Building Internet of Things Devices with AllJoyn

Ivan R. Judson
@irjudson
ivan.judson@microsoft.com
IoT Service Architecture
IoT Mapping Skills

- Things
  - Basic Computer Science
  - Electrical Engineering
  - Mobile Development
  - Mechanical Engineering
  - Embedded Systems
  - RTOS Constraints

- Gateway
  - Network Programming

- Ingest
  - High Scale Real-time Programming

- Register
  - Authentication & Authorization Services

- Transform
  - Low Latency Programming
  - Data Storage APIs

- Consume
  - Conversion between Document DBs and Relational DBs
  - REST API Development

- Store
  - Database Architecture
  - Database Administration

- Present and Act Upon
  - Statistics
  - Data Science
  - Machine Learning
  - User Experience
  - User Interface
What's Missing?
What is AllJoyn?

AllJoyn is...

a **system** that allows devices to advertise and share their abilities with other devices around them.

a **protocol** that your devices can use to interact intelligently.

a **library** you can include in your software to make smarter devices.
AllJoyn System Architecture

AllJoyn Gateway
Remote access and management
Interoperability
Supports open standards:
REST, XMPP, MQTT and TR-069.
AllJoyn Router
Connect Services and Clients.
Cache transient information.
Allow for arbitrary topology.
AllJoyn Service
Discoverable.
Exposé interfaces.
Interfaces are exposed in detail.
AllJoyn Client

Connect to Services

Expose Services to other Services

Expose Services to Users
<table>
<thead>
<tr>
<th><strong>Standard</strong></th>
<th><strong>Thin</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile, Laptop</td>
<td>RTOS</td>
</tr>
<tr>
<td>C++, C, Java, JavaScript, Objective-C</td>
<td>C</td>
</tr>
<tr>
<td>Router optional</td>
<td>No Router</td>
</tr>
</tbody>
</table>
Microsoft & AllJoyn
AllJoyn inside Windows 10!
Support in Visual Studio

