



THINGWORX COMPOSER

EINFÜHRUNG

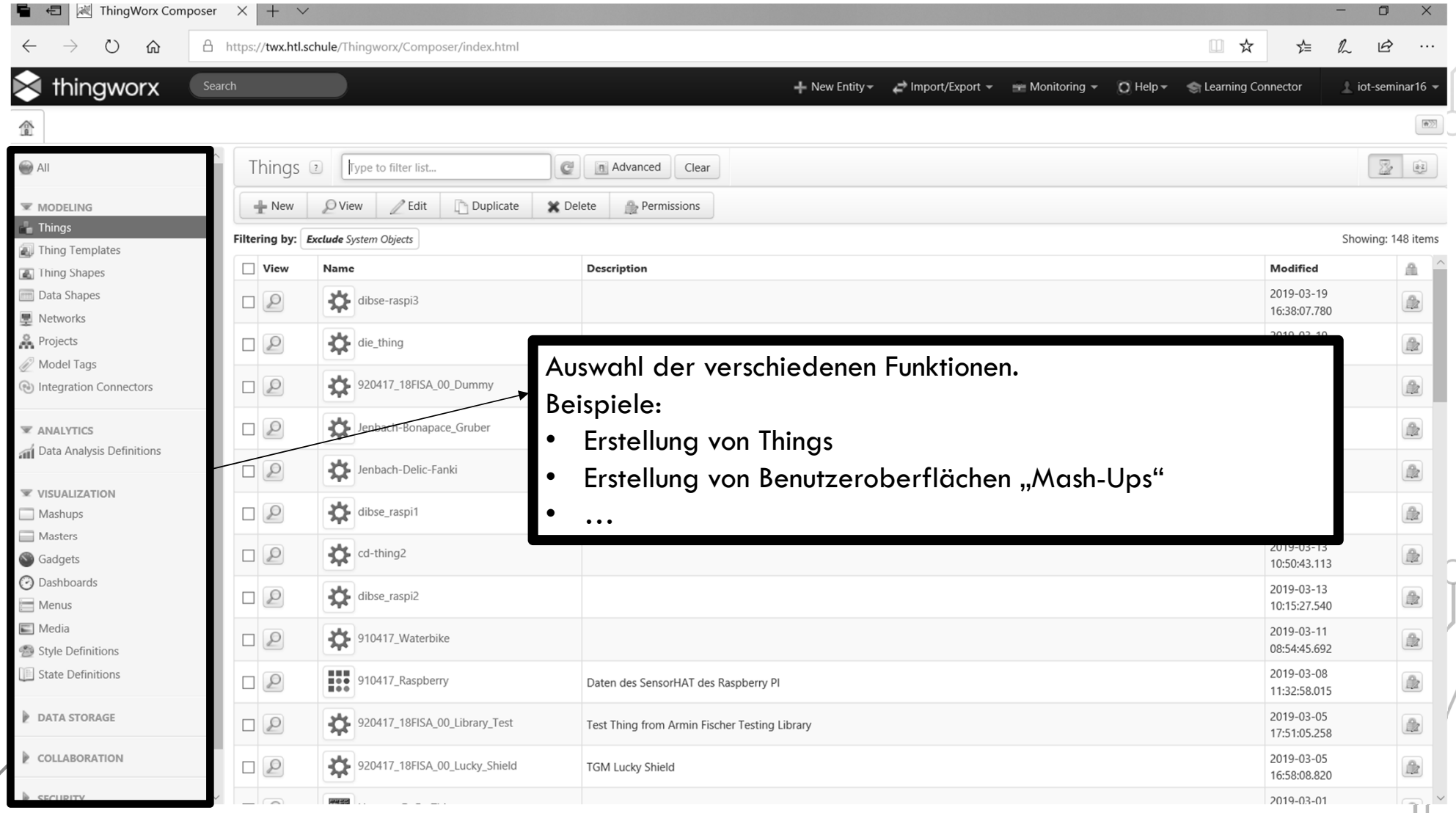
- Bis jetzt → Zugriff auf ein vorgegebenes Thing.
- Für weitere Projekte/Anwendungen muss man jedoch eigene Things und eigene Properties dazu erstellen.
- Dies ist mit der Software Thingworx Composer am Server möglich.



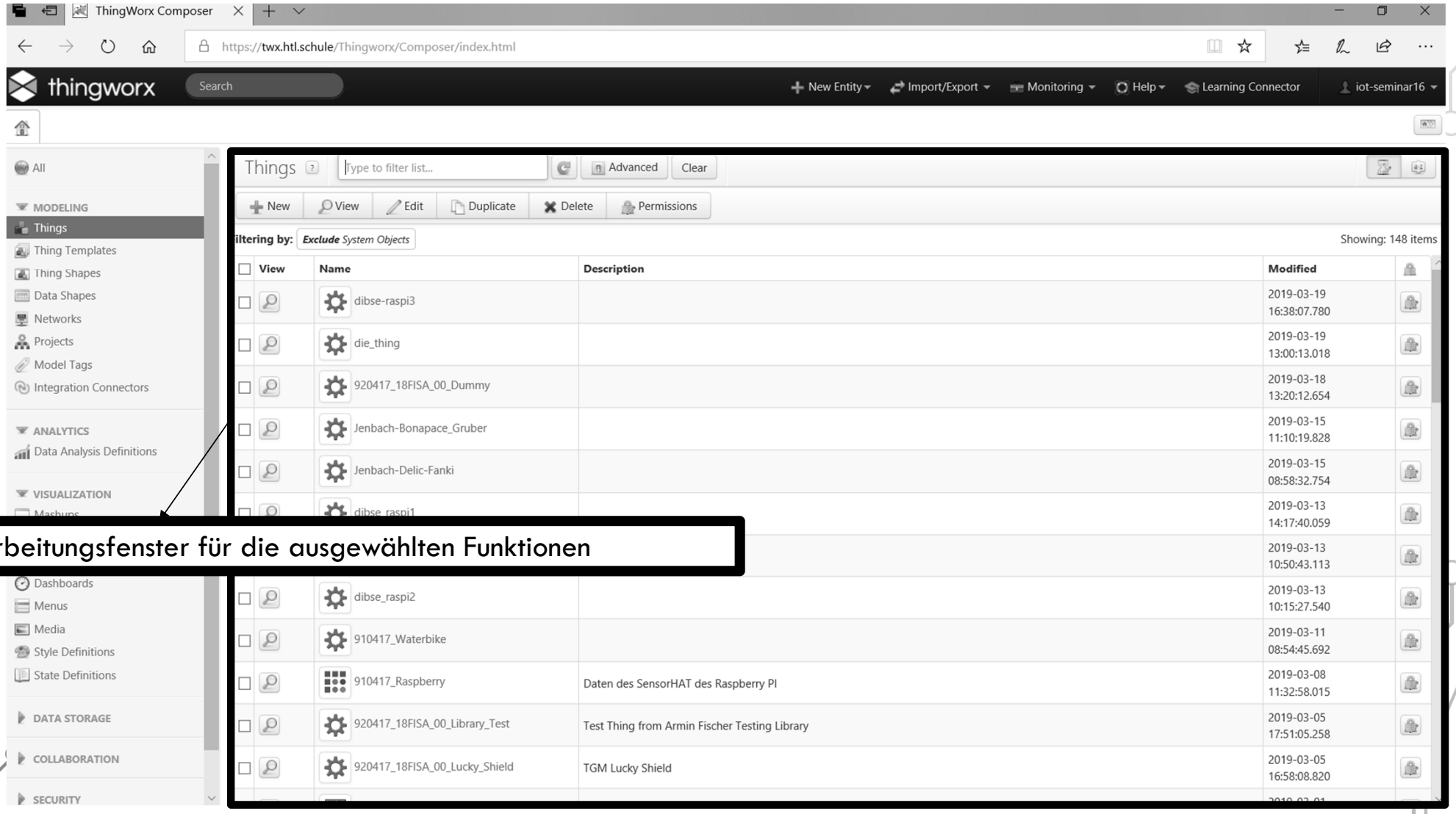
THINGWORX COMPOSER

- Software zur Verwaltung/Steuerung von „Things“
- Zugriff unter https://twx.htl.schule/
- Zugriffsdaten
  - Benutzer: iot-seminar16
  - Passwort: LiTec23-25

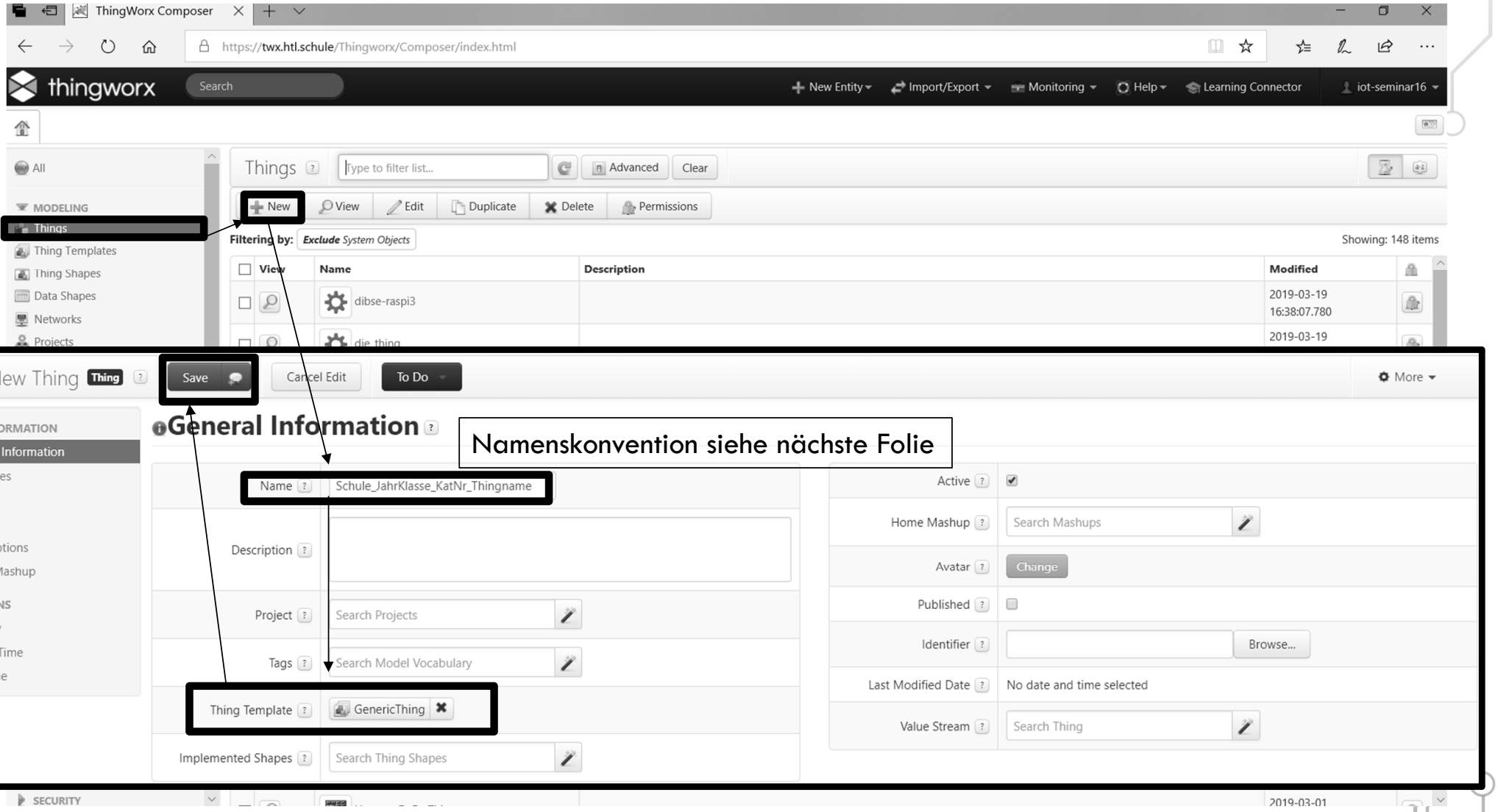
OBERFLÄCHE


ThingWorx Composer
https://twx.htl.schule/Thingworx/Composer/index.html
thingworx
Search
New Entity
Import/Export
Monitoring
Help
Learning Connector
iot-seminar16
All
MODELING
Things
Thing Templates
Thing Shapes
Data Shapes
Networks
Projects
Model Tags
Integration Connectors
ANALYTICS
Data Analysis Definitions
VISUALIZATION
Mashups
Masters
Gadgets
Dashboards
Menus
Media
Style Definitions
State Definitions
DATA STORAGE
COLLABORATION
SECURITY
Things
Type to filter list...
Advanced
Clear
New
View
Edit
Duplicate
Delete
Permissions
Filtering by: Exclude System Objects
Showing: 148 items
View
Name
Description
Modified
dibse-raspi3
2019-03-19 16:38:07.780
die_thing
920417_18FISA_00_Dummy
Jenbach-Bonapace_Gruber
Jenbach-Delic-Fanki
dibse_raspi1
cd-thing2
2019-03-13 10:50:43.113
dibse_raspi2
2019-03-13 10:15:27.540
910417_Waterbike
2019-03-11 08:54:45.692
910417_Raspberry
Daten des SensorHAT des Raspberry PI
2019-03-08 11:32:58.015
920417_18FISA_00_Library_Test
Test Thing from Armin Fischer Testing Library
2019-03-05 17:51:05.258
920417_18FISA_00_Lucky_Shield
TGM Lucky Shield
2019-03-05 16:58:08.820
Auswahl der verschiedenen Funktionen.
Beispiele:
• Erstellung von Things
• Erstellung von Benutzeroberflächen „Mash-Ups“
• …

