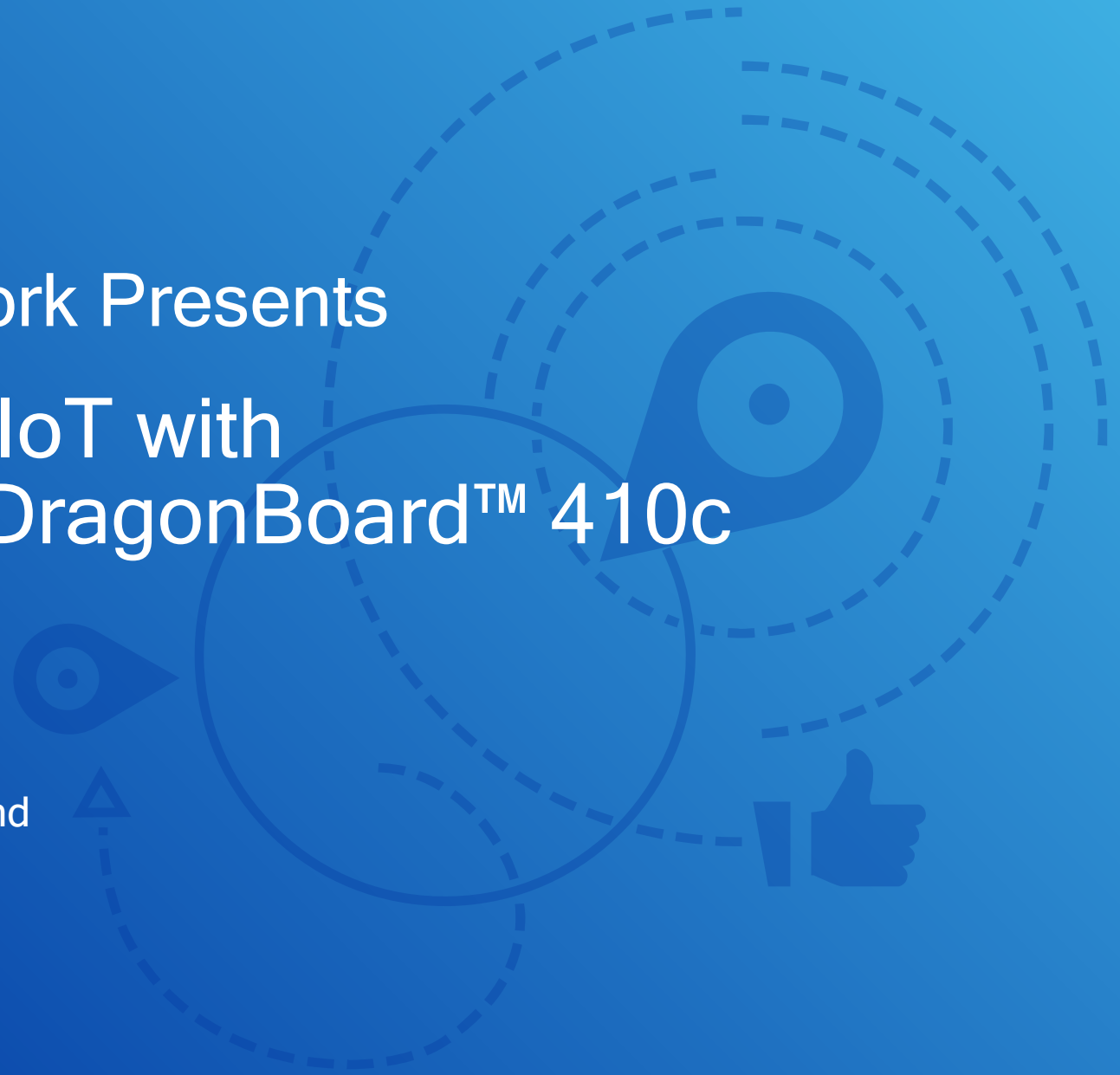
Qualcomm Developer Network Presents

Developing for Industrial IoT with Embedded Linux OS on DragonBoard™ 410c by Timesys University

