SUTRA — A workflow for documenting signals

Pavneet Arora

Abstract

Modern home design increasingly emphasizes electronic signals to carry not only data and entertainment, but also basic comfort and security functions throughout the house. The connected devices along these signal paths are almost never static, but are replaced or augmented regularly depending on the homeowners' needs or whims. This proliferation of signals, often implemented on an *ad hoc* basis, creates difficulties when it comes to effective implementation, but even more importantly, for diagnoses when things inevitably go wrong. The need for some form of documentation is clear; but what representations make sense both for capturing the implementation details as work progresses, and then as an aid in the discovery phase of diagnosis? SUTRA is a documentation workflow that builds upon the earlier work introduced in the YAWN framework. It relies on YAML for data representation, the Ruby language for processing, and, in this case, uses ConTEXt's natural tables mechanism to collate and present this information in a useful way to round out YAWN's model-view-controller projection onto the problem space.

1 Introduction

The humble abode has evolved from an unserviced shack bereft of either electricity or plumbing to an inter-connected set of systems — power, heating and cooling, electronic controls — that wind their way behind walls and through floor joists. Their scope is both dazzling and staggering in its complexity, and yet we assume they will instantaneously and reliably function.

On top of such basic services are layered audio and video distribution throughout the house, surveillance, alarm, data networking, and telephony. Not to mention that all of these services, primary and ancillary, are also being pushed outside into the garden and all the way to the lot perimeter. And these are just the elements that are the responsibility of the homeowner. They are in turn "lit" by the utility providers that have interfaces to the house systems.

Control and automation add yet another layer in an effort to bring these disparate systems under some semblance of manageability for the homeowner, but naturally each new layer adds even more complexity to the underlying infrastructure.

Plugging into these various systems can be fixed wire devices such as traditional thermostats, alarm keypads, computers, etc., as well as transient devices such as mobile phones, tablets, wand remotes, and others — both trusted and known, as well as interlopers brought in by guests who might be given temporary access.

Basic design documents used for construction or renovation almost always excludes any information about several of the layers mentioned above, as they do not come under the governance of building permits, or will leave the details of the implementation to the supplying vendor in a form of *late-binding* where details are expected to be discussed with the homeowner at some time in the future. As a result, it is common to go to tender for a generic "pre-wire" package with trust placed in the cabling vendor, often lowest-cost, without much attention paid to what this wiring will afford in terms of connectivity. Once the equipment selection is complete it is left to the implementation team to install the equipment using whatever wiring they find.

It is the rare consumer that even bothers to understand what is underneath the surface, satisfied as they are to interact through the various interfaces. And rightly so, as it should not concern them beyond being able to access the service they seek, any more so than a driver getting into their car and expecting it to start, move and turn under their command without requiring to know if their steering is driven hydraulically or electrically. Unfortunately, it is the equally rare installer that takes care to note down what they have done on site so that there might be an entry point of understanding in the future for either their next visit, or that of the next technician.

The end result is that there soon grows a vast gap between the actual as-built state of the wiring and the systems built upon it, and what is known about that state. This lack of information makes the systems fragile and susceptible to failure, a most unwelcome outcome given their increasing complexity and the utter dependence of the homeowner on them.

2 A specific example to illustrate the problem

To illustrate the rise in complexity of signal propagation in a residential setting, let us consider, in isolation, the example of music playback. The arrival of consumer stereos in the 1960s soon resulted in a new essential fixture in many homes, alongside televisions. However, even its more complex forms, e.g., a pre-amplifier driving a pair of mono-block power amplifiers which, in turn, would drive a pair of loudspeakers now seems laughable in its simplicity. A cursory visual inspection would suffice to make sense of which wire went where, and which did what.

The integrated amplifier which followed, incorporating the pre-amplifier and the power-amplifier in the same chassis, and then its successor the audio receiver which also included a tuner, simplified the wiring further, now reduced to just the speaker wiring and, perhaps, wiring from an additional source such as a turntable.

