Associate Professorship of Embedded Systems and Internet of Things
Department of Electrical and Computer Engineering
Technical University of Munich

Runtime Deployment, Management and Monitoring of Web of Things Systems

Miguel Romero Karam
miguel.romero@tum.de

Embedded Systems and Internet of Things

Department of Electrical and Computer Engineering

Munich | December 14, 2020

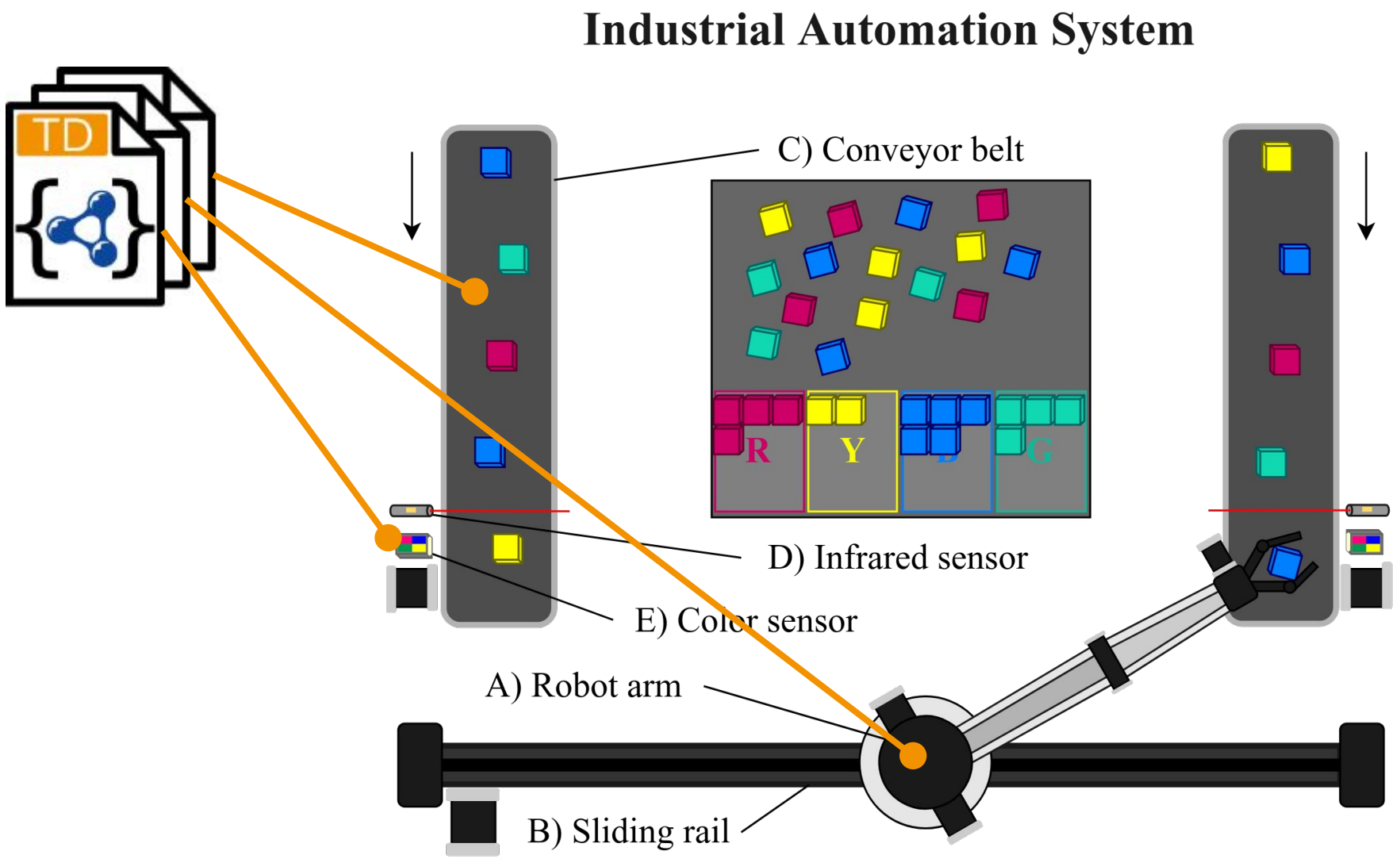
1. Introduction

2. Approach

3. Evaluation

4. Conclusion

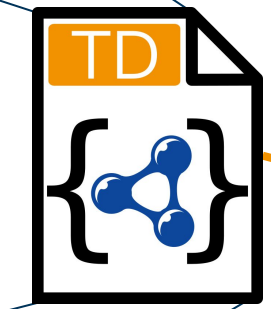
Motivation

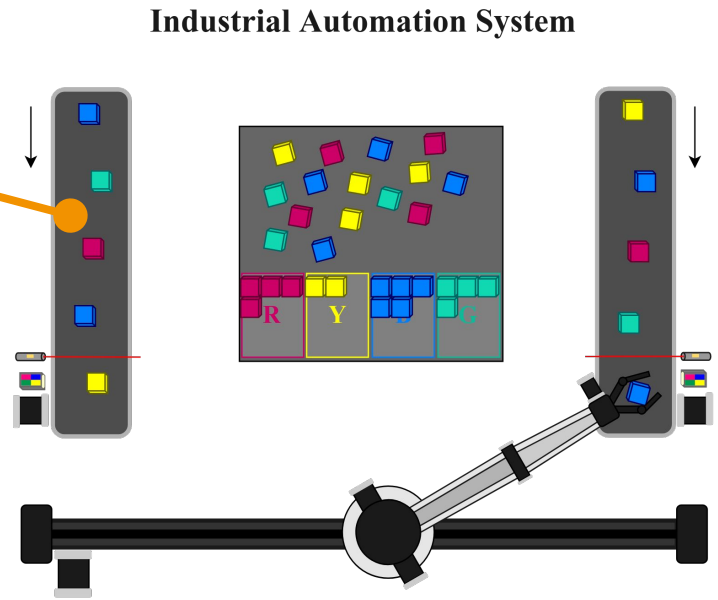

Industrial Automation System
TD
C) Conveyor belt
R
Y
G
D) Infrared sensor
E) Color sensor
A) Robot arm
B) Sliding rail

