IoT programming
with openSUSE
About this presentation

- What is IoT?
- Why is it important?
- Typical IoT devices
- Programming IoT devices
- Examples
About me

- SUSE veteran
- Open source veteran
  (Google knows it all)
- Father, Maker, Hacker
- Agilist
- Currently: Product Owner for SUSE Container platform offering
IoT
IoT – Internet of Things

“The Internet of things (IoT) is the extension of Internet connectivity into physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled.”

Source: https://en.wikipedia.org/wiki/Internet_of_things
Why is it important?

“IDC Forecasts Worldwide Spending on the Internet of Things to Reach $745 Billion in 2019”

¹ https://www.idc.com/getdoc.jsp?containerId=prUS44596319
IoT device examples
Raspberry Pi
ESP-201
ESP-01, ESP-201, WeMOS D1

- Espressif 8266 based (Xtensa CPU core)
  - Successor: ESP 32
- WLAN (IEEE 802.11 b/g/n)
- 80 MHz, RISC
- 32kB instruction, 80kB user data RAM
- 512kB - 4MB Flash
- < 5 US-$
IoT device characteristics
Characteristics of IoT devices

- Constrained
- Slow
- Low power (typ. 3.3V)
- Limited (analog/digital) I/O, I²C
- Small (local) storage
- Serial (programming) interface
- Networking capabilities
Sensors
IoT Programming ...
Programming model

Application

Firmware

Bootloader

CPU core
Default Firmware – Espressif AT firmware¹

- Modem Emulation
- Serial I/O
- AT² command set
- Needs ‘minicom’ or ‘picocom’

² https://en.wikipedia.org/?title=AT_Commands
Example – Espressif AT firmware

AT+GMR
AT version: 0.60.0.0 (Jan 29 2016 15:10:17)
SDK version: 1.5.2 (7e0e54f4)
Ai-Thinker Technology Co. Ltd.
May 5 2016 17:30:30
OK
AT+CWMODE_CUR=3
OK
AT+CWLAP
+CWLAP:(3,"Staycast-Device", -84, "e8:1d:a8:4c:b8:58", 1, -24, 0)
+CWLAP:(8,"@Hyatt_WiFi", -85, "34:8f:27:2c:f3:a8", 1, -16, 0)
+CWLAP:(3,"Staycast-Device", -84, "34:8f:27:6c:f3:a8", 1, -16, 0)
+CWLAP:(9,"Hyatt TV.b", -76, "fa:8f:ca:8a:04:57", 1, -17, 0)
+CWLAP:(3,"RWays", -93, "58:93:96:df:ce:b9", 1, -44, 0)
+CWLAP:(9,"@Hyatt_WiFi", -94, "e8:1d:a8:8d:69:68", 1, -21, 0)
+CWLAP:(8,"@Hyatt_WiFi", -86, "e8:1d:a8:0c:ee:88", 1, -26, 0)
+CWLAP:(3,"Staycast-Device", -84, "e8:1d:a8:4c:ee:88", 1, -26, 0)
+CWLAP:(8,"@Hyatt_WiFi", -90, "e8:1d:a8:0d:05:88", 1, -27, 0)
+CWLAP:(3,"Staycast-Device", -90, "e8:1d:a8:4d:05:88", 1, -26, 0)
+CWLAP:(0,"Renaissance_GUEST", -50, "f0:b0:52:3e:8d:28", 6, 32767, 0)
+CWLAP:(3,"RWays", -47, "f0:b0:52:fe:8d:29", 6, 32767, 0)
+CWLAP:(3,"Renaissance_Other", -48, "f0:b0:52:be:8d:2a", 6, 32767, 0)
+CWLAP:(8,"SUSECON2019", -49, "f0:b0:52:fe:8d:2a", 6, 32767, 0)
+CWLAP:(3,"RWays", -81, "34:8f:27:5f:63:8a", 6, -21, 0)
Alternative firmwares - microPython¹

“MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.”

- Analog/Digital IO
- UART, SPI, I2C, CAN bus support
- TCP/IP, HTTP(S) networking

¹ https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html
Example - microPython

>>> help()
Welcome to MicroPython!

