



# **Vom Chaos zur Korrektheit**

## **Zustände mit Minot gezielt verifizieren**

Mittwoch, 19.11.2025

Christopher Sieh

**Roboter haben Zustand.**

Position, Planung, Fortschritt, Joints...



# Stateful Nodes



ROS unterstützt Zustände mit Nodes.

Entwicklung komplexerer Systeme weiter herausfordernd.





# Stateful Nodes



ROS unterstützt Zustände mit Nodes.

Entwicklung komplexerer Systeme weiter herausfordernd.

1. **Netzwerkprobleme** Alle QoS außer Reliable können Nachrichten (lokal) verlieren.  
→ Netzwerk verändert Ergebnis
2. **Laufzeit** Node zu langsam für Sensordaten, dadurch Nachrichtenverlust.  
→ Hardware verändert Ergebnis
3. **Fehlerbehebung** Iterationen schwerfällig. Tooling breit aber nicht gezielt.  
→ Verifikation

**Für Reproduzierbarkeit haben wir Bagfiles**

**Für Reproduzierbarkeit haben wir Bagfiles**

Helfen die uns hier wirklich?



# Bagfiles





# Bagfiles



**Netzwerkprobleme** Publish mit QoS.



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.



# Bagfiles



✓ **Netzwerkprobleme** Publish mit QoS.

**Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.
- ✓ **Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.
  - ✓ **Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.
- Fehlerbehebung** Bagfiles kürzen, Filter über CLI, manuell auslesen...



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.
- ✓ **Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.
- ✓ **Fehlerbehebung** Bagfiles kürzen, Filter über CLI, manuell auslesen...



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.
- ✓ **Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.
- ✓ **Fehlerbehebung** Bagfiles kürzen, Filter über CLI, manuell auslesen...

Ja, es gibt Wege. Alle bringen neue Fehlerquellen.



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.
- ✓ **Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.
- ✓ **Fehlerbehebung** Bagfiles kürzen, Filter über CLI, manuell auslesen...

Ja, es gibt Wege. Alle bringen neue Fehlerquellen. Daher haben wir ✓.



# Bagfiles



- ✓ **Netzwerkprobleme** Publish mit QoS.
- ✓ **Laufzeit** Zeit verlangsamen oder Start/Stopp. Weiterhin Publish & Forget.
- ✓ **Fehlerbehebung** Bagfiles kürzen, Filter über CLI, manuell auslesen...

Ja, es gibt Wege. Alle bringen neue Fehlerquellen. Daher haben wir ✓.

Wir wollen aber ✓.



# Grüne Wiese



- ✓ **Netzwerkprobleme** QoS nur mit Opt-In.



# Grüne Wiese



- ✓ **Netzwerkprobleme** QoS nur mit Opt-In.
- ✓ **Laufzeit** Asynchrone → Synchrone Kommunikation.



- ✓ **Netzwerkprobleme** QoS nur mit Opt-In.
- ✓ **Laufzeit** Asynchrone → Synchrone Kommunikation.
- ✓ **Fehlerbehebung** Benutzerfreundliche, einheitliche, flexible Kontrolle über Input.



- ✓ **Netzwerkprobleme** QoS nur mit Opt-In.
- ✓ **Laufzeit** Asynchrone → Synchrone Kommunikation.
- ✓ **Fehlerbehebung** Benutzerfreundliche, einheitliche, flexible Kontrolle über Input.

→ **Minot**

Terminal Anwendung mit VSCode Integration und Sammlung von C bzw. Rust Bibliotheken.



*Minots Ledge Light* bei Boston 1978  
fotografiert von Kevin Cole.

## **'Fehlerbehebung'**

Bagfiles als Datenbank mit Cursor.



# Bagfiles



Nutzerschnittstelle ist **Code**.



# Bagfiles



Nutzerschnittstelle ist **Code**.

Integrierter Compiler für zielgerichtete Sprache.



# Bagfiles



Nutzerschnittstelle ist **Code**.

