



# ARDUINO – THINGWORX COMPOSER SCHULUNG

MÄRZ 2019

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

E-MAIL: [AFISCHER2@TGM.AC.AT](mailto: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



```
sketch_sep14a | Arduino 1.6.11 (Windows Store 1.6.11.0)
File Edit Sketch Tools Help
sketch_sep14a
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```





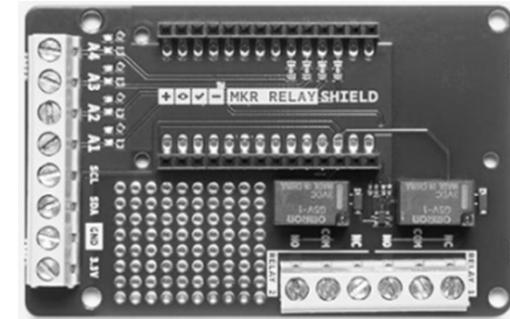
# SCHULUNGSMATERIAL

# 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>

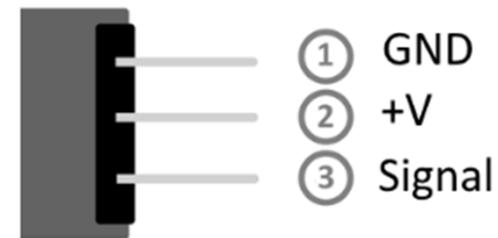
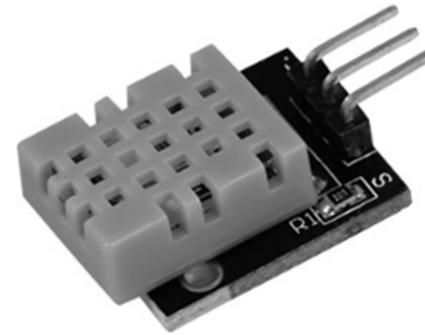
# ARDUINO MKR RELAY PROTO SHIELD



- 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](http://sensorkit.en.joy-it.net/index.php?title=KY-015_Combi-Sensor_Temperature%2BHumidity)



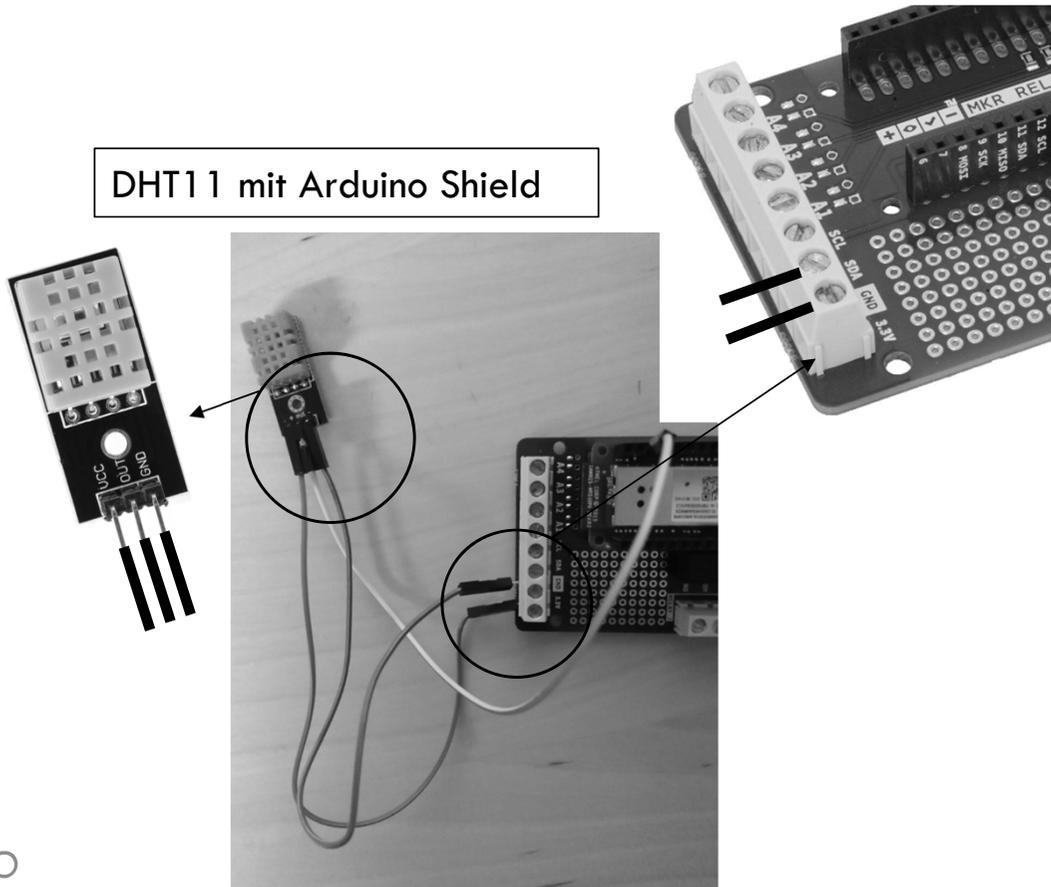
# VENTILATOR



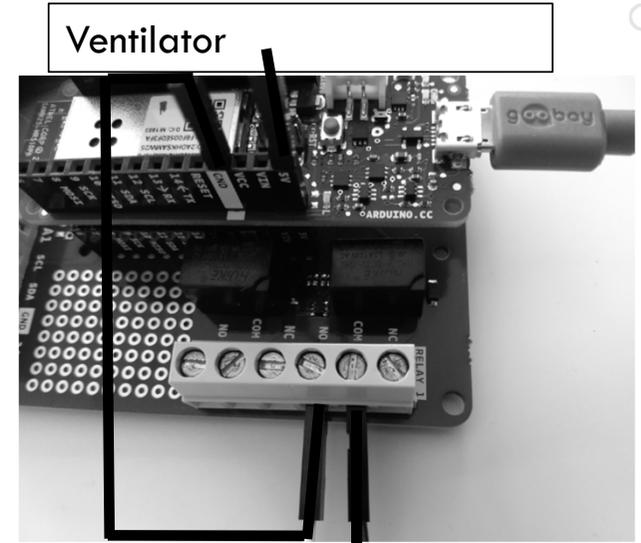
- Betriebsspannung 5VDC
- Weitere Informationen siehe [www.reichelt.at/](http://www.reichelt.at/) Produktnummer: RPI FAN 30X30

# VERBINDEN DER EINZELTEILE

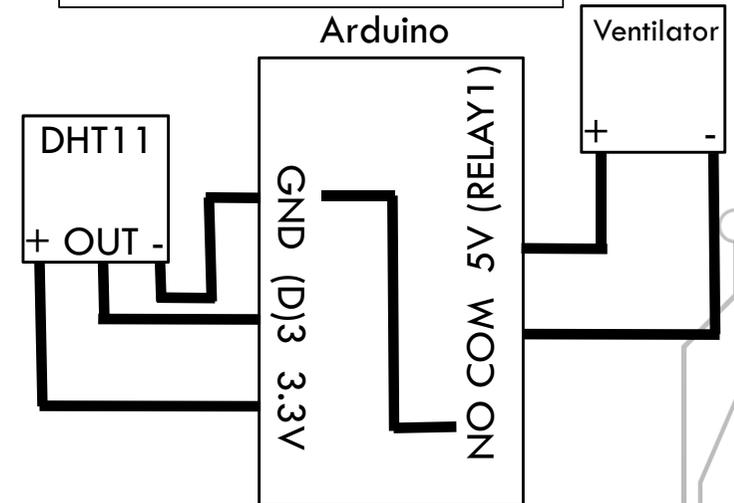
DHT11 mit Arduino Shield



Ventilator



Verbindungsplan





# AUFGABENSTELLUNG



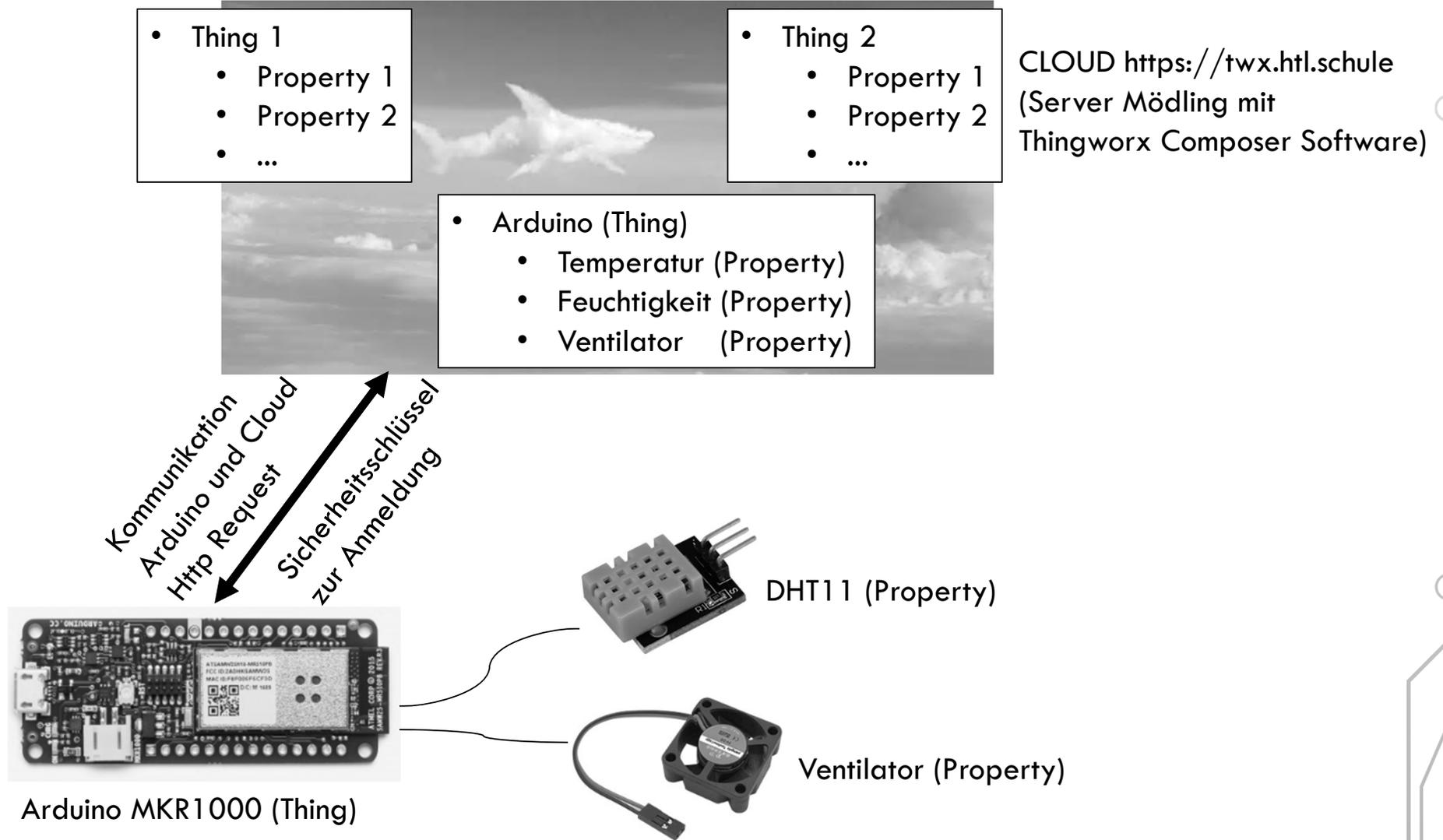
# AUFGABENSTELLUNG

- Messen der Raumtemperatur 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?



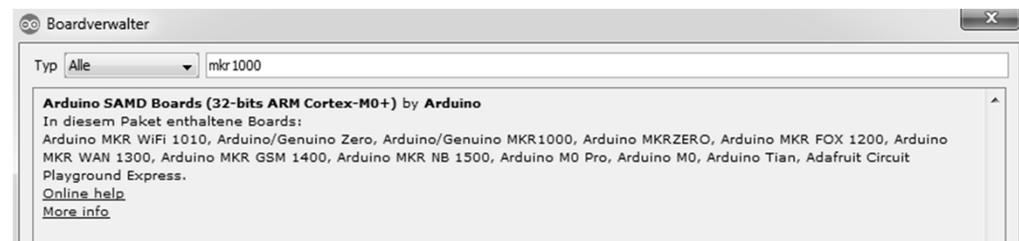
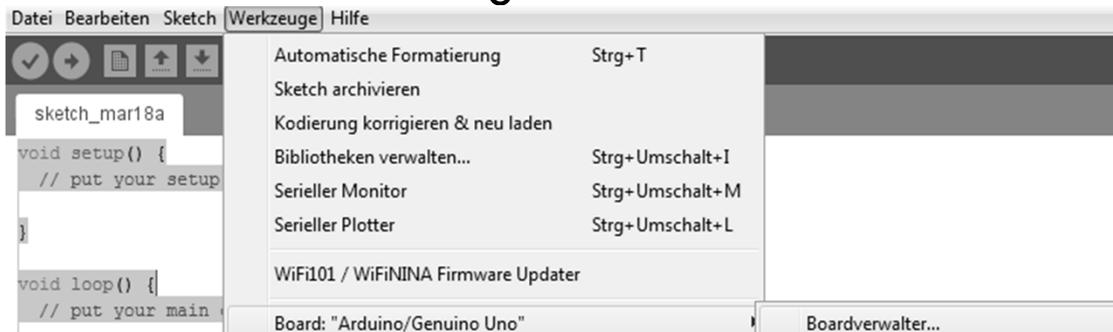




# REALISIERUNG DER DATENÜBERTRAGUNG MIT ARDUINO

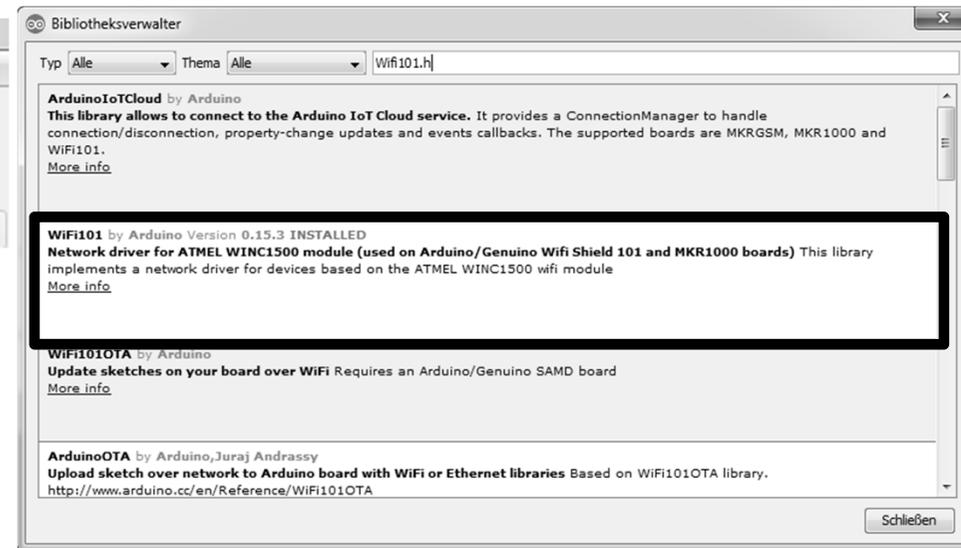
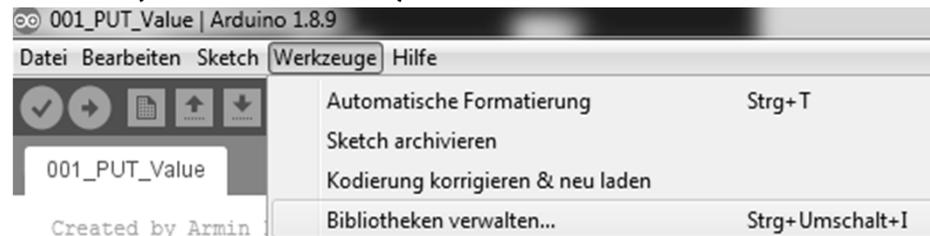
# ARDUINO IDE

- 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



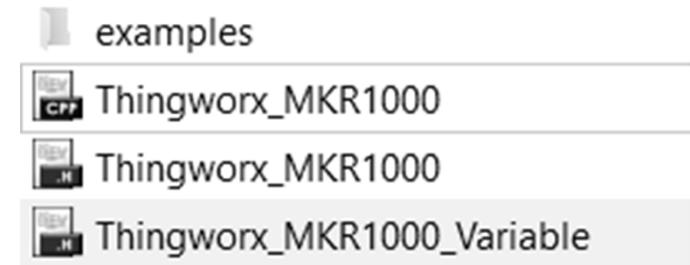
# INSTALLATION NÖTIGER LIBRARIES

- 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
```

