Qt day

Witekio
EMBEDDING SUCCESS

Qt for IoT: Present and Future

The process, and what Qt brings

Few words about IoT

- IoT
  - Internet of Things
- IIoT
  - Industrial IoT
- Edge IoT, Fog computing
  - Intelligent devices directly connected to the cloud

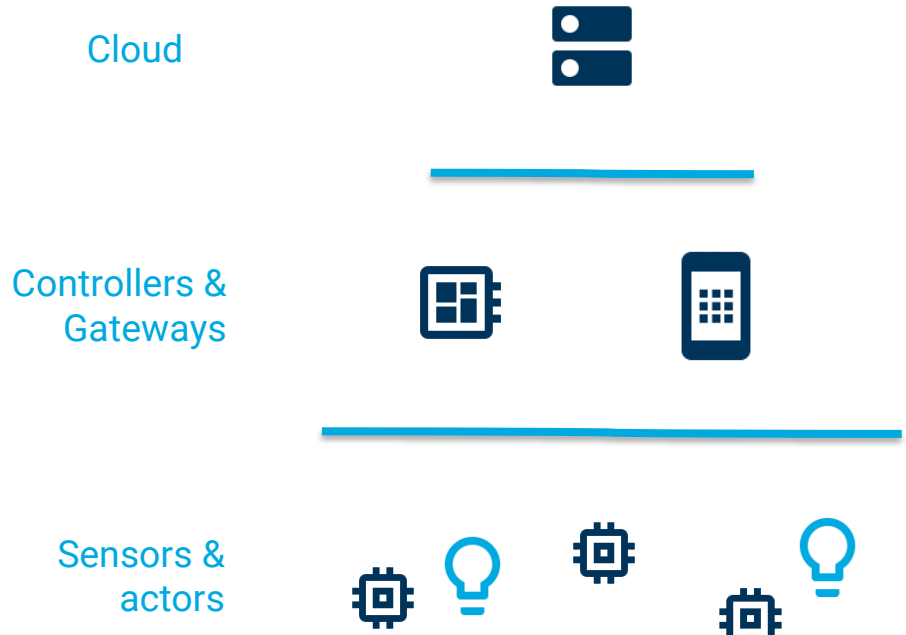
- How to go there?
- What is the place of Qt in that landscape?

- Adrien Leravat
  - Software Architect @Witekio

The IoT landscape

Light & Simple

- Cloud
  - High capacity and processing power
  - Storage, authentication, inter-sites bridge, main logic, web front/back, AI, …
- Edge
  - Decent capacity and processing power
  - Networking, control panel, phone
- Sensors/Actors
  - Very limited capacity and processing power
  - Limited connectivity options and roles
- Different roles and attributes
- Glue: set of technologies



Qt day

Witekio

From an Idea, to a product

- The key steps to a successful (IoT) product
- Don't underestimate their importance
- Anticipating them avoid project pitfalls

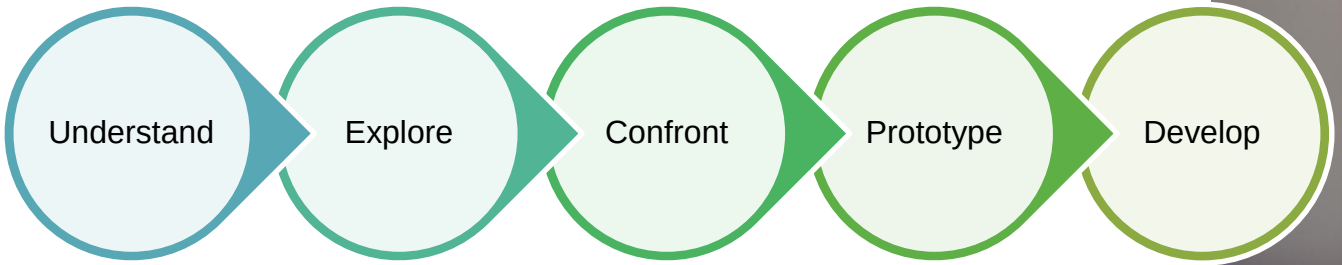
Ideation & Prototyping

Qt day

Ideation

Understand your real goals

- Success of a product = alignment with customers needs and pains
- Should be the first step
- Confront with customers expectations



Qt day

Prototype

Iterate… fast

- Target a MVP (Minimum Viable Product)

while (true) {

- Create-update prototype
- Have real customers test it, gather feedbacks

}

- Tons of tools
  - Adobe XD, proto.io, …
- Often HTML based
- What can Qt offer?