Co-sponsored by Qualcomm Technologies, Inc. and Arrow Electronics



Embedded Linux from a Trusted Source

timesys®

Session 3
Building a Cutting-Edge User Interface with Qt®

Maciej Halasz, Vice President of Technology
Timesys Corporation

Maurice Kalinowski, Principal Software Engineer
Qt Company

Webinar Series

- **Session 1:** Introduction to DragonBoard 410 SoC and Starting Development of Your Embedded Linux based “Industrial Internet of Things” (IIoT) Device
  - Setup for designing IIoT products
  - How to assemble and deploy initial BSP
- **Session 2**: Application Development for Embedded Linux
  - Application development environment setup
  - How to reflect product requirements in the BSP
  - Communication in the IIoT system
- **Session 3**: Building a Cutting-Edge User Interface with Qt®
  - Developing modern, rich UIs for factory terminals
- **Session 4**: Embedded Products Security
  - Designing security-rich devices

Session 2 recap

- **What we did**
  - Reflected API requirements in OpenEmbedded RPB Linux BSP
  - Talked about BSP customizations
    - New meta-layer
    - New recipe
    - Modified image
- **Application Development**
  - SDK setup on a host
  - Used IDE to develop/deploy/debug code on DragonBoard 410c
  - We looked briefly at a BLE protocol
  - Programmed an application that received info from a sensor board
    - Temperature, Luminosity, FreeFall
- **Key takeaways**
  - Its very straight forward to develop C/C++ application code for the 410c
  - IDEs accelerate development process
  - When working with Linaro BSP, customers can leverage meta layer library on OpenEmbedded