Benutzername und Passwort von WLAN

```
//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
```

Sicherheitsschlüssel und Thingname

```
//->Timing Vars
```

```
unsigned long lastConnectionTime = 0; //Last connection ms time between server requests
```

# 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);
  myThing.Wifi(ssid, password);
```

```
//Serial communications with computer at 9600 bauds for debug purposes
//Start the Wifi Connection
```

```
}

void loop() {
  if (millis() - lastConnectionTime > TPOST)
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");

    lastConnectionTime = millis();
  }
}
```

```
//Send request to server every TPOST seconds
//Get data with GET Request from Thingworx
//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);
  myThing.Wifi(ssid, password);
```

```
}

void loop() {
  if (millis() - lastConnectionTime > TPOST)
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");

    lastConnectionTime = millis();
  }
}
```

```
//Serial
//Start t
```

```
//Send req
//Get data with GET Request from Thingworx
//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)
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");

    lastConnectionTime = millis();
  }
}
```

```
//Serial
//Start t
```

```
for debug purposes
```

```
//Send request to server every TPOST seconds
//Get data with GET Request from Thingworx
//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);
  myThing.Wifi(ssid, password);
}
```

```
void loop() {
  if (millis() - lastConnectionTime > TPOST)
  {
    sensor_1=myThing.getjson(" Sensorwert_1 ");

    lastConnectionTime = millis();
  }
}
```

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