Now consider the modern audio/video receiver, which is tasked with not just taking source audio signal to speaker, amplifying it along the way, but also takes responsibility for video switching, sometimes transforming and scaling the video signal, processing digital audio signals into multi-channel audio, handling output to multiple zones while allowing repurposing of its amplifier channels to be assigned to these secondary zones. Accepted video signals can be selected from amongst a variety of interfaces, both analogue, e.g., component video, and digital, e.g., HDMI. The control interface handled by the receiver almost always includes infrared, RS-232, 12V trigger, and increasingly IP-based control. Add to this the variety of source equipment such as set top boxes, dedicated media players, games consoles, mobile devices, and dare I say, for some die-hards, a turntable and CD-player. Source signals need not even be local to the unit, but may also simply be connected via the data network. All of this capability and complexity — just for a single device! Now let's consider that this is just one such unit amongst many in a house, and I think that one can soon appreciate the need for both a way of capturing this information, and equally for representing it in a way that is both digestible and easy to navigate.

3 How to capture the information?

3.1 What information is needed?

In order to look at methods of information capture we must first understand just what information would be useful. The emphasis on the word *signals* in the present title gives a clue, in that what is of primary importance are the signals themselves, i.e., the connections between the different components that are the conduits for the signals. Components may act as either the source, destination, or intermediate processor of a signal but it is the connections that reveal the topology of the implementation, and are essential in understanding how the system behaves overall and how pieces within the system interact.

Of course, identifying equipment is also necessary, but we can think of equipment as a set of connection termini that happen to be physically grouped together.

The attributes of each connection that I feel essential to reveal its purpose are:

Identity Name of the connection. This could be just the physical port name on the component chassis or the assigned port name in the component's software configuration. Using the example of the A/V receiver mentioned earlier, this could be, for example, *Front Right Speaker* or *HDMI 1*.

Interface The type of physical connection.

Intent What is the purpose of the signal that the connection is carrying?

Implementation Details about the connection. An example might be: in connecting to the infrared interface, we might be using the *blue* and *green* pairs for *signal* and *ground* respectively of the pulled UTP cable between the location IR connecting block and the mouse emitter. Such implementation details should be captured as well.

Interaction To what component(s) does the signal travel? And to which port(s) of that destination equipment does it connect?

I call this set *5i* in counterpoint to the exclamation of frustration, "Ay-Ay-Ay-Ay-Ay", that usually accompanies confronting an undocumented equipment closet under time pressure, trying to fix a problem that is unknown in scope.

3.2 Layers

Another aspect that is important to note is that of *layers* or *buses*. From the audio example above, it is clear that an individual component can operate along different buses and may be involved in transforming a signal so that it moves from one bus to another. So the *interaction* attribute needs to encompass the movement of a signal along a single or multiple layers.

Sharing A connection operates within a single layer or bus.

Traversal A connection may switch or split a signal onto multiple layers. An example of this would be an HDMI audio extractor which decomposes the combined signal into its requisite video and digital audio components.

Isolation The component works in isolation. An example of this might be a power line conditioner that is IP-enabled to carry out scheduled or triggered reboot functions, but on the other hand it can also be left as a standalone unit.

3.3 A simple method to capture signals information

For the information capture I took inspiration from Kent Beck's and Ward Cunningham's seminal work

on CRC (*classes*, *responsibilities*, and *collaborators*) cards [7].

I rely on 3"x 5" index cards, and each time I or my team would come across a piece of equipment we would jot down whatever we found out about its connections using the *5i* framework. Our field experience indicated that discovery of a system's function is more effectively done bottom-up for the simple reason that there are rarely any maps that would facilitate a top-down approach. In other words, when *step-wise refinement* is not an option then the alternative is to turn to *step-wise abstraction* [2]. For that reason, we chose to put our attention on a single piece of equipment and its *outgoing* connections.

Our focus on outgoing connections exclusively came from our belief that much of the intent of the connection is revealed from its source. Once this informal information capture was processed, the software could resolve and illustrate the requisite connections on the corresponding input ports of the destination equipment.

4 How to present information about signals

When seeking to map out electronic connections, it is an easy assumption that a schematic is the natural and best representation. But is it always appropriate; moreover, is it even an effective representation for the domains described in this paper? There are several problems associated with schematics:

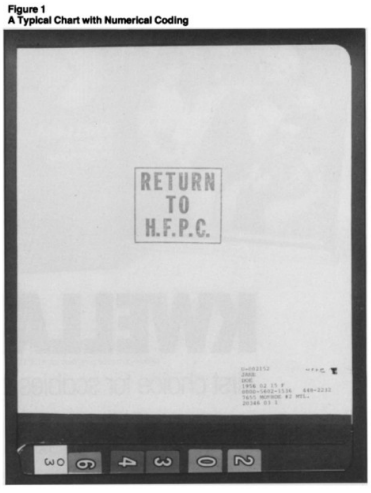
- They work best when there is a close mapping between the physical wiring layout and the logical design, e.g., a circuit board. When the critical information relates to the logical connection, there is a great deal of wasted real estate on the wire traces, all of which distracts from the essential information about the connection, and also consequently relegates this information to distant appendices.
- The above point relates to their use when the design goes through repeated refinements, and the results of this "stable" design are then produced in quantity. Unfortunately, almost every installation of the nature described in the paper is unique in that it is built up on an *ad hoc* basis, over considerable time.
- As the number of connections grows, the required size to present the layout soon becomes unwieldy. The schematic for the RaspberryPi, a relatively simple device, requires five pages, and this doesn't begin to include the underlying ARM Application Processor details.
- They also don't easily allow us to distinguish between the *physical* and *logical* location of components.

For some years we made a concerted effort to use Dia drawing software [5] under GNU/Linux to capture our own wiring installations. However, we struggled to keep our diagrams updated, and we still didn't have a methodology in place to quickly capture the information about the connections we found in the field. Upon analysis, the great shortcoming of this type of tool is that one spends an inordinate amount of time on layout, when one really wishes to enter the signal information in a raw form and have a suitable presentation generated for them. It was this need that led to the development of SUTRA.

When seeking archetypes for data representation the following characteristics seemed desirable:

- The ability to group connections into representative components.
- The ability to identify the layers on which a particular component might operate. This would also help in narrowing to a subset of components that operate on a layer or bus of interest during a diagnostic exercise.
- A clear differentiation between the physical location of a component and its logical location, i.e., the zone or area on which it operates.
- When reviewing a component and its constituent connections — both outputs and inputs — the ability to have enough information about the source and destination without having to flip to different parts of the document.
- Additional information about the component, e.g., photographs, or a manufacturer's wiring diagram that identifies terminals not labelled on the component itself.

I have always been intrigued by how physicians file patient records: the racks of colour-coded files [6] from which could be retrieved the file of a single patient or all the files of a family with seeming ease:



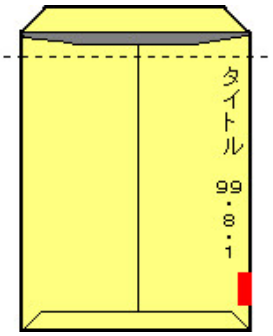


Figure 1: Sample of filing system by Prof. Yukio Noguchi.

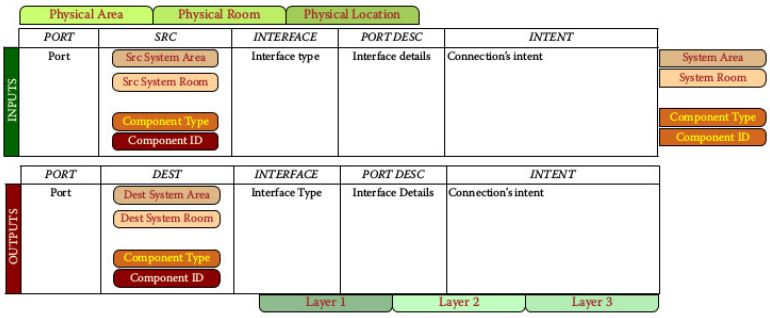


Figure 2: Hand-crafted prototype using ConTEXt's Natural Tables.

More recently, I came across a filing system developed by Prof. Yukio Noguchi [3]; sample image shown in fig. 1.

From these sources, I experimented with designs and eventually settled on a prototype constructed using ConTEXt's natural tables mechanism [4]. An early but representative version of the output is shown in fig. 2. One can see the coloured tabs around the perimeter of the table act as visual aids to classify the component. On the left are indications of input and output. On top is shown the *physical* location by *area*, *room*, *place*. At right is the analogous logical location (which defaults to the physical location unless overridden) and component identity. At bottom right are the layers on which the component operates.

The word *sutra* (सूत्र) is the Sanskrit word for a string. It seemed an apt name for our workflow since, unlike in a schematic where the physical layout dominates the presentation, it instead places emphasis on the source and destination, i.e., the connection. Of course, a large collection of strings can also invoke the image of a complex tapestry, which applies when reviewing large scale wiring diagrams. So the acronym SUTRA stands for its constituent components:

- Signals
- Unmasked using
- Table
- Representations with
- Annotations

5 What constitutes a signal?

While SUTRA's development initially arose from demands of electronic signals, its application has been successfully extended to domains where the definition of a signal can be a hybrid of connection types. There is no limitation within the workflow as to what constitutes a signal, and part of the appeal of its design is that it concerns itself only with connections and components, however the user might need to interpret those.

