Amazon FreeRTOS
IoT Operating System for Microcontrollers
Introduction

Richard Barry
Founder – FreeRTOS Project
Principal Engineer – Amazon Web Services
The FreeRTOS kernel, and Traditional Use Cases
  - Design Goals
  - Differences from General Purpose Operating Systems

IoT Relevant AWS Cloud Services

Amazon FreeRTOS Libraries

Amazon FreeRTOS, Connected Use Cases
(Very Brief) Introduction to the FreeRTOS Kernel

- It is a kernel, or scheduler, or RTOS, depending who you ask
- 15 Year Heritage
- MIT Licensed C Code and Pre-configured Projects
For MCUs

- μCLinux, eCOS
- Real Time Linux

Applicability
- 10s of K of RAM
- 100s of K of Flash

Processor power

No Scheduler
FreeRTOS
Technical Deep Dive? Removing Adoption Blockers

- Rapid Support
- Leadership
- Documentation
- Windows Host
- Customer Obsession
- Paid Options
- No IP Infringement
- Demonstrable Code Quality
- Robustness
- Knowledgeable Support
- Visible Activity
- Enterprise Friendly Licensing

Copyright 2018 Amazon Web Services
MIT Licensed C Source Code

- **Source File Downloads**
  - FreeRTOS.org
    - SVN
  - aws.amazon.com/FreeRTOS
    - GitHub
Usage Model

User (Application) Code

FreeRTOS
Kernel

[Middleware Libraries]

Vendor Supplied Libraries

Hardware
Use Cases
Use Cases – Syringe Pumps
Why?
Application Design Goals

- Depends on the application, but generally:
  - Meet real time requirements!
  - Maximize responsiveness
  - Use as little CPU/Power as possible
  - Maximize maintainability
  - Maximize portability (hardware change)
  - Simplicity!
  - Fast to market
  - Meet requirements with minimum expenditure
Physical Challenges

- Alarm Annunciator
- ADC Input
- Keypad Input
- Timer
- Motor Position
- CAN Request

- Convert and Display
- Commutation/Control
- Formatting
- CAN Driver/Protocol

- LCD
- Motor
- Flash Disk
- CAN Response
RTOS Operation

Function1_Run();

Function2_Run();

Function3_Run();

Function4_Run();

Application Design Goals:
- Depends on the application, but generally:
  - Meet real time requirements!
  - Maximize responsiveness
  - Use as little CPU/Power as possible
  - Maximize maintainability
  - Maximize portability (hardware change)
  - Simplicity!
  - Fast to market
  - Meet requirements with minimum expenditure
Multithreaded Design
Improved CPU Utilization
Improved CPU Utilization

xStreamBufferReceive()

Tick event if timeout / Trigger level hit if no timeout

Suspended

Ready

Running

Blocked

Copyright 2018 Amazon Web Services
Introduction to Amazon FreeRTOS
On-demand delivery of compute power, databases, applications, etc. via the internet, >125 services

Pay-as-you-go pricing
AWS IoT - In the Cloud

Cloud
Storage & Compute

Secure device connectivity and messaging
Fleet onboarding, management and SW updates
Fleet audit and protection
IoT data analytics and intelligence

Intelligence
Insights & Logic → Action

Copyright 2018 Amazon Web Services
AWS IoT – Cloud and Edge

Endpoints
- Fleet onboarding, management and SW updates
- Fleet audit and protection
- IoT data analytics and intelligence
- Secure device connectivity and messaging
- Secure local triggers, actions, and data sync

Cloud
- Storage & Compute

Things
- Sense & Act
- Cloud Storage & Compute
- Secure local triggers, actions, and data sync
- Secure device connectivity and messaging
- Fleet onboarding, management and SW updates
- Fleet audit and protection
- IoT data analytics and intelligence

Intelligence
- Insights & Logic → Action

Copyright 2018 Amazon Web Services
Why Amazon FreeRTOS?
Primary Functionality Vs Security, Connectivity

- Bootloader
- OTA
- Key Management
- Security Stack
- Communications Stack
- Functionality
The FreeRTOS Kernel

User (Application) Code

FreeRTOS Kernel

Vendor Supplied Libraries

Hardware

Copyright 2018 Amazon Web Services
Amazon FreeRTOS – Kernel and Libraries

Library Porting View

- User (Application) Code
- Amazon FreeRTOS Libraries
  - MQTT
  - MQTT I/O Port
  - Key Management
  - Secure Storage
  - TCP/IP
  - TLS Library
- Vendor Supplied Libraries
  - Wi-Fi
- FreeRTOS Kernel
- Hardware

Copyright 2018 Amazon Web Services
Design Goals
OTA on Amazon FreeRTOS – Operator Actions

1. Developer authors update
2. Upload to cloud and sign image
3. Schedule an update job
4. Notify device update is available
OTA on Amazon FreeRTOS – OTA Agent Actions

1. Developer authors update
2. Upload to cloud and sign image
3. Schedule an update job
4. Notify device update is available

5. Device downloads image (or streams over MQTT)
6. Write image to flash
7. Close file and verify signature
8. Notify application that new image is ready

Application activates when it is ready (set boot flags and reset)
OTA on Amazon FreeRTOS – Bootloader Actions

1. Developer authors update
2. Upload to cloud and sign image
3. Schedule an update job
4. Notify device update is available
5. Device downloads image (or streams over MQTT)
6. Write image to flash
7. Close file and verify signature
8. Notify application that new image is ready
9. Application activates when it is ready (set boot flags and reset)
10. Verify image at boot
11. Initialize OTA agent and confirm current image is latest
12. Hand control to application for self test
13. On passing self test, commit new image
14. Update cloud status to completed.
Design Goals
Amazon FreeRTOS – Kernel and Libraries
Application Writer’s View

User (Application) Code

FreeRTOS Kernel

MQTT Agent  Greengrass Discovery  OTA Agent

Amazon FreeRTOS Libraries

Vendor Supplied Libraries

Hardware

Copyright 2018 Amazon Web Services
Predictive Maintenance Example

Detect Anomalies to Predict Pump Failure

1. Run Amazon FreeRTOS on vibration sensors to securely collect data and connect to AWS Greengrass enabled device.

2. The AWS Greengrass enabled device runs the predictive model locally to identify when vibrations hit dangerous levels. AWS Greengrass triggers alert to maintenance staff when anomalies are detected. When Internet connectivity is available, the AWS Greengrass device sends data to the cloud for analytics filtering out “normal” data.

3. AWS IoT Analytics analyzes vibration data and adds time stamp and device information such as serial number pulling from AWS IoT Core. Sends updated model to the AWS Greengrass enabled device.
Thank You!

- https://www.FreeRTOS.org
- https://freertos.org/FreeRTOS-quick-start-guide.html
- https://aws.amazon.com/freertos
Predictive Quality Example

Predict Crop Quality

1. Soil sensors measure PH, moisture, nutrients, and gases

2. AWS IoT Analytics enriches soil sensor data with geolocation, rainfall, and weather information and predicts crop health and quality. Makes suggestions on watering and fertilization schedule to increase crop yield
Asset Condition Monitoring

Global Mining
Company Minimizes Unexpected Downtime

Problem
Needs to understand degradation equipment

Solution
Collect data from equipment, identify potholes and other problems on mining routes that can contribute to degradation

Impact
Monitor equipment status, health, and performance to detect issues in real-time. Detect road issues and identify equipment degradation and minimize downtime