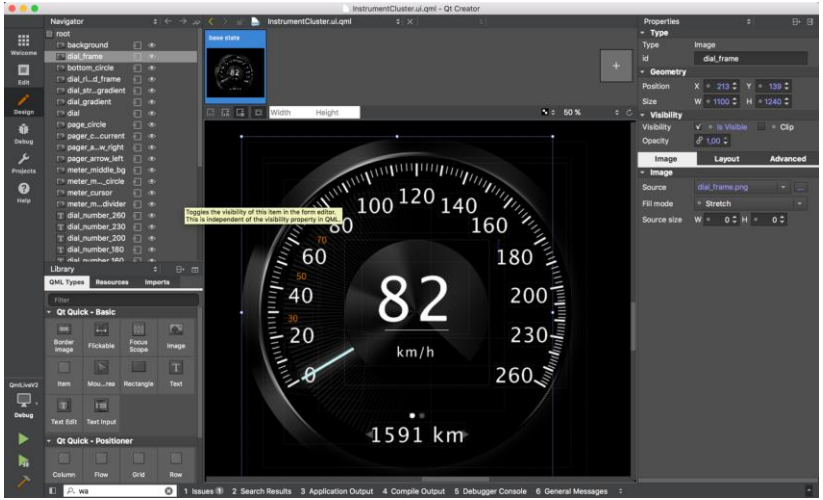
- Run web-based prototypes
  - WebView !

- Export Photoshop design to QML
- Use Qt Designer, Qt Quick UI Forms and Qt Quick Viewer
- Use Javascript for the logic

- Or both… and you're done!
- What's more: you can keep the same code

Prototype

Iterate… fast



From https://github.com/Pelagicore/QmlAssetExporter

- Run web-based prototypes
  - WebView !
- Export Photoshop design to QML
- Use Qt Designer, Qt Quick UI Forms and Qt Quick Viewer
- Use Javascript for the logic

- Or both… and you're done!
- What's more: you can keep the same code

Design

Qt day

Multiple design levels

… and questions !

- Device
  - Hardware, OS, libraries
- Network technologies
  - WiFi, BLE, SigFox
- Protocols
  - HTTP, MQTT, KNX