6 SUTRA example

For this example I have purposely chosen a more generalized application of SUTRA. Upon my return from the 2013 TUG conference, I arrived to find our hydronic heating system misbehaving. Parts of it worked, but the in-floor heating did not, in spite of the boiler functioning as expected and a continuous call for heat from the thermostats.

To reinforce the point made earlier of the unwitting homeowner, I myself had not paid much attention to the details when the system was installed, and the installing technician could not recall the specifics of the installation. So I was faced with the ubiquitous discovery and diagnostics steps taken in such situations.

As is common, this system operates on multiple layers:

1. Electronic low-voltage. In reality this represents several layers. It is used for the controls, and in some cases the sensors. Thermostats, water temperature loop sensors, ambient air sensors, etc., all use low-voltage signals, but they aren't uniform: both AC and DC are used, as are dry contacts feeding relays.
2. Line (mains) voltage. In some cases pumps are controlled indirectly by connection to a control board on the pump, e.g., variable speed pumps, while in others they are controlled directly by mains voltage.
3. Water. This drives domestic hot water, in-floor heating, towel warmers, area radiators, and a heat exchanging air handler in staged fashion.

So SUTRA came to the rescue.

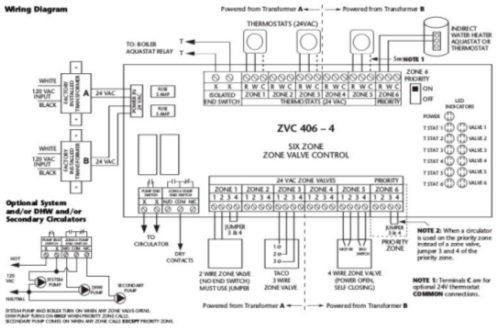
The implementation uses YAML for the input representation, following YAWN [1]. Here is some of the YAML for defining a single thermostat in the system:

```
:locAreas:
 - :locArea:
  :locAreaID: F1
  :locRooms:
   - :locRoom:
    :locRoomID: Master Bedroom
    :components:
     # THERMOSTATS
     - :component:
      :componentID: Tekmar 508
      :componentType: Thermostat
      :locPlaceID: Bottom of Steps
       (East Wall)
      :layers:
       - :layer: thermostat control
      :connections:
       - :connection:
        :type: thermostat
        :from:
         :port: 4-wire control
        :to:
         :sysAreaID: F0
         :sysRoomID: >
          Mechanical Room
         :componentID: Taco ZVC406
         :port: 4-wire control
         :portDesc: >
          Thermostat/Zone 1
        :desc: >
         F1 Master Bedroom in-floor
         heating
```

In a multi-pass operation, SUTRA builds a database of connections, and then resolves forward references to the destination components. In this case, the destination of the thermostat connection is the "Taco ZVC406" zone valve control.

The SUTRA output is shown in fig. 3.

With the `--showresources` flag, SUTRA includes any ancillary material associated with a component. Here, the wiring diagram for the zone valve control is included in the generated output:



To bring our story to a resolution: in the end it was discovered that the thermostat in the hot water tank had failed open, resulting in a continuous call for heat. Since this is the primary circuit, all hot water generated by the boiler was directed to the heat exchanger of the hot water tank, with none for the floor heating.

7 ConTEXt considerations

Turning now to the ConTEXt implementation of the output we have seen, here are a few of the cases that the generated code must consider:

- How to handle the case when there are no output connections, only input connections? The layers operated upon need to shift from the bottom of the output connections to the bottom of the input connections, if any.
- How to handle the case when there are no input connections, but there are output connections? The component's logical location and identity would shift to the output connections.
- How to handle the case when there are neither input nor output connections? An empty table needs to be generated so that the tabs can be affixed to it.

Some illustrative fragments of the ConTEXt code that generated fig. 3 are shown in fig. 4.

8 Conclusions

The problem of documenting system-wide signal paths in the construction industry can be overwhelming. The workflow SUTRA, built upon the YAWN framework [1], and utilizing ConTEXt's Natural Tables mechanism, offers a viable option to make this problem tractable.