OBERFLÄCHE


ThingWorx Composer
https://twx.htl.schule/Thingworx/Composer/index.html
thingworx
Search
New Entity
Import/Export
Monitoring
Help
Learning Connector
iot-seminar16
All
MODELING
Things
Thing Templates
Thing Shapes
Data Shapes
Networks
Projects
Model Tags
Integration Connectors
ANALYTICS
Data Analysis Definitions
VISUALIZATION
Dashboards
Menus
Media
Style Definitions
State Definitions
DATA STORAGE
COLLABORATION
SECURITY
Things
Type to filter list...
Advanced
Clear
New
View
Edit
Duplicate
Delete
Permissions
Showing: 148 items
View
Name
Description
Modified
dibse-raspi3
2019-03-19 16:38:07.780
die_thing
2019-03-19 13:00:13.018
920417_18FISA_00_Dummy
2019-03-18 13:20:12.654
Jenbach-Bonapace_Gruber
2019-03-15 11:10:19.828
Jenbach-Delic-Fanki
2019-03-15 08:58:32.754
dibse_raspi1
2019-03-13 14:17:40.059
2019-03-13 10:50:43.113
dibse_raspi2
2019-03-13 10:15:27.540
910417_Waterbike
2019-03-11 08:54:45.692
910417_Raspberry
Daten des SensorHAT des Raspberry PI
2019-03-08 11:32:58.015
920417_18FISA_00_Library_Test
Test Thing from Armin Fischer Testing Library
2019-03-05 17:51:05.258
920417_18FISA_00_Lucky_Shield
TGM Lucky Shield
2019-03-05 16:58:08.820
Bearbeitungsfenster für die ausgewählten Funktionen

ERSTELLUNG THING



NAMENSKONVENTION VON THINGS

- Warum? : Viele verschiedene Schulen erstellen verschiedene Things. Zur Zeit sehen alle Benutzer alle Things. Wenn keine Konvention vorhanden ist, dann wird eine Zuordnung der vielen Things nicht möglich sein.
- Namensaufbau:
  - Schulkennzahl_SchuljahrJahrgangsnummer_Katalognummer_Thingname
- Beispiel:

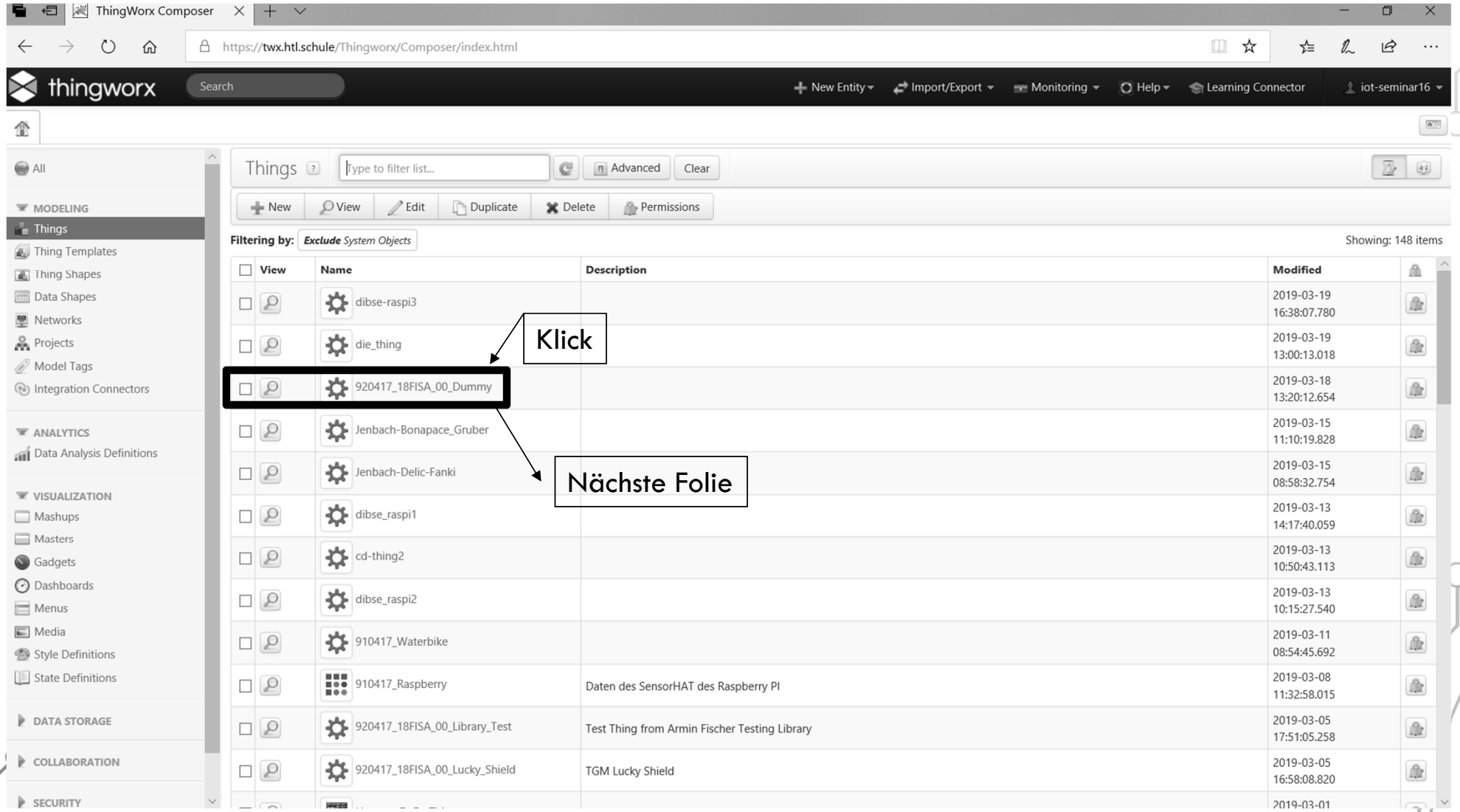
  Schüler des TGM im Jahr 2018/19 in der 5C mit der Katalognummer 10. Es wird eine Temperatursteuerung erstellt.

  920417_185C_10_Temperatursteuerung

  Lehrer des TGM im Jahr 2018/19. Das Kürzel in der Schule beträgt FISA. Es wird eine Temperatursteuerung erstellt.

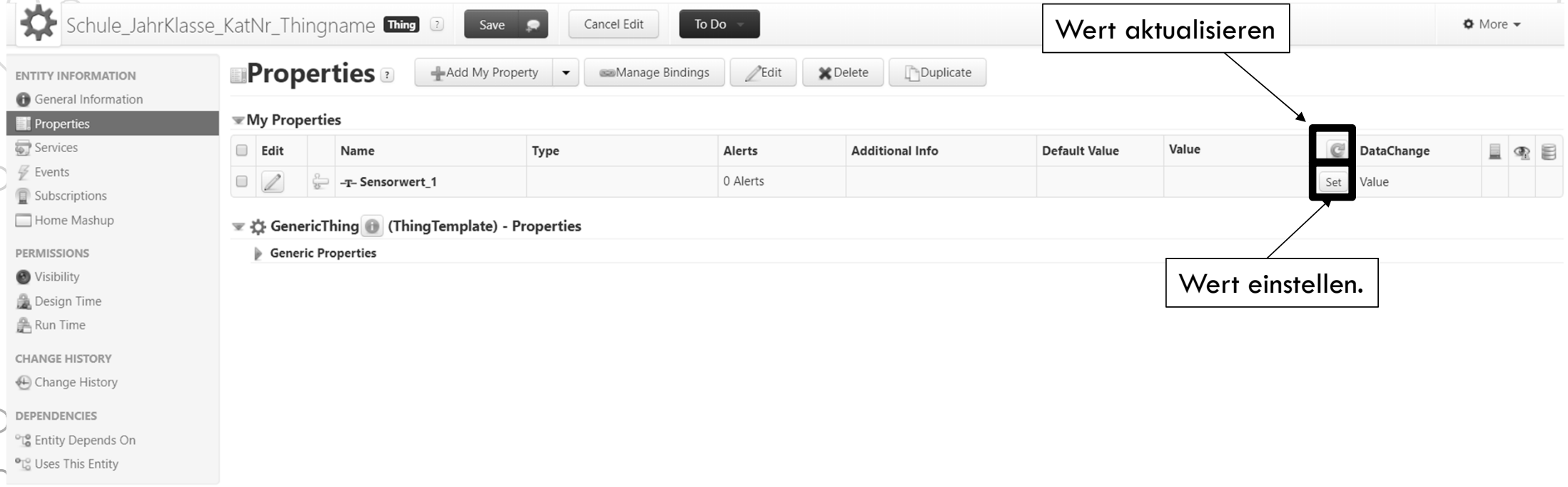
  920417_18FISA_00_Temperatursteuerung

ERSTELLUNG PROPERTY



ERSTELLUNG PROPERTY

PROPERTY – WERT EINSTELLEN ODER AKTUALISIEREN



- Wert einstellen: Wenn am Arduino ein Wert abgefragt wird, kann dieser hier eingestellt werden. Am Arduino wird sich im Serial Monitor die Zahl ändern.
- Wert aktualisieren: Wenn Sensorwerte von Arduino geschickt werden, dann muss bei einer Wertänderung hier händisch aktualisiert werden (sonst wird alter Wert angezeigt)

ÜBUNG

- Erstelle Dein eigenes Thing mit der Namenskonvention in Thingworx Composer
- Erstelle eine beliebige Property mit dem Datentyp Number
- Stelle bei der Property einen beliebigen Wert ein
- Konfiguriere die Datei Thingworx_MKR1000_Variable.h
- Frage den Wert mit dem Arduino ab und verändere diesen (Der Wert wird im Serial Monitor gesehen!)

STEUERUNG DES RELAIS MITTELS THINGWORX COMPOSER

ÜBUNG

- Mittels eines Things im Thingworx Composer soll das Relais am MKR1000 Proto Shield gesteuert werden.
- Die zugehörige Property ist vom Typ Integer. Wenn eine „1“ eingestellt ist, dann soll das Relais schalten. Bei einer „0“ wird das Relais geöffnet.
- Am Relais ist der Ventilator verbunden. Dieser wird ein- und ausgeschalten.
- Es wird dazu das Beispiel „004_GET_Relay“ verwendet
  - Bemerkung: Der Aufbau ist dem Programm „003_GET_Value“ ähnlich. Die gleichen Programmteile werden nicht nochmals beschrieben.

004_GET_RELAIS

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

//Relay 1 is on pin 1, relay 2 is on pin 2
#define RELAY_1 1
#define RELAY_2 2

void setup() {
  Serial.begin(9600);                    //Serial communications with computer at 9600 bauds for debug purposes
  myThing.Wifi(ssid, password);          //Start the Wifi Connection
  pinMode(RELAY_1,OUTPUT);               //Digital pin 1 is an output pin
  pinMode(RELAY_2,OUTPUT);               //Digital pin 2 is an output pin

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)       //
  {
    //Logic for the relay
    if( myThing.getjson("REL1") == 1.0)
    {
      digitalWrite(RELAY_1,HIGH);
    }
    else
    {
      digitalWrite(RELAY_1,LOW);
    }

    lastConnectionTime = millis();               //Refresh last connection time for if
  }
}
```

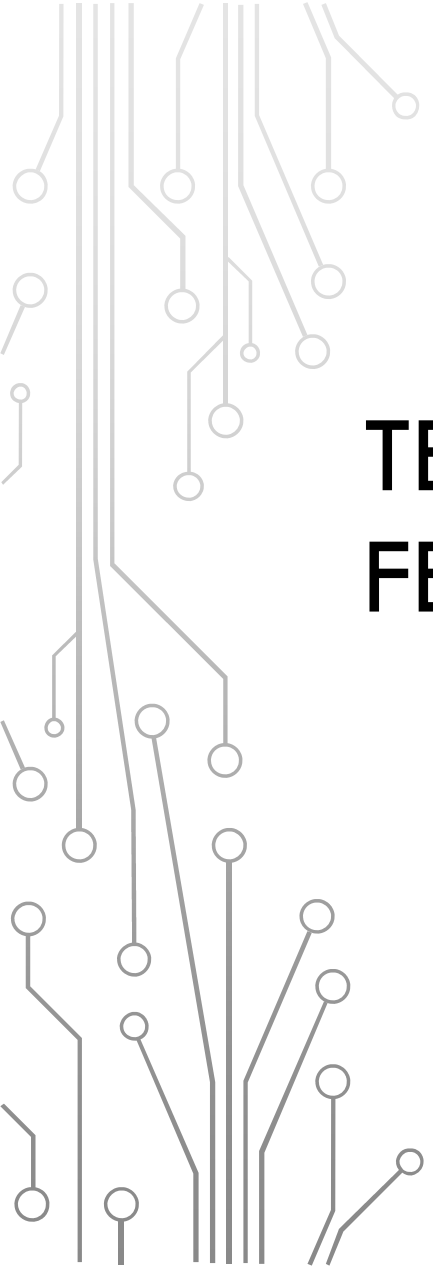
Das Relais 1/2 ist standardmäßig mit dem digitalen Pin 1/2 verbunden. Hier wird eine Konstante für die Pinzuweisung erstellt.

Der Digitalpin 1 und 2 wird als ein Ausgang gesetzt.

004_GET_RELAIS

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

//Relay 1 is on pin 1, relay 2 is on pin 2
#define RELAY_1 1
#define RELAY_2 2

void setup() {
  Serial.begin(9600);                          //Serial communications with computer at 9600 bauds for debug purposes
  myThing.Wifi(ssid, password);                //Start the Wifi Connection
  pinMode(RELAY_1,OUTPUT);                     //Digital pin 1 is an output pin
  pinMode(RELAY_2,OUTPUT);                     //Digital pin 2 is an output pin

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)   //Send request to server every TPOST seconds
  {
    //Logic for the relay
    if( myThing.getjson("REL1") == 1.0)
    {
      digitalWrite(RELAY_1,HIGH);
    }
    else
    {
      digitalWrite(RELAY_1,LOW);
    }
    lastConnectionTime = millis();             //Refresh last connection time for if
  }
}
```

If-Abfrage:
Die Property REL1 wird abgefragt. Ist der Wert 1 → der Pin wird auf HIGH gesetzt (das Relais wird geschalten). Sonst (else) wird das Relais geöffnet.