AllJoyn Studio plugin
AllJoyn Cordova Plugin & Apps

The Open Source Cordova Plugin

Android iOS Windows
Example

Heatworks
Heatworks Architecture
Heatworks Implementation
**Control Interface**

<table>
<thead>
<tr>
<th>Control</th>
<th>Range</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Point</td>
<td>75-130</td>
<td>Degrees F</td>
</tr>
<tr>
<td>Current Temperature</td>
<td>40-160</td>
<td>Degrees F</td>
</tr>
<tr>
<td>Current Amps</td>
<td>0-50</td>
<td>Amps</td>
</tr>
<tr>
<td>Total Time</td>
<td>0-MAXINT</td>
<td>Seconds</td>
</tr>
<tr>
<td>Total Amps</td>
<td>0-MAXINT</td>
<td>Amps</td>
</tr>
<tr>
<td>Special Modes</td>
<td>0-255</td>
<td>Bitmask</td>
</tr>
<tr>
<td>Presets</td>
<td>Undefined</td>
<td>Undefined</td>
</tr>
</tbody>
</table>
AllJoyn Interface

```xml
<node name="/control">
  <interface name="com.myheatworks.modell">
    <method name="setPoint">
      <arg name="temp" type="y" direction="in"/>
    </method>
    <method name="currentTemp">
      <arg name="temp" type="y" direction="out"/>
    </method>
    <method name="softCurrentLimit">
      <arg name="current" type="y" direction="in"/>
    </method>
    <method name="currentDrawInstant">
      <arg name="current" type="y" direction="out"/>
    </method>
    <method name="timeOdometerValue">
      <arg name="time" type="i" direction="out"/>
    </method>
    <method name="currentOdometerValue">
      <arg name="current" type="i" direction="out"/>
    </method>
  </interface>
</node>
```
Common Definitions

```c
#define CONNECT_ATTEMPTS 10

/* All times are expressed in milliseconds. */
#define CONNECT_TIMEOUT 1000 * 60
#define UNMARSHAL_TIMEOUT 1000 * 5
#define SLEEP_TIME 1000 * 2
#define METHOD_TIMEOUT 1000 * 10

static const char ServiceName[] = "com.myheatworks.modell";
static const char ServicePath[] = "/control";
static const uint16_t ServicePort = 25;

/* Buffer to hold the full service name. This buffer must be big enough to
* a possible 255 characters plus a null terminator (256 bytes)
*/
static char fullServiceName[AJ_MAX_SERVICE_NAME_SIZE];

/***
* The interface name followed by the method signatures.
*
* See also aj_introspect.h
*/
static const char* const sampleInterface[] = {
    "com.myheatworks.modell",       /* The first entry is the interface
    "?setPoint temp<y",             /* Set the water temperature */
    "?currentTemp temp>y",         /* Get the water temperature setting */
    "?softCurrentLimit current<y", /* Set the soft current limit */
    "?currentDrawInstant current>y",/* Get the current draw at this instant */
    "?timeOdometerValue time>i",   /* Get the number of seconds the unit
    "?currentOdometerValue current>i",/* Get the integral of amps where dt 
    NULL
```
Service Implementation

```c
#define AJ_MODULE_MODEL1_SERVICE
#include <stdio.h>
#include <aj_debug.h>
#include "alljoyn.h"
#include "model1.h"

uint8_t dbgMODEL1_SERVICE = 0;

/*
 * The value of the arguments are the indices of the
 * object path in AppObjects (above), interface in sampleInterfaces (above)
 * member indices in the interface.
 * The first index is 1 because the first entry in sampleInterface is the
 * This makes the first index (index 0 of the methods) the second string in
 * sampleInterface[].
 * See also aj_introspect.h
*/
#define BASIC_SERVICE_SETPOINT          AJ_APP_MESSAGE_ID(0, 0, 0)
#define BASIC_SERVICE_CURRENT_TEMP      AJ_APP_MESSAGE_ID(0, 0, 1)
#define BASIC_SERVICE_SOFT_CURRENT_LIMIT AJ_APP_MESSAGE_ID(0, 0, 2)
#define BASIC_SERVICE_CURRENT_DRAW_INSTANT AJ_APP_MESSAGE_ID(0, 0, 3)
#define BASIC_SERVICE_TIME_ODOMETER     AJ_APP_MESSAGE_ID(0, 0, 4)
#define BASIC_SERVICE_CURRENT_ODOMETER  AJ_APP_MESSAGE_ID(0, 0, 5)

int main(int argc, char **argv)
{
    AJ_Status status = AJ_OK;
    AJ_BusAttachment bus;
```
#include <stdio.h>
#include <stdlib.h>
#include <aj_debug.h>
#include <alljoyh.h>
#include "model1.h"

uint8_t dbgMODEL1_CLIENT = 0;

/*
 * The value of the arguments are the indices of the
 * object path in AppObjects (above), interface in sampleInterfaces (above)
 * member indices in the interface.
 * The first index is 1 because the first entry in sampleInterface is the interface.
 * This makes the first index (index 0 of the methods) the second string in
 * sampleInterface[].
 * See also aj_introspect.h
 */
#define BASIC_CLIENT_SETPOINT AJ_PRX_MESSAGE_ID(0, 0, 0)
#define BASIC_CLIENT_CURRENT_TEMP AJ_PRX_MESSAGE_ID(0, 0, 1)
#define BASIC_CLIENT_SOFT_CURRENT_LIMIT AJ_PRX_MESSAGE_ID(0, 0, 2)
#define BASIC_CLIENT_CURRENT_DRAW_INSTANT AJ_PRX_MESSAGE_ID(0, 0, 3)
#define BASIC_CLIENT_TIME_ODOMETER AJ_PRX_MESSAGE_ID(0, 0, 4)
#define BASIC_CLIENT_CURRENT_ODOMETER AJ_PRX_MESSAGE_ID(0, 0, 5)

void SetPoint(AJ_BusAttachment *bus, uint32_t sessionIId, uint8_t setPoint) {
    AJ_Status status = AJ_OK;
    
}
Bringing the Peer to Peer Web to all your Devices with Node.js

Wedensday, April 6th, 2016, 11:05am - 11:55am, in Harbor F

In other news...

- **Bash on windows**...
- **Toyota Connected**...
Thanks!

Ivan R. Judson, PhD

LinkedIn: irjudson
Twitter: @irjudson
Email: ivan.judson@microsoft.com