- Each adds strengths, weaknesses and constrains to the solution

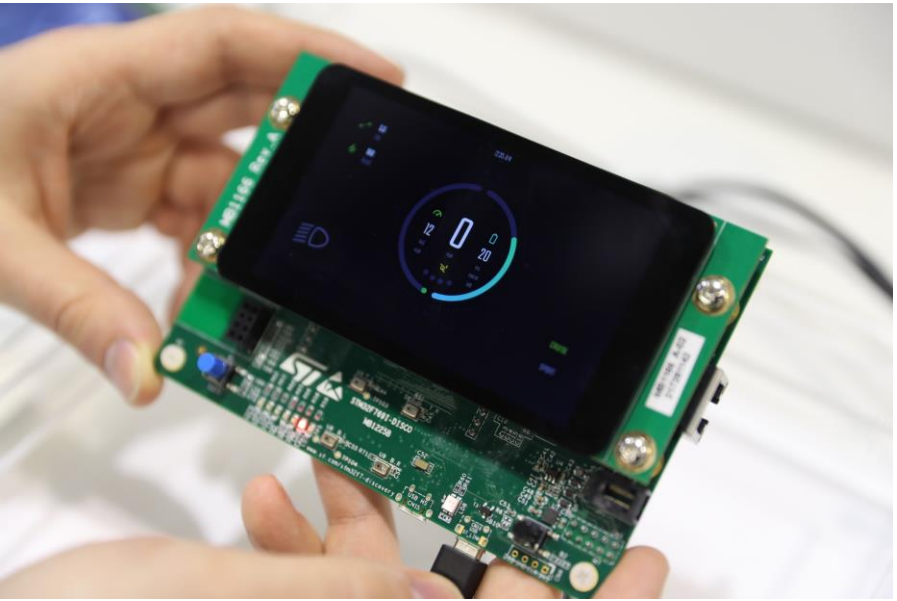
Device

Hardware and software

- Lower end CPU/MCU
  - Cheaper
  - Smaller form factor
  - More energy efficient
- Qt typically runs of Cortex-A
  - e.g.: i.MX6
  - 256 MB of RAM
- What's new here?
- Qt Lite
  - Only features you need
  - Smaller, faster application
- Qt on MPU
  - Cortex-M ! STM32F4
  - RTOS
- Linux Yocto Qt layer
- Boot2Qt
  - Services to boot on your application

Device

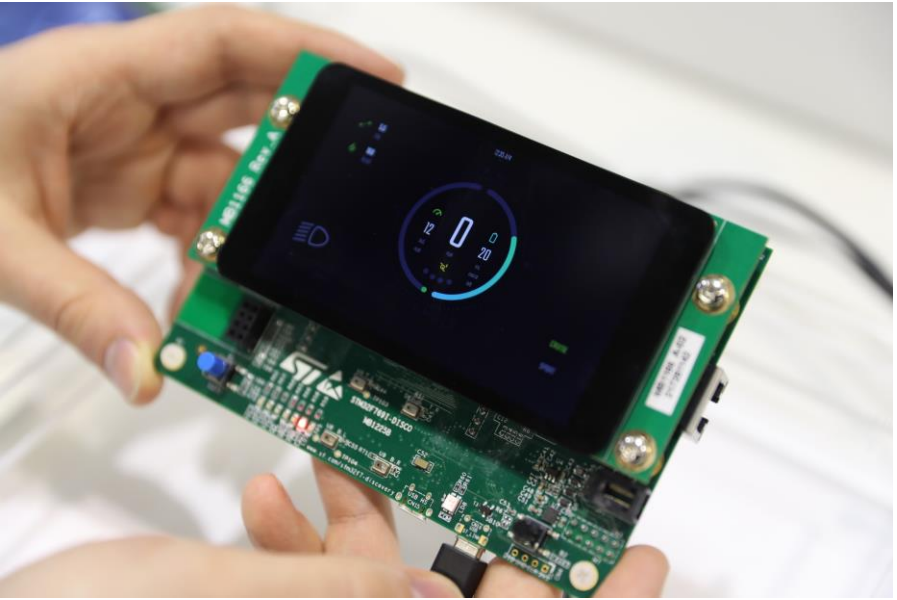
Hardware and software



- Qt Lite
  - Only features you need
  - Smaller, faster application
- Qt on MPU
  - Cortex-M ! STM32F4
  - RTOS
- Linux Yocto Qt layer
- Boot2Qt
  - Services to boot on your application

Device

Hardware and software



- Build in release, -O3/-Os
- Use static linking
  - Commercial license!
- Use QML Compiler
- Disable unused Qt modules
- Don’t neglect Widgets for GUIs
- Don’t bloat resource files
  - Keep large files on the FS
- Tailor resources to target
  - Resolution, format, depth
  - Optimize png/jpg
  - SVG for large or scaled resources
- Use plugin and service-based architecture