Integrierter Compiler für zielgerichtete Sprache.

```
1  _bag.{  
2      cloud.{  
3          _topic = "/ouster/points"  
4          _type = Cloud  
5      }  
6      imu.{  
7          _topic = "/ouster/imu"  
8          _type = Imu  
9      }  
10 }  
11 _start_locked = false
```

Setze einige Filter.

```
13  reset! ./bagfile  
14  pf! cloud 1
```

Lade die Bagfile  
und spiele die **nächste** LiDAR Nachricht.

## 'Laufzeit'

Asynchrones Pub/Sub synchronisieren.



# Minot-Loop



Node gibt vor, **wann** Bagfile publiziert.

```
pf! [imu, cloud], cloud 1 tick_end  
pf! cloud 1 tick_end  
pf! imu 1 tick_end
```

Registriere Abfragen bei **Auslösen** von „tick\_end“.

```
rat_init("my_lio", -1);  
  
// your code here...  
  
unsigned char* tick_end = NULL;  
tick_end = malloc(sizeof(*tick_end));  
rat_bacon_u8("tick_end", tick_end, 1, 1);  
free(tick_end);  
  
// your code here...
```

Löse die Abfragen und das Publishen aus (C).



# Interface



## Visual Studio Code oder Terminal

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with files like 'debug.mt' and 'debug.mt'. The main editor area displays a C++ code snippet:

```
1 #--- init
2 _bag.{ 
3     _topic = "/velodyne_points"
4     _short = "1"
5     _type = Cloud
6 }
7
8 imu.{ 
9     _topic = "/imu/data"
10    _short = "1"
11    _type = Imu
12 }
13
14 _start_locked = false
15
16 pf! [i, 1], 1 tick_end
17 pf! 1 tick_end
18 pf! i 1 tick_end
19 reset! ./wachsbleiche1
20 ---
```

The status bar at the bottom shows various toolbars and settings.

VSCode Erweiterung für Bagfile Query.

A screenshot of a terminal window comparing variables from two datasets, labeled 'rat3' and 'rat2↑'. The table has two columns of numerical values. The first column is labeled 'var1[1]' and the second is 'var1[2]'. The terminal also shows command-line history and a status bar at the bottom.