https://layers.openembedded.org

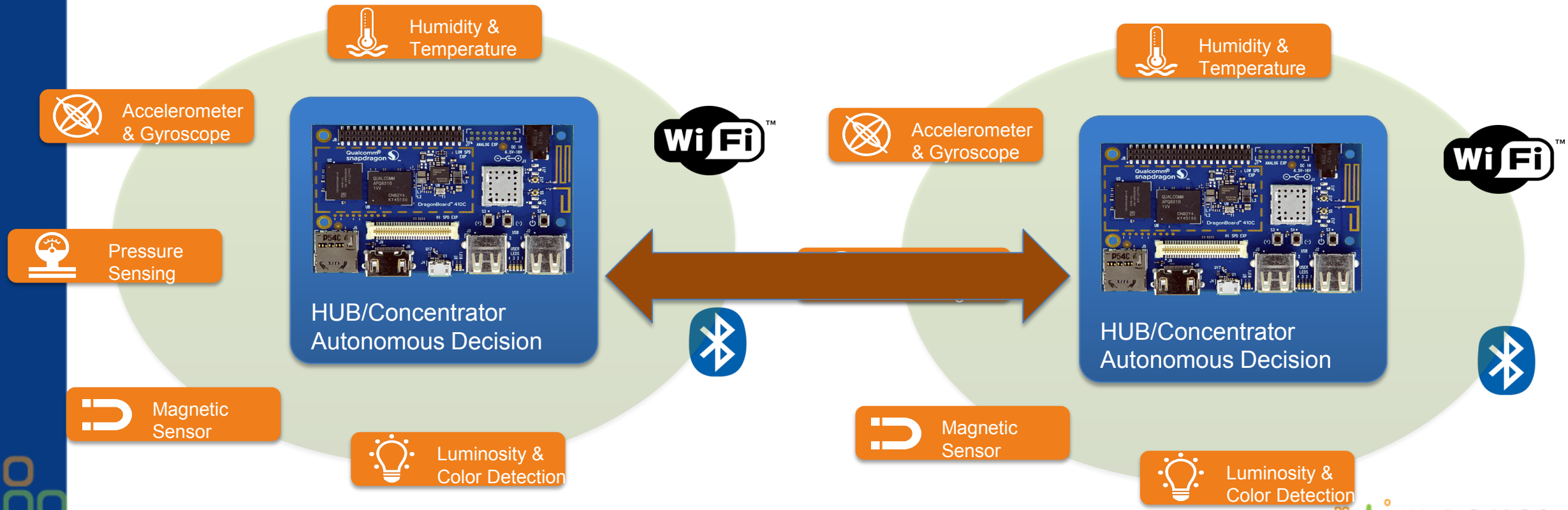
Session 3 — Agenda

- Sharing information in an IoT system
- Introduction to the Qt Software
- Integrating Qt in the OpenEmbedded RPB Linux BSP
- The Qt Software structure
  - Qt for Application Development
  - Qt for Device Creation
- Qt Development Environment setup
- Writing first Qt application
  - Programing language options
- Qt for Automation
- Licensing considerations
- Q & A

How to share/publish information?

Publishing information

- **Why need for sharing information?**
  - System can have multiple Hubs collecting data
- **Information may be needed to make a wider scope decision**

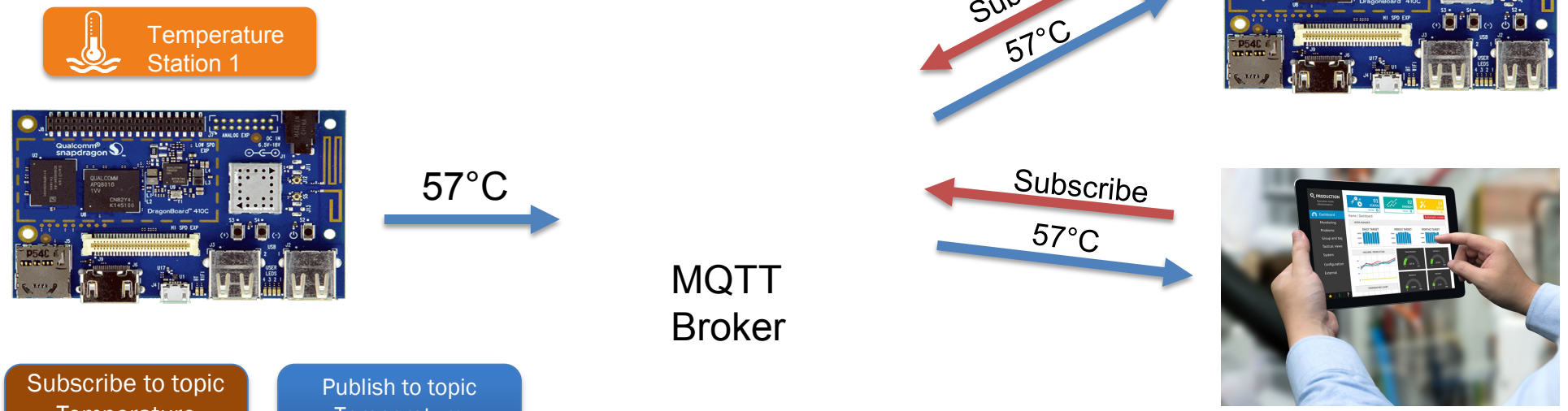


Publishing, energy efficient protocol

- **MQTT is a lightweight publish/subscribe protocol**
  - reliable bi-directional message delivery.
- **Invented by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999**
- **Why is MQTT gaining popularity?**
  - Runs on top of TCP/IP
    - Both client and server (aka broker) leverage TCP/IP stack
  - Can handle high volumes of data even in low bandwidth networks
  - Small code footprint
  - Event oriented.
  - Implementation of the protocol provides publish/subscribe and also recovery
  - Wide use
    - POS
    - Asset tracking
    - Automation
    - more

MQTT mechanism

- **Implements publish <-> subscribe mechanism**
  - Server – Broker – has subscribed clients connected at all time
  - Clients can be both subscribers and publishers
  - A client sends a message (publisher role)
  - One or more clients can receive the posted message (subscriber role)
  - MQTT protocol uses “topics” to determine which message is routed to which client
    - Client must be subscribed to a “topic”
  - A topic is a “hierarchical structured string”



MQTT Software

- **Client**
  - Small code base API
  - Available in C, C++, Go, Java, JavaScript
- **Broker**
  - Handles authorization of clients
  - Scalable
  - Easy to integrate
  - Several options available
    - Mosquitto
      - Recipe available in meta-oe
- **Broker can be installed on**
  - HUB
    - Smart sensors
  - Cloud Server
    - Messages submitted from HUBs

What is Qt?

