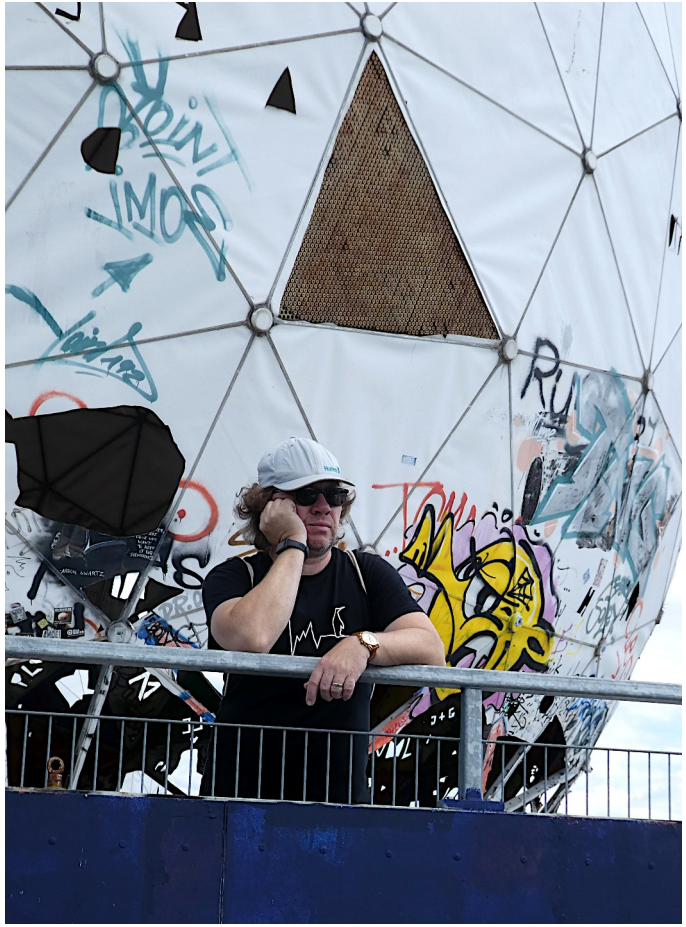
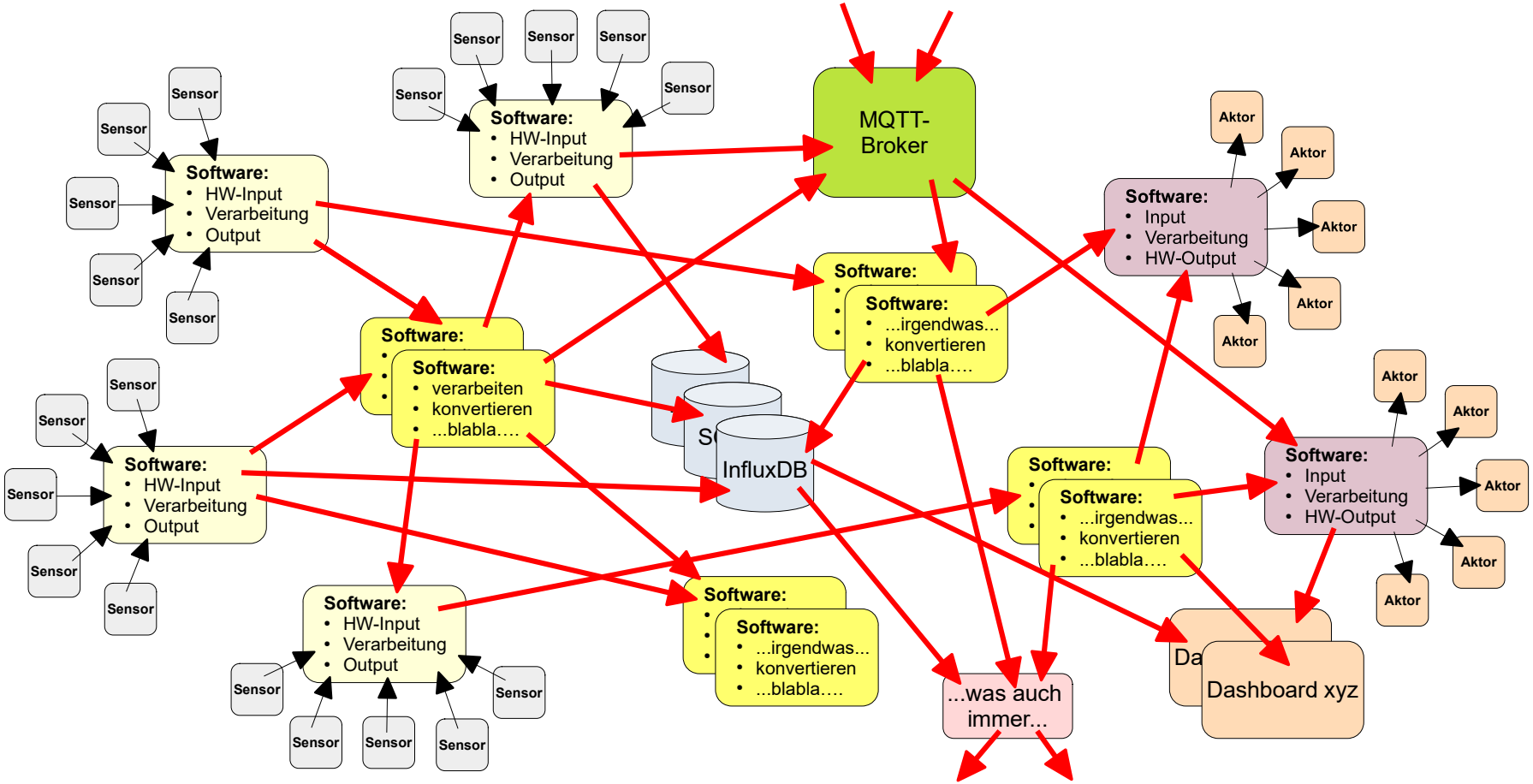
Datenkreuzung Telegraf