```
//Serial
//Start t
for debug purposes
```

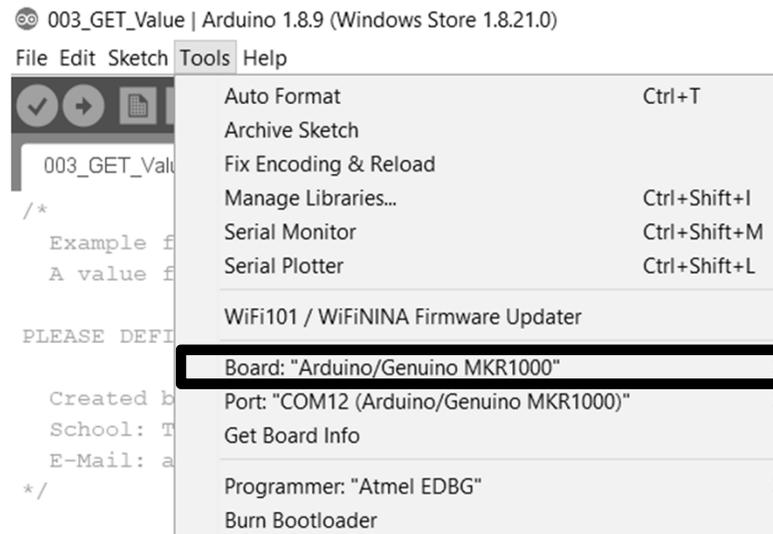
```
//Send req
//Get data with GET Request from Thingworx
//Refresh last connection time for if
```



# THINGWORX MKR1000 LIBRARY

## 003\_GET\_VALUE – ABFRAGEN EINES WERTES

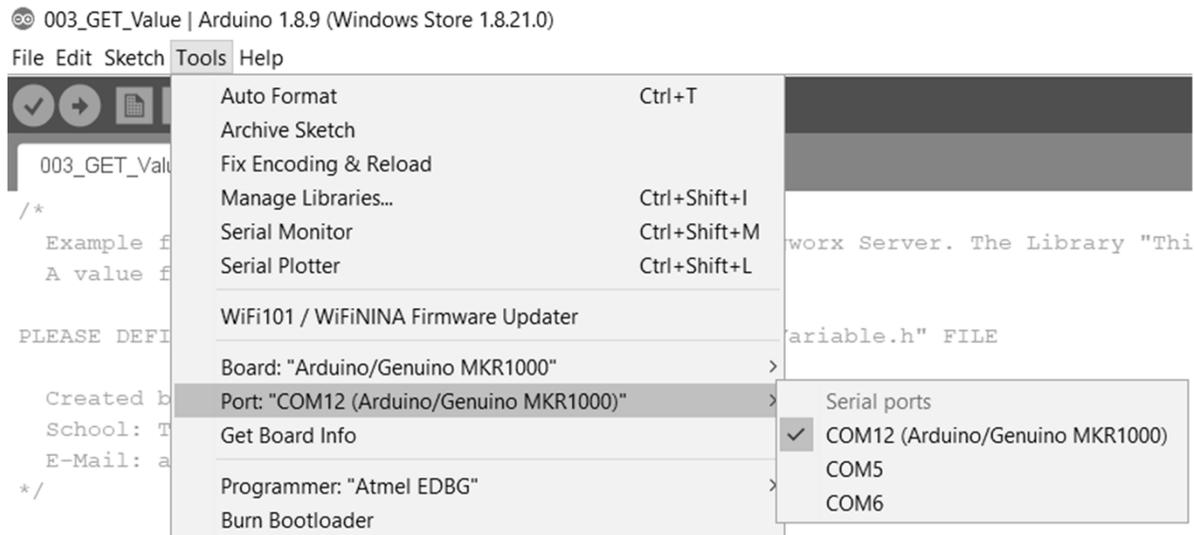
- Einstellung des MKR1000 Mikrocontrollers



# 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



003\_GET\_Value | Arduino 1.8.9 (Windows Store 1.8.21.0)

File Edit Sketch Tools Help

003\_GET\_Value

```
/*  
  Example for using an Arduino MKR1000 with a PTC Thingworx Serv  
  A value from the Thingworx Server is obtained.  
  
  PLEASE DEFINE ALL VARIABLES IN THE "Thingworx_MKR1000_Variable.h"  
  
  Created by Armin Fischer, Jan 2019.  
  School: TGM Vienna  
  E-Mail: afischer2@tgm.ac.at  
*/  
  
//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);
```

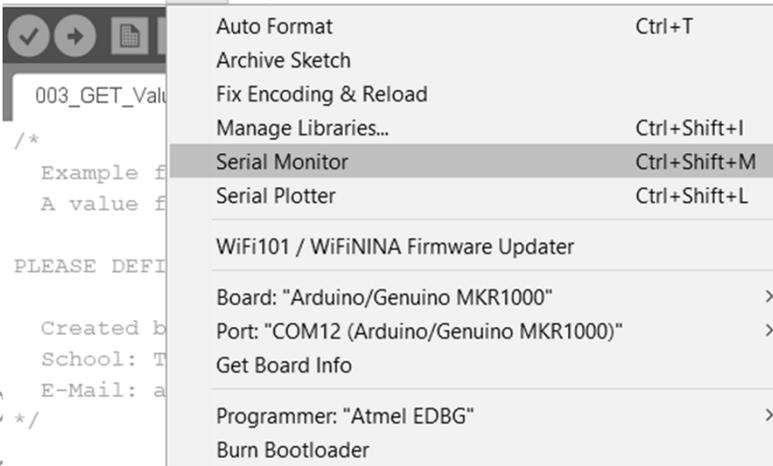
# THINGWORX MKR1000 LIBRARY

## 003\_GET\_VALUE – ABFRAGEN EINES WERTES

- Einschalten des Serial Monitor um den Ablauf des Programmes nachzuvollziehen

003\_GET\_Value | Arduino 1.8.9 (Windows Store 1.8.21.0)

File Edit Sketch Tools Help



The screenshot shows the 'Tools' menu in the Arduino IDE. The 'Serial Monitor' option is highlighted, and an arrow points from it to the Serial Monitor window on the right. Other options in the menu include Auto Format (Ctrl+T), Archive Sketch, Fix Encoding & Reload, Manage Libraries... (Ctrl+Shift+I), Serial Plotter (Ctrl+Shift+L), WiFi101 / WiFiNINA Firmware Updater, Board: "Arduino/Genuino MKR1000", Port: "COM12 (Arduino/Genuino MKR1000)", Get Board Info, Programmer: "Atmel EDBG", and Burn Bootloader.



The screenshot shows the Serial Monitor window titled 'COM12'. The output text is as follows:

```
15:41:27.866 -> Connected to: twx.htl.schule:8080
15:41:27.866 -> GET twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Properties/Sensorwert_1
15:41:27.866 -> Host: twx.htl.schule
15:41:27.866 -> Thing/Property: 920417_18FISA_00_Dummy/Sensorwert_1
15:41:27.866 -> appKey: 4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:41:27.866 ->
15:41:28.917 -> Value: 25.00
```

The window also shows a 'Send' button, a scroll bar, and settings at the bottom: 'Autoscroll' and 'Show timestamp' are checked, 'Both NL & CR' is selected, '9600 baud' is set, and there is a 'Clear output' button.

# 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