TEMPERATUR- UND FEUCHTIGKEITSMESSUNG

ÜBUNG

- Es wird mittels dem angeschlossenen DHT11 Sensor Temperatur und Feuchtigkeit gemessen.
- Diese Werte werden an 2 Properties eines Things zum Thingworx Composer Server geschickt.
- Es wird dazu das Beispiel „002_PUT_DHT11_Value“ verwendet
  - Bemerkung: Die Thing- und Propertynamen müssen angepasst werden.

003_PUT_DHT11_VALUE

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"
#include <dht.h>

//Definition for DHT11 sensor
dht DHT;
#define DHT11_PIN 3

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

void setup() {
  pinMode(DHT11_PIN, INPUT);
  Serial.begin(9600);
  myThing.Wifi(ssid, password);

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)
  {
    //Collect DHT11 Data
    int chk = DHT.read11(DHT11_PIN);
    //Send data with PUT Request to Thingworx
    myThing.put("BME280_TEMP2",DHT.temperature);  //Send temperature to server platform
    myThing.put("BME280_HUM",DHT.humidity);      //Send humidity to server platform

    lastConnectionTime = millis();               //Refresh last connection time for if
  }
}
```

Einfügen der Library dht.h
Dadurch können vorgefertigte Befehle für das Messen von Temperatur und Feuchtigkeit verwendet werden.

Es wird wieder (wie bei ThingWorx) eine Klasse für den Sensor erstellt. Die Klasse heißt DHT.
Die Signalleitung des DHT11 ist am Pin 3 angeschlossen

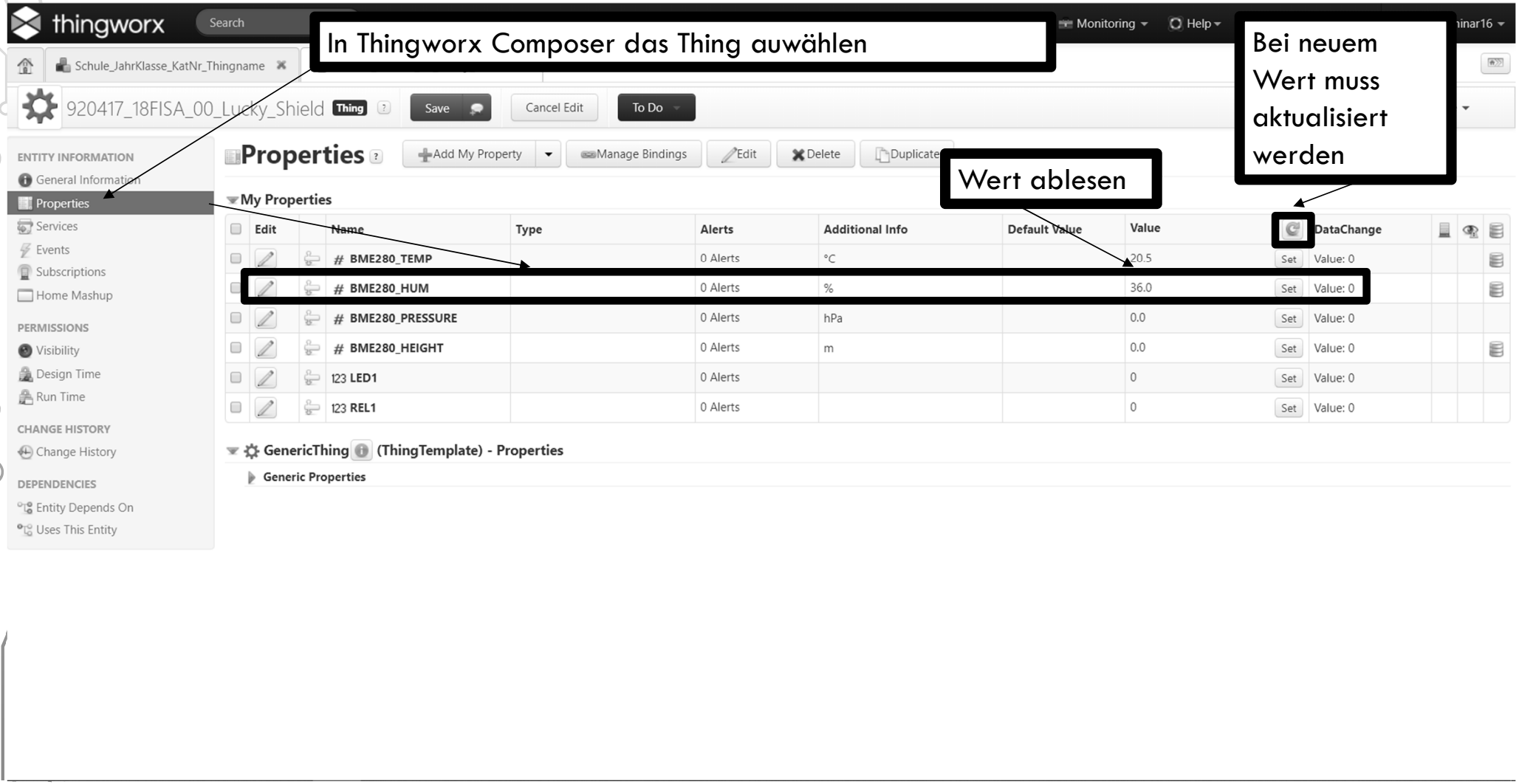
Der Pin 3 ist ein Input Pin.

003_PUT_DHT11_VALUE

```
//Definition of used Libraries
#include "Thingworx_MKR1000.h"
#include "Thingworx_MKR1000_Variable.h"
#include <dht.h>

//Definition for DHT11 sensor
dht DHT;
#define DHT11_PIN 3                                    //

// Define Thingworx Class (1 per Thing)
ThingWorx myThing(host, port, appKey, thingName);

void setup() {
  pinMode(DHT11_PIN, INPUT);                           // set DHT11_PIN to input
  Serial.begin(9600);                                  //Serial communications with computer at 9600 bauds for debug purposes
  myThing.Wifi(ssid, password);                        //Start the Wifi Connection

}

void loop() {
  if (millis() - lastConnectionTime > TPOST)      //Send request to server every TPOST seconds
  {
    //Collect DHT11 Data
    int chk = DHT.read11(DHT11_PIN);
    //Send data with PUT Request to Thingworx
    myThing.put("BME280_TEMP2",DHT.temperature);  //Send temperature to server platform
    myThing.put("BME280_HUM",DHT.humidity);      //Send humidity to server platform