Uwe Berger
bergeruw@gmx.net

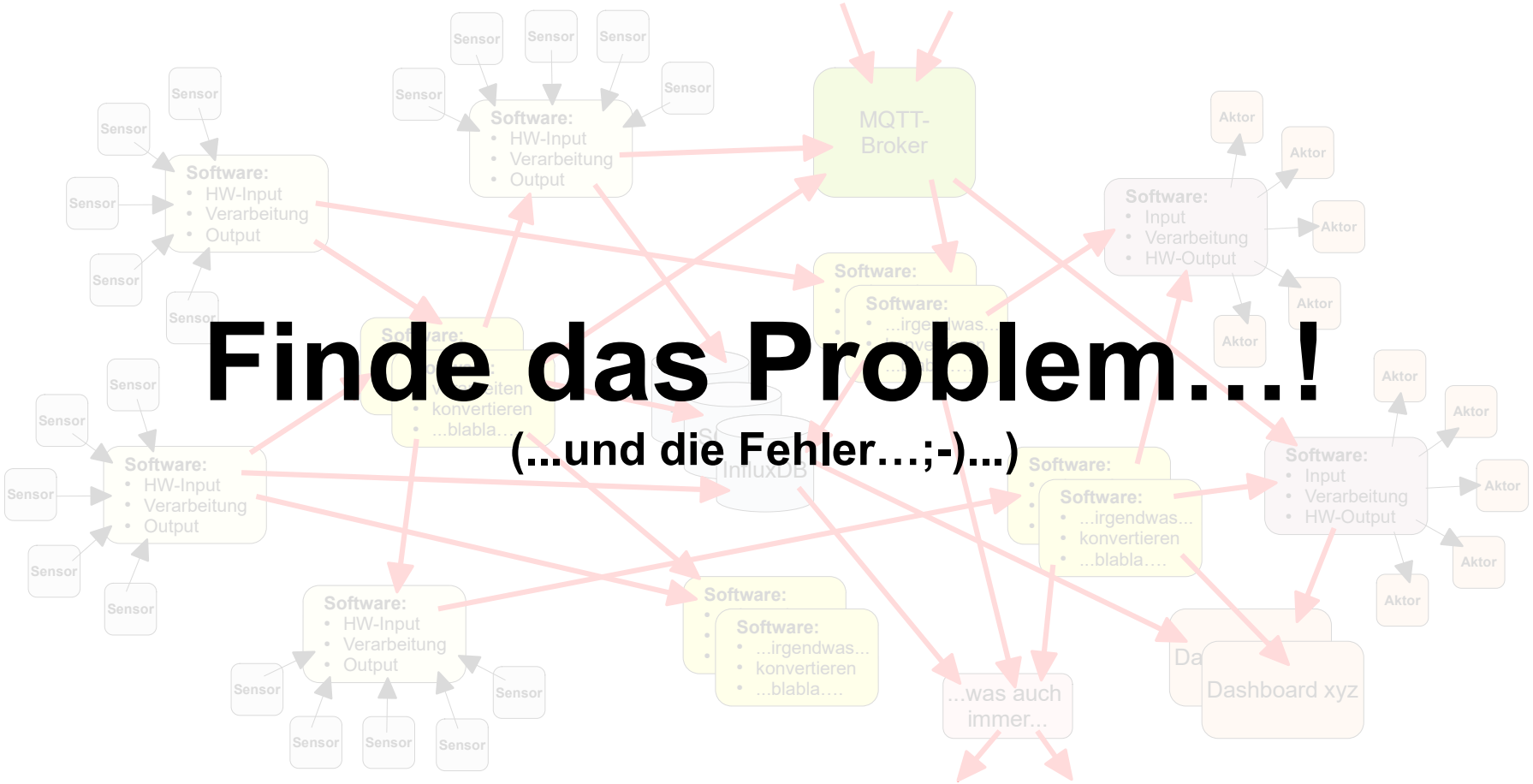
Uwe Berger



Motivation


Sensor
Software:
• HW-Input
• Verarbeitung
• Output
MQTT-Broker
Software:
• verarbeiten
• konvertieren
• ...blabla....
Software:
• ...irgendwas...
• konvertieren
• ...blabla....
Software:
• Input
• Verarbeitung
• HW-Output
Aktor
InfluxDB
...was auch immer...
Dashboard xyz

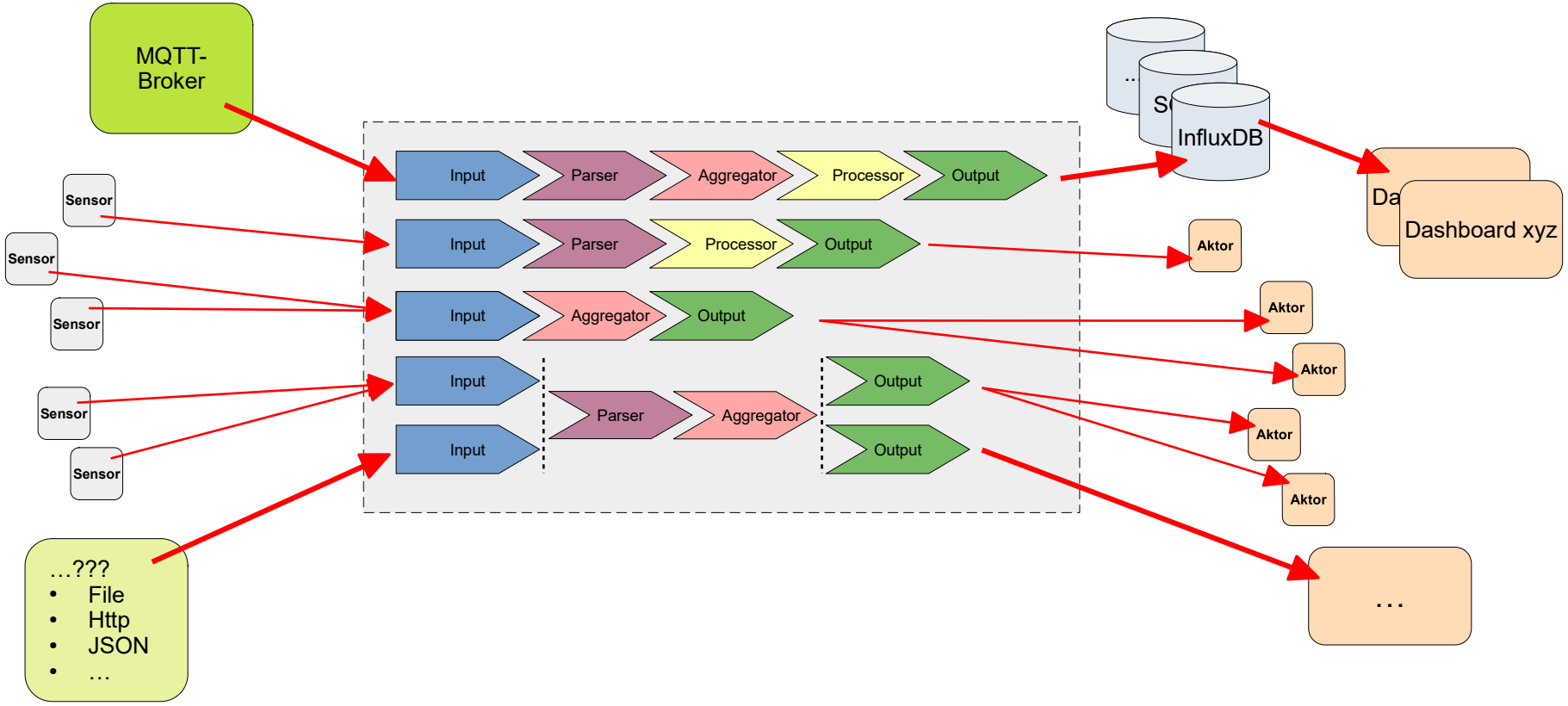
Motivation



Finde das Problem…!