WoT Thing Description

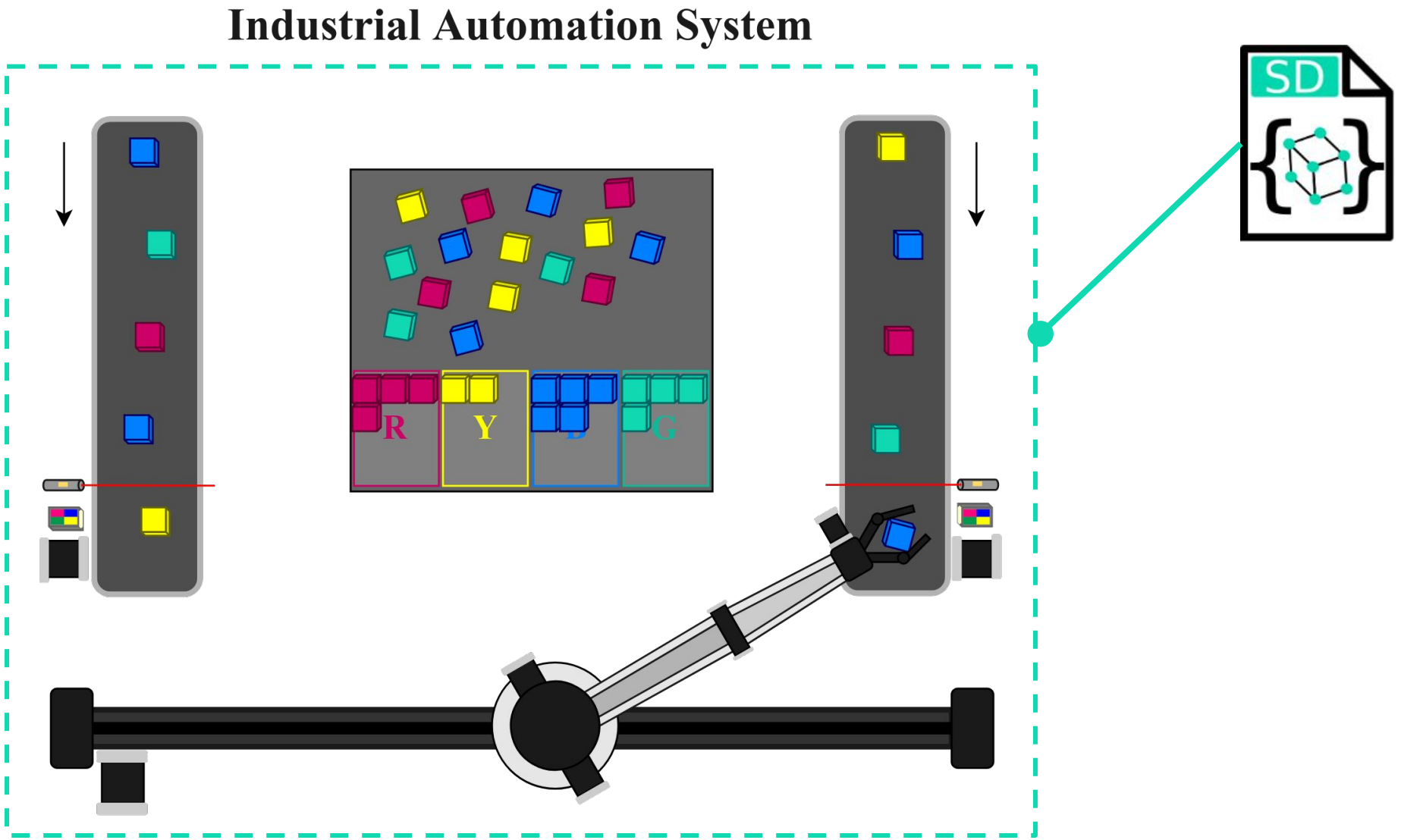
TD of Conveyor Belt

```
"name": "LeftConveyorBelt",
"@type": "conveyor-belt",
"properties": [
   "speed": {
        "type": "number",
        "minimum": -100.0,
        "maximum": 100.0
    },
   "status": {{
        "type": "string",
        "enum": [ "on", "off" ],
    }
],
"actions: [
   "start": {...},
   "stop": {...},
],
"events: [
   "emergencystop": {...},
],
"security": [...]
```





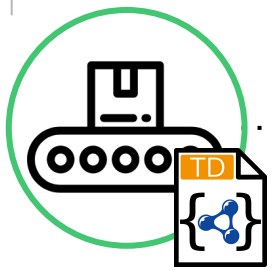
WoT System Description



Problem Statement

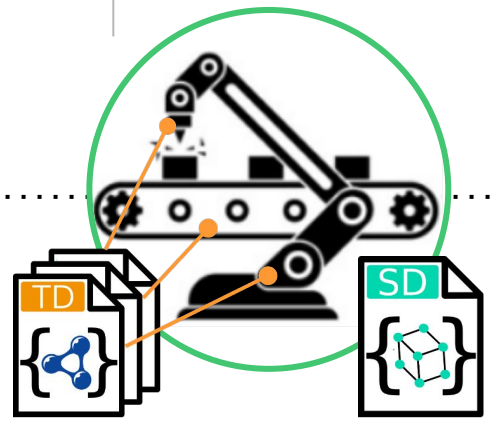
Thing Description

Allows modelling simple and complex **standalone Things**



System Description

Allows describing interaction logic between Things, in order **to create WoT Systems**

