Towards an Open Framework for Home Automation Development

Pham Huu Dang Nhat
Department of Computer Engineering
Ho Chi Minh City University of Technology
23 November 2015
Outline

- Introduction
- Conventional Approaches
- Proposed Framework
- Prototype
- Demo
Introduction

General purpose computing platforms

Sensors & Actuators

Current problems
Zigbee, Z-Wave, etc.
Inconvenient in connecting to the Internet

Conventional Approaches

Support only a specific set of end devices

Automatic operations
Only system administrator/maintainer could configure
Towards an Open Framework for Home Automation Development

Based on 6LoWPAN (IPv6)

General framework for device drivers layer
Support various kinds of end-devices & **runtime configuration**.

**Our Approaches**

- **6LoWPAN**
  - IPv6-based Low-power Wireless Personal Area Networks

Both system administrators and users could configure scenarios for automatic operations.

Remotely control end devices & setup scenarios.
System Architecture

- Control Center (CC)
- Host Board (HB)

- All data and control decisions are centralized → CC.
- Each HB could host many end points (OS’s threads).
- End point → any supported end device. (Hot plugging)
- Shell (CLI) via USART → communicate with CC, HB.
Towards an Open Framework for Home Automation Development

- Device Drivers Layer

  **Device instance**

  - Switch/Button
  - PIR
  - On/Off Bulb
  - Buzzer
  - Temperature Sensor
  - Gas Sensor
  - Light Sensor
  - RGB led
  - On-Off Input
  - On-Off Output
  - Dimmer
  - ADC Sensor
  - Servo
  - Dimmable Bulb

  **Device class**

  - GPIO device class
  - ADC device class
  - PWM device class

  **Inheritance**

  - GPIO
  - ADC
  - PWM

  **Platform**

- Adopt OOP & Inheritance model to achieve
  - Support many different kinds of end devices.
  - Runtime configuration

- Higher layers don’t access directly to hardware. \(\rightarrow\) **reusability**.

- C++ on Cortex-M microcontrollers.
Runtime Configuration

- **Hot plugging** new end device.
- **Start running** new end device using **Shell commands**.
  - What is the newly plugged in end device?
  - How does it work?
  - Where does it placed?
    - Which GPIOs?
    - Which ADC/TIM/… channels? etc.

<table>
<thead>
<tr>
<th>Command</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>senadc</td>
<td>-e [EP_id] -s [sensing condition]</td>
</tr>
<tr>
<td></td>
<td>-p [port] -n [pin] -a [adc] -c [channel]</td>
</tr>
<tr>
<td></td>
<td>-t [equations] -P [parameters]</td>
</tr>
<tr>
<td></td>
<td>-f [noise filtering value]</td>
</tr>
<tr>
<td></td>
<td>-u [alert under bound] -o [alert upper bound]</td>
</tr>
</tbody>
</table>
Scenarios

- **Scenarios**: default scenario, user scenarios.
- **Supported conditions**:
  - The value of a device is =, <, <=, >, >= to a threshold.
  - The value of a device was changed.
  - The value of a device was changed over a threshold.
  - In a specific time period.
  - In a specific time period every day.
- **Supported action**:
  - Adjust the value of an end-device.
Prototype

- Up to **25 rules** for each scenarios, **3 inputs** and **3 outputs** for each rule.
- **28 end devices** of **15 different types**.
- **Over 20 shell commands** to manage network, scenarios, device configuration.
Towards an Open Framework for Home Automation Development

Android Application

- Remotely control end devices, manage (create/delete/rename) multiple scenarios and zones (rooms).
Available Resource

• Source code: freely available on Github under LGPL v2.1 license at https://github.com/dangnhat/HA-project

Analog sensor example

- \( Y = F_n \left( F_{n-1} \left( \ldots \left( F_1(X) \right) \right) \right) \)
- Supported equation types:
  - \( Y = a \cdot X^b + c \)
  - \( Y = \frac{1}{a \cdot X + b} + c \)
  - Look-up table.
- Combination of these functions.
Analog sensor example

- Final output: \( L \text{ (lux)} = F_2(F_1(V_{adc})) \)
- \( Rs = F_1(V_{adc}) = \frac{3.3 \times R_1}{V_{adc}} - R_1(\Omega) = \frac{1}{0.303 \times V_{adc}} - 1 \text{ (k}\Omega) \)
- \( L = F_2(Rs) = 10^{2.75} \times Rs^{-1.25} \text{ (lux)} \)
  - \( L \) : light intensity (lux)
  - \( Rs \) : photo-cell’s resistor

Figure 8. Schematic (left) and characteristic (right) of a photoresistor