(...und die Fehler…;-)...)

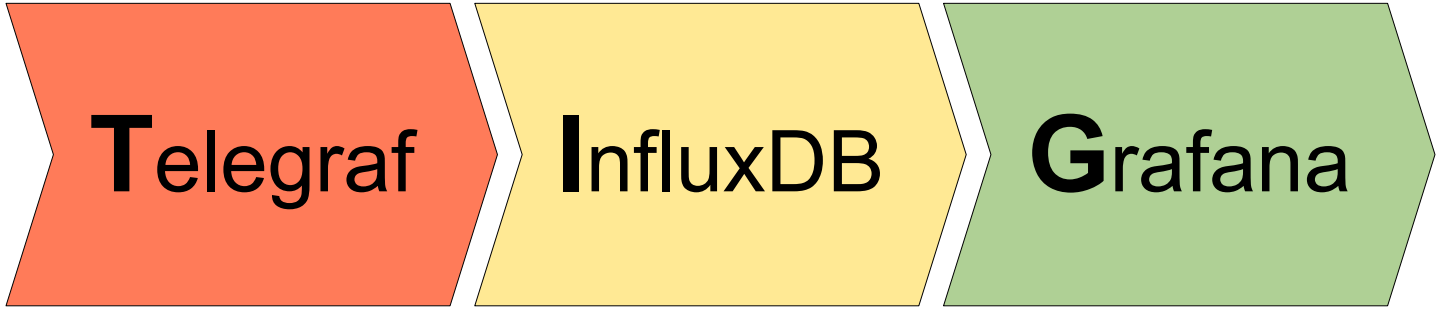
Motivation


MQTT-Broker
Sensor
Sensor
Sensor
Sensor
Sensor
...???
• File
• Http
• JSON
• ...
Input
Parser
Aggregator
Processor
Output
Input
Parser
Processor
Output
Input
Aggregator
Output
Input
Input
Parser
Aggregator
Output
Output
InfluxDB
Dashboard xyz
Aktor
Aktor
Aktor
Aktor
Aktor
...

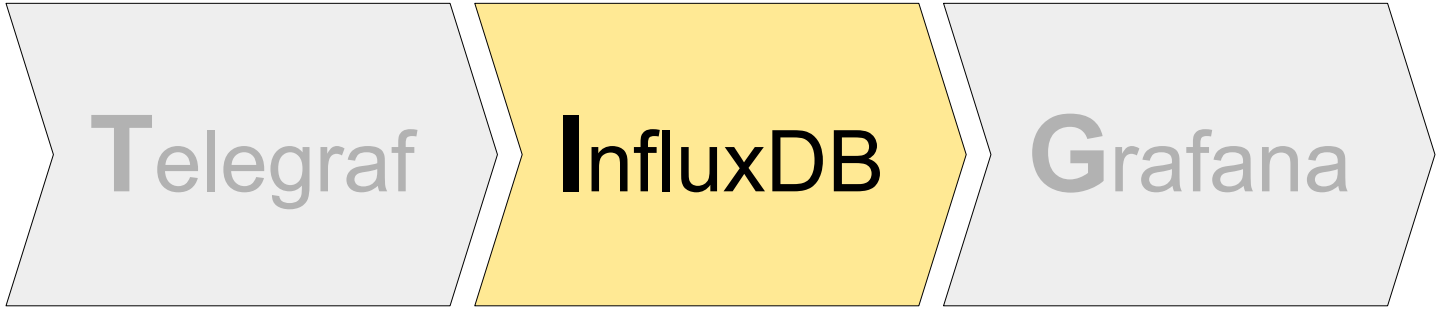
Was erzähle ich heute?

- Telegraf:
  - Features, Kennzahlen, Installation, Konfiguration
  - Plugins („out of the Box“)
  - Eigene Telegraf-Plugins

Kleiner Ausflug: TIG-Stack?



InfluxDB...



InfluxDB: eine Zeitreihendatenbank

- Zeitreihen: Serien von Messungen, Beobachtungen, Zuständen o.ä. zu aufeinanderfolgenden Zeitpunkten
- Zeitreihendatenbanken:
  - Ein Datensatz beinhaltet in der Regel:
    Zeitstempel, Messwert(e), Metainformation(en)
  - Typischerweise auch:
    - Zeitbezogene Abfragen/Berechnungen
    - Hohe Anzahl von (parallelen) Schreibvorgängen (Echtzeit…)
    - Hohe Flexibilität bei Definition/Typisierung von Dateninhalten

InfluxDB: Kennzahlen

- InfluxData Inc. → https://www.influxdata.com/
- Home:
  → https://www.influxdata.com/products/influxdb-overview/
- Lizenz: abhängig von der Variante
- Versionen: 1.x; 2.x
- Vorträge:
  → https://programm.froscon.org/2020/events/2579.html
  → https://media.ccc.de/v/froscon2020-2579-influxdb_eine_einfuhrung/

InfluxDB: Begrifflichkeiten

- *InfluxDB arbeitet mit einem schemalosen Datenmodell*
- *Databases* → sind Container für eine/mehrere Zeitreihen
  - *Measurements* → Messreihen, die eigentlichen Zeitreihen
    - ein *Datapoint* einer Messreihereihe wird beschrieben durch:
      - *Tag Values* → beschreiben/kennzeichnen die Datenpunkte
      - *Field Values* → die eigentlichen Messwerte
      - *Timestamp* → der Zeitpunkt der Messung

InfluxDB: „Influx-Format“

```
temperatur,ort=Bad,... wert=24.2,... 1583157255
```

Measurement	Tag-Value(s)	Field-Value(s)	Timestamp
(Name der Messreihe)	(Beschreibung des Messpunktes)	(die eigentlichen Messwerte)	(<u>könnte</u> man angeben...)

