OneDM@WoT Plugfest

Summary

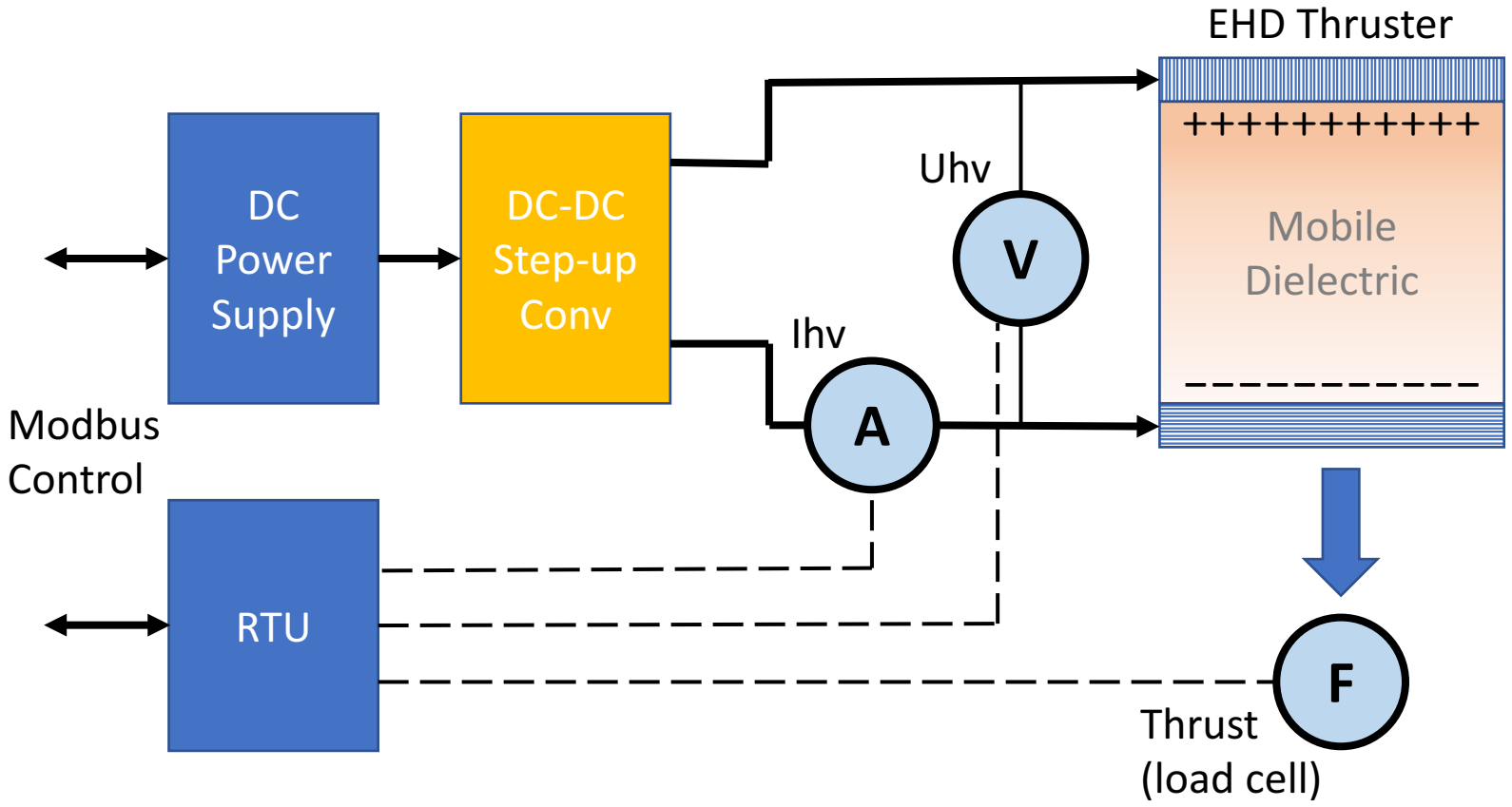
March 24, 2021

Michael J. Koster

Goals

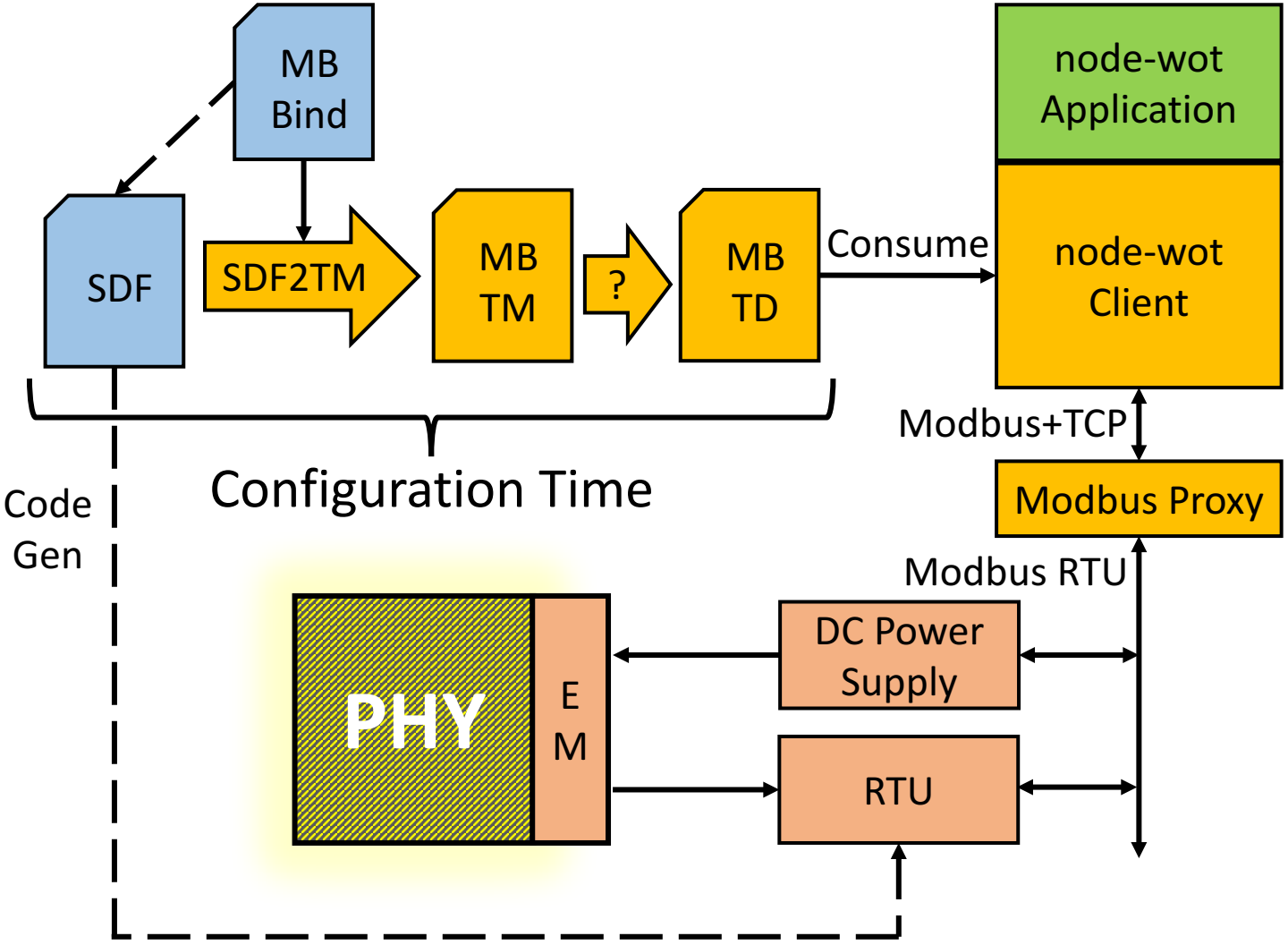
- Explore SDF => TM/TD Generation
- Define how semantic annotation from SDF works
- Test the Modbus protocol binding