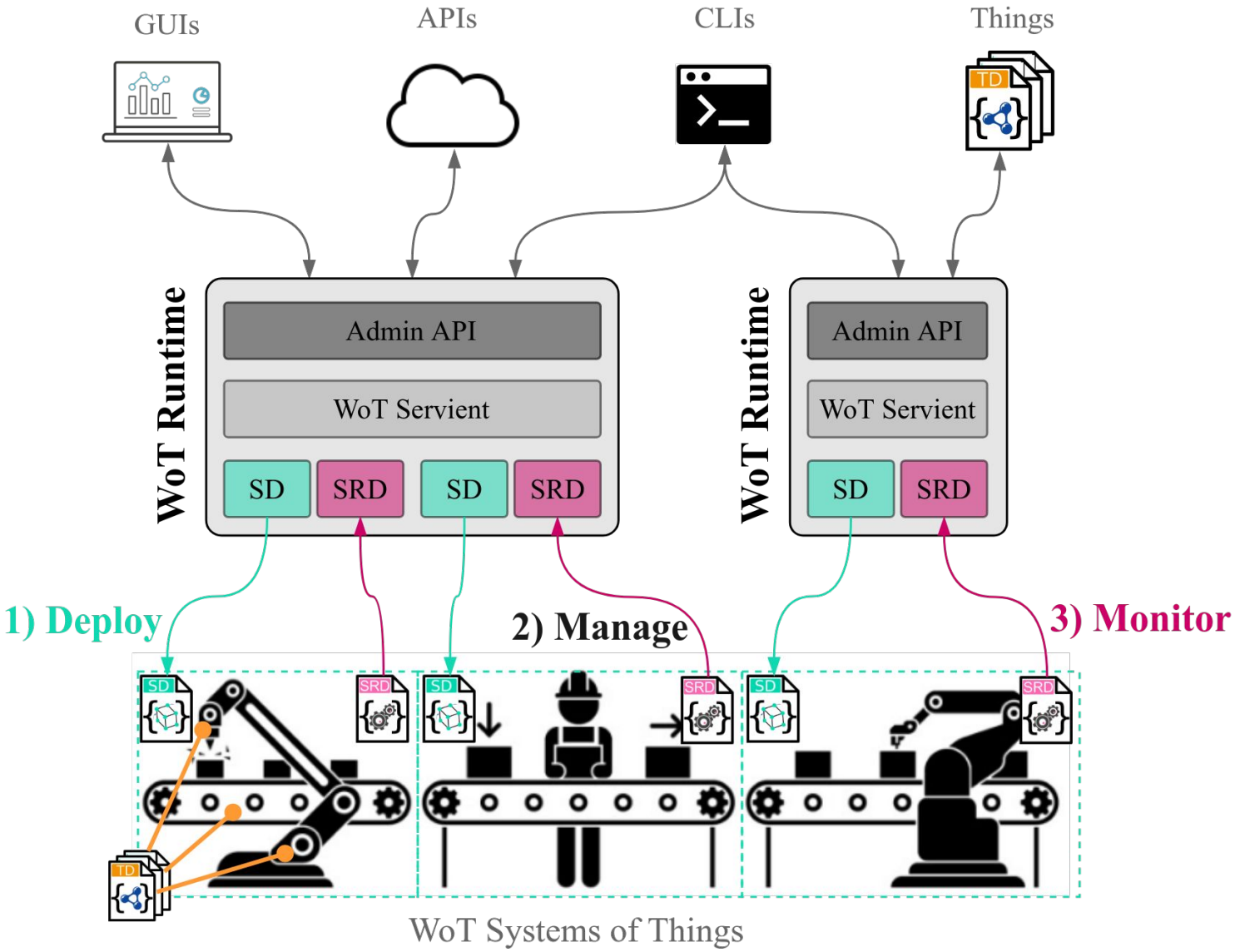


Development Tools

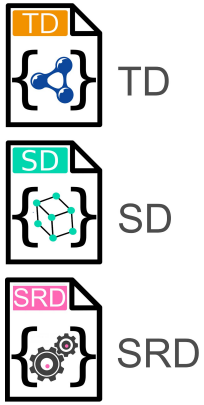
For **deployment, management, and monitoring** of WoT Systems