```
COM12
15:45:41.591 -> SSID: UPC3F7C124
15:45:41.591 -> IP Address: 192.168.0.185
15:45:41.591 -> Signal strength (RSSI):-41 dBm
15:45:41.625 -> Connected to: twx.htl.schule:8080
15:45:41.625 -> GET twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Properties/Sensorwert_1?appKey=4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:41.625 -> Host: twx.htl.schule
15:45:41.625 -> Thing/Property: 920417_18FISA_00_Dummy/Sensorwert_1
15:45:41.625 -> appKey: 4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:41.625 ->
15:45:42.674 -> Value: 25.00
15:45:47.724 -> Connected to: twx.htl.schule:8080
15:45:47.724 -> GET twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Properties/Sensorwert_1?appKey=4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:47.724 -> Host: twx.htl.schule
15:45:47.724 -> Thing/Property: 920417_18FISA_00_Dummy/Sensorwert_1
15:45:47.724 -> appKey: 4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:47.724 ->
15:45:48.766 -> Value: 25.00
15:45:53.797 -> Connected to: twx.htl.schule:8080
```

Autoscroll  Show timestamp

Both NL & CR 9600 baud Clear output

Erfolgreiches Verbinden mit dem WLAN  
„UPC3F7C124“.  
Zugewiesene IP-Adresse ist  
192.168.0.185

# 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

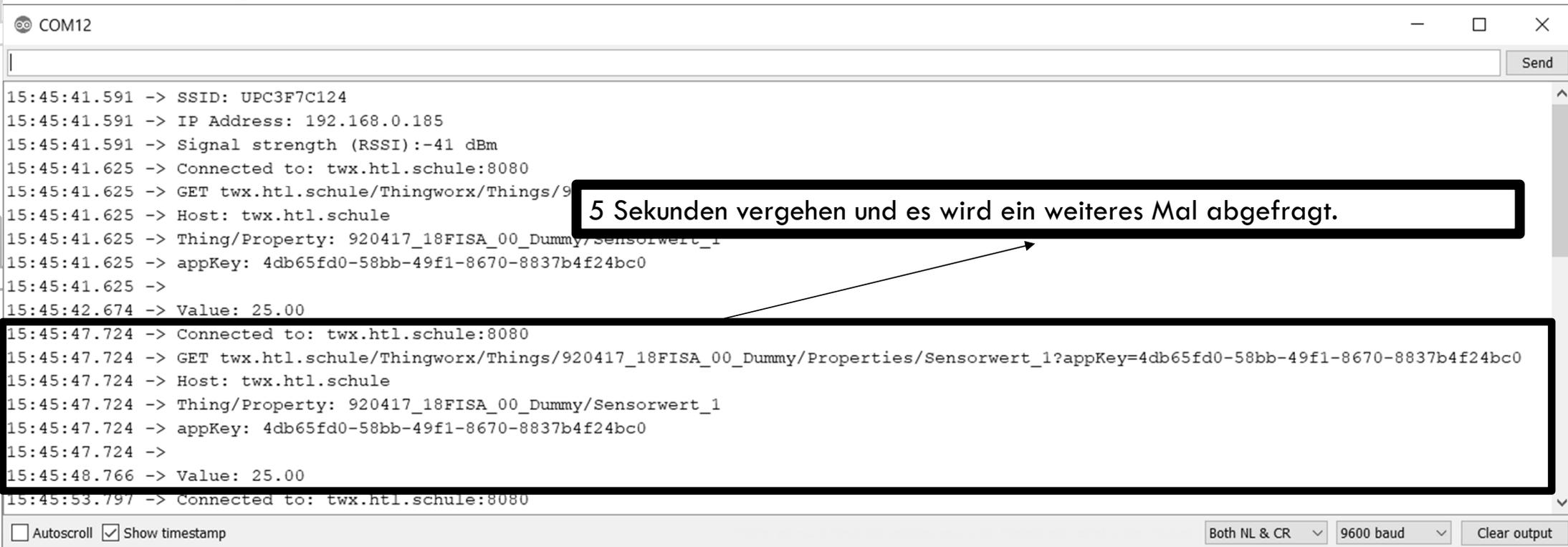
```
COM12
15:45:41.591 -> SSID: UPC3F7C124
15:45:41.591 -> IP Address: 192.168.0.185
15:45:41.591 -> Signal strength (RSSI):-41 dBm
15:45:41.625 -> Connected to: twx.htl.schule:8080
15:45:41.625 -> GET twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Properties/Sensorwert_1?appKey=4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:41.625 -> Host: twx.htl.schule
15:45:41.625 -> Thing/Property: 920417_18FISA_00_Dummy/Sensorwert_1
15:45:41.625 -> appKey: 4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:41.625 ->
15:45:42.674 -> Value: 25.00
15:45:47.724 -> Connected to: twx.htl.schule:8080
15:45:47.724 -> GET twx.htl.schule/Thingworx/Things
15:45:47.724 -> Host: twx.htl.schule
15:45:47.724 -> Thing/Property: 920417_18FISA_00_Du
15:45:47.724 -> appKey: 4db65fd0-58bb-49f1-8670-883
15:45:47.724 ->
15:45:48.766 -> Value: 25.00
15:45:53.797 -> Connected to: twx.htl.schule:8080
```

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