Network technologies

Hardware and software

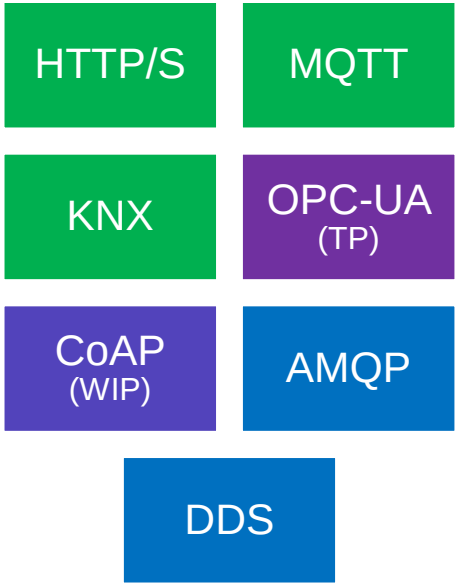
Wired	Serial Modbus, CAN, RS232	Ethernet TCP, UDP
PAN	NFC	BT/BLE
	Zigbee	Thread
LAN	Wi-Fi	
WAN	Sigfox	LoRa

- Support at OS level
  - mbed, FreeRTOS, Linux
- Qt provides higher level layer
- Others: Rely on your OS driver
  - or implement your own!

Protocols

Hardware and software

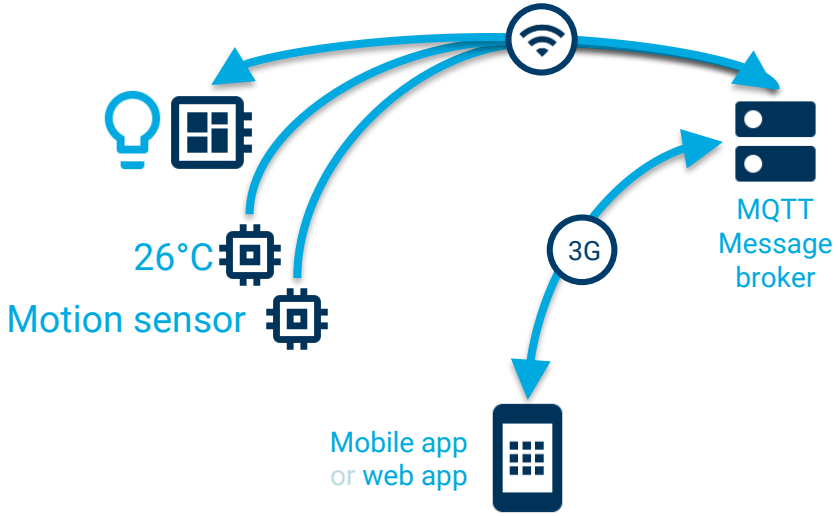
- Multipurpose, point to point
  - HTTP/S, CoAP
- Multipurpose broker based
  - MQTT, AMQP, STOMP
- Industry: OPC-UA
- Home automation: KNX



MQTT

Light & Simple

- Requires a message broker
  - Dispatch all messages
- Features
  - Light and simple
  - Publish-subscribe on topics
    - Event driven
  - Support “last will”
  - QoS
- Limited features and extensibility
- Request-answer pattern hard to implement