Telegraf

Telegraf: Features

- Serverbasierter Agent zum Sammeln, Verarbeiten, Erzeugen und Weiterleiten von Daten
- Programmiert in Go; keine Abhängigkeiten; geringer Speicherverbrauch
- Internes, konfigurierbares Scheduling
- Spezialisiert auf das IoT-Umfeld, perfekt für Zeitreihen
- 300+ Plugins „out of the box“
- Zentral konfigurierbar

Telegraf: „Kennzahlen“

- InfluxData Inc.
  → https://www.influxdata.com/
- Home:
  → https://www.influxdata.com/time-series-platform/telegraf/
- Lizenz: MIT
  → https://github.com/influxdata/telegraf/blob/master/LICENSE
- Source:
  → https://github.com/influxdata/telegraf
- aktuelle Version: 1.23.x

Telegraf: Installation

- Für alle gängigen Betriebssysteme verfügbar
- Siehe:
  → https://docs.influxdata.com/telegraf/v1.21/introduction/installation/
- Beispiel Debian/Ubuntu:
  - InfluxData-Repository einbinden
  - apt update
  - apt install telegraf

Telegraf: Konfiguration

- Konfigurationsdateien (Default):
  - /etc/telegraf/telegraf.conf
  - /etc/telegraf/telegraf.d/*.conf
- „Konfigurationshülle“ erzeugen:
  → telegraf config > ~/telegraf/telegraf.conf

Telegraf: Konfiguration

→ https://github.com/influxdata/telegraf/blob/master/docs/CONFIGURATION.md

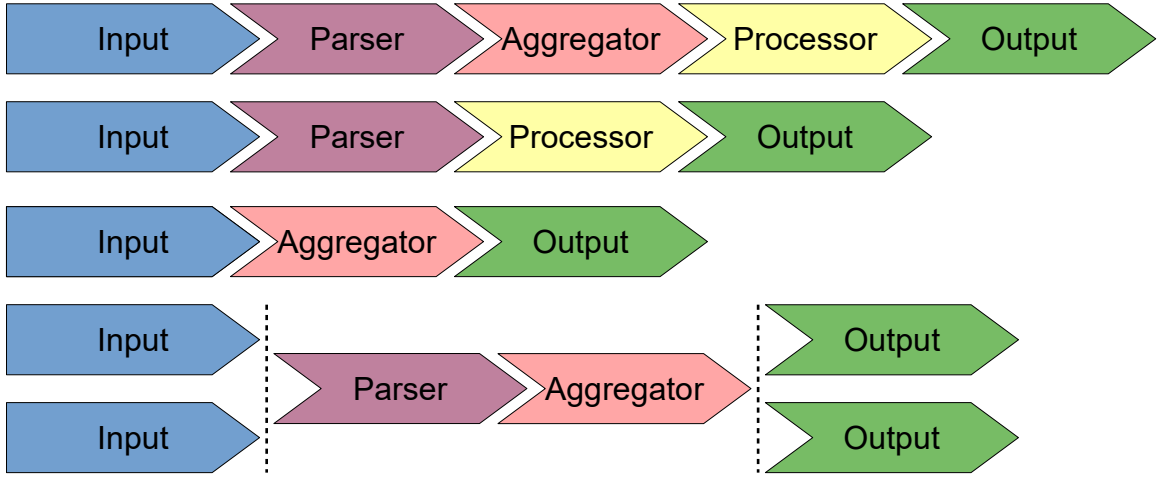
```
# Global tags can be specified here in key="value" format.
[global_tags]
...

# Configuration for telegraf agent
[agent]
...

##### OUTPUT PLUGINS  ########
[[outputs.influxdb]]
...

####  INPUT PLUGINS  #########
[[inputs.cpu]]
...

####  PROCESSOR PLUGINS  #####
[[processors.aws_ec2]]
...

####  AGGREGATOR PLUGINS  ####
[[aggregators.basicstats]]
...
```

Telegraf: Konfiguration

- Konfiguration testen:
  → telegraf --test
  → telegraf --config ~/telegraf/my_telegraf.conf --test
  → telegraf --config-directory ~/telegraf/telegraf.d --test

Telegraf: Plugins

- Plugins:
  - Inputs
  - Parsers
  - Processors
  - Aggregators
  - Outputs



- Plugins sind beliebig kombinierbar/konfigurierbar
- https://docs.influxdata.com/telegraf/v1.23/plugins/

Telegraf: Input-Plugins

- Zur Zeit über 230 Input-Plugins, z.B.:
  - MQTT-Consumer
  - Diverse Datenbanken (z.B. InfluxDB, MySQL, MS-SQL, CouchDB, Oracle, PostgreSQL, ...)
  - File
  - Host-Systeminfos (CPU, Mem, Kernel, Net, HDD, Syslog, ...)
  - OPC-UA, Siemens S7 (...hallo Arbeit...)
  - HTTP/HTTPS
  - *Exec/Execd*

Telegraf: Process/Aggregation-Plugins

- Derzeit 9 Aggregator-Plugins, z.B.:
  - Merge
  - BasicStats
- ...und 27 Processor-Plugins, z.B.:
  - Convert, String
  - Parser, Regex
  - Clone, Dedup
  - GeoIP
  - *Execd*