```
COM12
15:45:41.591 -> SSID: UPC3F7C124
15:45:41.591 -> IP Address: 192.168.0.185
15:45:41.591 -> Signal strength (RSSI):-41 dBm
15:45:41.625 -> Connected to: twx.htl.schule:8080
15:45:41.625 -> GET twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Sensorwert_1
15:45:41.625 -> Host: twx.htl.schule
15:45:41.625 -> Thing/Property: 920417_18FISA_00_Dummy/Sensorwert_1
15:45:41.625 -> appKey: 4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:41.625 ->
15:45:42.674 -> Value: 25.00
15:45:47.724 -> Connected to: twx.htl.schule:8080
15:45:47.724 -> GET twx.htl.schule/Thingworx/Things/920417_18FISA_00_Dummy/Properties/Sensorwert_1?appKey=4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:47.724 -> Host: twx.htl.schule
15:45:47.724 -> Thing/Property: 920417_18FISA_00_Dummy/Sensorwert_1
15:45:47.724 -> appKey: 4db65fd0-58bb-49f1-8670-8837b4f24bc0
15:45:47.724 ->
15:45:48.766 -> Value: 25.00
15:45:53.797 -> Connected to: twx.htl.schule:8080
```

5 Sekunden vergehen und es wird ein weiteres Mal abgefragt.

Autoscroll  Show timestamp

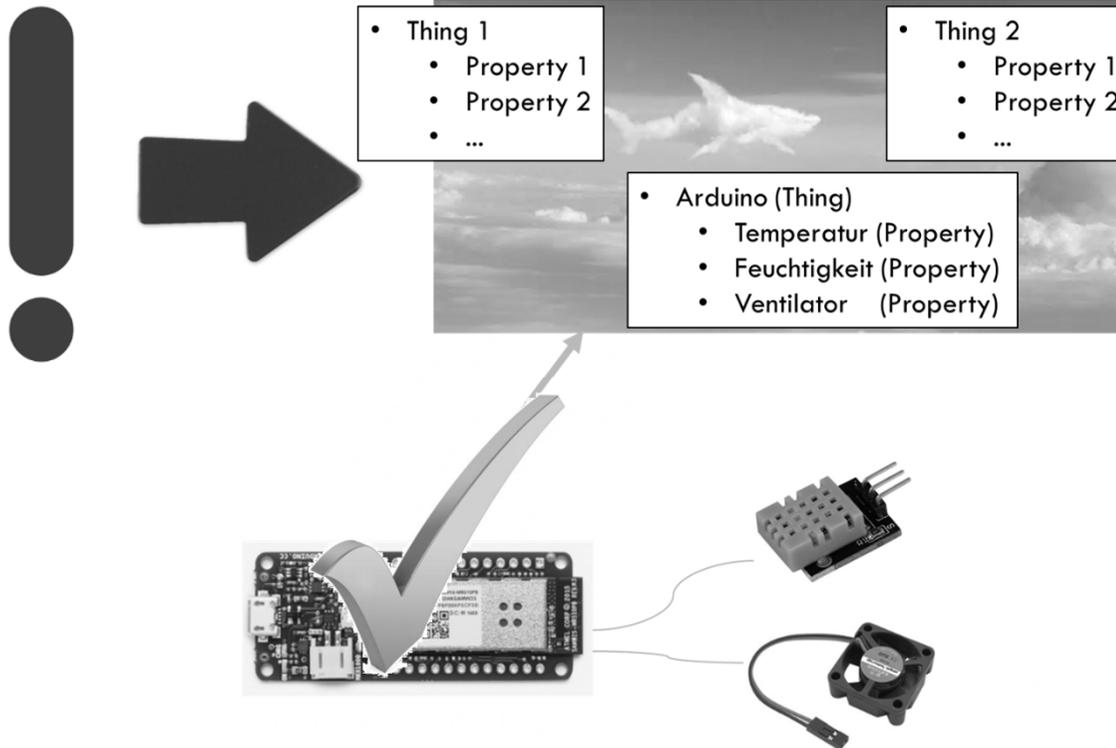
Both NL & CR 9600 baud Clear output



# 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

The screenshot displays the ThingWorx Composer web interface. At the top, the browser address bar shows the URL `https://twx.htl.schule/Thingworx/Composer/index.html`. The ThingWorx logo and search bar are visible on the left. The main navigation menu on the left is categorized into several sections: 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, and SECURITY.

The central area shows a list of Things. The list is filtered by "Exclude System Objects" and displays 148 items. The table columns are View, Name, Description, and Modified. The list includes items such as "dibse-raspi3", "die\_thing", "920417\_18FISA\_00\_Dummy", "Jenbach-Bonapace\_Gruber", "Jenbach-Delic-Fanki", "dibse\_raspi1", "cd-thing2", "dibse\_raspi2", "910417\_Waterbike", "910417\_Raspberry" (with description "Daten des SensorHAT des Raspberry PI"), "920417\_18FISA\_00\_Library\_Test" (with description "Test Thing from Armin Fischer Testing Library"), and "920417\_18FISA\_00\_Lucky\_Shield" (with description "TGM Lucky Shield").

A callout box with a black border and white background is overlaid on the right side of the interface. It contains the text "Auswahl der verschiedenen Funktionen. Beispiele:" followed by a bulleted list: "Erstellung von Things", "Erstellung von Benutzeroberflächen „Mash-Ups“", and "...". An arrow points from the "die\_thing" row in the table to the callout box.

**Auswahl der verschiedenen Funktionen.  
Beispiele:**

- Erstellung von Things
- Erstellung von Benutzeroberflächen „Mash-Ups“
- ...

# OBERFLÄCHE

The screenshot displays the ThingWorx Composer web interface. The browser address bar shows the URL `https://twx.htl.schule/Thingworx/Composer/index.html`. The interface includes a top navigation bar with the ThingWorx logo, a search bar, and several menu items: `+ New Entity`, `Import/Export`, `Monitoring`, `Help`, and `Learning Connector`. The user's name `iot-seminar16` is visible in the top right corner.

The main content area is titled "Things" and features a search bar with the placeholder text "Type to filter list...". Below the search bar are buttons for `+ New`, `View`, `Edit`, `Duplicate`, `Delete`, and `Permissions`. A filter dropdown is set to `Exclude System Objects`. The interface indicates that there are 148 items shown.

A table lists the Things, with columns for `View`, `Name`, `Description`, and `Modified`. The table contains the following entries:

View	Name	Description	Modified
<input type="checkbox"/>	dibse-raspi3		2019-03-19 16:38:07.780
<input type="checkbox"/>	die_thing		2019-03-19 13:00:13.018
<input type="checkbox"/>	920417_18FISA_00_Dummy		2019-03-18 13:20:12.654
<input type="checkbox"/>	Jenbach-Bonapace_Gruber		2019-03-15 11:10:19.828
<input type="checkbox"/>	Jenbach-Delic-Fanki		2019-03-15 08:58:32.754
<input type="checkbox"/>	dibse_raspi1		2019-03-13 14:17:40.059
<input type="checkbox"/>			2019-03-13 10:50:43.113
<input type="checkbox"/>	dibse_raspi2		2019-03-13 10:15:27.540
<input type="checkbox"/>	910417_Waterbike		2019-03-11 08:54:45.692
<input type="checkbox"/>	910417_Raspberry	Daten des SensorHAT des Raspberry PI	2019-03-08 11:32:58.015
<input type="checkbox"/>	920417_18FISA_00_Library_Test	Test Thing from Armin Fischer Testing Library	2019-03-05 17:51:05.258
<input type="checkbox"/>	920417_18FISA_00_Lucky_Shield	TGM Lucky Shield	2019-03-05 16:58:08.820

A callout box with a black border and white background contains the text "Bearbeitungsfenster für die ausgewählten Funktionen". An arrow points from this box to the "View" column header in the table.

# ERSTELLUNG THING

The screenshot shows the ThingWorx Composer web interface. The browser address bar displays `https://twx.htl.schule/Thingworx/Composer/index.html`. The top navigation bar includes the ThingWorx logo, a search bar, and menu items for 'New Entity', 'Import/Export', 'Monitoring', 'Help', and 'Learning Connector'. The user is logged in as 'iot-seminar16'. The main content area shows a 'Things' list with a search bar and a '+ New' button highlighted with a black box. Below the list, there are buttons for 'View', 'Edit', 'Duplicate', 'Delete', and 'Permissions'. The list itself has columns for 'View', 'Name', 'Description', and 'Modified'. Two items are visible: 'dibse-raspi3' and 'die.thing', both with a 'Modified' date of '2019-03-19 16:38:07.780'.

This screenshot shows the 'New Thing' configuration form. The title bar reads 'New Thing Thing'. The 'Save' button is highlighted with a black box. The form is divided into 'ENTITY INFORMATION' and 'General Information'. Under 'ENTITY INFORMATION', there are sections for 'General Information', 'Properties', 'Services', 'Events', 'Subscriptions', and 'Home Mashup'. Under 'PERMISSIONS', there are sections for 'Visibility', 'Design Time', and 'Run Time'. The 'General Information' section contains several fields: 'Name' (with value 'Schule\_JahrKlasse\_KatNr\_Thingname'), 'Description', 'Project' (with value 'Search Projects'), 'Tags' (with value 'Search Model Vocabulary'), 'Thing Template' (with value 'GenericThing'), and 'Implemented Shapes' (with value 'Search Thing Shapes'). A text box with the text 'Namenskonvention siehe nächste Folie' is positioned above the 'Name' field. The 'Active' checkbox is checked. Other fields include 'Home Mashup' (with value 'Search Mashups'), 'Avatar' (with a 'Change' button), 'Published' (checkbox), 'Identifier' (with a 'Browse...' button), 'Last Modified Date' (with value 'No date and time selected'), and 'Value Stream' (with value 'Search 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\_SchuljahrJahrgangnummer\_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

The screenshot shows the ThingWorx Composer web interface. The browser address bar displays `https://twx.htl.schule/Thingworx/Composer/index.html`. The interface includes a navigation sidebar on the left with categories like MODELING, ANALYTICS, VISUALIZATION, DATA STORAGE, COLLABORATION, and SECURITY. The main area shows a list of Things, filtered by 'Exclude System Objects'. The table has columns for View, Name, Description, and Modified. The entry '920417\_18FISA\_00\_Dummy' is highlighted with a black box. A callout box labeled 'Klick' points to the gear icon of this entry, and another callout box labeled 'Nächste Folie' points to the right side of the table.

View	Name	Description	Modified
<input type="checkbox"/>	dibse-raspi3		2019-03-19 16:38:07.780
<input type="checkbox"/>	die_thing		2019-03-19 13:00:13.018
<input type="checkbox"/>	920417_18FISA_00_Dummy		2019-03-18 13:20:12.654
<input type="checkbox"/>	Jenbach-Bonapace_Gruber		2019-03-15 11:10:19.828
<input type="checkbox"/>	Jenbach-Delic-Fanki		2019-03-15 08:58:32.754
<input type="checkbox"/>	dibse_raspi1		2019-03-13 14:17:40.059
<input type="checkbox"/>	cd-thing2		2019-03-13 10:50:43.113
<input type="checkbox"/>	dibse_raspi2		2019-03-13 10:15:27.540
<input type="checkbox"/>	910417_Waterbike		2019-03-11 08:54:45.692
<input type="checkbox"/>	910417_Raspberry	Daten des SensorHAT des Raspberry PI	2019-03-08 11:32:58.015
<input type="checkbox"/>	920417_18FISA_00_Library_Test	Test Thing from Armin Fischer Testing Library	2019-03-05 17:51:05.258
<input type="checkbox"/>	920417_18FISA_00_Lucky_Shield	TGM Lucky Shield	2019-03-05 16:58:08.820
<input type="checkbox"/>			2019-03-01

# ERSTELLUNG PROPERTY

**General Information**

Name ? Schule\_JahrKlasse\_KatNr\_Thingname

Description ?

Active ?

Home Mashup ? Search Mashups

Avatar ? Change

Schule\_JahrKlasse\_KatNr\_Thingname Thing ? Save Cancel Edit To Do 2 More ▾

**Properties** + Add My Property Manage Bindings Edit Delete Duplicate

**My Properties**

Edit	Name	Type	Alerts	Additional Info	Default Value	Value	DataChange
<input type="checkbox"/>	New Property		0 Alerts				Set Value

**New Property** My Property ▾

**General Property Info** ?

Name ?

Description ?

Category ?

Alerts ?

**BaseType Info** ?

Base Type ? -T- STRING ▾

Has Default Value ?

- 123 INTEGER
- 123 LONG
- JSON
- LOCATION
- MASHUPNAME
- MENUNAME
- NOTIFICATIONCONTENTNAME
- NOTIFICATIONDEFINITIONNAME
- # NUMBER
- PASSWORD

**Aspects** ?

Persistent ?

Read-only ?

Logged ?

**Data Change Info** ?

Data Change Type ? Value ▾

Cancel Done Done and Add

**Ganze Zahl Kommazahl**

# PROPERTY – WERT EINSTELLEN ODER AKTUALISIEREN

Schule\_JahrKlasse\_KatNr\_Thingname Thing Save Cancel Edit To Do More

Properties ? + Add My Property Manage Bindings Edit Delete Duplicate

My Properties

Edit	Name	Type	Alerts	Additional Info	Default Value	Value	DataChange
	-T- Sensorwert_1		0 Alerts				Value

GenericThing (ThingTemplate) - Properties

Generic Properties

Wert aktualisieren

Wert einstellen.

- 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 ausgeschaltet.
- 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);
  myThing.Wifi(ssid, password);
  pinMode(RELAY_1,OUTPUT);
  pinMode(RELAY_2,OUTPUT);
}

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();
}
```

//Serial communications with computer at 9600 bauds for debug purposes  
//Start the Wifi Connection  
//Digital pin 1 is an output pin  
//Digital pin 2 is an output pin

//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.

```
//DHT11 Sensor is on Pin 3
```

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.

