ObjectFlow for Microcontrollers

Simple and Reliable Data
Flow Graph programming
for embedded applications

ObjectFlow

- Data-driven programming for tiny microcontrollers
- Arduino Uno class (2KB/32KB) and larger
- Small library, no central executive, timer- and communication event-driven
- Similar to IEC61499, Node-RED, etc. based on Data Flow Graphs (DFG)
- Uses the LWM2M data model and semantics with an event-driven communication protocol
- Communication is implemented in the application layer using a set of well-known LWM2M types

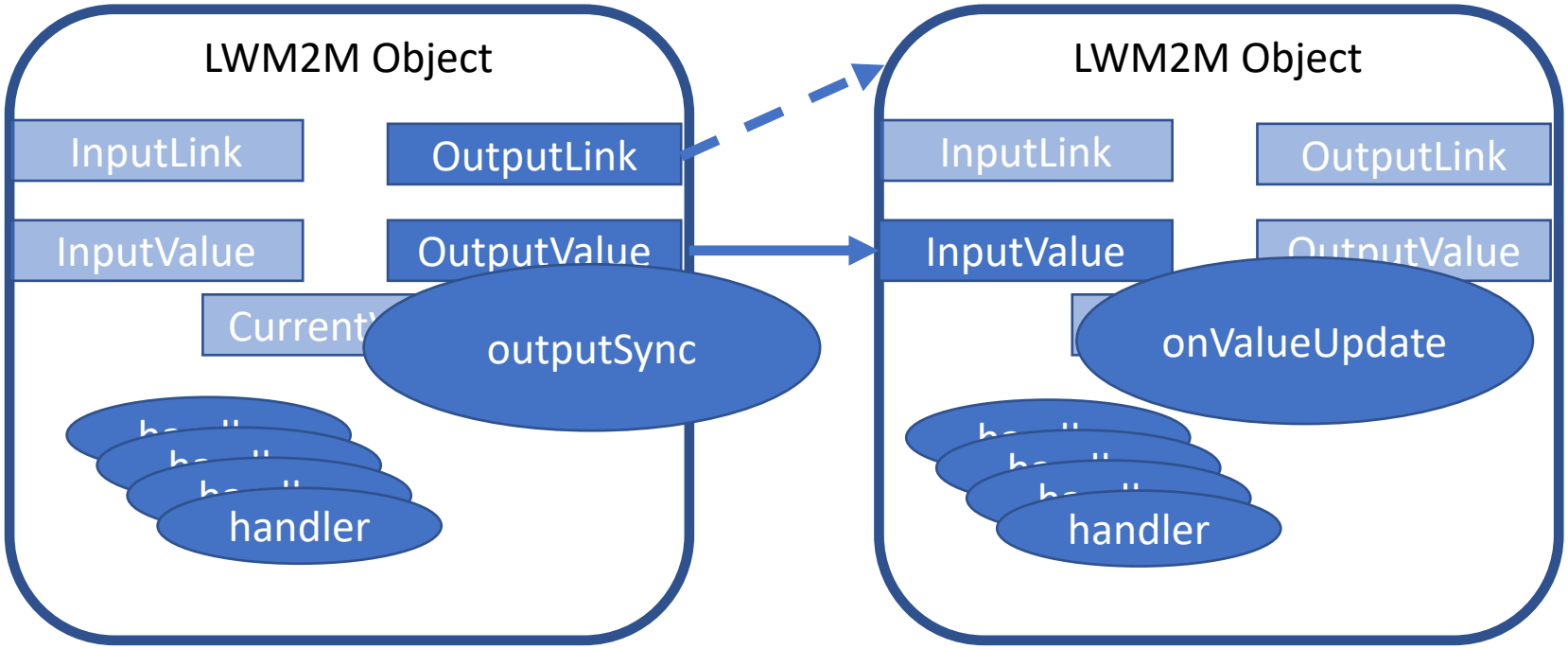
ObjectFlow - Digital Twin for Embedded Code

- On-device code is simple, minimal, and reliable
- A small set of primitives implemented as static C++ wrapper classes for Object and Resource
- Graphs are build from a standardized JSON format that models Object and Resource Instances on the embedded device – a digital twin of the device code
- Tools construct a C++ header file template that is built with the device code application handler bundle using the standard IDE (e.g. Arduino IDE), and downloaded to the device in the usual way.

ObjectFlow – DFG wrapper

- A small set of Resource types for communication
  - Time types for Current, Interval, and LastActivation
  - Input and Output link types based on LWM2M Objlink
  - Subtypes for Input, Output, and Current Value
- DFG - communication is event driven and initiated according to application logic
  - Bound methods implement application handlers
  - InputSync (data pull) and OutputSync (data push)
  - onValueUpdate, onValueSync, onInterval handlers

ObjectFlow – DFG and Reactive Communication



ObjectFlow – DFG architecture

- An ObjectFlow application is a Data Flow Graph
- DFG Nodes are implemented as LWM2M Objects
- A DFG Node is a collection of one or more Objects
- LWM2M Object Links (Objlink) are used to group Objects into a DFG Node and to implement the data flow connections between Nodes (arcs, edges)

ObjectFlow - Tools

- The DFG model tools do all of the heavy lifting so the embedded code can be extremely simple and lightweight, e.g. type checking is done in the model
- Code generation involves serialization of the objects and resources into a C++ header file, and packaging of the implementation code for the application objects (time and data event handlers)
- The result is a standard C++ package that can be built by any C++ toolchain (Arduino, etc.)