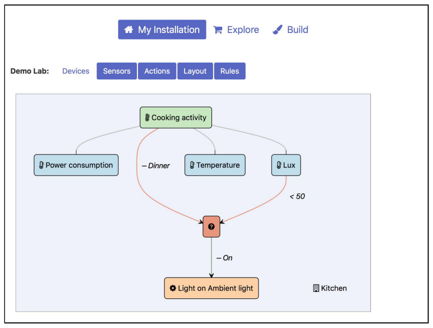
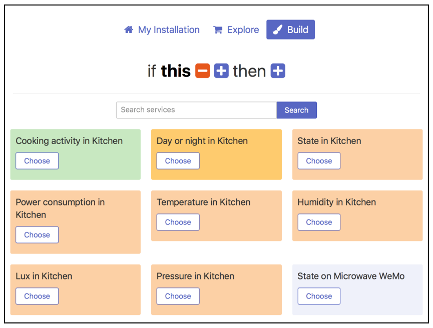
CharIoT: An end-user programming environment for the IoT

Matúš Tomlein
Aarhus University
Aarhus, Denmark
matus@cs.au.dk

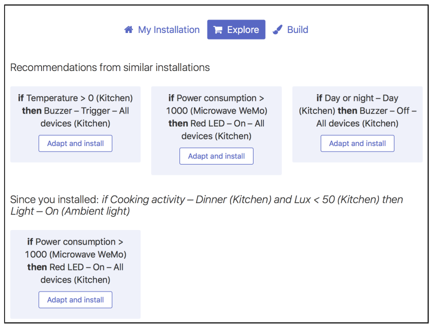
Sudershan Boovaraghavan, Yuvraj Agarwal, Anind K. Dey
Carnegie Mellon University
Pittsburgh, PA, USA
sud335@gmail.com, [yuvraj, anind]@cs.cmu.edu



((a)) A graph-based overview of the installation showing entities in the installations as nodes, grouped by their placements and linked by their relations.



((b)) An end-user programming interface inspired by IFTTT with support for describing higher-level events using virtual sensors.



Recommendations from similar installations

if Temperature > 0 (Kitchen) **then** Buzzer – Trigger – All devices (Kitchen)
Adapt and install

if Power consumption > 1000 (Microwave WeMo) **then** Red LED – On – All devices (Kitchen)
Adapt and install

if Day or night – Day (Kitchen) **then** Buzzer – Off – All devices (Kitchen)
Adapt and install

Since you installed: *if Cooking activity – Dinner (Kitchen) and Lux < 50 (Kitchen) then Light – On (Ambient light)*

if Power consumption > 1000 (Microwave WeMo) **then** Red LED – On – All devices (Kitchen)
Adapt and install

((c)) Rules from other installations recommended using content-based and collaborative filtering approaches.

Figure 1: Three main parts of the interface: overview of the installation, building automation rules and exploring shared rules.

ABSTRACT
Despite the breadth of related work, enabling end-users of varying technical ability to leverage sensor data to control their Internet of Things (IoT)-enabled installations remains a challenge. This work proposes a unified interface that provides three building blocks to support the end-user configuration of IoT environments: capturing higher-level events in the installation through virtual sensors, construction of automation rules with a visual overview of the current configuration and support for sharing configuration between end-users using a recommendation mechanism.

ACM Classification Keywords
D.1.7 Software: Programming Techniques: Visual Programming

Author Keywords
End-user programming; visual programming.

INTRODUCTION
Supporting end-users in making use of the context data generated by the numerous sensors in smart environments and use it to enable actuation using heterogeneous devices is a task that provides challenges on multiple levels. CharIoT builds on the GIoTTO software stack [1] and provides a user information for the configuration of IoT-enabled installations that addresses three open challenges: supporting end-users in making use of raw sensor data, providing overview of the current configuration within the smart home and supporting users to share configuration among each other.

IoT '17 October 22–25, 2017, Linz, Austria
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5318-2/17/10.
DOI: https://doi.org/10.1145/3131542.3140261

VIRTUAL SENSORS
Virtual sensors are components of the GIoTTO stack that can capture higher-level patterns from raw sensor data. Virtual sensors can range in complexity from simple analyses such as averaging sensor data to complex patterns evaluated through machine learning. For instance, virtual sensors may be used to capture events and states such as window open or someone knocking on a door. CharIoT provides two types of virtual sensors: programmed and demonstrated virtual sensors.

Programmed virtual sensors can provide a more accessible and understandable abstraction over the raw sensor data. For instance, a user may program conditions on the temperature in the room to say that the room is cold if the temperature falls below 20℃, warm if it remains between 20 to 24℃, and hot if the temperature raises above 24℃. In addition to making the data more understandable, such an abstraction enables describing how the data should be processed for different settings (e.g., what does "warm" mean in the living room vs. in the garage).