MQTT

Light & Simple

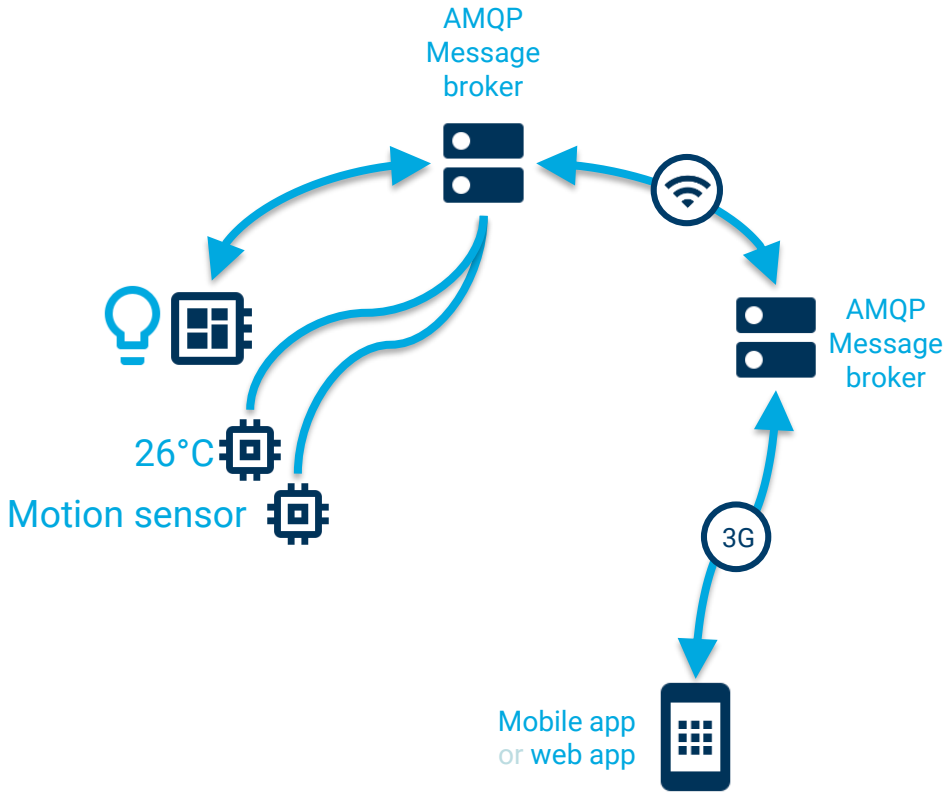
- Requires a message broker
  - Dispatch all messages
- Features
  - Light and simple
  - Publish-subscribe on topics
    - Event driven
  - Support “last will”
  - QoS
- Limited features and extensibility
- Request-answer pattern hard to implement

```
m_client = new QMqttClient();
m_client->setHostname(“192.168.10.22”);
m_client->setPort(1883);

//...

QMqttSubscription *sub =
                m_client->subscribe(“qtday/session”);

//...

sub->unsubscribe();
```

AMQP

Extensible

- Often uses message broker(s)
- Extensible easily with headers
  - ex: “reply-to: device543”
- Features
  - Publish-subscribe
  - Request-answer
  - Queues
  - Flexible typology
- Heavier
- Headers support may vary



AMQP

Extensible

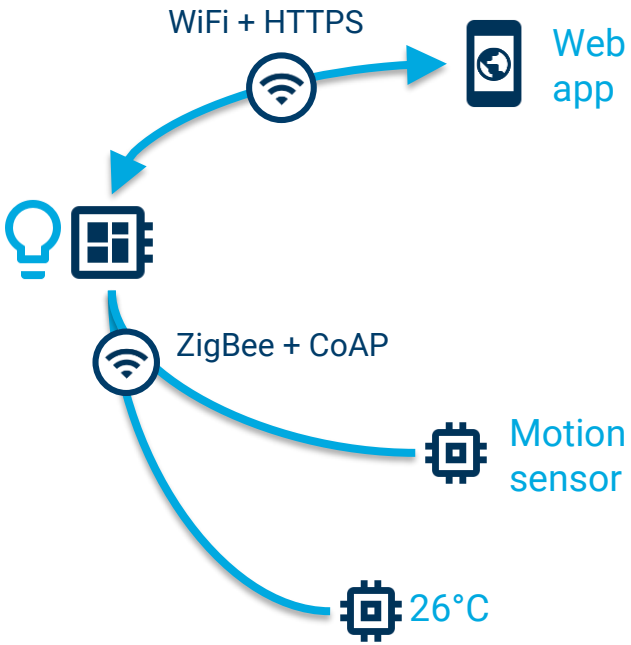
- Often uses message broker(s)
- Extensible easily with headers
  - ex: “reply-to: device543”
- Features
  - Publish-subscribe
  - Request-answer
  - Queues
  - Flexible typology
- Heavier
- Headers support may vary