Approach

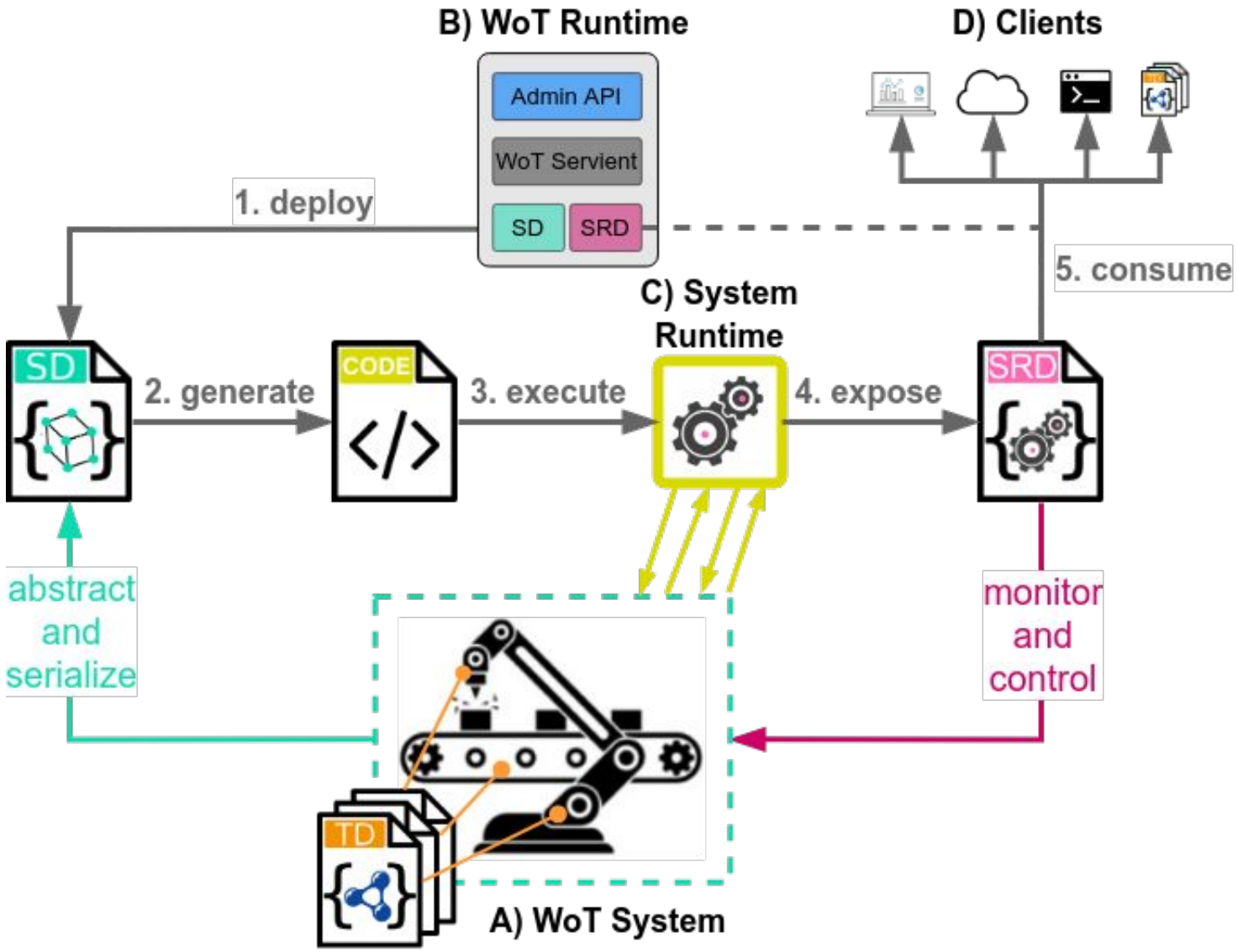
Overview



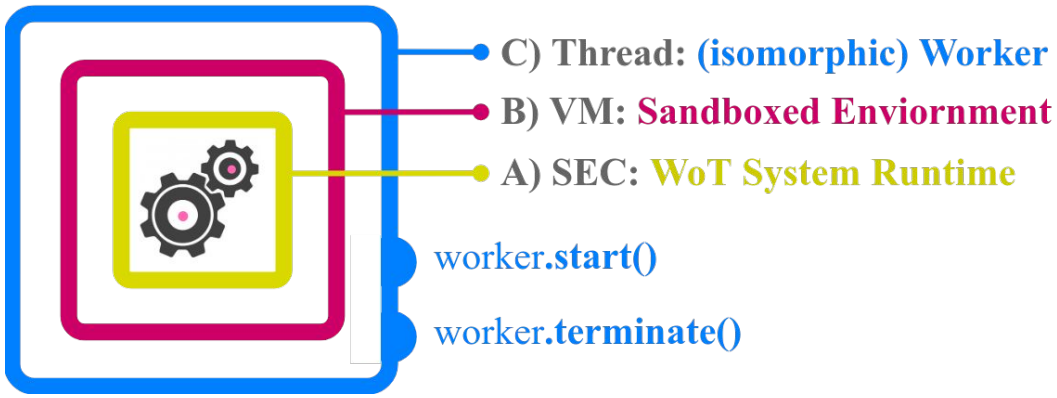
KEY