	rat3	rat2↑	var1[1]
↓	8 9 10 11 12 13 14 15 16	↓	8 9 10 11 12 13 14 15 16
24	8.4 3.5 3.9 9.3 3.2 5.9 6.7 6.5 1.2	24	0.8 2.0 3.1 3.0 5.0 3.3
25	8.4 2.2 8.4 6.6 3.1 8.7 3.4 4.7 6.3	25	5.6 3.3 4.0 6.8 1.4 2.6 2.4
26	1.4 9.8 0.7 9.0 7.2 7.2 8.4 5.5 9.4	26	7.4 8.1 7.6 5.2 1.3 1.9 1.5 5.3
27	7.3 3.7 9.5 3.6 4.6 7.2 9.6 2.1 3.1	27	0.1 2.9 3.6 2.4 1.8 0.7 3.8 2.4 0.8
28	8.5 5.6 2.2 7.2 4.2 3.0 9.9 0.3 0.9	28	4.4 2.8 2.7 5.3 0.2 0.4 0.2 7.1 6.2
29	0.0 6.7 0.6 0.2 5.2 0.1 0.8 3.1 0.6	29	2.0 3.9 2.6 4.0 7.4 2.8 4.6 2.4
30	9.2 5.7 6.5 1.2 1.6 4.4 6.9 2.2 0.0	30	7.0 2.4 1.2 3.8 5.1 0.3 6.1
31	5.9 9.6 9.2 3.5 5.5 4.3 4.5 7.7 7.6	31	0.4 8.0 4.7 1.8 0.4 3.5 6.8 6.6
32	5.7 8.7 4.9 5.9 3.0 3.4 7.3 7.4 8.5	32	0.4 1.0 1.2 0.1 2.4 0.9 1.5 6.9
33	7.7 4.5 9.7 8.3 3.1 7.2 7.7 9.6 8.6	33	1.7 4.1 1.7 0.7 5.8 3.4 3.3
34	6.7 7.3 7.0 6.1 9.7 9.4 7.3 1.7 2.9	34	0.9 6.3 0.3 5.1 1.6 2.4 0.9
35	2.2 4.5 0.1 1.4 4.0 8.2 2.0 9.1 8.7	35	1.3 3.6 4.1 1.9 1.6 1.2 2.5 0.3
36	8.4 9.8 1.0 5.1 9.7 9.9 1.8 7.3 3.9	36	1.2 1.3 7.7 1.2 2.8 1.1 2.0
37	0.3 3.8 7.4 9.2 1.7 0.0 4.2 6.8 1.2	37	8.1 0.3 0.7 7.1 6.1 0.8 1.5 2.4 5.6
38	6.0 8.6 9.8 1.7 3.9 5.1 5.3 2.0 8.4	38	0.4 4.2 1.9 4.0 0.4 2.2 0.7 5.2 4.0
39	0.7 7.6 8.5 0.7 7.5 2.4 6.2 8.4 2.2	39	4.7 4.3 2.1 2.9 1.5 5.0 1.9 4.8
40	9.3 0.5 7.4 5.7 4.1 1.6 6.9 1.1 3.5	40	6.4 1.9 1.6 3.5 0.4 4.0 5.3
41	0.7 2.6 6.0 1.7 8.9 7.9 7.0 8.3 9.7	41	0.1 2.6 3.1 2.8 4.9 4.8 2.1 0.6 5.6
42	5.3 6.4 3.7 1.6 3.3 6.0 5.9 0.2 2.9	42	4.4 3.3 0.8 3.1 7.2 2.2
43	2.1 8.3 6.6 1.6 9.5 4.1 0.2 3.1 9.8	43	6.6 4.9 7.4 5.1 0.1 0.6 3.3
44	0.5 1.4 1.1 5.1 5.4 1.0 2.4 7.6 9.9	44	6.3 1.2 0.9 1.5 2.7 2.3 0.3
45	0.7 5.1 6.4 9.3 2.5 2.0 5.8 5.6 3.9	45	8.2 2.1 0.3 0.8 3.5 3.8
46	8.1 0.8 2.9 7.4 6.9 5.2 3.7 7.7 3.9	46	7.0 5.3 2.0 6.1 3.8 2.9 5.1 5.4 0.9
47	2.2 7.5 7.0 1.9 5.4 4.3 3.2 3.7 9.0	47	8.8 3.1 1.9 0.7 3.0 0.4 1.4
48	2.7 6.7 5.3 6.9 3.8 4.9 8.5 2.1 2.3	48	3.2 2.1 4.0 3.6 5.3 6.3
49	6.1 8.2 4.2 3.4 3.2 3.2 6.5 4.4 9.2	49	0.4 0.7 2.8 2.0 5.5 2.0 0.5 4.0
50	3.6 0.7 2.9 4.7 6.4 8.3 7.8 7.8 4.5	50	0.9 2.3 1.2 3.4 4.0 2.1 0.4
51	9.0 0.9 9.9 4.3 4.7 2.8 6.1 0.1 8.1	51	4.8 7.8 4.9 0.1 1.5 2.6 4.1 3.0 1.4

Terminal UI mit erweiterter Funktionalität,  
hier Vergleich von Variablen.



# Beispiel LIO (VSCode)



Intuitives, **selektives Ausführen** von Zeilen.

Der **Zustand wird gehalten** (vgl. Jupyter Notebook).

```
_bag.{  
  . . .  
  cloud.{  
    . . . _topic := "/ouster/points"  
    . . . _type := Cloud  
    . . . }  
    . . .  
  imu.{  
    . . . _topic := "/ouster/imu"  
    . . . _type := Imu  
    . . . }  
  . . .  
}  
  
_start_locked = false  
  
reset! ./bagfile  
pf! cloud 1
```



# Beispiel LIO (VSCode)



Log bestätigt: Bagfile wurde gelesen.