    lastConnectionTime = millis();                //Refresh last connection time for if
  }
}
```

Es wird mit dem Befehl DHT.read11 Befehl der Bibliothek dht.h die Temperatur und Feuchtigkeitswerte abgefragt. Danach wird mit dem Befehl myThing.put(…,DHT.temperature) die Temperatur auf das Thing mit der Property „BME280_TEMP2“ geschickt. Danach wird mit dem Befehl myThing.put(…,DHT.humidity) die Feuchtigkeit auf das Thing mit der Property „BME280_HUM“ geschickt.

003_PUT_DHT11_VALUE – THINGWORX COMPOSER

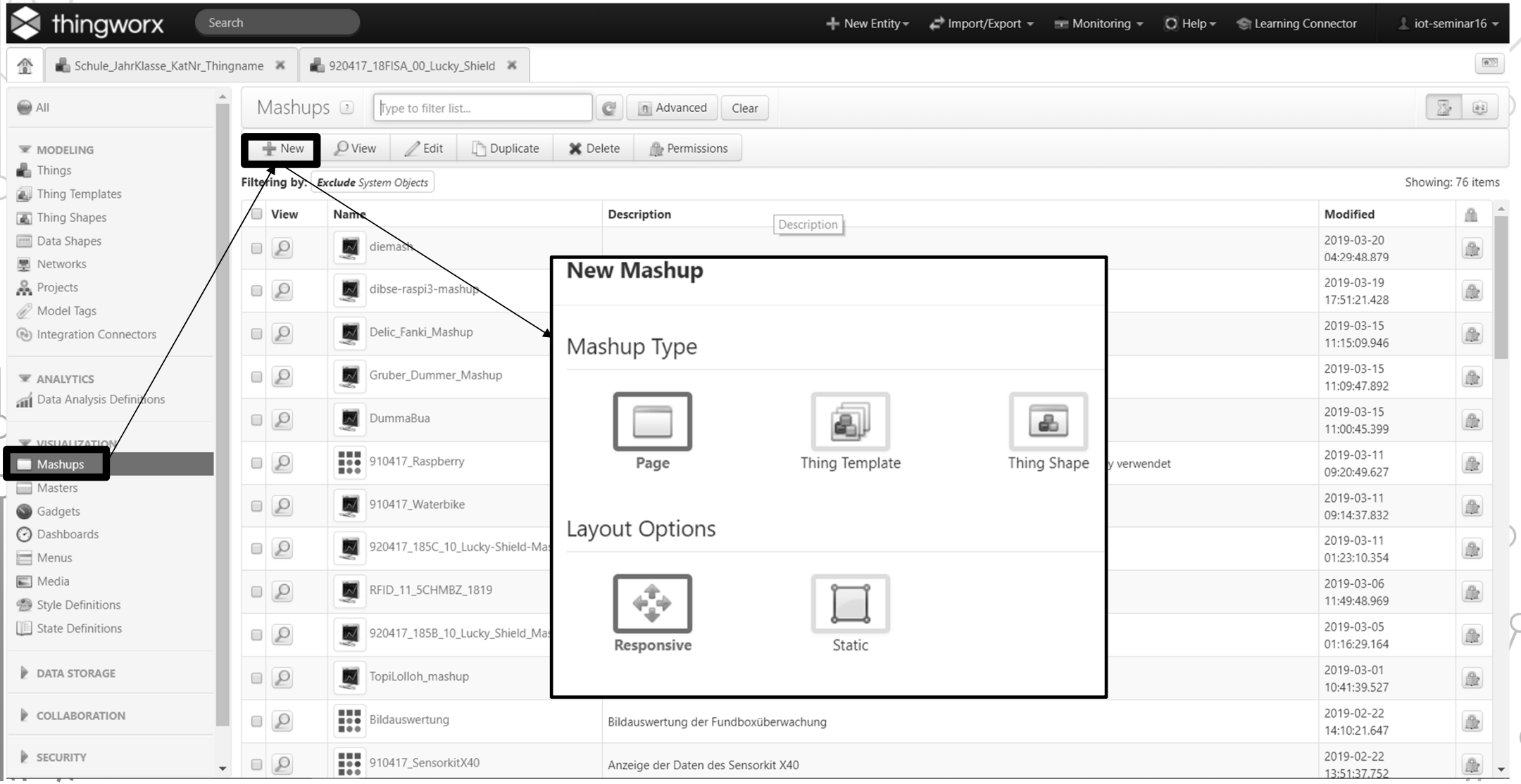


ERSTELLUNG BEDIENPANEL („MASH UP“) IN THINGWORX COMPOSER