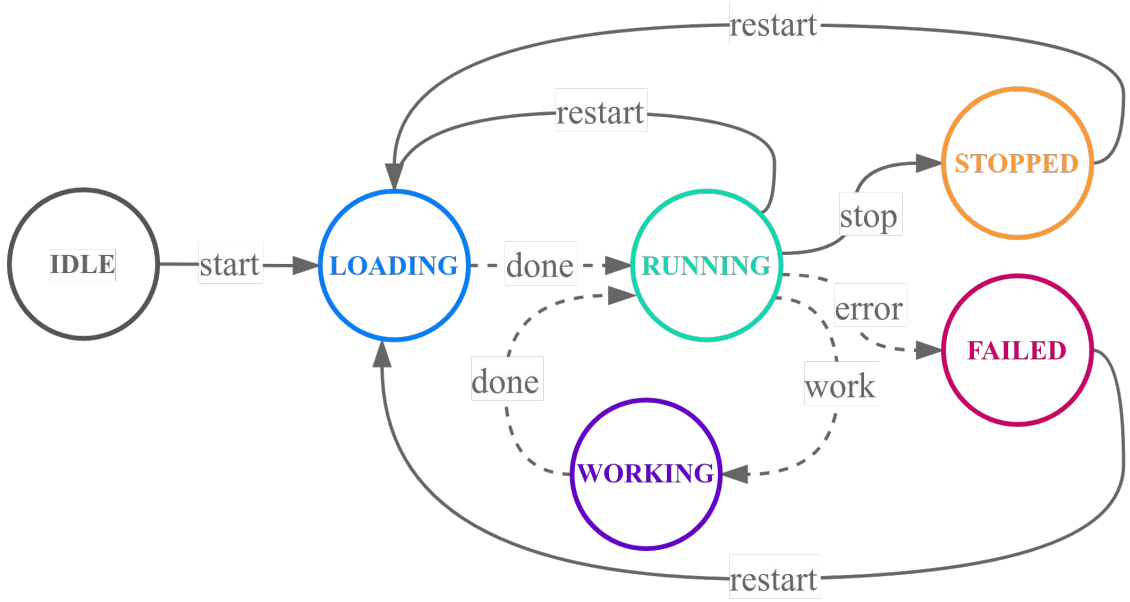


Methodology

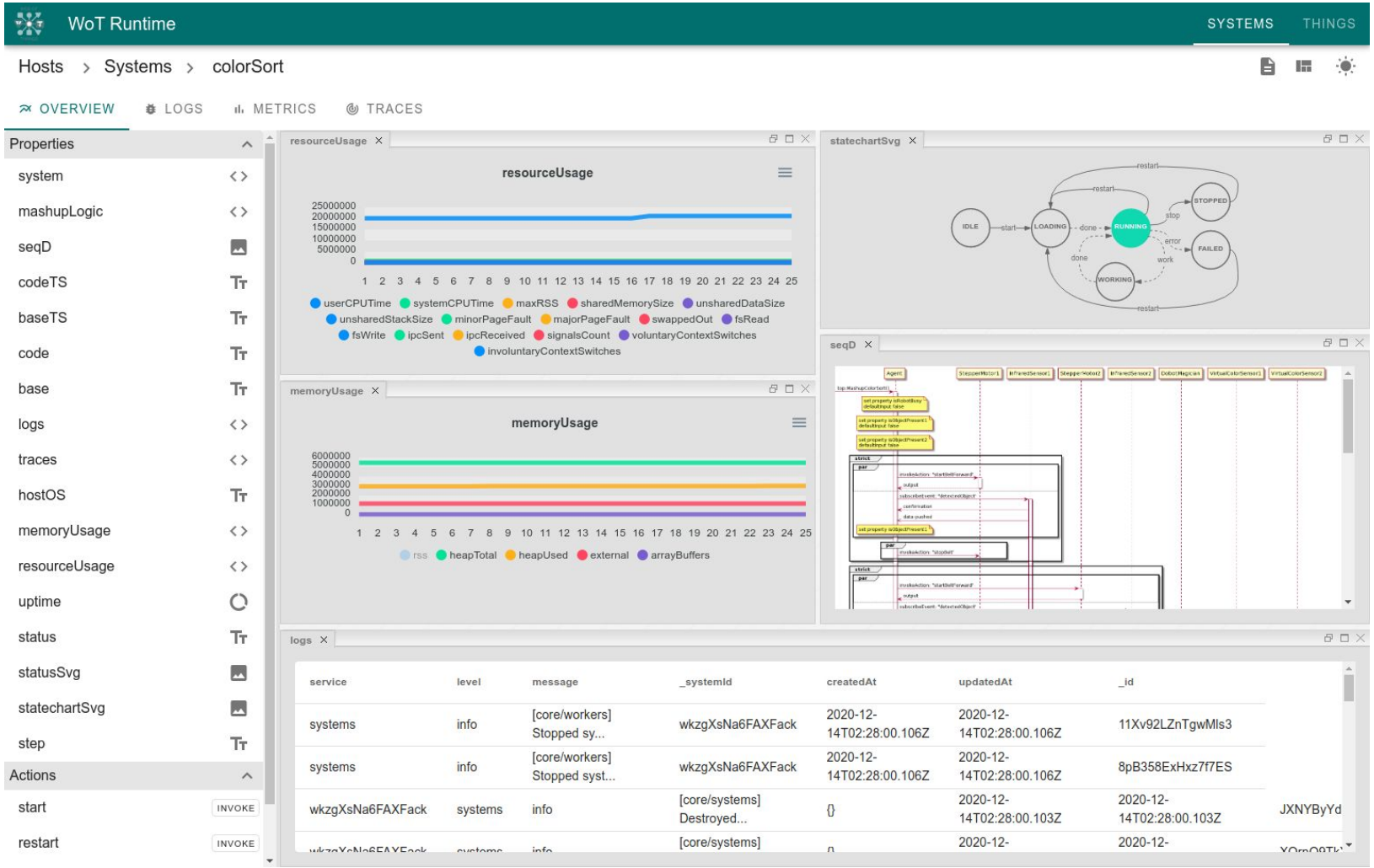


Execution Environment