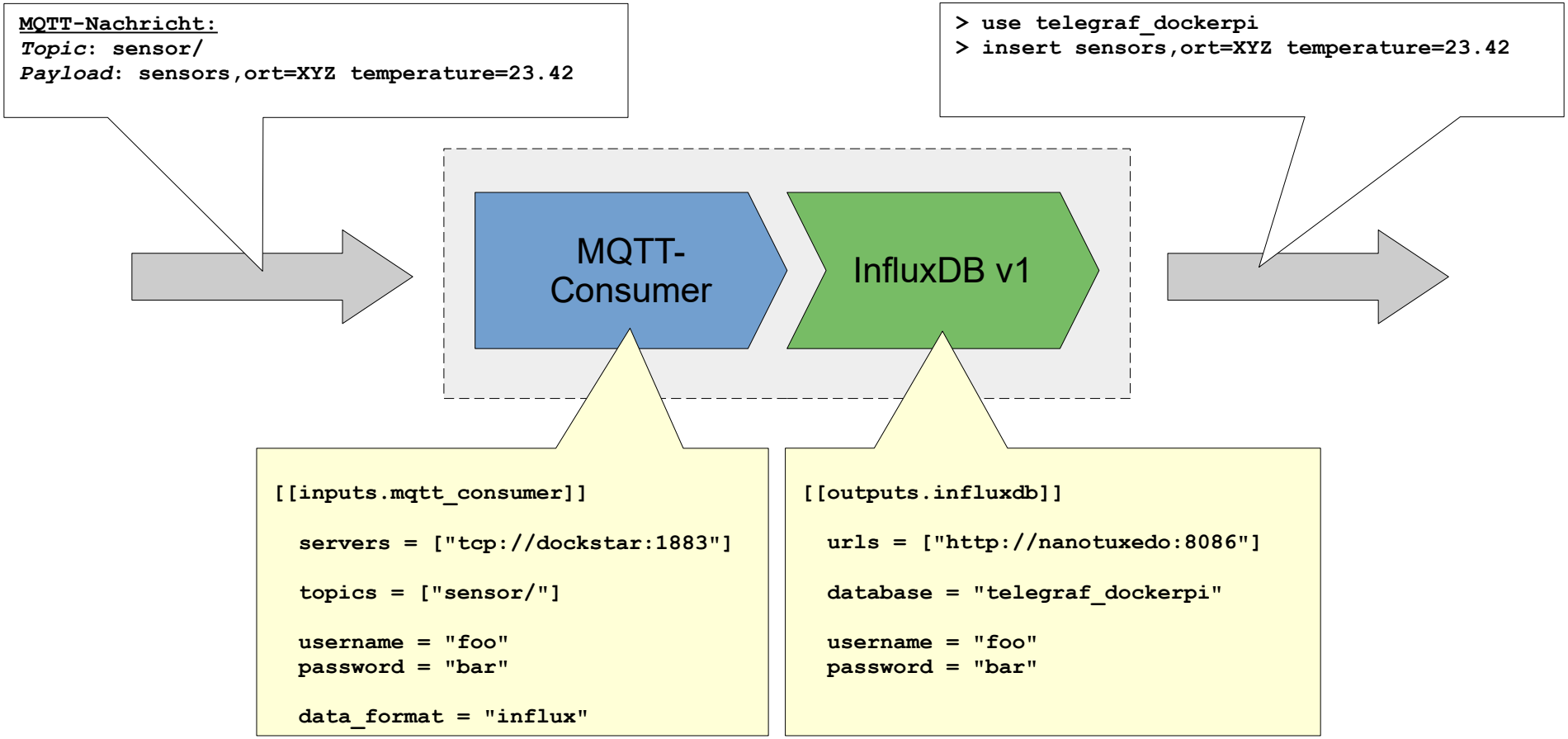
Telegraf: Output-Plugins

- Momentan fast 60 Output-Plugins, z.B.:
  - MQTT-Producer
  - Diverse Clouds
  - Diverse Datenbanken
  - Syslog
  - Socket Writer
  - HTTP
  - *Exec, Execd*

1. Usecase

- Abonnieren eines Topics von einem MQTT-Broker
  - Payload (Messpunkt) wird im „influx“-Format publiziert
- Einfügen des Messpunktes in eine InfluxDB

1. Usecase


MQTT-Nachricht:
Topic: sensor/
Payload: sensors,ort=XYZ temperature=23.42
> use telegraf_dockerpi
> insert sensors,ort=XYZ temperature=23.42
MQTT-Consumer
InfluxDB v1
[[inputs.mqtt_consumer]]
servers = ["tcp://dockstar:1883"]
topics = ["sensor/"]
username = "foo"
password = "bar"
data_format = "influx"
[[outputs.influxdb]]
urls = ["http://nanotuxedo:8086"]
database = "telegraf_dockerpi"
username = "foo"
password = "bar"

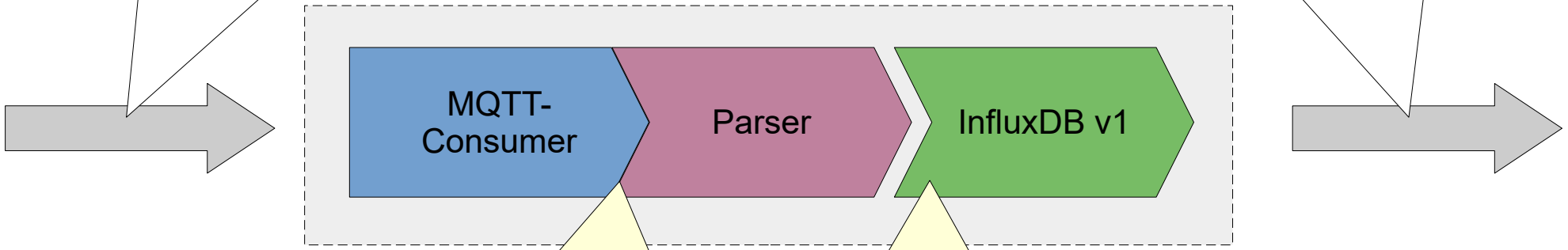
2. Usecase

- Abonnieren eines Topics von einem MQTT-Broker
  - Payload (Messpunkt) wird im „json“-Format publiziert
- Format-Konvertierung: „json“ → „influx“
- Einfügen des Messpunktes in eine InfluxDB

2. Usecase

MQTT-Nachricht:
Topic: sensor/
Payload: {"ort":"XYZ", "temperature"=23.42}

InfluxDB:
```
> use telegraf_dockerpi
> insert sensors,ort=XYZ temperature=23.42
```