EHD Propulsion Experiment "Ionic Wind Tunnel"

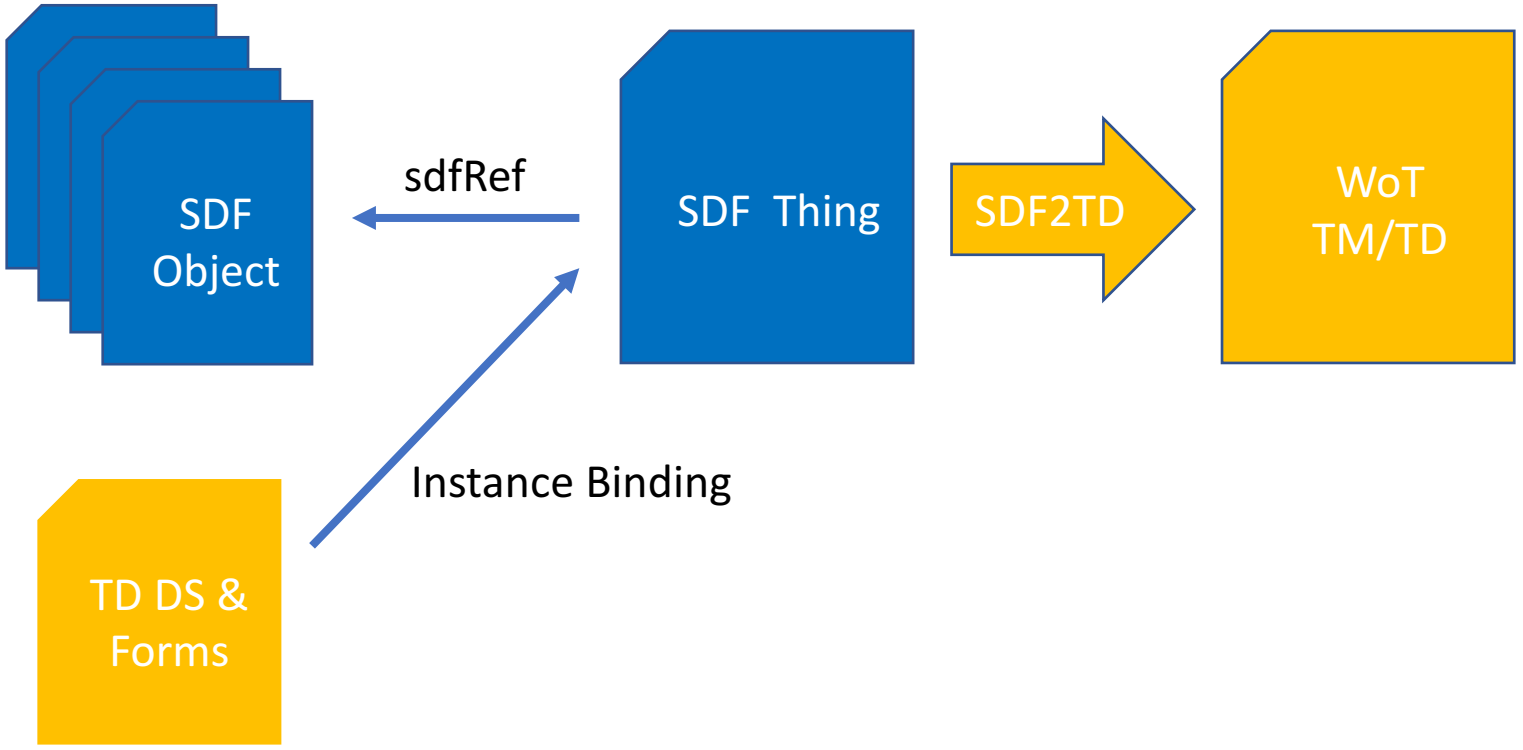


Ref: https://www.researchgate.net/publication/235189622_A_Model_of_an_Ideal_Electrohydrodynamic_Thruster