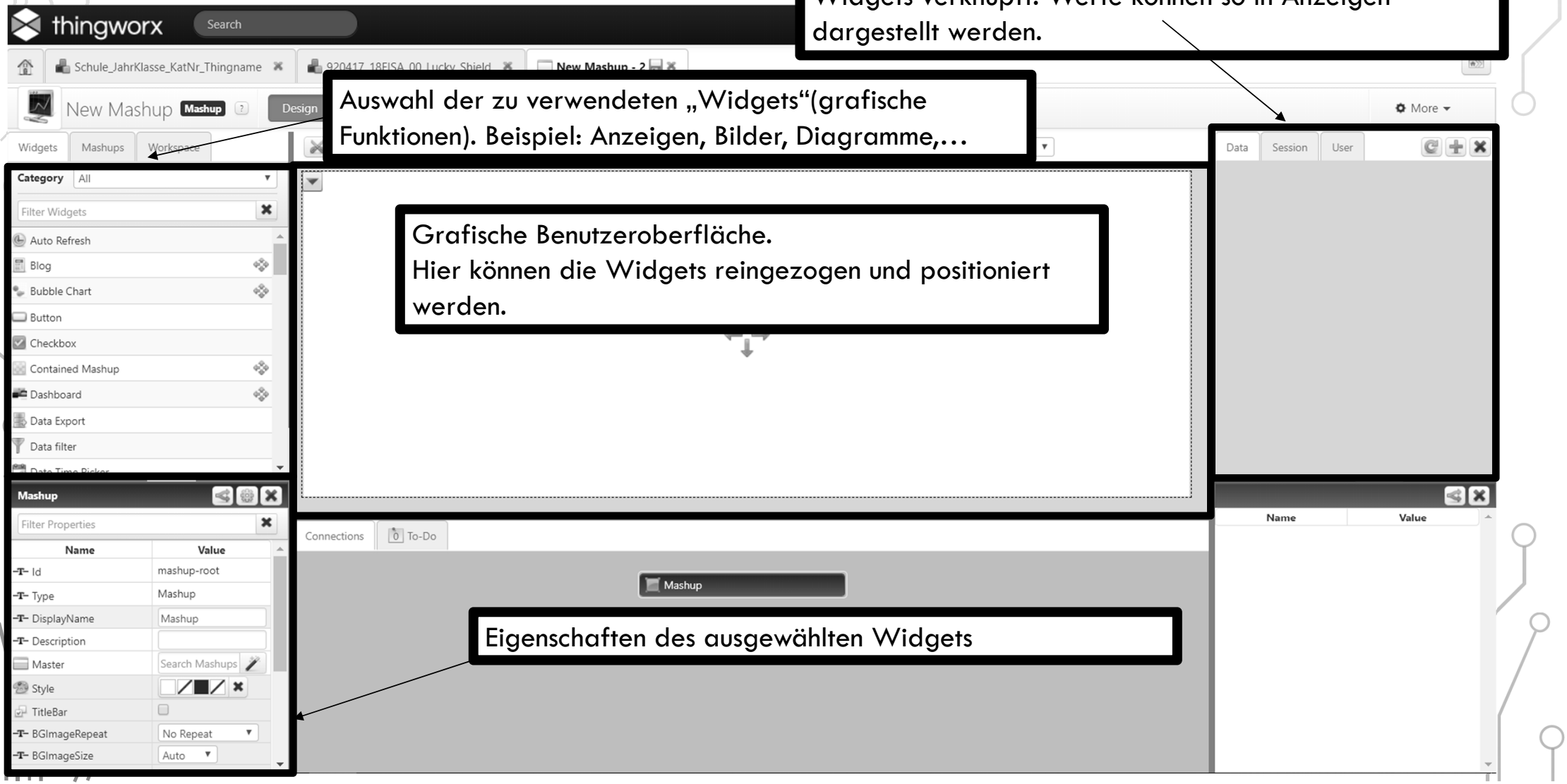
MASH UP

- In Thingworx Composer können Bedienpanele oder sogenannte Mash Ups erstellt werden.
- Diese erlauben das Erstellen einer GUI (Graphical User Interface)
- Es können somit Sensorwerte visualisiert und getrackt werden.
- Das Mash Up ist per digitalem Endgerät (Tablet, Smart Phone, PC,…) abrufbar.

ERSTELLUNG MASH UP



MASH UP OBERFLÄCHE



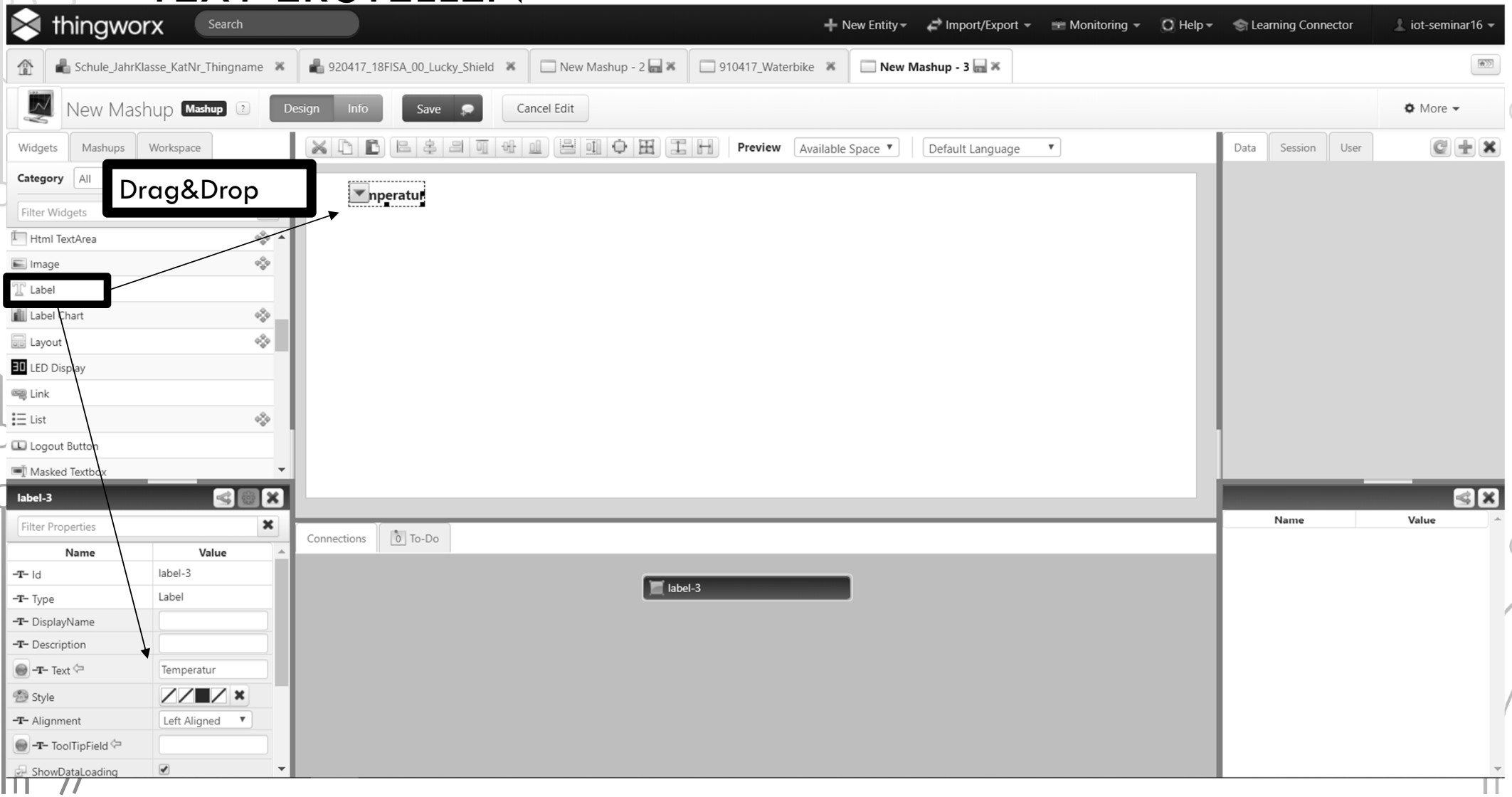
Hier können Things und Properties ausgewählt werden. Im späteren Verlauf werden die Properties dann mit den Widgets verknüpft. Werte können so in Anzeigen dargestellt werden.