and Control

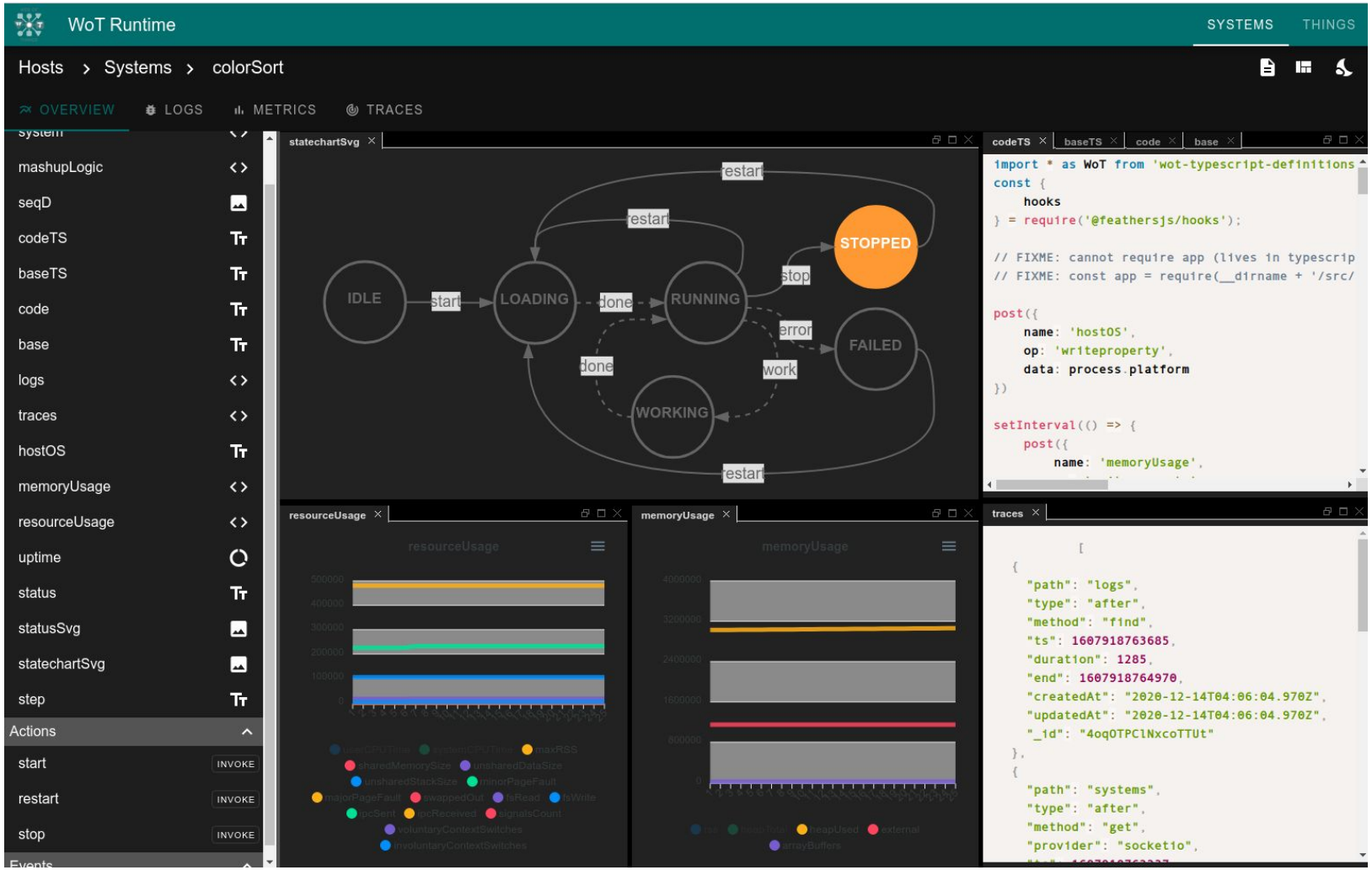




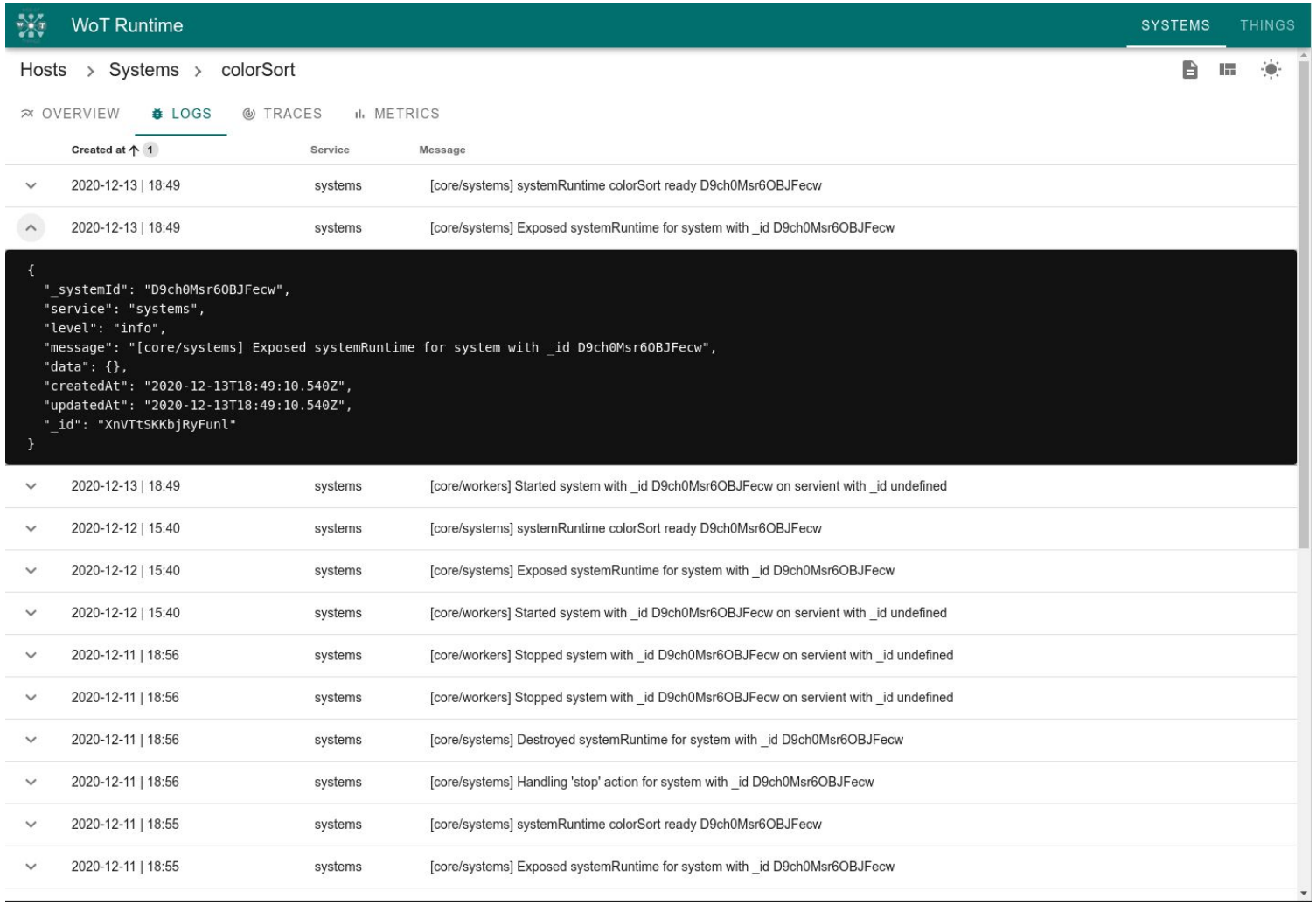