Serial.begin(9600); //Serial prints for debug purposes

# 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) //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

In Thingworx Composer das Thing auswählen

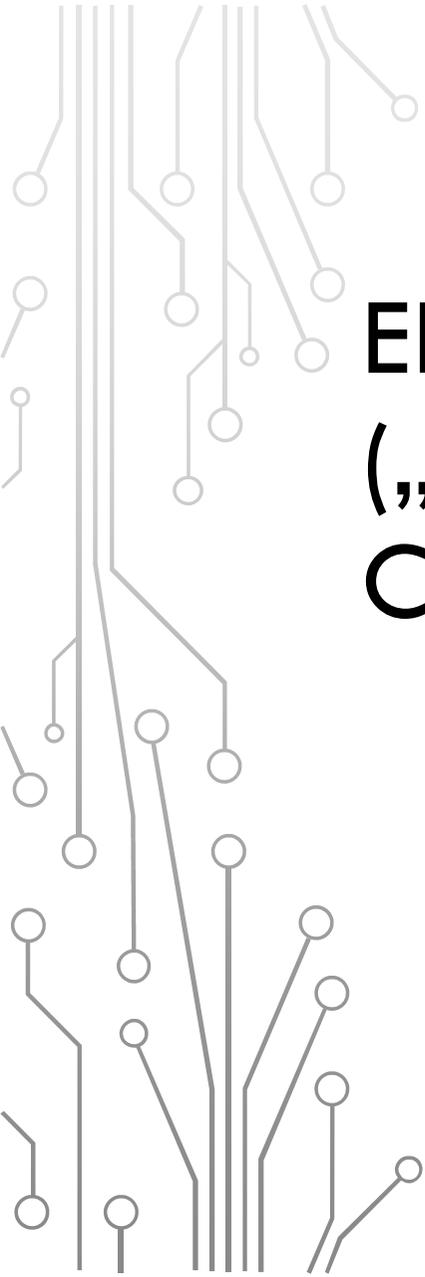
Bei neuem Wert muss aktualisiert werden

Wert ablesen

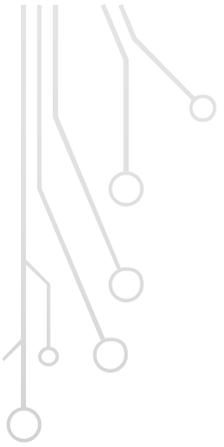
Edit	Name	Type	Alerts	Additional Info	Default Value	Value	DataChange
<input type="checkbox"/>	# BME280_TEMP		0 Alerts	°C		20.5	Set Value: 0
<input type="checkbox"/>	# BME280_HUM		0 Alerts	%		36.0	Set Value: 0
<input type="checkbox"/>	# BME280_PRESSURE		0 Alerts	hPa		0.0	Set Value: 0
<input type="checkbox"/>	# BME280_HEIGHT		0 Alerts	m		0.0	Set Value: 0
<input type="checkbox"/>	123 LED1		0 Alerts			0	Set Value: 0
<input type="checkbox"/>	123 REL1		0 Alerts			0	Set Value: 0

GenericThing (ThingTemplate) - Properties

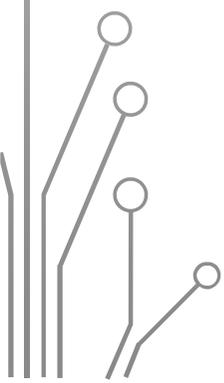
- Generic Properties



# 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

The screenshot displays the ThingWorx user interface for managing Mashups. The top navigation bar includes the ThingWorx logo, a search bar, and menu items for New Entity, Import/Export, Monitoring, Help, and Learning Connector. The user is logged in as 'iot-seminar16'. The left sidebar shows various categories: 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, and SECURITY. The 'Mashups' category is selected and highlighted.

The main content area shows the Mashups management interface. At the top, there is a search bar and a toolbar with buttons for '+ New', 'View', 'Edit', 'Duplicate', 'Delete', and 'Permissions'. Below the toolbar, there is a filter option 'Exclude System Objects' and a 'Showing: 76 items' indicator. The main area contains a table of Mashups with columns for View, Name, Description, and Modified. A 'New Mashup' dialog box is open, showing options for Mashup Type (Page, Thing Template, Thing Shape) and Layout Options (Responsive, Static).

View	Name	Description	Modified
	diemash		2019-03-20 04:29:48.879
	dibse-raspi3-mashup		2019-03-19 17:51:21.428
	Delic_Fanki_Mashup		2019-03-15 11:15:09.946
	Gruber_Dummer_Mashup		2019-03-15 11:09:47.892
	DummaBua		2019-03-15 11:00:45.399
	910417_Raspberry		2019-03-11 09:20:49.627
	910417_Waterbike		2019-03-11 09:14:37.832
	920417_185C_10_Lucky-Shield-Ma		2019-03-11 01:23:10.354
	RFID_11_5CHMBZ_1819		2019-03-06 11:49:48.969
	920417_185B_10_Lucky_Shield_Ma		2019-03-05 01:16:29.164
	TopiLolloh_mashup		2019-03-01 10:41:39.527
	Bildauswertung	Bildauswertung der Fundboxüberwachung	2019-02-22 14:10:21.647
	910417_SensorkitX40	Anzeige der Daten des Sensorkit X40	2019-02-22 13:51:37.752

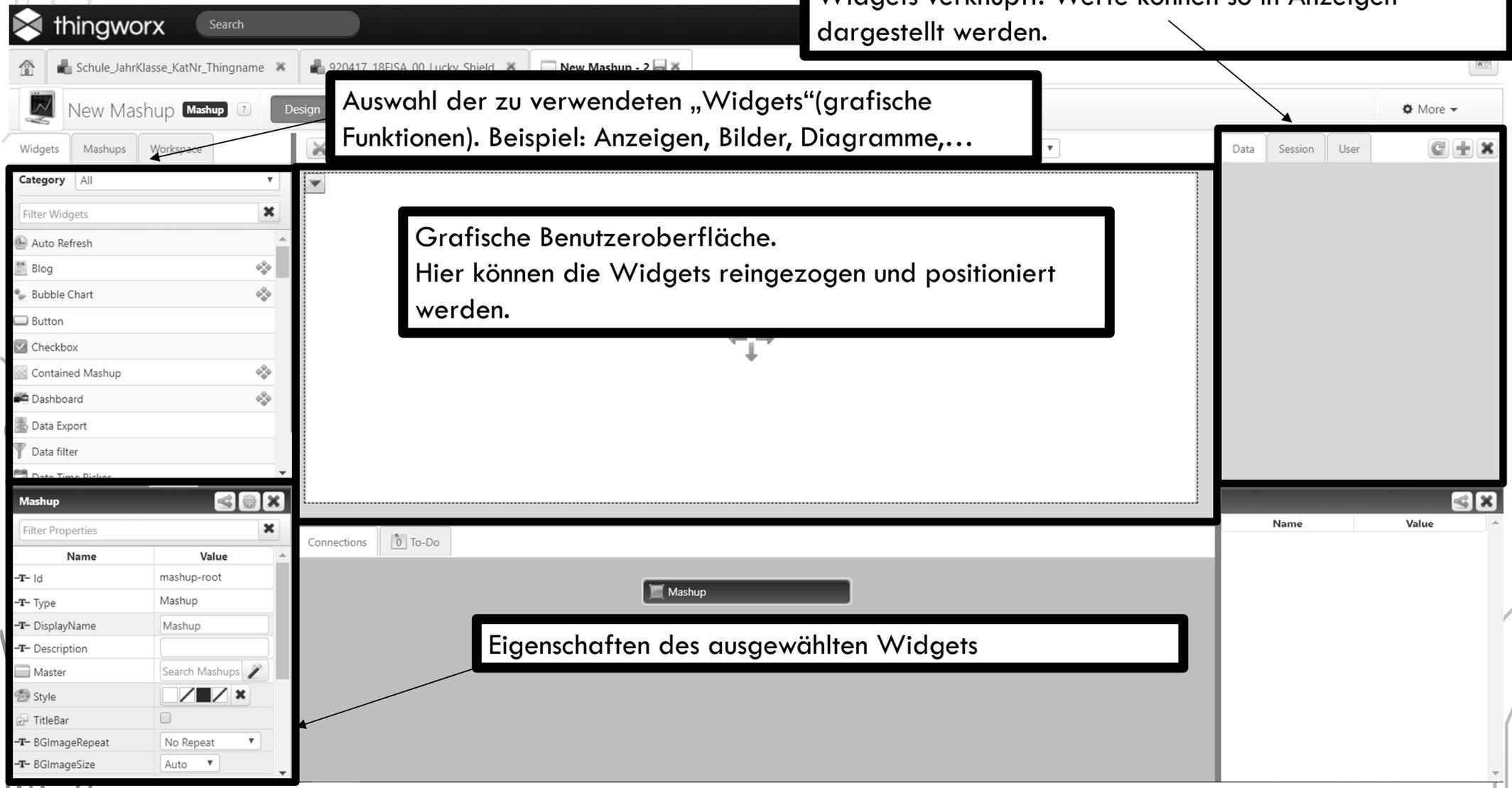
# MASH UP OBERFLÄCHE

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

The screenshot displays the ThingWorx IDE interface. At the top, the 'thingworx' logo and a search bar are visible. The main workspace is titled 'New Mashup' and contains a 'Design' tab. A 'Label' widget is being dragged from the 'Widgets' palette onto the workspace. A black box with the text 'Drag&Drop' is positioned over the widget in the palette, with an arrow pointing to the widget in the workspace. The 'Properties' panel for the selected widget, 'label-3', is open at the bottom left, showing various attributes and their values.