Auswahl der zu verwendeten „Widgets“(grafische Funktionen). Beispiel: Anzeigen, Bilder, Diagramme,…

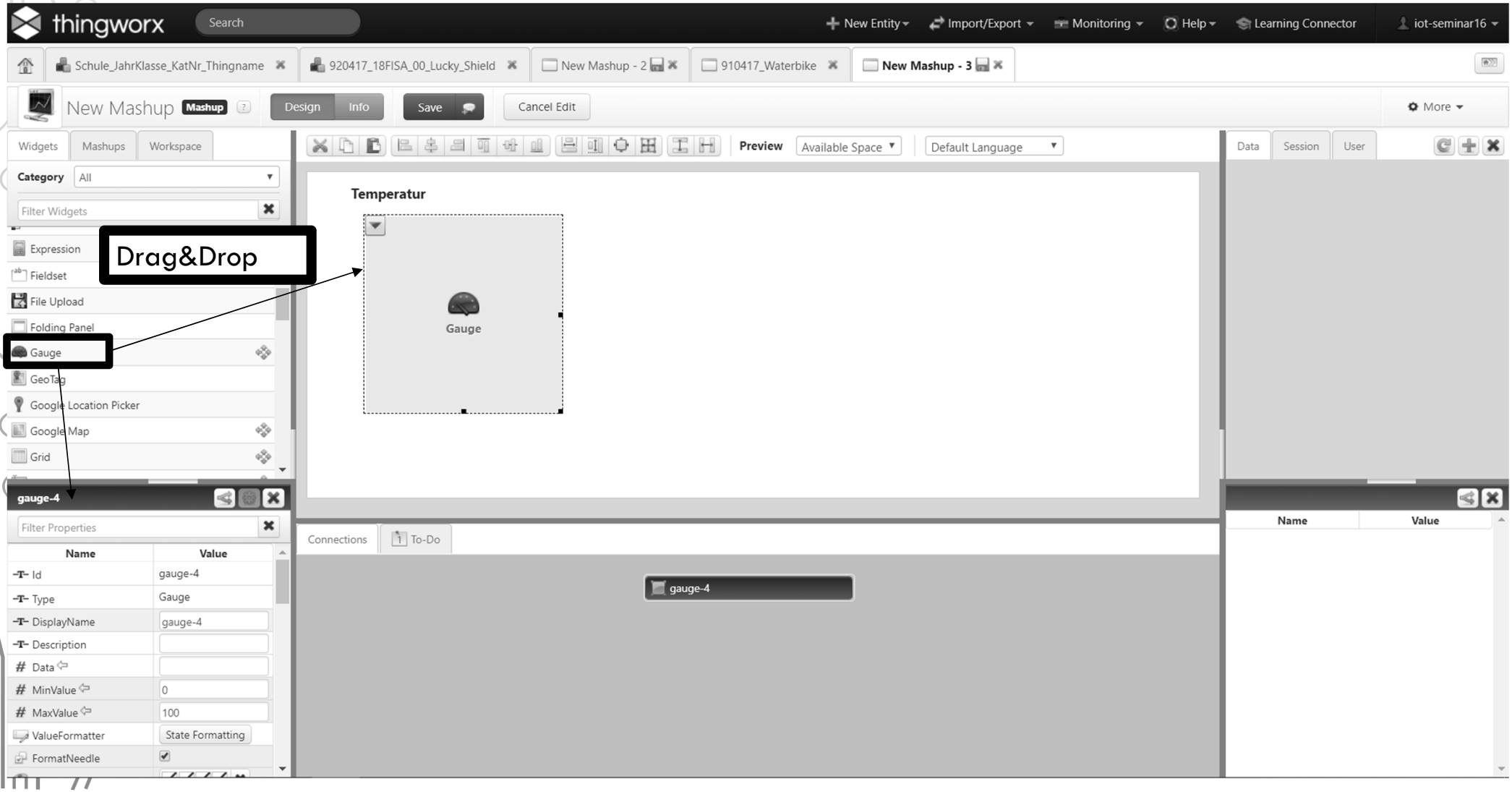
Grafische Benutzeroberfläche.
Hier können die Widgets reingezogen und positioniert werden.

Eigenschaften des ausgewählten Widgets

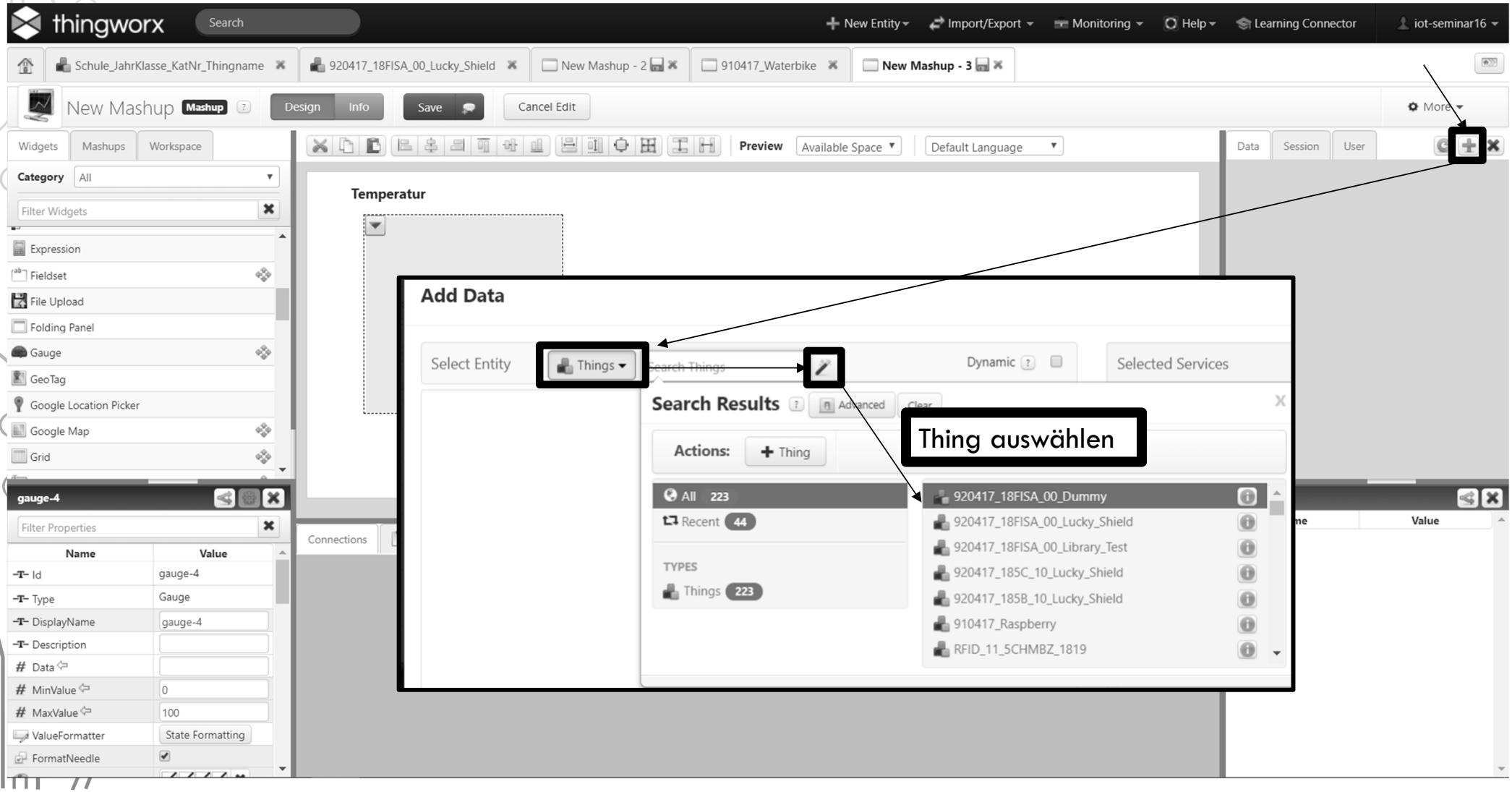
TEXT ERSTELLEN



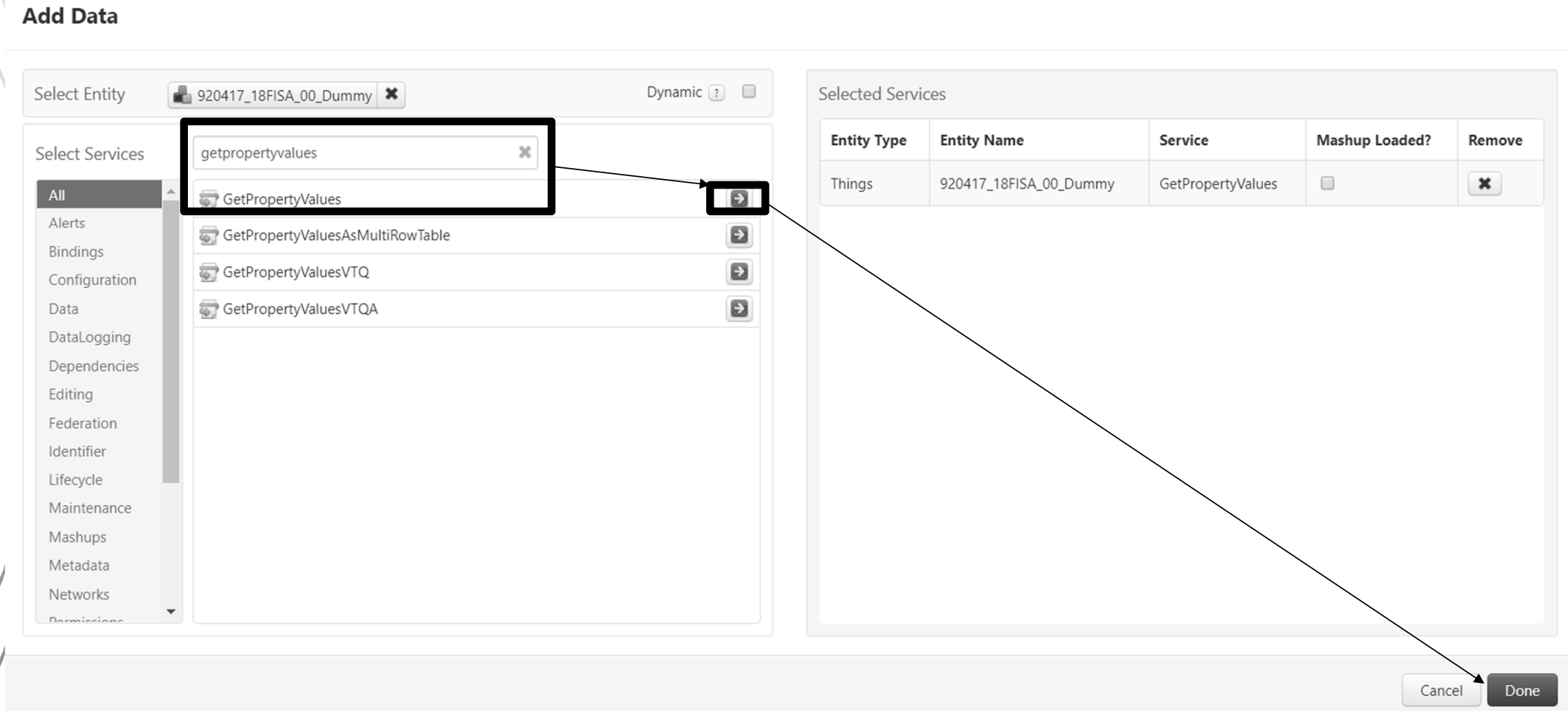
ANZEIGE ERSTELLEN



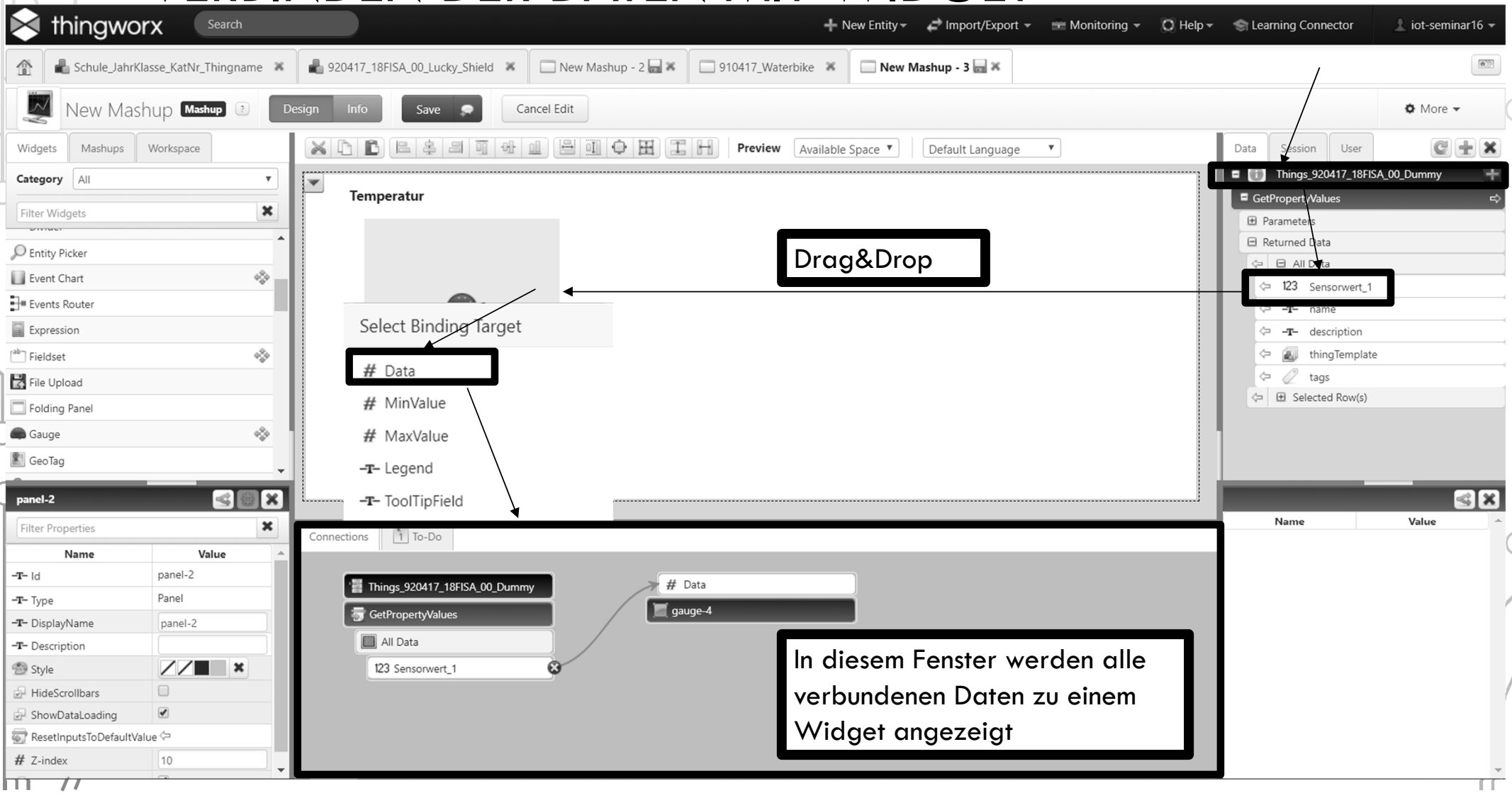
HINZUFÜGEN VON DATEN



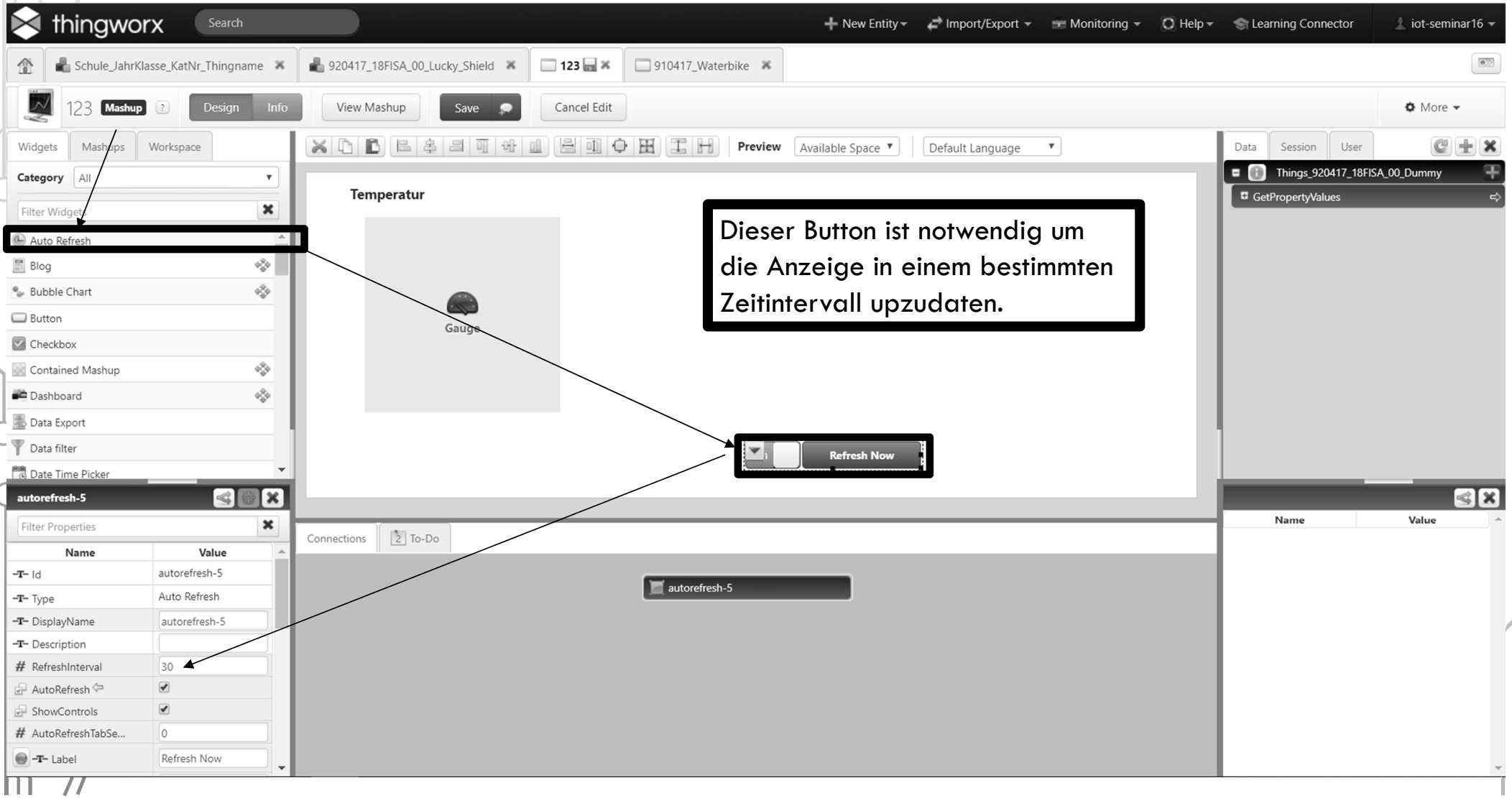
HINZUFÜGEN VON DATEN