WoT Runtime UI - System Dashboard



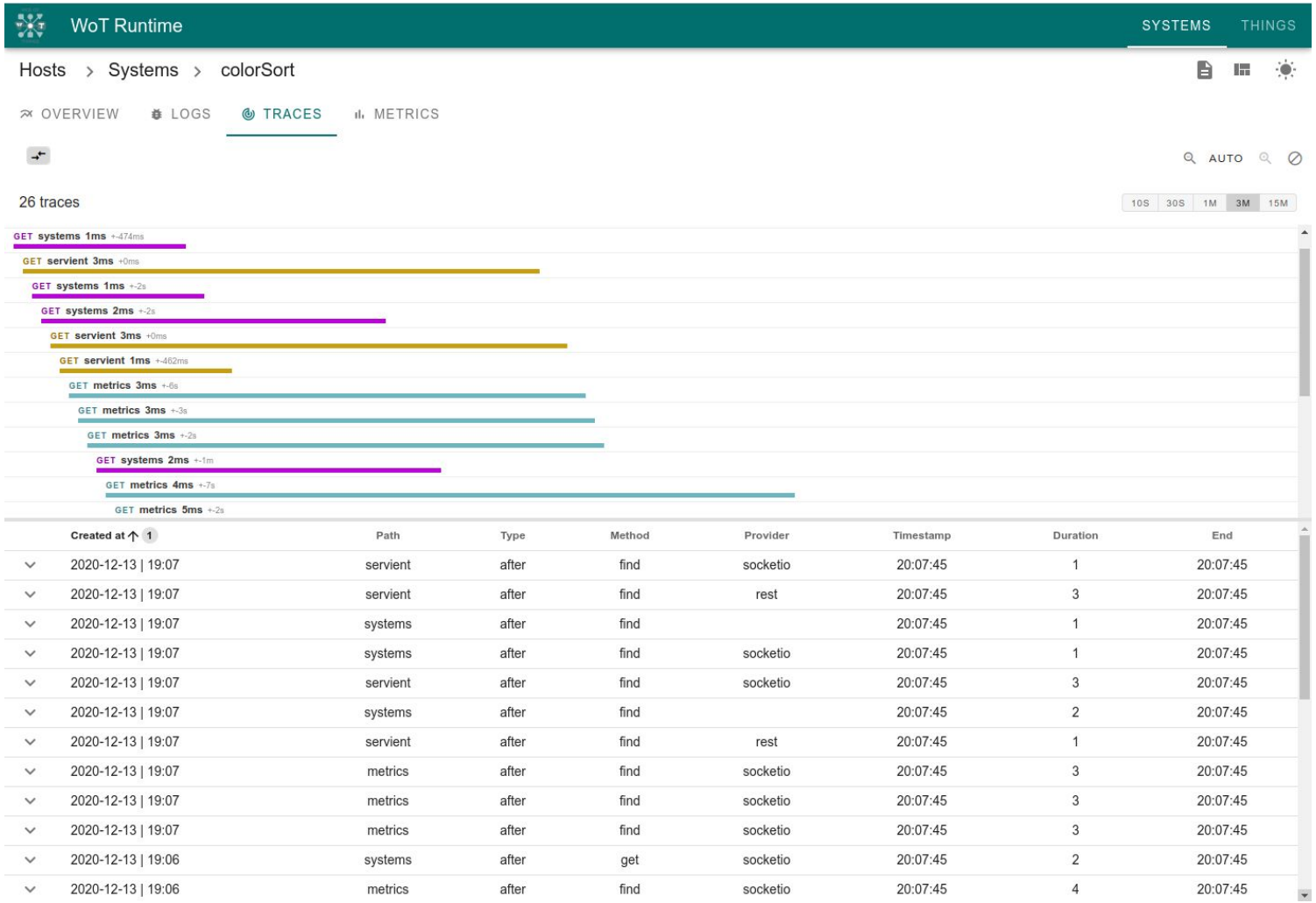
WoT Runtime UI - System Dashboard (dark)