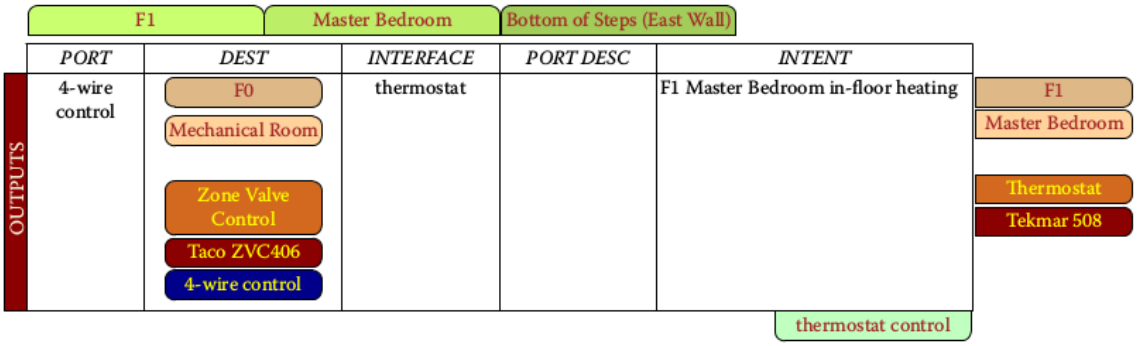


Figure 3: SUTRA table for Tekmar 508 thermostat.

```
\setupTABLE[r][first][leftframe=off,...
\setupTABLE[c][1][align={middle,lohi}]
...
\framed[frame=off,location=top]{
\bTABLE
\bTR
  \bTD
  \eTD
  \bTD
  ...
  \framed
    [width=1.5in,
    align={middle},
    corner=03,
    background=color,backgroundcolor=darkolivegreen1,
    foreground=color,foregroundcolor=brown]
    {\switchtobodyfont[9pt] F1}%
  ...
    {\switchtobodyfont[9pt] Bottom of Steps (East Wall)}
  \eTD
\eTR
\bTR [align=middle] ...
  \bTD
    {\em PORT}
  ...
    {\em INTENT}
  \eTD
\eTR
\bTR
  \bTD [nr=1,background=color,backgroundcolor=darkred,
    foreground=color,foregroundcolor=beige]
    \framed
    [frame=off]
    {\rotate[rotation=90]{OUTPUTS}}
  \eTD
  \bTD
    4-wire control
  \eTD
  \bTD
    \framed
    [width=1in,
    align={middle},
    corner=00,
    background=color,backgroundcolor=darkred,
    foreground=color,foregroundcolor=yellow]
    {Taco ZVC406}
    ...
  \eTD
...
  \bTD
    {F1 Master Bedroom in-floor heating}
  \eTD
  \bTD
    \framed
    ...
    {F1}
  \eTD
  ...
\eTR
...
  \bTD
    [nc=5,
    align=flushright,
    offset=0in]
  \dontleavehmode
  \framed
    [width=1.25in,
    corner=01,
    background=color,backgroundcolor=darkseagreen1,
    foreground=color,foregroundcolor=brown]
    {\switchtobodyfont[9pt] thermostat control}
  \eTD
  \bTD
  \eTD
\eTR
\eTABLE}}
```

Figure 4: Abridged ConTEXt Natural Tables source for fig. 3.

References

[1] Pavneet Arora. YAWN — A TEX-enabled workflow for project estimation. *TUGboat*, 33(2), October 2012. http://tug.org/TUGboat/tb33-2/tb104arora.pdf.

[2] Ramesh Chandra, AT&T Bell Laboratories. Private conversation.

[3] Dave Gray. The Noguchi filing system. http://communicationnation.blogspot.ca/2005/12/noguchi-filing-system.html, December 2005.

[4] Hans Hagen. Natural tables. http://wiki.contextgarden.net/TABLE.

[5] Alexander Larsson et al. http://live.gnome.org/Dia

[6] Cheryl Levitt. A patient record-filing system for family practice. *Canadian Family Physician*, 34, October 1988. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2219171.

[7] Nancy M. Wilkinson. *Using CRC Cards — An Informal Approach to Object-Oriented Development.* SIGS Books, New York, 1995.

◇ Pavneet Arora
pavneet_arora (at) bespokespaces dot com
http://blog.bansisworld.org