Demonstrated virtual sensors aim to capture more complex states and events by letting the end-users demonstrate them. To demonstrate an event, such as knocking, a user provides demonstrations both for a knocking state and a time period without knocking. After a few examples (at least three for each state), the system takes data from all available sensors in the installations during the demonstrated periods, extracts and selects relevant features and trains a Random Forest classifier. Afterwards, windows of new data with length corresponding to the length of the demonstrations are periodically sampled and used to predict the current state.

AUTOMATION RULES

Events captured by virtual sensors are useful human-understandable triggers for automating changes in the installation. Our interface provides support for building automation rules that create conditions from the virtual sensors and specify actions to be performed in the installation when those conditions are met (see Figure 1(b)). The interaction is inspired by the popular IFTTT service[1]. In addition, rules may specify multiple conditions and trigger multiple actions. Actions in rules can either refer to specific devices (e.g., turn on lights with certain label) or to locations (e.g., sound alarm on any/all devices in the kitchen).

Once smart environments are configured, their configuration often becomes hidden or forgotten. In CharIoT, we wanted to support the visibility of the automation rules using a graph-based interface (see Figure 1(a)). The graph provides an overview of the sensors and actuators in the installation grouped based on their locations (e.g., rooms or locations within rooms). The automation rules are shown as nodes with labeled links to the sensors and actions that they use. By clicking the nodes in the graph, the user may make changes to the configuration.

SHARING RULES ACROSS INSTALLATIONS

Although user-driven software ecosystems have emerged for other platforms, support for sharing and disseminating software and configurations in IoT is still rudimentary [2]. This is also shown by the duplication of rules on the IFTTT service and the low adoption of sharing [6]. In CharIoT, we aim to support sharing rules among users by recommending relevant rules.

Rules are shared between individual installations of CharIoT that connect to the GIoTTO stack. New rules are recommended based on rules from other CharIoT installations. In the first step, rules from the available installations are matched to the targeted installation. This is done using semantic reasoning, where the conditions and actions of each rule are expressed using a semantic query in the Notation3 language and a semantic reasoner tries to match the conditions and actions to the devices available in the targeted installation. This generates a set of rules applicable for the targeted installation. In the next step, the generated rules are ranked based on two criteria: similarity of the sensor values that the rules build on and similarity of the installations. They are presented along with the recommendation criteria (see Figure 1(c).

[1]https://ifttt.com/

To evaluate the similarity of sensors, they are compared using a content-based approach inspired by [5]. Sensor profiles containing fuzzy sets of sensor values are built for both compared installations. The fuzzy sets represent distributions of values produced by the sensors within a certain time frame (previous day in our case). The fuzzy sets of corresponding sensors are compared using cosine similarity. The similarity gives an indication of how similar the two sensor environments are. The similarities are combined and used to rank the rules.

To recommend rules based on the similarity of installations, an item-to-item collaborative filtering approach is used [3]. New rules are recommended based on other rules that the user configured, similarly to product recommendations on Amazon. Thus, if a user configures a rule similar to a rule configured by another user, additional rules from the other user are recommended.

Since automation rules are based on virtual sensors, when a user decides to install a recommended rule, the underlying virtual sensor needs to be transferred as well. This is simple for programmed virtual sensors. However, for demonstrated virtual sensors, the trained demonstrations need to be transferred as well. Since the demonstrations from the source installation may not always fit the targeted installation, in our ongoing work we are investigating how to decide when a transfer is possible and how to choose the best representation for transfer using transfer learning.

CONCLUSION

CharIoT is an end-user programming environment that enables capturing higher-level events using virtual sensors, provides a graph-based overview of configurations and recommends rules across IoT installations. It was tested using IoT sensors (e.g., TI SensorTag, Mites) and actuators (e.g., Philips Hue, WeMo Switch). CharIoT is implemented using Web technologies (NodeJS, ReactJS) and available on Github [4].

REFERENCES

1. Y. Agarwal and A. K. Dey. 2016. Toward Building a Safe, Secure, and Easy-to-Use Internet of Things Infrastructure. *Computer* 49, 4 (Apr 2016), 88–91.
2. G. Kortuem and F. Kawsar. 2010. Market-based user innovation in the Internet of Things. In *2010 Internet of Things (IOT)*. 1–8.
3. G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (Jan 2003), 76–80. DOI: http://dx.doi.org/10.1109/MIC.2003.1167344
4. Matúš Tomlein. 2017. CharIoT source code. https://github.com/matus-tomlein/CharIoT/. (Sep 2017).
5. C. Truong and K. Römer. 2013. Content-based sensor search for the Web of Things. In *2013 IEEE Global Communications Conference (GLOBECOM)*. 2654–2660.
6. B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman. 2014. Practical Trigger-action Programming in the Smart Home. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 803–812.