- Use existing standards where ever possible
- Functional abstraction vs Data abstraction
  - Functional abstraction better for legacy and performance (JavaScript: Geolocation example)
- Connectivity: URI naming
  - Addressing must work with non IP and P2P networks
- Security is essential: prefer explicit consumer centric models
  - Again functional abstraction: grant access to data instead of giving away data
  - Peer to peer security model challenges to be addressed
- Driver – Hub Model primary focus – with option for native to device
- Open source: develop before specification– must map to multiple bearers/app protocols

- Specifics: to be standardised
  - URI Schema that can resolve to non IP devices
  - APIs
    - Discovery API
    - Sensor API – ala Geolocation: real–time semantics for data updates
    - Database API – somewhere to store data
  - Security: simple but good enough – MUST support distributed model
- Advanced topics: distributed JavaScript, Remote JavaScript deployment
WOT is the scope?

Hard use case Direct with limited footprint

All connections TLS mutually authenticated with PKI certs

Peer to Peer connections

Embedded server

Multiplexing hub using driver metaphor to convert to common language

Serial
RF
Blue
Zigbe
MQTT
Driver
Questions

- What are the challenges for adapting W3C approaches to IOT/WOT issues?
- How important and CoAP, MQTT to and approach?
- How far should we go in defining a security model for WOT work?
- Why are functional abstractions so important for IOT?