For online docs please visit http://docs.micropython.org/en/latest/esp8266/.
For diagnostic information to include in bug reports execute 'import port_diag'.

Basic WiFi configuration:

```python
import network
sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
sta_if.scan()  # Scan for available access points
sta_if.connect("<AP_name>", "<password>")  # Connect to an AP
sta_if.isconnected()  # Check for successful connection

# Change name/password of ESP8266's AP:
ap_if = network.WLAN(network.AP_IF)
ap_if.config(essid="<AP_NAME>", authmode=network.AUTH_WPA_WPA2_PSK, password="<password>")
```

Control commands:
CTRL-A  -- on a blank line, enter raw REPL mode
CTRL-B  -- on a blank line, enter normal REPL mode
CTRL-C  -- interrupt a running program
CTRL-D  -- on a blank line, do a soft reset of the board
CTRL-E  -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)

```python
>>> import network
```

```python
>>> sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
#5 ets_task(4020f4d0, 28, 3fff9e30, 10)
```

```python
>>> sta_if.scan()  # Scan for available access points
```

```python
[(b'@Hyatt_WiFi', b'\$y\x05\x08\x08', 1, -72, 0, 0), (b'Staicast-Device', b'\$yE\x84\x08', 1, -71, 3, 0), (b'Staicast-Device', b'4\x8f
```
Alternative firmwares

- Espruino¹ – JavaScript
- NodeMCU² - Lua
- ESPBasic³ – Basic
- and more ...

¹ https://www.espruino.com
² https://nodemcu.readthedocs.io/en/master
³ https://www.esp8266basic.com
Firmware update

- Requires serial interface (USB/Serial)
- Download new firmware
  - python\{2,3\}-esptool
  - Wire for “programming” (GPIO0 to GND)

```
esptool.py --port /dev/ttyUSB0 erase_flash
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_size=detect 0 firmware.bin
```
No Linux!
Linux in IoT

The Service Delivery Edge
Linux in IoT (2)

- Software development
- Cross compilation
- Debugging
- Terminal
Cross compilation
Software development

- Edit
- Compile
- Link
- Run

No platform change
Cross development

- Edit
- Cross-Compile
- Cross-Link
- Upload
- Run
Cross toolchain

- Cross-GCC
- Cross-Headers
- Cross-Binutils
  - Assembler, Linker
- Cross-Libraries
Cross toolchain

- *devel: gcc* on build.opensuse.org
- Arm, Avr, Power, Risc-V, …

- Kudos to the maintainers!
Arduino
Arduino MKR WiFi 1010

- ARM (M0) core, 48 MHz
- ESP 32 WiFi module
- 256 kB Flash, 32 kB RAM
Arduino IDE

- Integrated Development Environment
- Java application
- Edit, Compile, Upload
- Processing, C
- Huge range of supported hardware
- Huge software ecosystem
- Extensible
/* Simple ESP8266 test. Requires SoftwareSerial and an ESP8266 that's been 
flashed with recent 'AT' firmware operating at 9600 baud. Only tested 
w/Adafruit-programmed modules: https://www.adafruit.com/product/2292 

The ESP8266 is a 3.3V device. Safe operation with 5V devices (most 
Arduino boards) requires a logic-level shifter for TX and RX signals. 

#include <Adafruit_ESP8266.h>
#include <SoftwareSerial.h>

#define ESP_RX  2
#define ESP_TX  3
#define ESP_RST 4

SoftwareSerial softser(ESP_RX, ESP_TX);

// Must declare output stream before Adafruit_ESP8266 constructor; can be 
// a SoftwareSerial stream, or Serial/Seriali/etc. for UART.
Adafruit_ESP8266 wifi(&softser, &Serial, ESP_RST);
// Must call begin() on the stream(s) before using Adafruit_ESP8266 object.