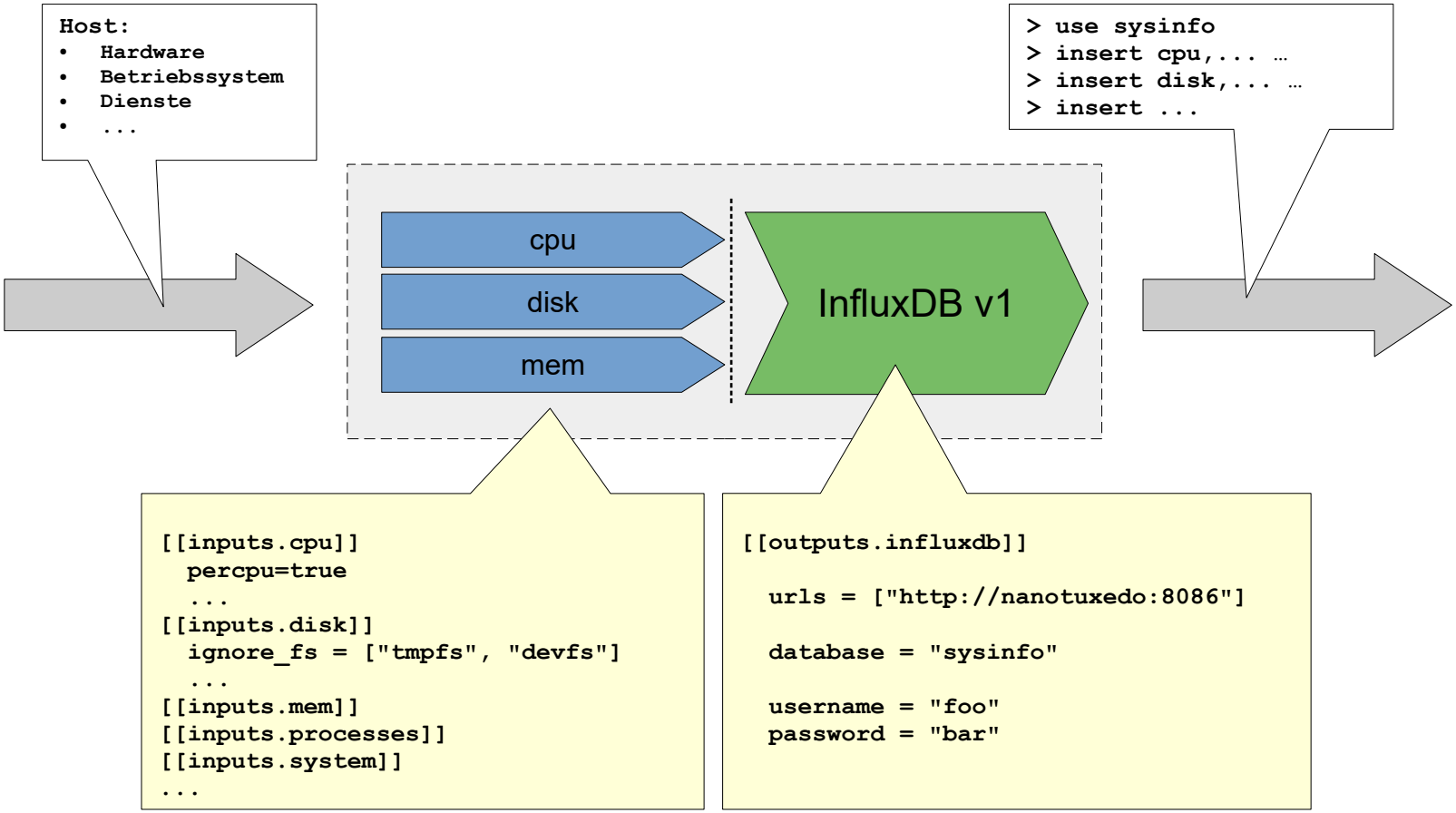

```
[[inputs.mqtt_consumer]]
  servers = ["tcp://dockstar:1883"]
  topics = ["sensor/"]
  data_format = "json_v2"
  [[inputs.mqtt_consumer.json_v2]]
    measurement_name = "sensors"
    [[inputs.mqtt_consumer.json_v2.tag]]
      path = "ort"
    [[inputs.mqtt_consumer.json_v2.field]]
      path = "temperature"
      type = "float"
```

```
[[outputs.influxdb]]

  urls = ["http://nanotuxedo:8086"]

  database = "telegraf_dockerpi"

  username = "foo"
  password = "bar"
```

3. Usecase

- Diverse Systeminformationen zu Hardware, Betriebssystem, Software des Host-Systems ermitteln
- Einfügen der Messpunkte der einzelnen Metriken in eine InfluxDB

3. Usecase


Host:
• Hardware
• Betriebssystem
• Dienste
• ...
> use sysinfo
> insert cpu,... …
> insert disk,... …
> insert ...
cpu
disk
mem
InfluxDB v1
[[inputs.cpu]]
percpu=true
...
[[inputs.disk]]
ignore_fs = ["tmpfs", "devfs"]
...
[[inputs.mem]]
[[inputs.processes]]
[[inputs.system]]
...
[[outputs.influxdb]]
urls = ["http://nanotuxedo:8086"]
database = "sysinfo"
username = "foo"
password = "bar"

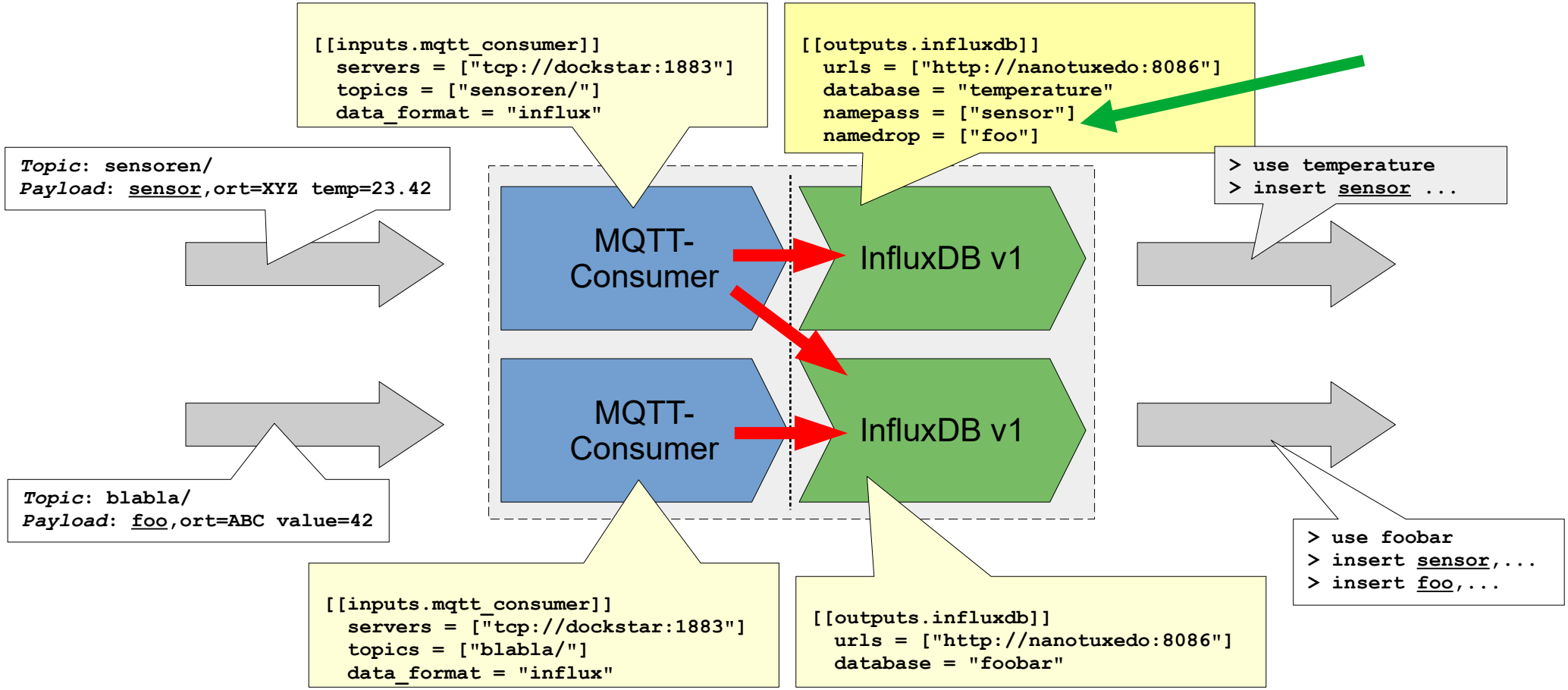
Telegraf: „Verkehrspolizist“

- Input-/Process-/Aggregate-/Output-Plugins können jeweils mehrmals definiert/kombiniert werden…
- ...wir brauchen ein „Leitsystem“