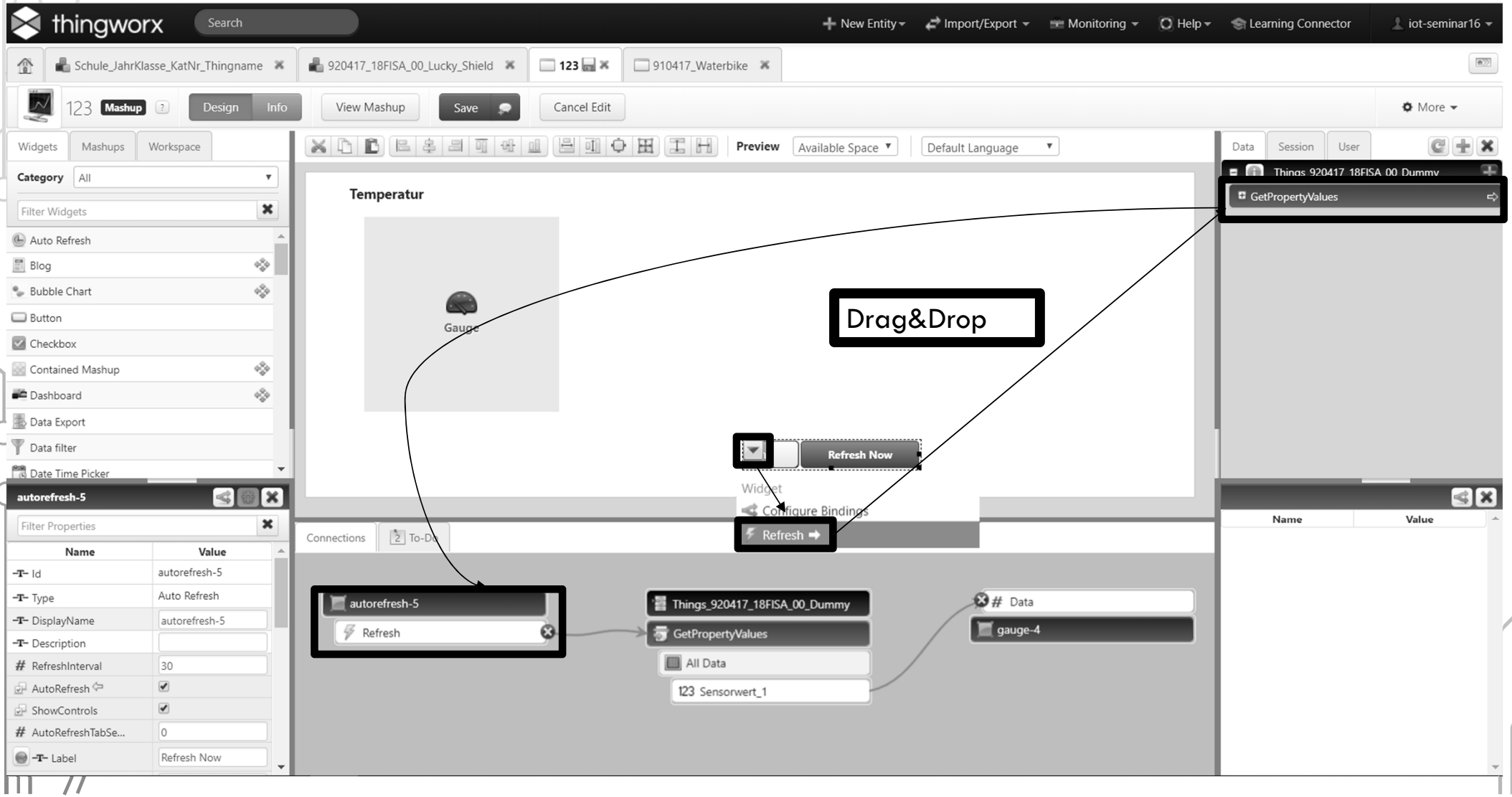
VERBINDEN DER DATEN MIT WIDGET



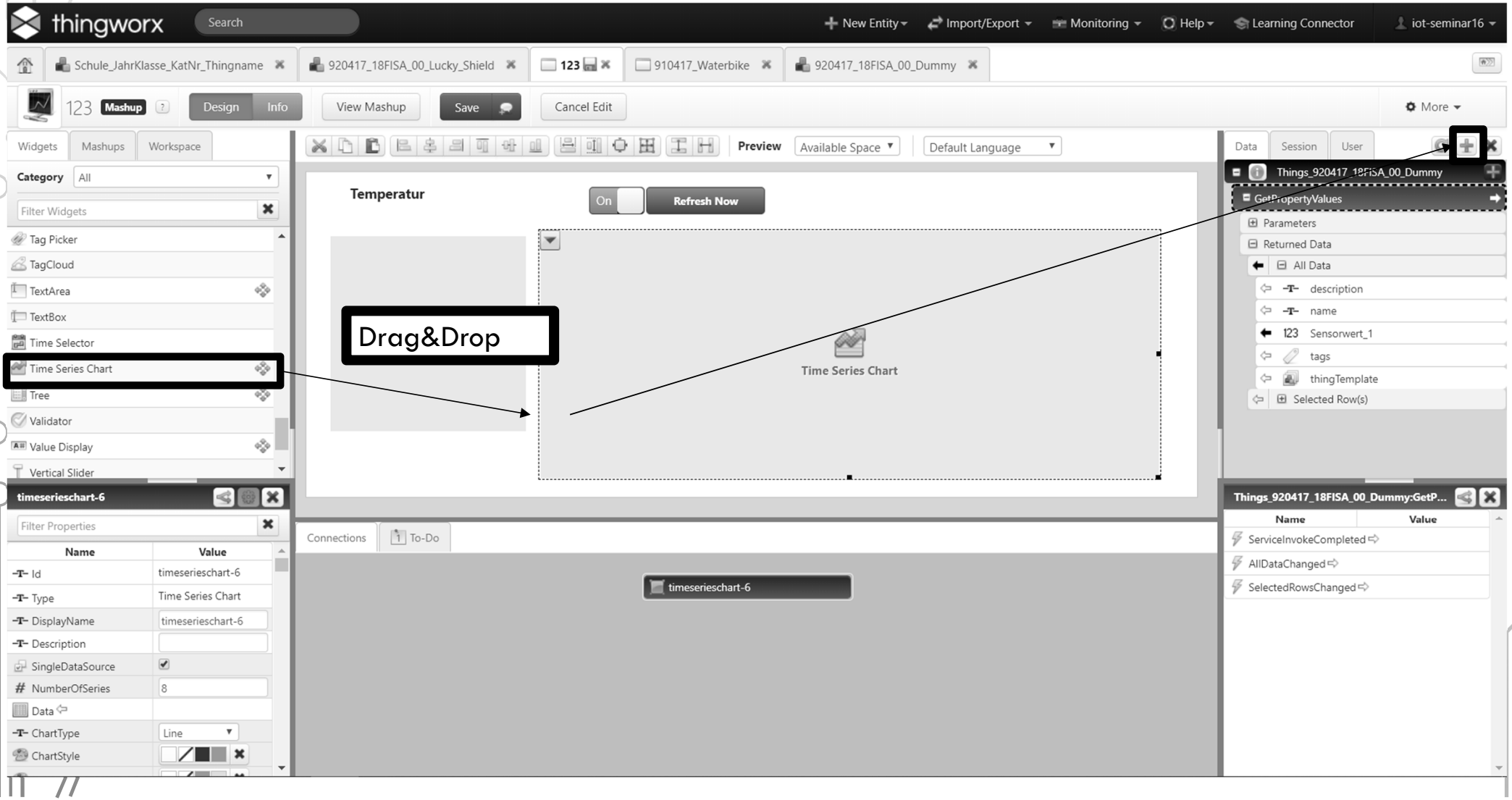
HINZUFÜGEN EINES REFRESH BUTTONS



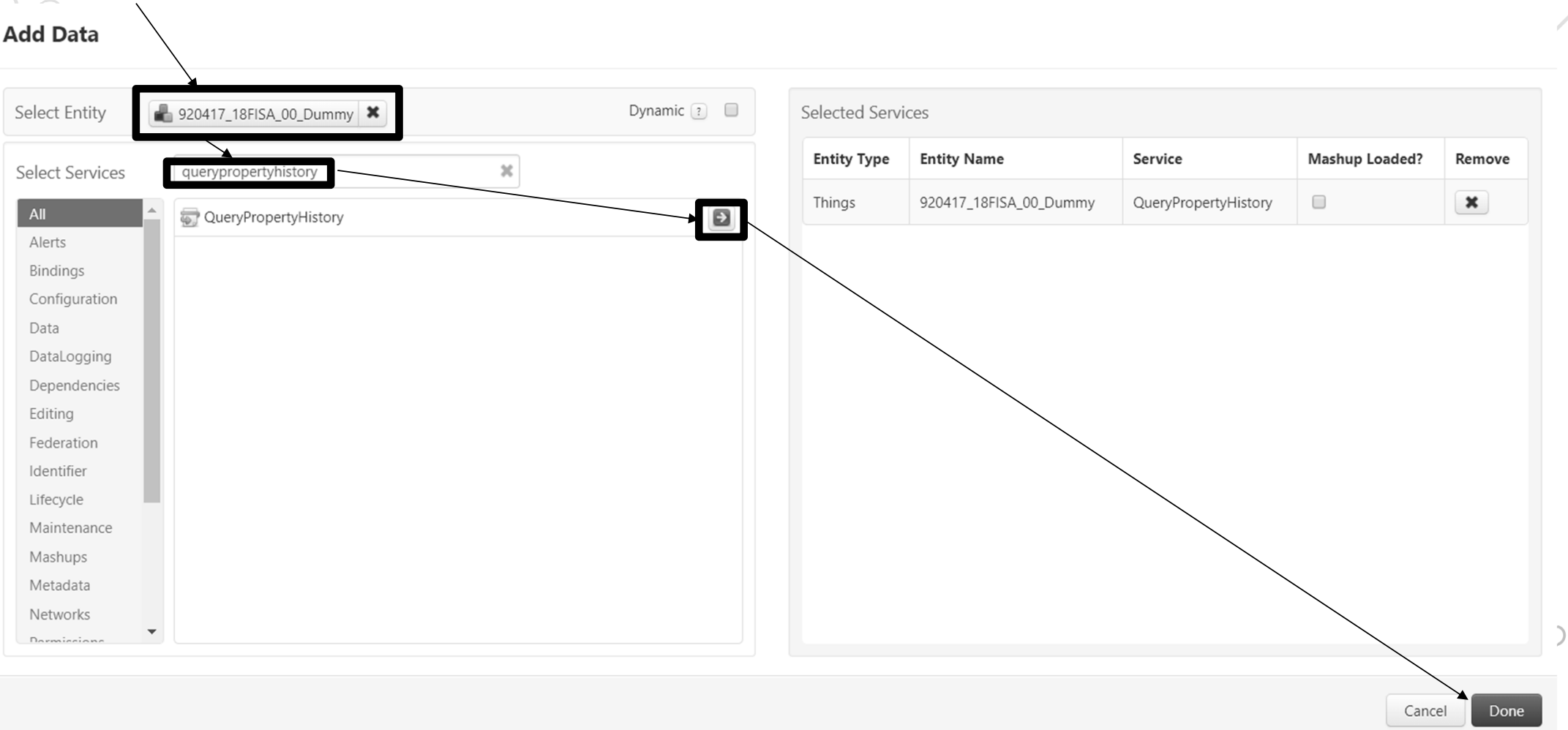
HINZUFÜGEN EINES REFRESH BUTTONS



HINZUFÜGEN EINES DIAGRAMMES



HINZUFÜGEN EINES DIAGRAMMES



HINZUFÜGEN EINES DIAGRAMMES

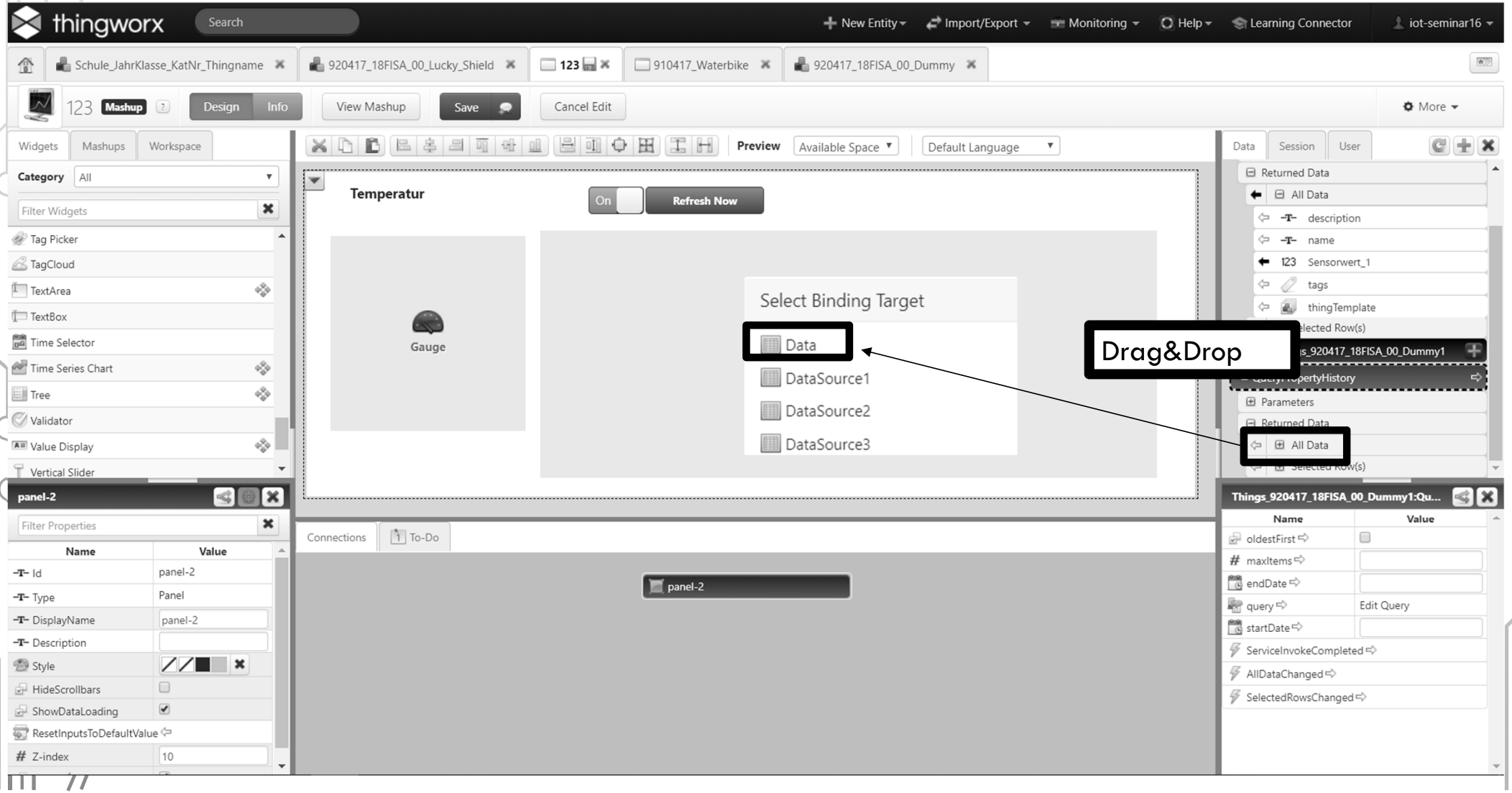
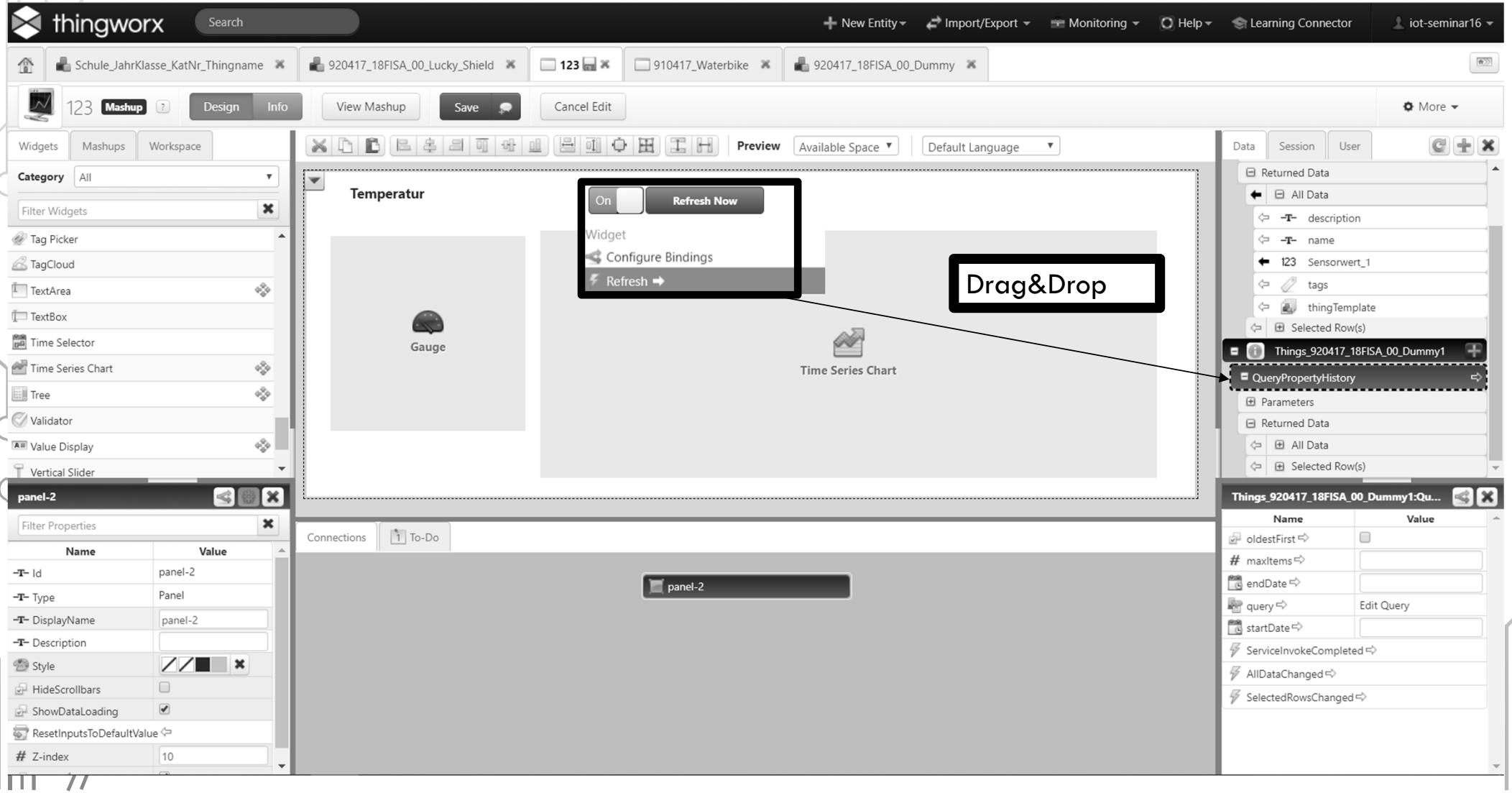
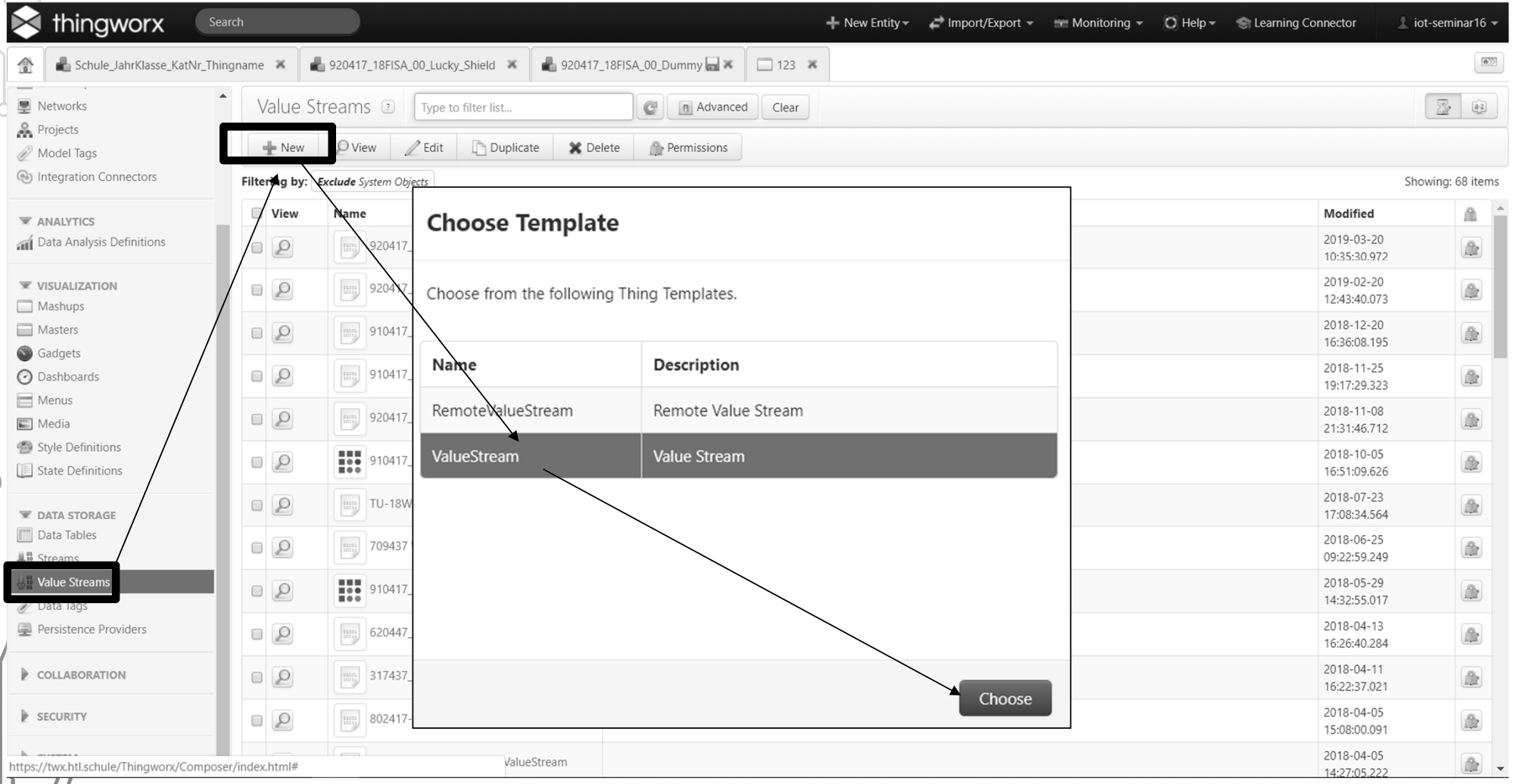


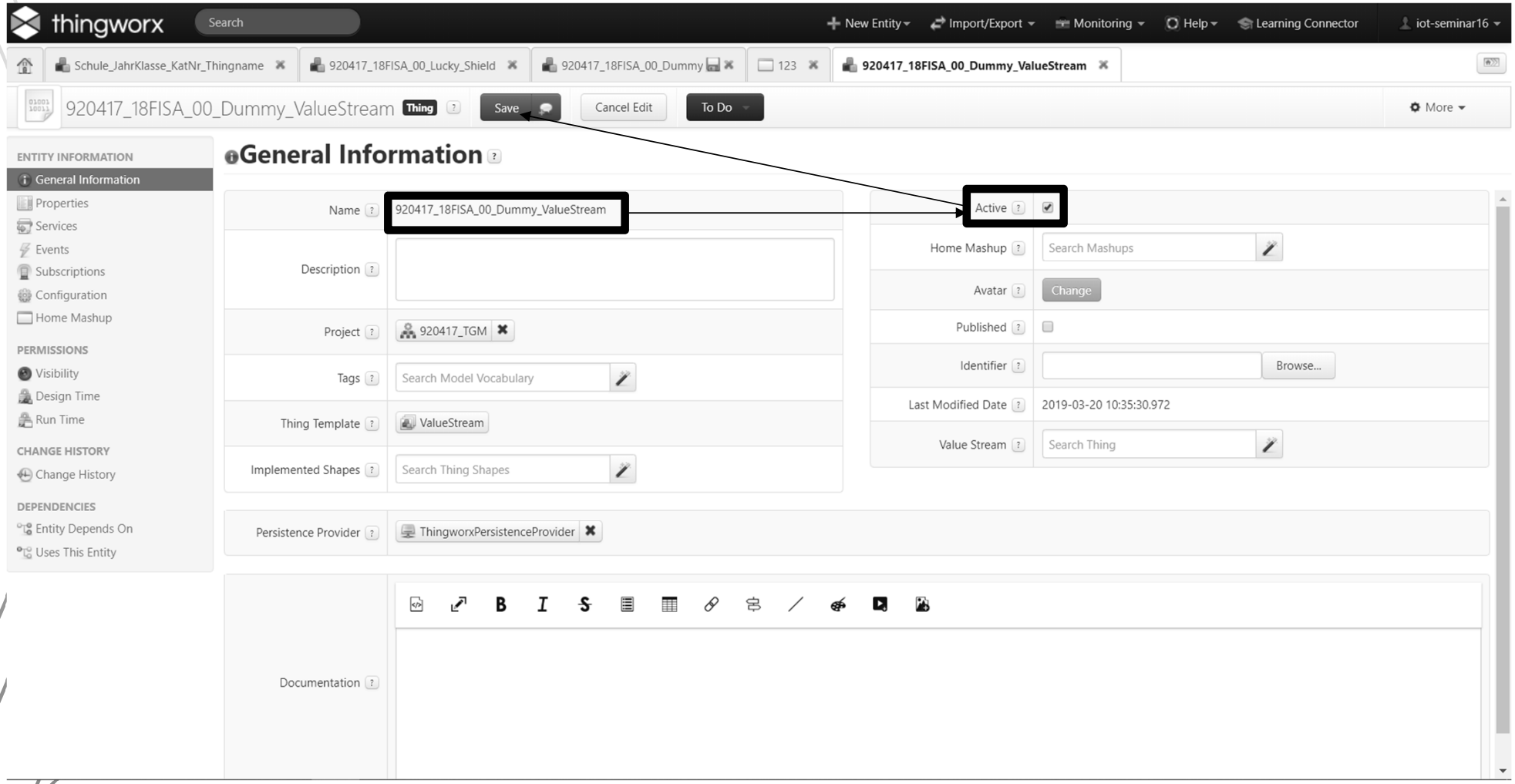
DIAGRAMM AKTUALISIEREN



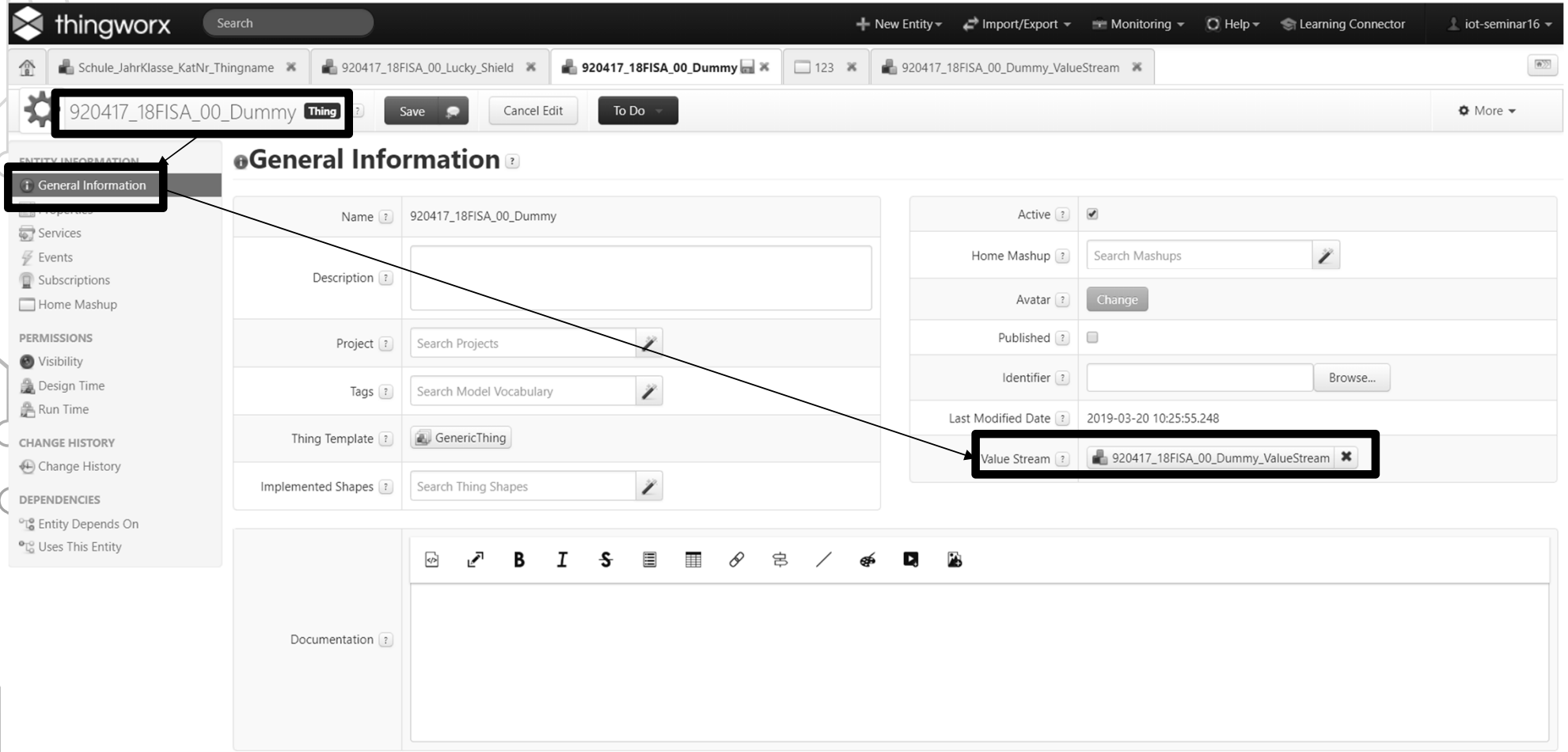
EINSTELLUNGEN AM THING FÜR DAS SPEICHERN VON DATEN – ERSTELLUNG VALUE STREAM