- Compatible clients
- QAmqp
  - https://github.com/mbroadst/qamqp
  - AMQP v0.9.1
- RabbitMQ
- Apache Qpid
- Amazon, Azure

CoAP

HTTP, but lighter

- Simple, based on HTTP
  - But lighter, and uses UDP
- Features
  - Request/answer pattern
  - RESTful
  - Discover resources, observe
    - Detect endpoints and be notified of resource change
  - Easy to map to HTTP
- Difficult in NAT environment
- Not optimal for events



CoAP

HTTP, but lighter

- Simple, based on HTTP
  - But lighter, and uses UDP
- Features
  - Request/answer pattern
  - RESTful
  - Discover resources, observe
    - Detect endpoints and be notified of resource change
  - Easy to map to HTTP
- Difficult in NAT environment
- Not optimal for events

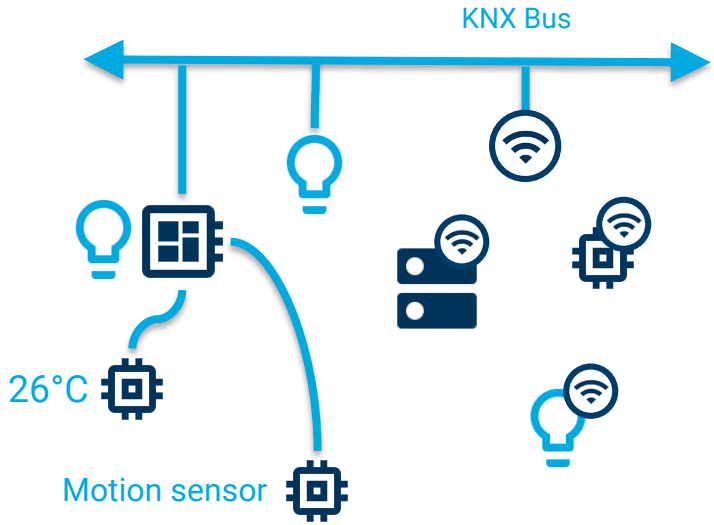
```
QCoapClient* coap = new QCoapClient(this);
connect(coap, &QCoapClient::finished,
        this, &MyClass::onFinished);

coap->get(QCoapRequest(QUrl("coap://<ip>/sensor/temp")));

coap->put(QCoapRequest(QUrl("coap://<ip>/sensor/poll-rate")),
                       QByteArray("10")); // or QIODevice
```

- Status
- Work in process
  - 5.12 tech preview?
- LibCoap (C)
- CantCoap (C++)

KNX

Home automation standard

- Main home automation and building control standard
  - Sensors, actuators, …
  - Only certified devices
- Support different physical media
  - Ethernet (KNXnet/IP), radio, twisted pair, powerline, infrared
- Different installation modes
  - Automatic, easy, system
- Versatile



KNX

Home automation standard

- Main home automation and building control standard
  - Sensors, actuators, …
  - Only certified devices
- Support different physical media
  - Ethernet (KNXnet/IP), radio, twisted pair, powerline, infrared
- Different installation modes
  - Automatic, easy, system
- Versatile

```
QKnxNetIpTunnelConnection connection;
connection.connectToHost(QHostAddress(...), port);

const QKnxSwitch switch { QKnxSwitch::State::On };
const QKnxAddress address { QKnxAddress::Type::Group, "2.6.4" };
QKnxTunnelFrame frame =
QKnxTunnelFrameFactory::GroupValue::WriteRequest(address,
    switch.bytes());
connection.sendTunnelFrame(frame);
```

OPC-UA

Industrial standard

- Discovery features
- Scalable
- Secure
  - Authentication and encryption
- Client/Server
- Binary and HTTP protocol
- Relies on different backends
  - Open62541, Unified Automation

```
bool QOpcUaReader::init()
{
    QOpcUaProvider p;
    qDebug() << "Available backends:" << p.availableBackends();

    m_client = p.createClient(p.availableBackends()[0]));

    QObject::connect(m_client.data(), &QOpcUaClient::stateChanged,
                     this, &QOpcUaReader::handleStateChanged);
    m_client->connectToEndpoint(
                     QUrl("opc.tcp://domain.com:51210/UA/SampleServer"));

    return true;
}
```