Modbus Integration Test Case



Model Construction Workflow



SDF to TM Conversion Workflow

- Inserted WoT DataSchema and Forms construct into SDF as extensions
- Created an sdfThing that customizes its sdfObject components and inserts the WoT extensions:

https://github.com/mjkoster/onefb-sdk/blob/main/control-models/modbus/sdfthing-modbus-ehd-rtu.sdf.json

- Generated a TD manually from the SDF file:

https://github.com/mjkoster/onefb-sdk/blob/main/control-models/modbus/modbus-rtu.td.jsonld

- TBD Create a set of transformation rules/handlers for creating TD from SDF Thing
- Extend the conversion tool and framework
  - Read Input => Invoke Transforms => Write Output

Semantic Annotation using SDF

- SDF URIs have json-pointer fragments
- application/sdf+json contentType needs to be indicated where SDF is referenced

```
"@type": {
   "https://onedm.org/playground/#/sdfObject/Voltage",
   "contentType": "application/sdf+json"
}
```

- Semantic annotation for sdfObjects using "links" construct in the TD, since there is no TD Object class
  - We need a link relation like "implements" to describe this pattern

sdf conversion gaps

- SDF1.1 changes
  - New pattern for input and output, JSON object
  - JSON Schema required for input and output elements
  - sdfChoice + mapping for enum generation
- Generate composed TD
  - Working with composed SDF Thing => TD
  - Add annotation to the generated TD based on SDF source references

TD Vocabulary – value scaling

- We need to represent the data pattern which encodes a fixed point decimal number using an integer
- Also other similar use cases where a linear re-scaling is needed
- scaleMinimum and scaleMaximum will be deprecated in SDF 1.1
- digitalMinimum, digitalMaximum proposed as instance/protocol binding to describe the encoding
- unit, minimum, maximum, and multipleOf will be for the modeled engineering values

Modbus binding

- Hosted 2 modbus devices for the test case and plugfest
- Modbus vocabulary suggestions
  - use address and quantity instead of offset and length
  - review "entity" concept to collapse read/write/obs
- Modbus URI construction feedback
- Modbus requires read multiple entities
  - Should we use array property or TD readmultiple?
  - Entity blocks need to be defined separately in the TD
  - Entity size needs to be defined in a dataschema extension
  - How do we semantically identify the individual entities?

Backup

- Example node-RED application
- Mapping file and binding extensions to SDF
- Semantic Proxy

Example application using Node-RED

- Sequences the power supply output voltage control through a series of steps and records the electrode voltage, electrode current, and thrust
- Value scaling reflected in TD extension parameters
- Setup parameters for maximum voltage and current, DC voltage step, and step time
- Start button is an example of a long-running action (if the experiment had its own TD)

Mapping Files, Bindings, and Extension Points

- Examples of extension points based on auxiliary schemas
  - DataSchema, ProtocolBinding, Hyperlink
- Declaration Format - "extensionPoint" in info block
- How to identify points in the base schema that the extension point is valid for

Extension Declaration Format

```
 "extensionPoint": {
   "DataSchema" : {
     "description": "Additional data schema constraints and mappings",
     "sdfRef": "#/sdfExtensionSchema/DataSchema"
   },
   "ProtocolBinding": {
     "description": "Protocol Binding Form compatible with WoT TD Form",
     "sdfRef": "#/sdfExtensionSchema/ProtocolBinding"
   }

"sdfAction": {
  "SetAmperage": {
    "sdfInputData":{
      "sdfRef": "#/sdfThing/DCPowerSupply_LW3010E/sdfData/AmperageData",
      "DataSchema": {
        "WidthInBits": 16
      }
    },
    "ProtocolBinding": {
      "href": "tcp+modbus://192.168.1.1:502/",
      "modv:unitID": 1,
      "modv:regID": 1001,
      "modv:function": "WriteHoldingRegister"
    }
```

Semantic Proxy