  ⟶ https://github.com/influxdata/telegraf/blob/master/docs/CONFIGURATION.md#metric-filtering

  - **`namepass, namedrop`**
  - **`tagpass, tagdrop`**
  - **`name_overide, name_prefix, name_sufix`**

Telegraf: „Verkehrspolizist“ (Beispiel)



Telegraf: eigene Plugins schreiben

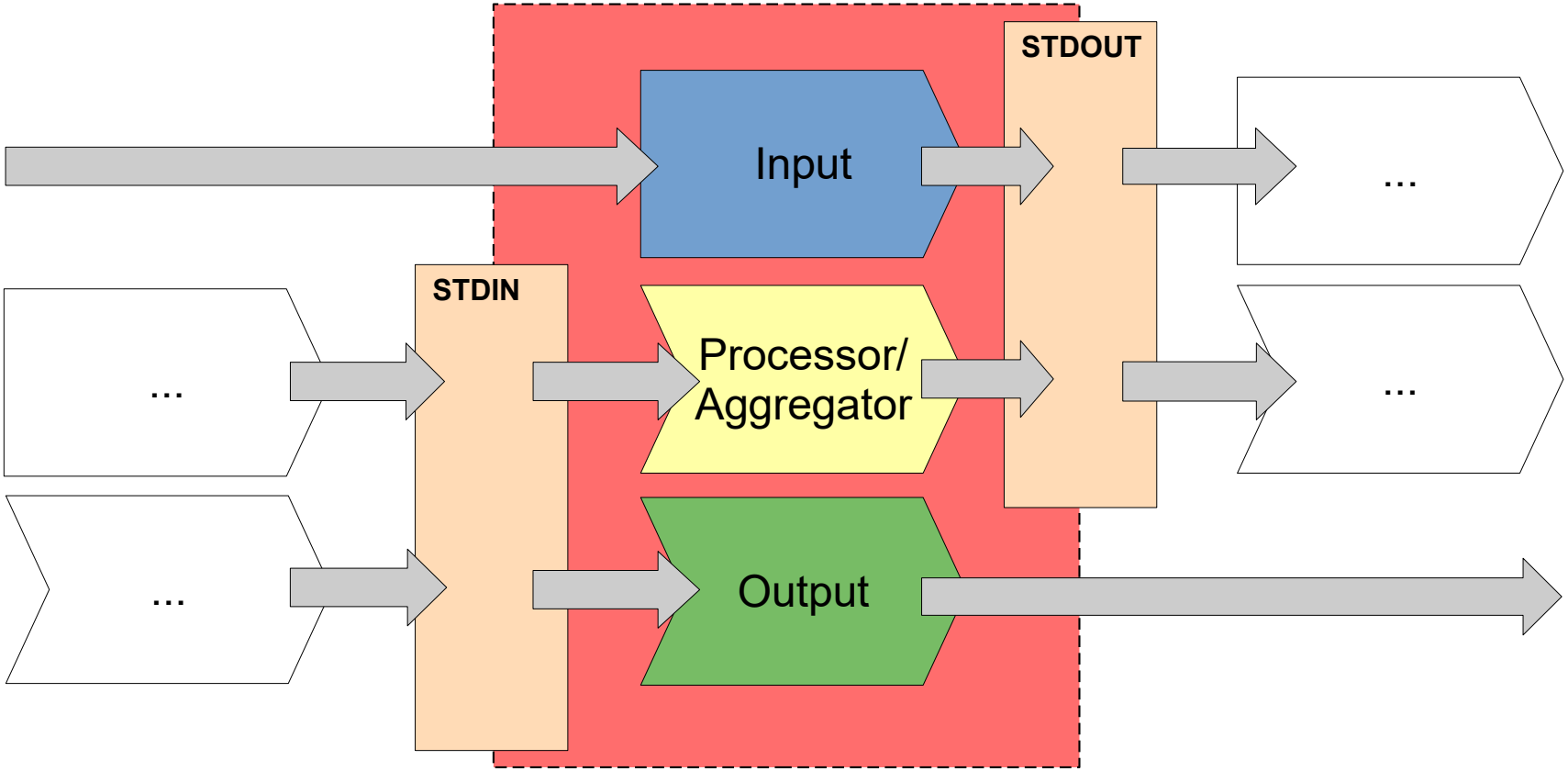
Varianten:

- Internes Telegraf-Plugin (in Go)
  → https://www.influxdata.com/blog/how-to-write-telegraf-plugin-beginners/
  → https://www.slideshare.net/influxdata/write-your-own-telegraf-plugin
  → ...
- **Externes Telegraf-Plugin**

Telegraf: eigene, externe Plugins

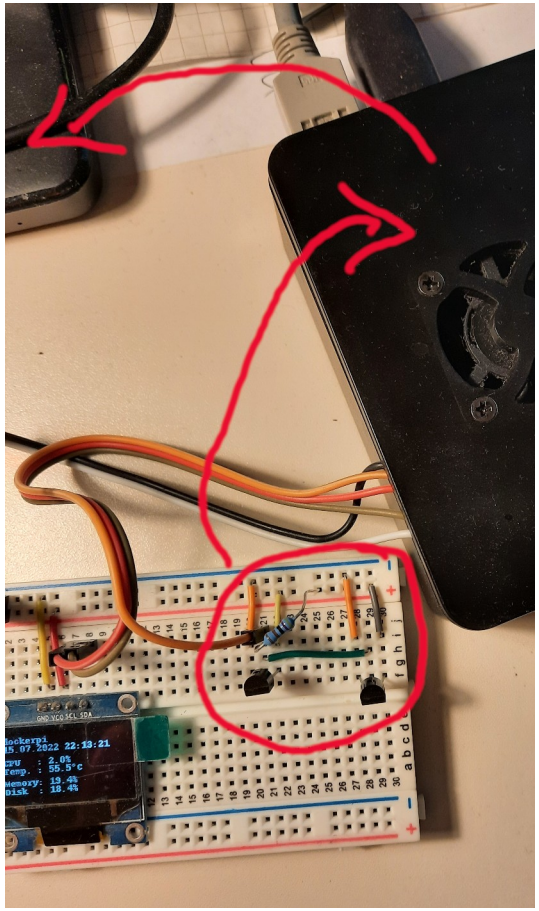
- Lauffähiges Programm in der Sprache deiner Wahl:
  - einzige Bedingung: Lesen/Schreiben von/nach STDIN/STDOUT
- Konfiguration (innerhalb Telegraf):
  - [[inputs.exec]], [[inputs.execd]]
  - [[processors.execd]]
  - [[outputs.exec]], [[outputs.execd]]