From https://blog.basyskom.com/2018/want-to-give-qt-opcua-a-try/

Make it reliable

Qt day

Secure

Communications

SSL/TLS
up to TLS 1.2

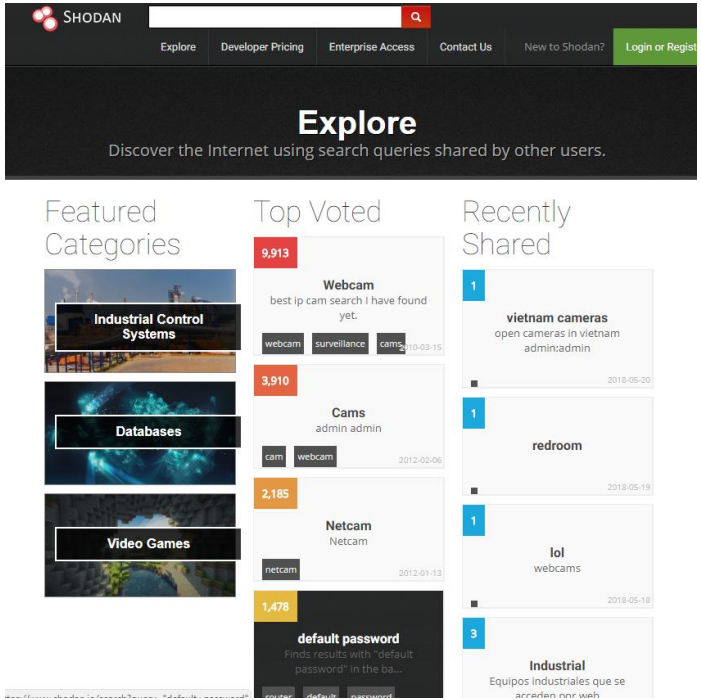
DTLS
(TP)

Authentication
OAuth1-2

- Communications secured with
  - SSL/TLS for TCP
  - DTLS for UDP
- Authenticate with OAuth1 & OAuth2 services
  - Google, Facebook, Twitter
- Mandatory for most
  - Impacts CPU/MCU usage
  - Look for hardware cryptographic acceleration

Secure

Communications



- Communications secured with
  - SSL/TLS for TCP
  - DTLS for UDP
- Authenticate with OAuth1 & OAuth2 services
  - Google, Facebook, Twitter
- Mandatory for most
  - Impacts CPU/MCU usage
  - Look for hardware cryptographic acceleration

Test

For the greater good

- Testing is Key
- Devices not easily accessible and updatable
- Software should be designed for tests
  - Big difference!
- Follow SOLID principles
  - Single responsibility
  - Open/closed
  - Liskov substitution
  - Interface segregation
  - Dependency inversion

- Already integrated in Qt Creator
  - Qt Tests
  - Qt Quick tests
  - Google tests

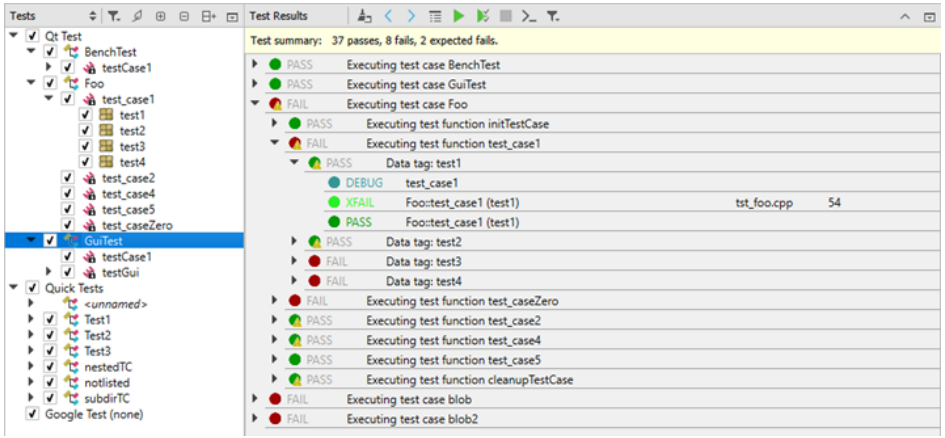
```
void TestQString::toUpper()
{
    QString str = "Hello";
    QCOMPARE(str.toUpper(), QString("HELLO"));
}
```