#define ESP_SSID  "SSIDNAME"  // Your network name here
#define ESP_PASS  "PASSWORD"  // Your network password here

#define HOST  "www.adafruit.com"  // Host to contact
#define PAGE  "/testwifi/index.html"  // Web page to request
#define PORT  80  // 80 = HTTP default port
<table>
<thead>
<tr>
<th>Type</th>
<th>All</th>
<th>Topic</th>
<th>All</th>
<th>esp</th>
</tr>
</thead>
</table>

**Arduino Uno WiFi Dev Ed Library by Arduino**
This library allows users to use network features like rest and mqtt. Includes some tools for the ESP8266. Use this library only with Arduino Uno WiFi Developer Edition.

[More info](#)

**Esplora Built in by Arduino Version 1.0.4 INSTALLED**
Grants easy access to the various sensors and actuators of the Esplora. For Arduino Esplora only. The sensors available on the board are: 3-Axis analog joystick with center push-button, 4 push-buttons, microphone, light sensor, temperature sensor, 3-axis accelerometer, 2 TinkerKit input connectors. The actuators available on the board are: bright RGB LED, piezo buzzer, 2 TinkerKit output connectors.

[More info](#)

**ACAN2515 by Pierre Mollinaro**
**Driver for MCP2515 CAN Controller**
Arduino CAN network driver for the MCP2515 CAN Controller. Compatible with ACAN, ACAN2517, ACAN2517FD libraries. The default configuration enables to receive all the frames. User can easily define reception filters. Runs on ESP32 from version 1.1.2.

[More info](#)

**ACAN2517 by Pierre Mollinaro**
**Driver for MCP2517FD CAN Controller (CAN 2.0B mode)**
This library is an Arduino CAN network driver for the MCP2517FD CAN Controller, in CAN 2.0B mode (CANFD is not handled by this driver). Compatible with ACAN, ACAN2515, ACAN2517, ACAN2517FD libraries. Default configuration sends and receives any frame - no default filter to provide. Reception filters (up to 32) can be easily defined. Runs on ESP32 from version 1.1.0.

[More info](#)

**ACAN2517FD by Pierre Mollinaro**
**Driver for MCP2517FD CAN Controller (CAN FD mode)**
This library is an Arduino CAN network driver for the MCP2517FD CAN Controller, in CAN FD mode. Compatible with ACAN, ACAN2515, ACAN2517 libraries. Default configuration sends and receives any frame - no default filter to provide. Reception filters (up to 32) can be easily defined. Compatible with ESP32 from version 1.1.0.

[More info](#)

**AcoRoutine by Brian T. Park**
A low-memory, fast-switching, cooperative multitasking library using stackless coroutines on Arduino platforms. Supported macros include COROUTINE(), COROUTINE_BEGIN(), COROUTINE_YIELD(), COROUTINE_DELAY(), COROUTINE_WAIT(), COROUTINE_LOOP(), and COROUTINE_END(). Verified to work on AVR (Nano, UNO, etc), Teensy ARM, ESP8266 and ESP32. Unit tested using AUnit.

[More info](#)

**ACROBOTIC SSD1306 by ACROBOTIC**
**Library for SSD1306-powered OLED 128x64 displays**
This is a library for displaying text and images in SSD1306-powered OLED 128x64 displays; includes support for the ESP8266 SoC.

[More info](#)
Arduino IDE in openSUSE

- CrossToolchain:avr on build.opensuse.org
- Package ‘Arduino’, command ‘arduino’
- Compiled from source
- Toolchain from devel:gcc
Conclusion
IoT programming with openSUSE

- Low initial investment
- Sensors, sensors, sensors
- Many programming languages, libraries, and examples
- Simple

- openSUSE comes with ‘batteries included’
  - Low level tooling
  - Build toolchain
  - IDE
License
This slide deck is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license. It can be shared and adapted for any purpose (even commercially) as long as Attribution is given and any derivative work is distributed under the same license.

Details can be found at https://creativecommons.org/licenses/by-sa/4.0/

General Disclaimer
This document is not to be construed as a promise by any participating organisation to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. openSUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for openSUSE products remains at the sole discretion of openSUSE. Further, openSUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All openSUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE LLC, in the United States and other countries. All third-party trademarks are the property of their respective owners.

Credits
Template
Richard Brown
rbrown@opensuse.org

Design & Inspiration
openSUSE Design Team
http://opensuse.github.io/branding-guidelines/