Externe Telegraf-Plugins

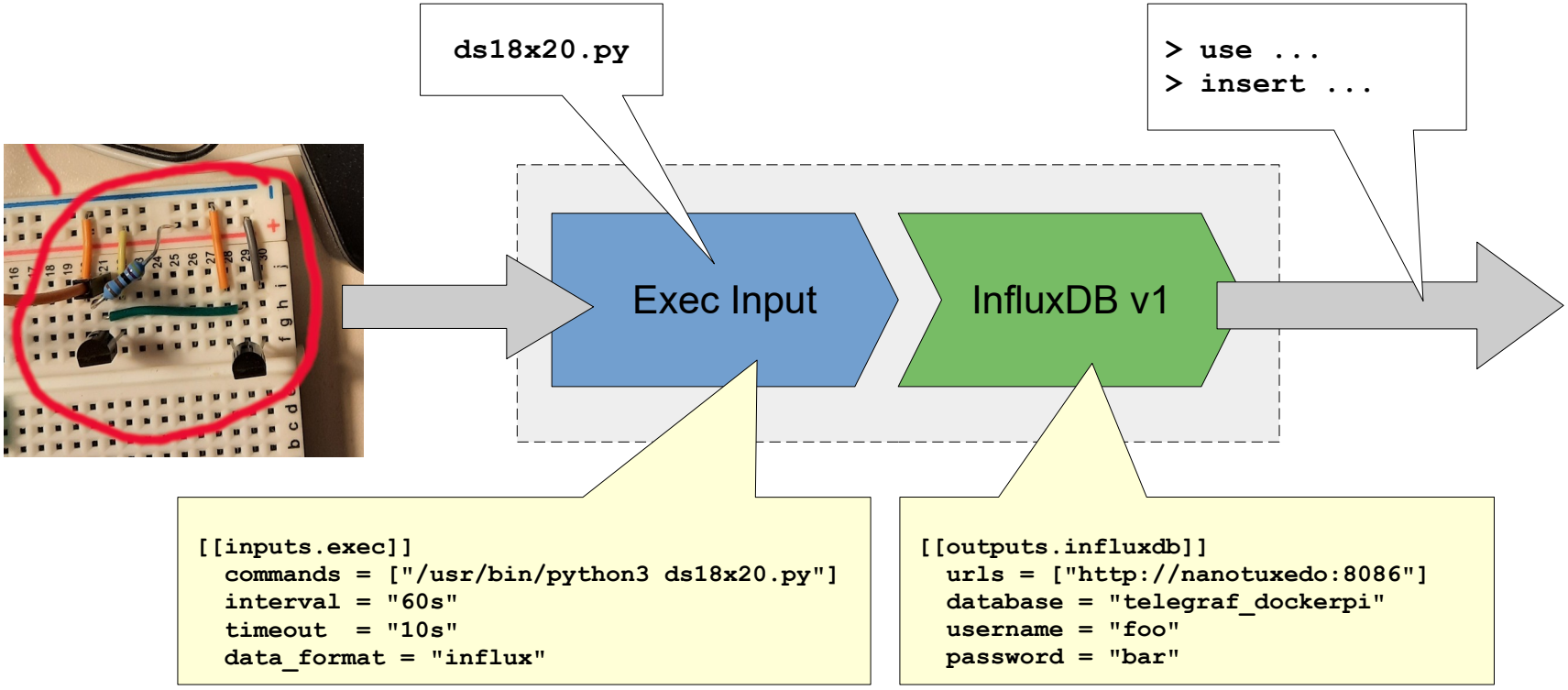


Usecase „DS18x20“

- DS18x20-Temperatursensoren an einem Raspberry Pi via OneWire angeschlossen
- Sensoren zyklisch mittels eines externen Programms (ds18x20.py) auslesen
- Messergebnisse jeweils in eine InfluxDB schreiben



Eigene Plugins: Usecase „DS18x20“



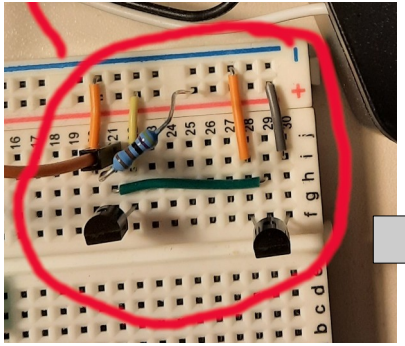
Usecase „DS18x20“ (ds18x20.py)

```
import glob
def read_raw(d):
  f = open(d, 'r')
  line = f.readlines()
  f.close()
  return line

def read_temperature(d):
  lines = read_raw(F"{d}/w1_slave")
  while lines[0].strip()[-3:] != 'YES':
    time.sleep(0.2)
    lines = read_raw(F"{d}/w1_slave")
  equals_pos = lines[1].find('t=')
  temp_string = lines[1][equals_pos+2:]
  temp = float(temp_string) / 1000.0
  return(d.split('/')[5], temp)

# ***********************************************************************************
for dev_dir in glob.glob('/sys/bus/w1/devices/' + '28*'):
  address, temperature = read_temperature(dev_dir)
  print(F"python_ds18x20,address=\"{address}\" temperature={temperature}", flush=True)
```

Usecase „DS18x20“ ...plus OLED???

```
[[outputs.influxdb]]
  urls = ["http://nanotuxedo:8086"]
  database = "telegraf_dockerpi"
  username = "foo"
  password = "bar"
```

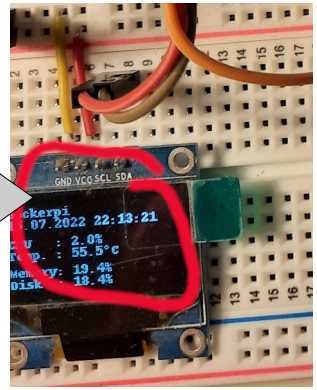
```
> use ...
> insert ...
```



Exec Input

InfluxDB v1

Exec Output



```
[[inputs.exec]]
  commands = ["/usr/bin/python3 ds18x20.py"]
  interval = "60s"
  timeout  = "10s"
  data_format = "influx"
```

```
[[outputs.execd]]
  commands = ["/usr/bin/python3 oled.py"]
  timeout  = "5s"
```

oled.py (???)

Fragen?

...ansonsten Danke & Ende!