Quick introduction

- **Qt is a cross-platform application framework**
  - Supports development of applications for desktop, embedded (including headless) and mobile
- **Qt is versatile and feature rich**
  - Qt 5.7, 5.9 includes Qt 3D, Qt Quick UI Controls 2.0 for embedded and mobile, modern C++ 11 support, graphics improvements for embedded (EGLFS and Wayland), and HTML5/hybrid user interfaces with the Qt WebEngine module.
    - https://www.qt.io/qt5-7/
  - New licensing options – 5.6 LTS with LGPLv2; 5.7, 5.9 with commercial, LGPLv3 or GPLv3
    - http://doc.qt.io/qt-5/licensing.html
    - http://doc.qt.io/qt-5/licensing.html#licenses-used-in-qt
    - http://doc.qt.io/qt-5/qtwebengine-licensing.html
- **The features and different licensing options available with Qt 5.7/5.9 creates many questions in developers' minds such as:**
  - What Qt modules (Essentials, Qt Quick, Add-Ons, Value-Add, …) do I need for my application development?
  - How do I install Qt in my BSP?
  - Should I choose Qt 5.6 with LGPLv2, or should I first develop with Qt 5.9 GPL, and based on the requirements, choose GPL or commercial or LGPLv3?
  - Which package should I start with?
    - Boot to Qt, Qt for Application Development, Qt for Device Creation

Adding Qt Software to the OpenEmbedded RPB BSP for the DragonBoard 410c

Qt in OpenEmbedded RPB

- **Qt software availability**
  - Recipes are already present in the OpenEmbedded RPB BSP
    - meta-qt5 fetched with the RPB BSP
  - BSP image definition
    - Part of meta-rpb
    - Contains qt-5.7
    - Demos
      - Cinematic experience
      - Standard QtDemos
  - Options to build in desktop environment
    - Wayland, X11
- **Possible modifications**
  - Newer Qt version
  - SDK with Qt libraries for application development

OpenEmbedded RPB customizations

- **Modifications**
  - Extended further the rpb-qt5-image recipe
    - custom-qt5-image.bb
    - Added X11 package groups
    - Sato Desktop
- **Build**
  - Custom image

```
$ bitbake custom-qt5-image
```

  - QT SDK

```
$ bitbake meta-toolchain-qt5
```

  - Installed SDK under /opt/rpb-qt5

```
# by default and without x11-base in IMAGE_FEA
SYSTEMD_DEFAULT_TARGET = "multi-user.target"

CORE_IMAGE_BASE_INSTALL += " \
        96boards-tools \
        alsa-utils-aplay \
        networkmanager \
        networkmanager-nmtui \
        coreutils \
        gps-utils \
        gpsd \
        gptfdisk \
        gstreamer1.0-plugins-bad-meta \
        gstreamer1.0-plugins-base-meta \
        gstreamer1.0-plugins-good-meta \
        ${@bb.utils.contains("LICENSE_FLAGS_WH
        hostapd \
        iptables \
        kernel-modules \
        sshfs-fuse \
        cinematicexperience \
        qt5everywheredemo \
        qtbase-examples \
        packagegroup-core-x11-base \
        packagegroup-core-x11-sato \
        packagegroup-core-boot \
        ${@bb.utils.contains( MACHINE_FEATURES
"
```

LAB 1 — Getting ready for the Qt Development

- **Generate deployment images with Qt5**
- **Generate Qt SDK**
- **Install Qt SDK on a host PC**
- **Download Qt software**
  - IDE – QtCreator
  - Qt5 software stack

Questions?

Source code, deployment images and SDK can be downloaded from linuxlink.timesys.com

---

developer.qualcomm.com 96boards.org arrow.com timesys.com

timesys®

Thank you

Follow us on:

For more information, visit us at:
www.qualcomm.com & www.qualcomm.com/blog