Name	Value
-T- Id	label-3
-T- Type	Label
-T- DisplayName	
-T- Description	
-T- Text	Temperatur
Style	
-T- Alignment	Left Aligned
-T- ToolTipField	
ShowDataLoading	<input checked="" type="checkbox"/>

# ANZEIGE ERSTELLEN

thingworx

Search

+ New Entity Import/Export Monitoring Help Learning Connector iot-seminar16

Schule\_JahrKlasse\_KatNr\_Thingname 920417\_18FISA\_00\_Lucky\_Shield New Mashup - 2 910417\_Waterbike **New Mashup - 3**

New Mashup Mashup Design Info Save Cancel Edit More

Widgets Mashups Workspace

Category All

Filter Widgets

Expression

Fieldset

File Upload

Folding Panel

**Gauge**

GeoTag

Google Location Picker

Google Map

Grid

gauge-4

Filter Properties

Name	Value
-T- Id	gauge-4
-T- Type	Gauge
-T- DisplayName	gauge-4
-T- Description	
# Data	
# MinValue	0
# MaxValue	100
ValueFormatter	State Formatting
FormatNeedle	<input checked="" type="checkbox"/>

Connections To-Do

gauge-4

Temperatur

Gauge

Name	Value
------	-------

# HINZUFÜGEN VON DATEN

The screenshot shows the ThingWorx interface with the 'Add Data' dialog box open. The dialog has a 'Select Entity' dropdown menu set to 'Things'. Below this is a 'Search Results' section with a search bar containing 'Things' and a 'Dynamic' checkbox. The search results show 223 items, with the first item, '920417\_18FISA\_00\_Dummy', highlighted. A callout box with the text 'Thing auswählen' points to this item. The 'Things' dropdown menu is also highlighted. The background shows a 'Temperatur' widget and a 'gauge-4' widget.

**Add Data**

Select Entity: Things

Search Results

Actions: + Thing

920417\_18FISA\_00\_Dummy

920417\_18FISA\_00\_Lucky\_Shield

920417\_18FISA\_00\_Library\_Test

920417\_185C\_10\_Lucky\_Shield

920417\_185B\_10\_Lucky\_Shield

910417\_Raspberry

RFID\_11\_5CHMBZ\_1819

**Thing auswählen**

Name	Value
-T- Id	gauge-4
-T- Type	Gauge
-T- DisplayName	gauge-4
-T- Description	
# Data	
# MinValue	0
# MaxValue	100
ValueFormatter	State Formatting
FormatNeedle	<input checked="" type="checkbox"/>

# HINZUFÜGEN VON DATEN

## Add Data

Select Entity 920417\_18FISA\_00\_Dummy Dynamic

Select Services

- All
- Alerts
- Bindings
- Configuration
- Data
- DataLogging
- Dependencies
- Editing
- Federation
- Identifier
- Lifecycle
- Maintenance
- Mashups
- Metadata
- Networks
- Permissions

getpropertyvalues ✕

- GetPropertyValues ➔
- GetPropertyValuesAsMultiRowTable ➔
- GetPropertyValuesVTQ ➔
- GetPropertyValuesVTQA ➔

Selected Services

Entity Type	Entity Name	Service	Mashup Loaded?	Remove
Things	920417_18FISA_00_Dummy	GetPropertyValues	<input type="checkbox"/>	<span>✕</span>

Cancel Done

# VERBINDEN DER DATEN MIT WIDGET

The screenshot displays the ThingWorx development environment. The main workspace shows a 'Temperatur' widget with a 'Select Binding Target' dialog box. The dialog lists several binding options: '# Data', '# MinValue', '# MaxValue', '-T- Legend', and '-T- ToolTipField'. The '# Data' option is highlighted with a black box. A 'Drag&Drop' label with an arrow points from the '# Data' option to the widget's binding area.

On the right side, the 'Data' panel is open, showing a table of data for 'Things\_920417\_18FISA\_00\_Dummy'. The table has columns for 'Name' and 'Value'. The first row shows '123 Sensorwert\_1', which is also highlighted with a black box. An arrow points from this data row to the '# Data' binding target in the dialog.

At the bottom, the 'Connections' window is visible, showing a flow from the 'Things\_920417\_18FISA\_00\_Dummy' data source through a 'GetPropertyValues' action to the 'All Data' widget, which then feeds into the '123 Sensorwert\_1' widget. A black box highlights this entire connection flow, with a text box stating: 'In diesem Fenster werden alle verbundenen Daten zu einem Widget angezeigt'.

On the left, the 'panel-2' properties window is open, showing various configuration options for the widget, such as 'Name', 'Type', 'Style', and 'Z-index'.

# HINZUFÜGEN EINES REFRESH BUTTONS

The screenshot shows the ThingWorx design environment. On the left, the widget palette is open, and the 'Auto Refresh' widget is selected. Below it, the 'autorefresh-5' properties panel is visible, showing the 'RefreshInterval' set to 30. In the center, a 'Temperatur' gauge is displayed on a dashboard. A 'Refresh Now' button is placed on the dashboard, and an arrow points from the 'Auto Refresh' widget in the palette to this button. A text box explains the button's function.

Dieser Button ist notwendig um die Anzeige in einem bestimmten Zeitintervall upzudaten.

Name	Value
Id	autorefresh-5
Type	Auto Refresh
DisplayName	autorefresh-5
Description	
RefreshInterval	30
AutoRefresh	<input checked="" type="checkbox"/>
ShowControls	<input checked="" type="checkbox"/>
AutoRefreshTabSe...	0
Label	Refresh Now

# HINZUFÜGEN EINES REFRESH BUTTONS

The screenshot displays the ThingWorx IDE interface. At the top, the 'thingworx' logo and search bar are visible. The main workspace shows a 'Temperatur' widget with a 'Gauge' component. A 'Refresh Now' button is being added to the widget's control area. A 'Refresh' button is also visible in the 'autorefresh-5' widget's configuration area. A 'Drag&Drop' label with arrows indicates the process of moving the 'Refresh' button from the configuration area to the widget's control area.

The 'autorefresh-5' widget configuration table is as follows:

Name	Value
-T- Id	autorefresh-5
-T- Type	Auto Refresh
-T- DisplayName	autorefresh-5
-T- Description	
# RefreshInterval	30
AutoRefresh	<input checked="" type="checkbox"/>
ShowControls	<input checked="" type="checkbox"/>
# AutoRefreshTabSe...	0
-T- Label	Refresh Now

The 'Connections' panel shows the following connections:

- Things\_920417\_18FISA\_00\_Dummy
- GetPropertyValues
- All Data
- 123 Sensorwert\_1
- # Data
- gauge-4

# HINZUFÜGEN EINES DIAGRAMMES

The screenshot displays the ThingWorx development environment. At the top, the 'thingworx' logo and search bar are visible. The main workspace is titled '123 Mashup' and shows a 'Temperatur' mashup with a 'Time Series Chart' widget. A 'Drag&Drop' callout box points to the widget in the workspace. On the left, the 'Widgets' panel lists various components, with 'Time Series Chart' highlighted. On the right, the 'Data' panel shows the 'GetPropertyValue' operation for the 'Things\_920417\_18FISA\_00\_Dummy' entity, displaying returned data including 'Sensorwert\_1'.

**Widgets Panel:**

- Tag Picker
- TagCloud
- TextArea
- TextBox
- Time Selector
- Time Series Chart**
- Tree
- Validator
- Value Display
- Vertical Slider

**Data Panel:**

Name	Value
ServiceInvokeCompleted	⇒
AllDataChanged	⇒
SelectedRowsChanged	⇒

# HINZUFÜGEN EINES DIAGRAMMES

Add Data

Select Entity: 920417\_18FISA\_00\_Dummy

Select Services: querypropertyhistory

Selected Services

Entity Type	Entity Name	Service	Mashup Loaded?	Remove
Things	920417_18FISA_00_Dummy	QueryPropertyHistory	<input type="checkbox"/>	<input type="button" value="X"/>

Cancel Done

# HINZUFÜGEN EINES DIAGRAMMES

The screenshot displays the ThingWorx IDE interface. The main workspace shows a 'Temperatur' widget with a 'Gauge' and a 'Refresh Now' button. A 'Select Binding Target' dialog box is open, listing 'Data', 'DataSource1', 'DataSource2', and 'DataSource3'. The 'Data' option is highlighted with a black box. An arrow points from a 'Drag&Drop' label to this box. The right sidebar shows a data table with columns 'Name' and 'Value'.

Name	Value
oldestFirst	
# maxItems	
endDate	
query	Edit Query
startDate	
ServiceInvokeCompleted	
AllDataChanged	
SelectedRowsChanged	

# DIAGRAMM AKTUALISIEREN

The screenshot displays the ThingWorx IDE interface. At the top, the 'thingworx' logo and search bar are visible. The main workspace shows a mashup titled '123' in 'Design' mode. The central diagram, titled 'Temperatur', contains a 'Gauge' and a 'Time Series Chart'. A context menu is open over a widget, showing options: 'On', 'Refresh Now', 'Widget', 'Configure Bindings', and 'Refresh'. A callout box labeled 'Drag&Drop' points to the 'QueryPropertyHistory' widget in the right-hand data panel. The data panel shows a table with columns 'Name' and 'Value'.

Name	Value
oldestFirst	
# maxItems	
endDate	
query	Edit Query
startDate	
ServiceInvokeCompleted	
AllDataChanged	
SelectedRowsChanged	

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

The screenshot shows the Thingworx interface with the 'Value Streams' section selected in the left sidebar. A 'New' button is highlighted in the top toolbar. A 'Choose Template' dialog is open, displaying a table of templates. The 'ValueStream' template is selected, and the 'Choose' button is highlighted.