The screenshot shows a terminal window in VS Code with the following log output:

```
BE  ... Filtern Minot
FOUND COMPATIBLE MINOT VERSION: 0.1.0RC.0
Server reported ready. Sending "Init" message...
[INFO] Waiting for 'Init' message.
[INFO] Looking for coordinator...
[INFO] Rat("#minot-tui") Registering for network...
[INFO] Listening 0.0.0.0:6594
[INFO] Rat("#minot-tui") Registered.
[INFO] Connection established. Minot initialized.
[INFO] compiled in 3.448292ms
[INFO] read bag ./crash_bag
```

At the bottom of the terminal window, there are several status indicators: Run Selection, Step, and two circular icons with numbers (0 and 0). To the right, it shows Leerzeichen: 2, UTF-8, and LF.



# Beispiel LIO (VSCode)



LIO wird über Minot Pub/Sub verbunden.

```
[INFO] Send channel closed, send_to_socket task exiting.  
[INFO] Send channel closed, send_to_socket task exiting.  
[INFO] Registering for topics ("pelorus_node", "/tf", Subscribe)  
[INFO] Registering for topics ("pelorus_node", "/tf_static", Subscribe)  
[INFO] Registering for topics ("pelorus_node", "/velodyne_points", Subscribe)  
[INFO] Registering for topics ("pelorus_node", "/imu/data", Subscribe)  
[INFO] Registering for topics ("pelorus_node", "registered", Publish)  
[INFO] Registering for topics ("pelorus_node", "odom", Publish)  
[INFO] Registering for topics ("pelorus_node", "path", Publish)  
[INFO] Registering for topics ("pelorus_node", "imu_odom", Publish)  
[INFO] Registering for topics ("pelorus_node", "imu_path", Publish)
```

Run Selection Step ⚡ 0 ⚠ 0

Leerzeichen: 2 UTF-8 LF { } Minot 🐸 ⚡ Prettier 🎵



# Beispiel LIO (VSCode)



LIO wird über Minot Pub/Sub verbunden.

```
pf! [imu, cloud], cloud 1
pf! cloud 1
pf! imu 1
```

Stoße **Loop** an.

```
[23:50:53 INFO] [INFO] read bag ./crash_bag
[23:51:00 INFO] [INFO] compiled in 158.667µs
[23:51:00 INFO] [INFO] iterating on bagfile...
[23:51:00 INFO] [INFO] got 13 msgs in 6.137625ms
[23:51:00 INFO] [INFO] iterating on bagfile...
[23:51:00 INFO] [INFO] Registering for topics ("embedded_ratpub_turbine_pub", "/ouster/imu",
Publish)
[23:51:00 INFO] [INFO] got 1 msg in 14.040375ms
[23:51:00 INFO] [INFO] iterating on bagfile...
[23:51:00 INFO] [INFO] got 1 msg in 10.495875ms
[23:51:00 INFO] [INFO] Registering for topics ("embedded_ratpub_turbine_pub", "/ouster/
points", Publish)
```