WoT Runtime UI - Logs

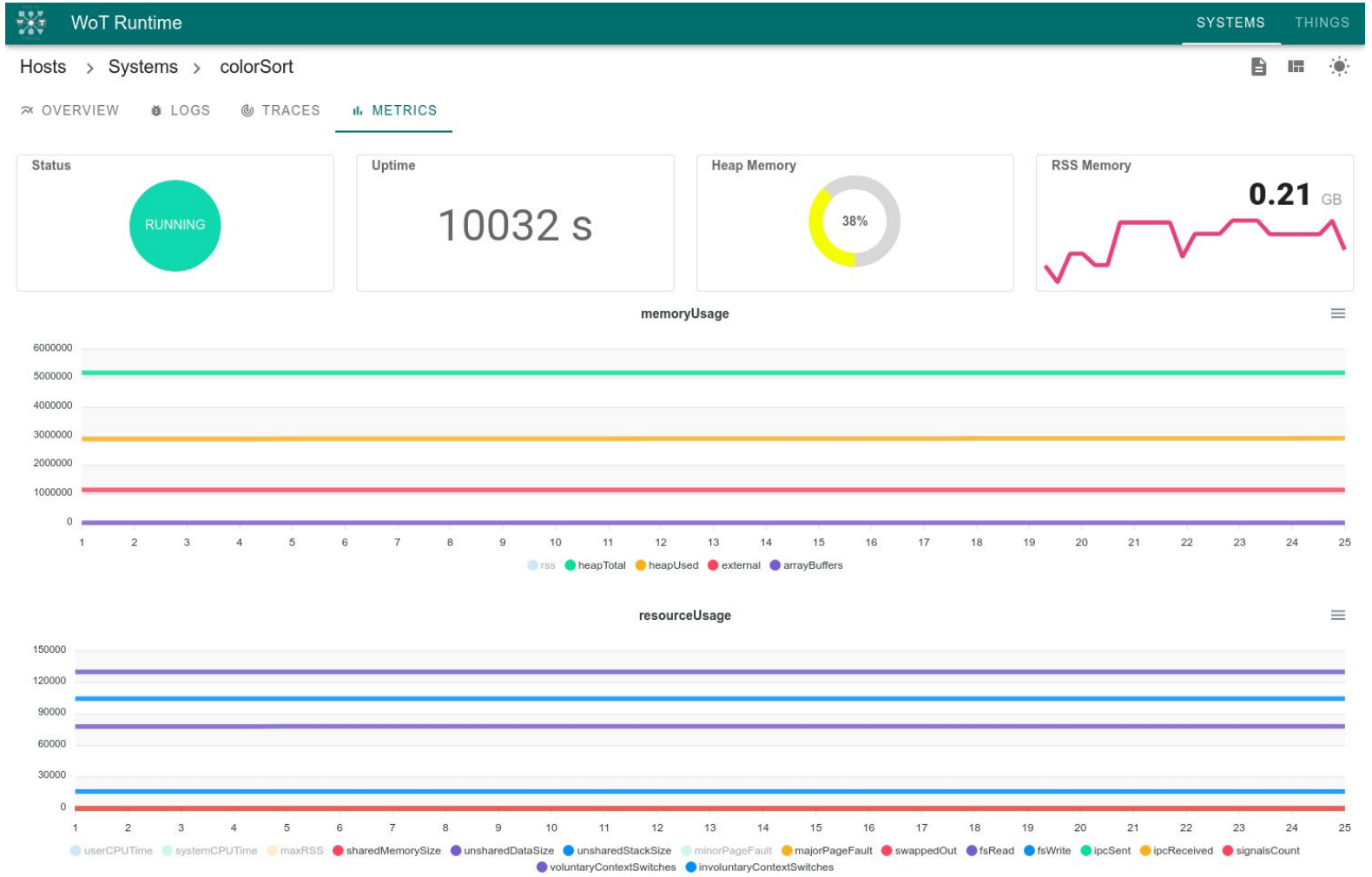


WoT Runtime UI - Traces



Created at	Path	Type	Method	Provider	Timestamp	Duration	End
2020-12-13 | 19:07	servient	after	find	socketio	20:07:45	1	20:07:45
2020-12-13 | 19:07	servient	after	find	rest	20:07:45	3	20:07:45
2020-12-13 | 19:07	systems	after	find		20:07:45	1	20:07:45
2020-12-13 | 19:07	systems	after	find	socketio	20:07:45	1	20:07:45
2020-12-13 | 19:07	servient	after	find	socketio	20:07:45	3	20:07:45
2020-12-13 | 19:07	systems	after	find		20:07:45	2	20:07:45
2020-12-13 | 19:07	servient	after	find	rest	20:07:45	1	20:07:45
2020-12-13 | 19:07	metrics	after	find	socketio	20:07:45	3	20:07:45
2020-12-13 | 19:07	metrics	after	find	socketio	20:07:45	3	20:07:45
2020-12-13 | 19:07	metrics	after	find	socketio	20:07:45	3	20:07:45
2020-12-13 | 19:06	systems	after	get	socketio	20:07:45	2	20:07:45
2020-12-13 | 19:06	metrics	after	find	socketio	20:07:45	4	20:07:45

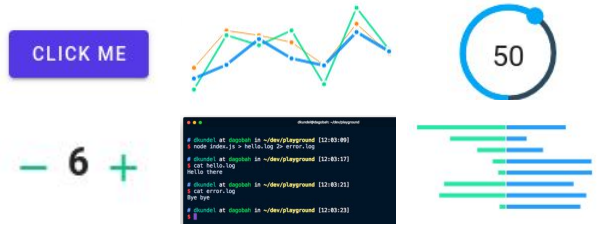
WoT Runtime UI - Metrics



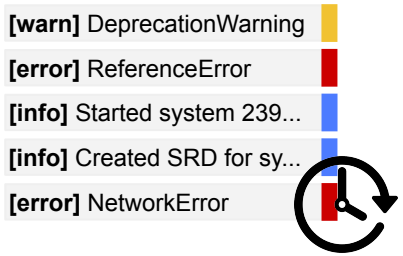
Results

Smart Farm Simulation Use Case

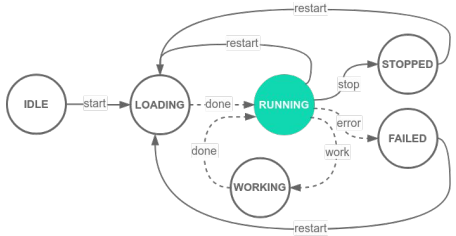
C) Mashup **monitoring and control** via auto-generated GUI

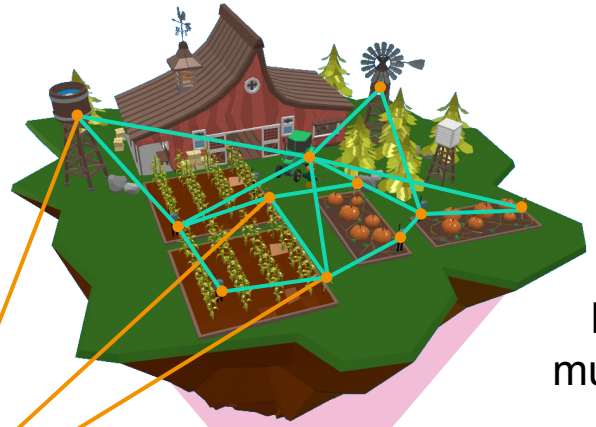


B) Runtime monitoring via **logs, metrics and traces**

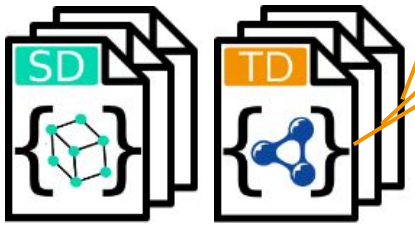


D) **Safe and secure** execution environment

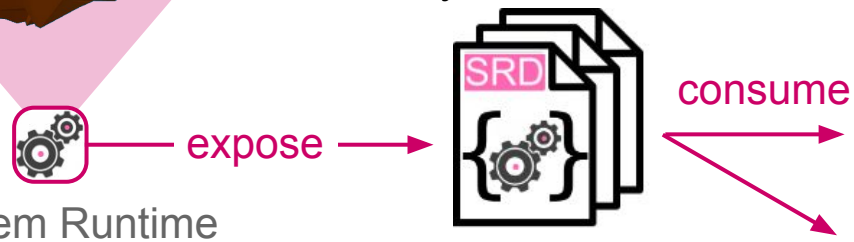




A) **Management** of multiple **SDs and TDs** via API



E) **SRD** for composing multi-layered architectures



System Runtime

Conclusion

In conclusion:

- The WoT Runtime Framework introduces an additional WoT building block for:
  - remote deployment of SDs into safe execution Environments,
  - management of WoT System Runtimes,
  - and monitoring and control through the auto-generated SRD
- The SRD allows composing multi-layered WoT architecutes
- The approach is evaluated in two use cases
  - An industrial automation scenario
  - and a smart farm simulation

  demonstrating how the proposed solution works in practice
- The WoT Runtime framework lays the basis for future in distributed WoT architectures

Thank you for your attention