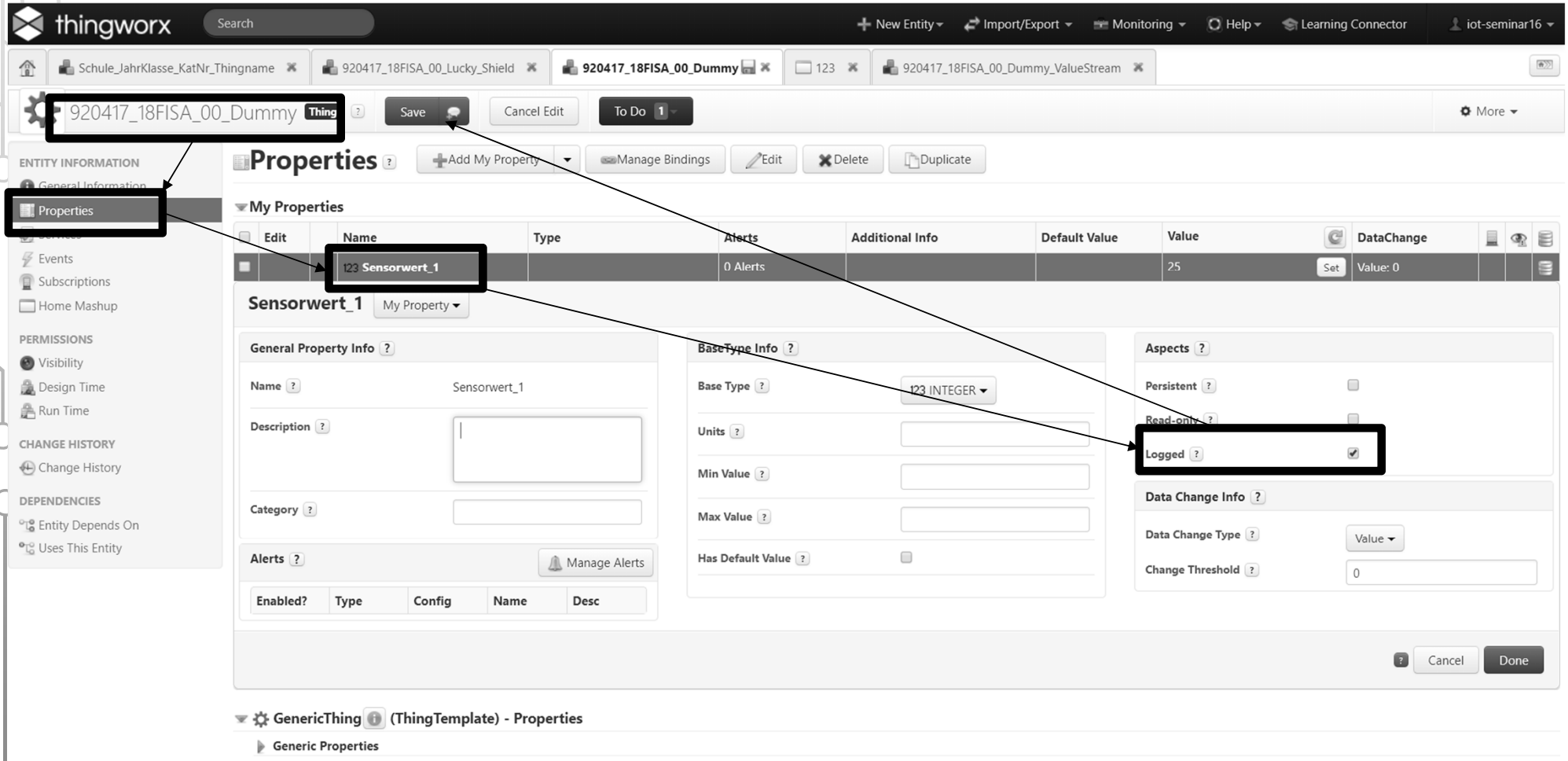
EINSTELLUNGEN AM THING FÜR DAS SPEICHERN VON DATEN – ERSTELLUNG VALUE STREAM



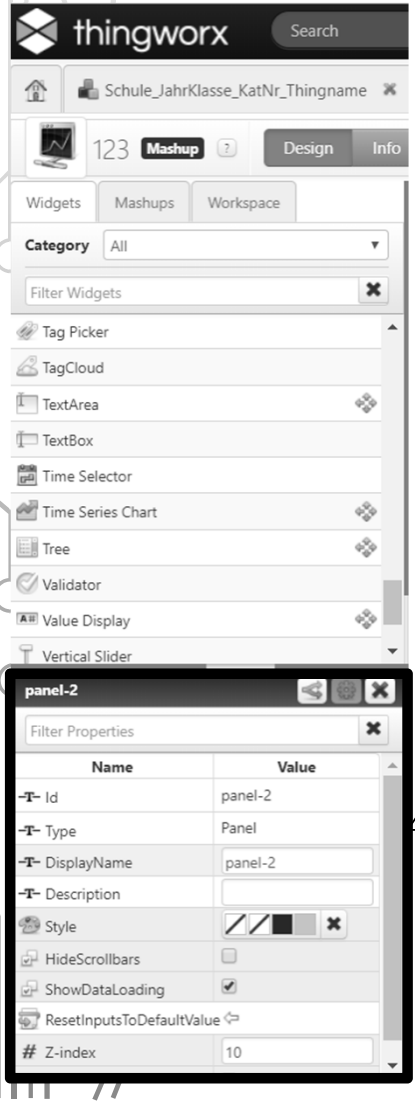
EINSTELLUNGEN AM THING FÜR DAS SPEICHERN VON DATEN

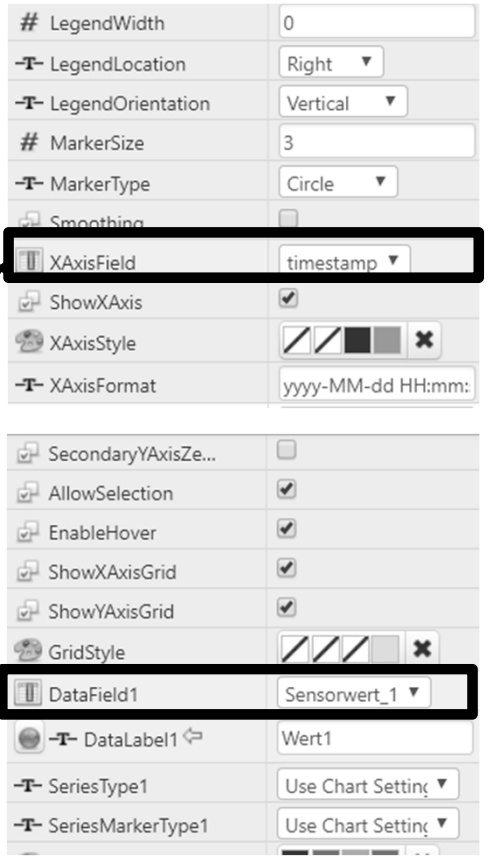


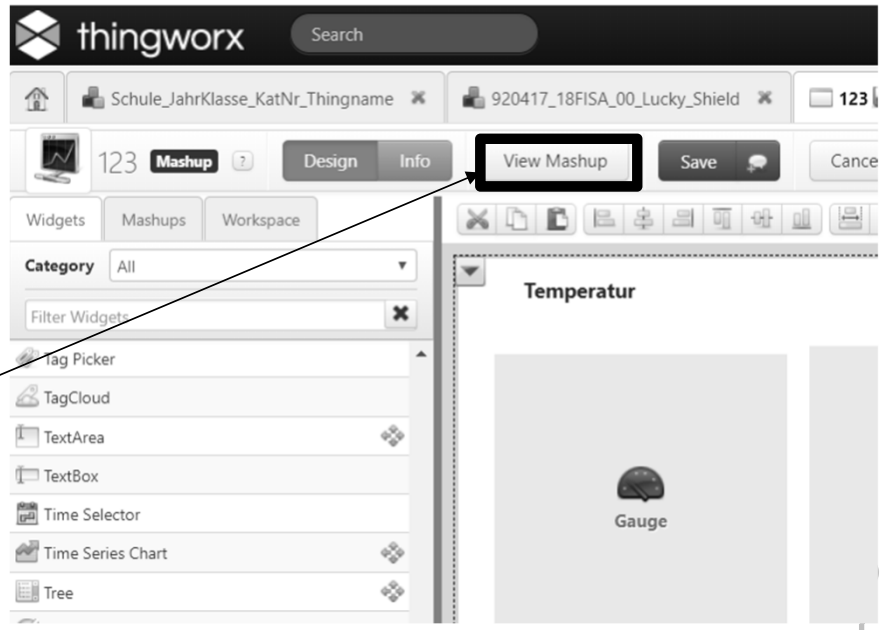
EINSTELLUNGEN AM THING FÜR DAS SPEICHERN VON DATEN



LETZTE EINSTELLUNGEN AM DIAGRAMM







FERTIGES MASHUP IM BROWSER

Diese Adresse kann in einem Browser eines digitalen Endgerätes eingegeben werden. Nach der Eingabe der Logindaten ist diese Seite sichtbar.