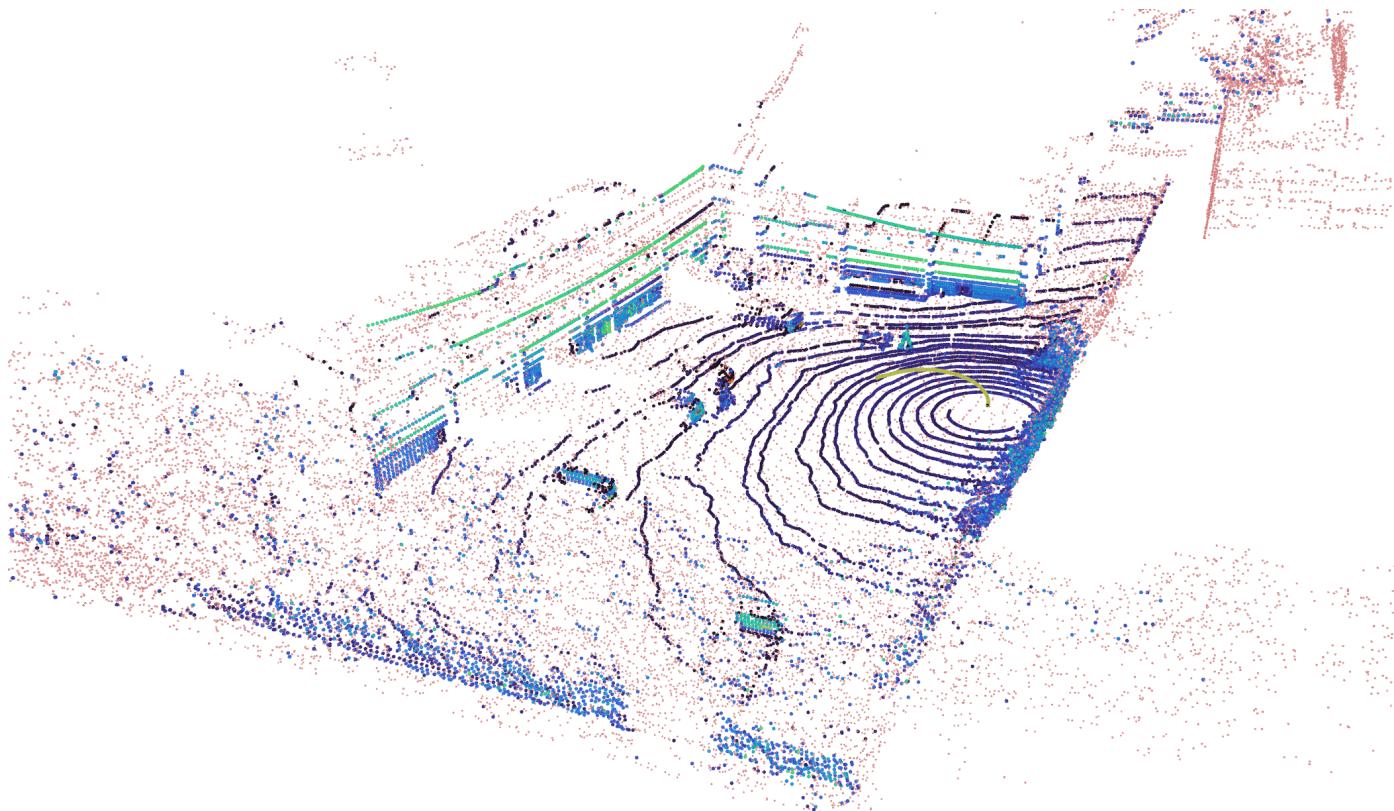
Minot publisiert zu LIO.



# Beispiel LIO (VSCode)



LIO läuft.