- Complemented by GUI testing tools
  - Froglogic Squish, Ranorex

Test

For the greater good

- Testing is Key
- Devices not easily accessible and updatable
- Software should be designed for tests
  - Big difference!
- Follow SOLID principles
  - Single responsibility
  - Open/closed
  - Liskov substitution
  - Interface segregation
  - Dependency inversion



Test

For the greater good

- Build and run tests run automatically
  - Software factory
- Continuous integration system
  - Jenkins, Pipelines, TeamCity

Ultimate integration test

- Test on the real hardware
- Concept of “farm”
  - Testing directly on the final hardware
  - Simulate power on/off, SD card, physical buttons

Witekio, United Kingdom

Deploy & maintain

Qt day

Manufacture

And provisioning

- Devices need to be tested for QA
- "Manufacturing mode" for testing

- System needs to be secured
  - Authorized servers
  - Authorized devices
- = Provisioning
  - Provide servers certificates
  - Provide and share device certificate
  - Serial number, …

- Typically automatically done at manufacturing

- Requires dedicated tools and infrastructure

Update

Over land and sea

- OTA
- Progressively deploy features
  - Less planning stress
  - Test population
- Fix security breaches
  - Connected = at risk

- Req: Dedicated management
- Req: High bandwidth
  - Hundreds of kB to tens to MB
- Req: Redundancy
  - There is no way back if bricked!

- Dedicated management
  - swupdate, vendor specific, or custom mechanism
- High bandwidth
  - Alternative connection: USB, BLE, Wi-Fi
- Redundancy
  - Dual bank

- Don’t forget to secure it too!

Maintain

What's going on?

- Be able to know what happens
  - Get notified in case of problem
  - Get usage information and statistics
  - Detect suspicious and malicious behavior
  - Predictive maintenance
- Application logs
  - Override logging to filter, transform and forward logs
  - Leverage Qt logging categories

```
qInstallMessageHandler(myLogHandler);

QLoggingCategory category("temp-sensor");
qCDebug(category) << "debug message";
```

- Format logs for parsing
  - This is the goal!
  - Use JSON if possible: QJson!

```
{
    type: "warning",
    timestamp: "2018-23-05 13:05:00 -0100",
    action: "get-temperature",
    sensor: "fridge"
    result: "temperature-high"
    temperature: 56
}
```

- Catch and report errors
  - Use exceptions and errors
  - Services error logs

Maintain

What's going on?

- Logging and statistics services
  - Parse: Log Stash, Loggly
  - Store: MongoDB, Elastic search
  - Visualize: Kibana
- … plus lot of tools from Microsoft Azure, Amazon, and Google

Obsolescence

This is the end

- One day your device will be outdated
  - Hardware and software
- Goal: minimize the transition cost
  - Plan for the worst, hope for the best!
- Are we making the right choices?

- Qt = long lasting and stable API
  - Seems like a sensible choice
- Other pillars
  - SOLID principles
  - Service architecture
  - Hardware abstraction
- Ask the same question for
  - Hardware
  - Other services

Last words

- Qt is a very good candidate for IoT
  - Main protocols and features already supported, continuing
  - Leaning toward smaller and smaller devices
- Continue building great software
- … and think ahead
- What is the future of it?

Come talk to us, or hit “follow”!

- Adrien Leravat
- Witekio