Name	Description
RemoteValueStream	Remote Value Stream
<b>ValueStream</b>	<b>Value Stream</b>

The background shows a list of Value Streams with columns for View, Name, and Modified. The 'Value Streams' menu item in the sidebar is also highlighted.

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

The screenshot displays the Thingworx web interface for configuring a 'Value Stream' entity. The top navigation bar includes the Thingworx logo, a search bar, and menu items for 'New Entity', 'Import/Export', 'Monitoring', 'Help', 'Learning Connector', and a user profile 'iot-seminar16'. The breadcrumb trail shows the path: 'Schule\_JahrKlasse\_KatNr\_Thingname' > '920417\_18FISA\_00\_Lucky\_Shield' > '920417\_18FISA\_00\_Dummy' > '123' > '920417\_18FISA\_00\_Dummy\_ValueStream'. Below the breadcrumb, there are 'Save', 'Cancel Edit', and 'To Do' buttons. The main content area is titled 'General Information' and contains several sections:

- ENTITY INFORMATION:** A sidebar menu with options like 'General Information', 'Properties', 'Services', 'Events', 'Subscriptions', 'Configuration', and 'Home Mashup'.
- PERMISSIONS:** Options for 'Visibility', 'Design Time', and 'Run Time'.
- CHANGE HISTORY:** A 'Change History' link.
- DEPENDENCIES:** Options for 'Entity Depends On' and 'Uses This Entity'.
- General Information Fields:**
  - Name:** '920417\_18FISA\_00\_Dummy\_ValueStream' (highlighted with a black box).
  - Description:** An empty text area.
  - Project:** '920417\_TGM'.
  - Tags:** 'Search Model Vocabulary'.
  - Thing Template:** 'ValueStream'.
  - Implemented Shapes:** 'Search Thing Shapes'.
  - Persistence Provider:** 'ThingworxPersistenceProvider'.
  - Active:** A checkbox that is checked (highlighted with a black box).
  - Home Mashup:** 'Search Mashups'.
  - Avatar:** 'Change' button.
  - Published:** An unchecked checkbox.
  - Identifier:** 'Browse...' button.
  - Last Modified Date:** '2019-03-20 10:35:30.972'.
  - Value Stream:** 'Search Thing'.
- Documentation:** A rich text editor with a toolbar containing icons for bold, italic, strikethrough, list, table, link, unlink, and other editing functions.

# EINSTELLUNGEN AM THING FÜR DAS SPEICHERN VON DATEN

The screenshot displays the ThingWorx user interface for configuring a 'Thing'. The top navigation bar includes the ThingWorx logo, a search bar, and various utility buttons like '+ New Entity', 'Import/Export', 'Monitoring', 'Help', and 'Learning Connector'. The user is logged in as 'iot-seminar16'. The breadcrumb trail shows the current entity: '920417\_18FISA\_00\_Dummy Thing'. The main content area is titled 'General Information' and contains several sections:

- ENTITY INFORMATION:** A sidebar menu with options like 'General Information', 'Properties', 'Services', 'Events', 'Subscriptions', and 'Home Mashup'.
- PERMISSIONS:** Options for 'Visibility', 'Design Time', and 'Run Time'.
- CHANGE HISTORY:** A 'Change History' button.
- DEPENDENCIES:** A section for 'Entity Depends On' and 'Uses This Entity'.
- General Information Form:**
  - Name:** 920417\_18FISA\_00\_Dummy
  - Description:** (Empty text area)
  - Project:** Search Projects
  - Tags:** Search Model Vocabulary
  - Thing Template:** GenericThing
  - Implemented Shapes:** Search Thing Shapes
  - Active:**
  - Home Mashup:** Search Mashups
  - Avatar:** Change
  - Published:**
  - Identifier:** Browse...
  - Last Modified Date:** 2019-03-20 10:25:55.248
  - Value Stream:** 920417\_18FISA\_00\_Dummy\_ValueStream
- Documentation:** A rich text editor with a toolbar containing icons for undo, redo, bold, italic, strikethrough, bulleted list, numbered list, link, unlink, insert link, insert image, and insert video.

# EINSTELLUNGEN AM THING FÜR DAS SPEICHERN VON DATEN

The screenshot displays the ThingWorx user interface for configuring a property. The top navigation bar includes the ThingWorx logo, a search bar, and various utility buttons like 'New Entity', 'Import/Export', 'Monitoring', 'Help', and 'Learning Connector'. The user is logged in as 'iot-seminar16'. The main workspace shows the configuration for a 'Thing' named '920417\_18FISA\_00\_Dummy'. The 'Properties' tab is active, showing a table of 'My Properties' with one entry: '123 Sensorwert\_1'. Below the table, the configuration form for 'Sensorwert\_1' is visible. The 'General Property Info' section shows the name 'Sensorwert\_1'. The 'Base Type Info' section shows the base type set to '123 INTEGER'. The 'Aspects' section has the 'Logged' checkbox checked. The 'Data Change Info' section shows the 'Data Change Type' set to 'Value' and the 'Change Threshold' set to '0'. The 'Save' button at the top is highlighted with a red box. The 'Properties' tab in the left sidebar and the 'Sensorwert\_1' entry in the table are also highlighted with red boxes. Arrows point from these boxes to the 'Save' button and the 'Logged' checkbox, respectively.

thingworx Search

+ New Entity Import/Export Monitoring Help Learning Connector iot-seminar16

Schule\_JahrKlasse\_KatNr\_Thingname 920417\_18FISA\_00\_Lucky\_Shield 920417\_18FISA\_00\_Dummy 123 920417\_18FISA\_00\_Dummy\_ValueStream

920417\_18FISA\_00\_Dummy Thing Save Cancel Edit To Do 1 More

ENTITY INFORMATION

- General Information
- Properties**
- Events
- Subscriptions
- Home Mashup

PERMISSIONS

- Visibility
- Design Time
- Run Time

CHANGE HISTORY

- Change History

DEPENDENCIES

- Entity Depends On
- Uses This Entity

**Properties** + Add My Property Manage Bindings Edit Delete Duplicate

**My Properties**

Edit	Name	Type	Alerts	Additional Info	Default Value	Value	DataChange
	123 Sensorwert_1		0 Alerts			25	Set Value: 0

**Sensorwert\_1** My Property

**General Property Info**

Name ? Sensorwert\_1

Description ?

Category ?

**Alerts** Manage Alerts

Enabled?	Type	Config	Name	Desc
----------	------	--------	------	------

**Base Type Info**

Base Type ? 123 INTEGER

Units ?

Min Value ?

Max Value ?

Has Default Value ?

**Aspects**

Persistent ?

Read-only ?

**Logged** ?

**Data Change Info**

Data Change Type ? Value

Change Threshold ? 0

Cancel Done

GenericThing (ThingTemplate) - Properties

- Generic Properties

# LETZTE EINSTELLUNGEN AM DIAGRAMM

The screenshot shows the ThingWorx interface with the widget palette on the left and a properties panel for 'panel-2' at the bottom. The widget palette includes items like Tag Picker, TagCloud, TextArea, TextBox, Time Selector, Time Series Chart, Tree, Validator, Value Display, and Vertical Slider. The properties panel for 'panel-2' shows various settings such as Id, Type, DisplayName, Description, Style, HideScrollbars, ShowDataLoading, ResetInputsToDefaultValue, and Z-index.

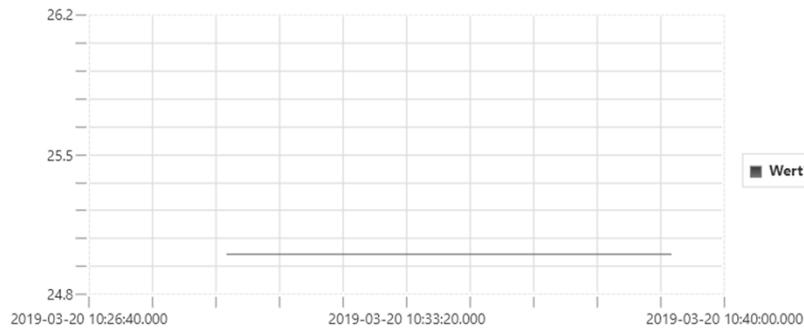
The screenshot shows the properties panel for a Time Series Chart widget. The 'XAxisField' property is set to 'timestamp' and the 'DataField1' property is set to 'Sensorwert\_1'. Other visible properties include LegendWidth, LegendLocation, LegendOrientation, MarkerSize, MarkerType, Smoothing, ShowXAxis, XAxisStyle, XAxisFormat, SecondaryYAxisZe..., AllowSelection, EnableHover, ShowXAxisGrid, ShowYAxisGrid, GridStyle, DataLabel1, SeriesType1, and SeriesMarkerType1.

The screenshot shows the ThingWorx interface with the 'View Mashup' button highlighted in the top right corner. The interface displays a mashup titled 'Temperatur' with a 'Gauge' widget. The 'View Mashup' button is located in the top right corner of the design view.

# FERTIGES MASHUP IM BROWSER



Temperatur



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