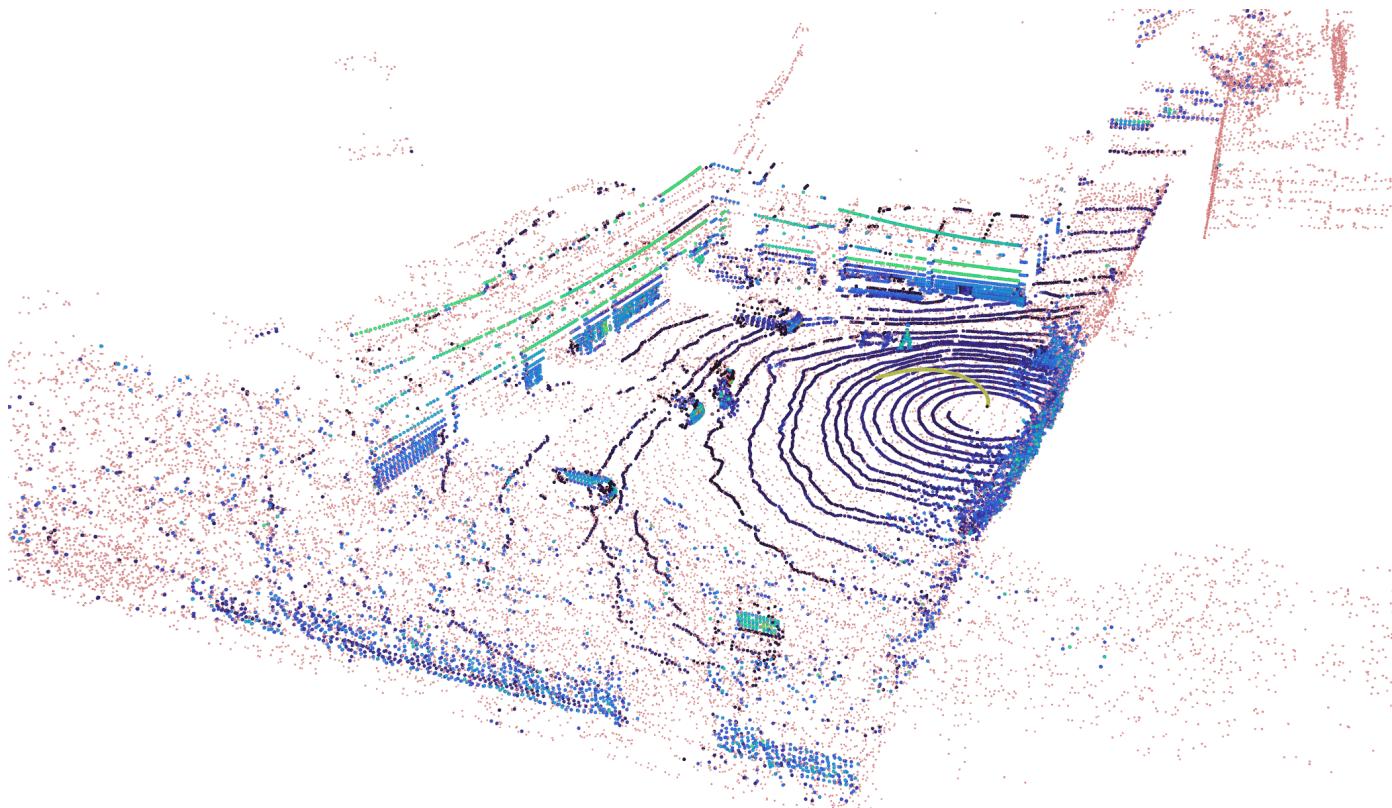




# Beispiel LIO (VSCode)



LIO läuft.

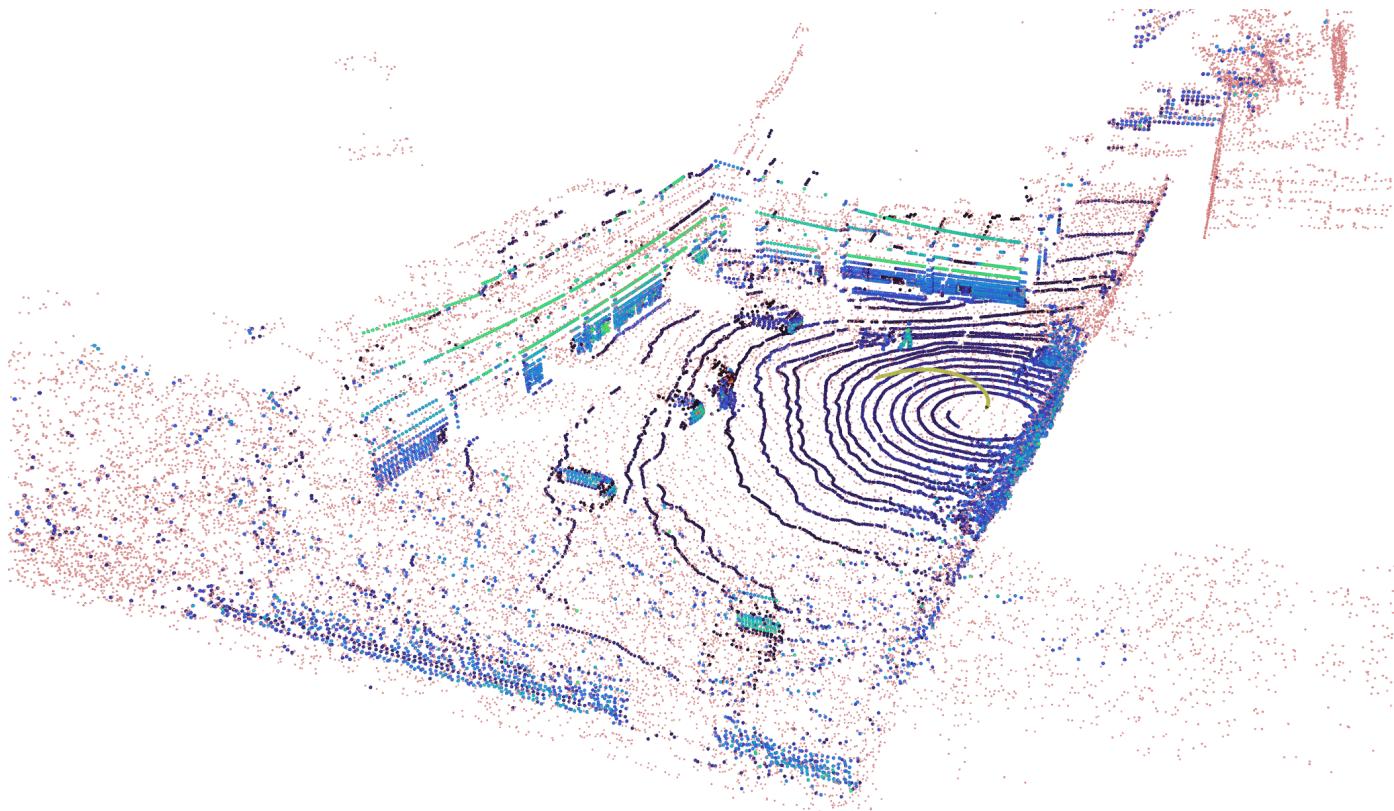




# Beispiel LIO (VSCode)



LIO läuft.



## Und wie wird verifiziert?

Der Auslöser (im Beispiel ‚tick\_end‘) ist technisch die Anfrage, ob die Variable ‚tick\_end‘ synchronisiert werden soll.



# Variablen Synchronisieren



Speicher übers Netzwerk **synchronisieren**.

```
# ...
reset! ./bagfile
# declare sharing rules
rule! (myvar
    # Send myvar from node1 to node2, overwriting node2
    node1 -> node2

    # node3 overwrite both variables of node1 and node2
    node3 <- node1, node2

    # Both nodes send their myvar to Minot where they
    # can be compared with each other
    blue == yellow

    # node1 sends its myvar to the Minot TUI for displaying it
    node1 -> LOG
)
pf! cloud 1 # ... continue normal execution
# ...
```



# Variablen Synchronisieren



Ein **Prozess im Netzwerk** kann Werte zu einem Zeitpunkt synchron **testen**.

```
rule! (var
| lio -> e2e
)

rule! (var3
| lio -> e2e
)

rule! (var4
| lio -> e2e
)
```

```
ret = rat_bacon_u8("var", var, 1, 1);
if (ret != 0) {
    printf("Error while bacon var");
    return ret;
}
assert(*var == true);
free(var);
```

Asserte das Ergebnis.

Sende alle Zwischenergebnisse zum Testprozess.



# Und Minot kann noch vieles mehr...



- Läuft überall: Windows, Linux, Mac, iOS, Android...
- Pub/Sub ohne QoS (nur Rust) – „Netzwerkprobleme“
- Gleichzeitig publishen nach ROS1, ROS2, Minot Pub/Sub (Ratpub)
- Komplett verteilte, modulare Architektur
- Open Source (MIT oder Apache-2) und dokumentiert
- Mehr **Szenarien gesucht!**

Informationen auf <https://uos.github.io/minot>

# Danke!

**Christopher Sieh** – **chsieh@uos.de**

Wissenschaftlicher Mitarbeiter

AG Technische Informatik

Prof. Dr.-Ing. Mario Porrmann

<https://inf.uos.de/ce>

Universität Osnabrück

Institut für Informatik

Wachsbleiche 27

49090 Osnabrück