Wrocaw University
of Science and Technology

Faculty of Electronics
Embedded Robotics

Intermediate project

Rule engine for distributed home automation system

Author:
Tomasz Bartos

Supervisor:
Witold Paluszyski, Ph.D

Abstract

The goal of the project was to implement rule based engine for distributed home automation system, which can apply custom user logic. Logic can be understood as set of states, which can occur in the system and if all states are matched as expected, then some predefined action will be performed.

Wrocaw 2018

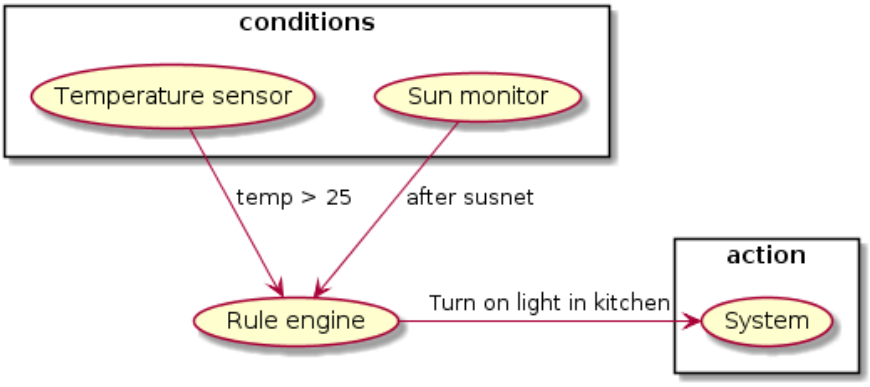

Figure 1: Model of rule engine based on neural network neuron[4]

1 Introduction

Project proposes method of control based on *rule engine*, which allows to customize control logic over home automation system. It is an extension of bachelors thesis *Home automation system based on event driven architecture*[1], it means that most of the assumptions and implementation details have not been described in this report.

Defined logic can be understood by rules, which have following form:

If *temperature in kitchen is below 25 Celsius and* ***if*** *it is after sunset* ***then*** *turn on light in kitchen*
If *sensor in garage detect move and* ***if*** *alarm is armed* ***then*** *raise alarm*

As can be seen, presented set of sample rules is composed from some common parts, which can be used to build user defined logic:

- **Bolded text** represents if-then statements, which is a standard form of rule representation [2][3].
- Underlined text represents states of the system given by set of predefined values and possible actions.

This form of representation gives possibility to divide problem into smaller parts. First one responsible for monitoring state changes in the system and filtering under given conditions, second one responsible for performing user defined actions. Presented model has been shown on Figure 1.

Sample rule defined from comercial *Zipato*[5] home automation system can be observed on Figure 2. As can be seen, in particular example green boxes are representing conditions which should be fulfilled. Orange box is representing action that will be performed when all conditions are correct. Similar method has been used by Polish *Fibaro* group, what has been shown on Figure 3.

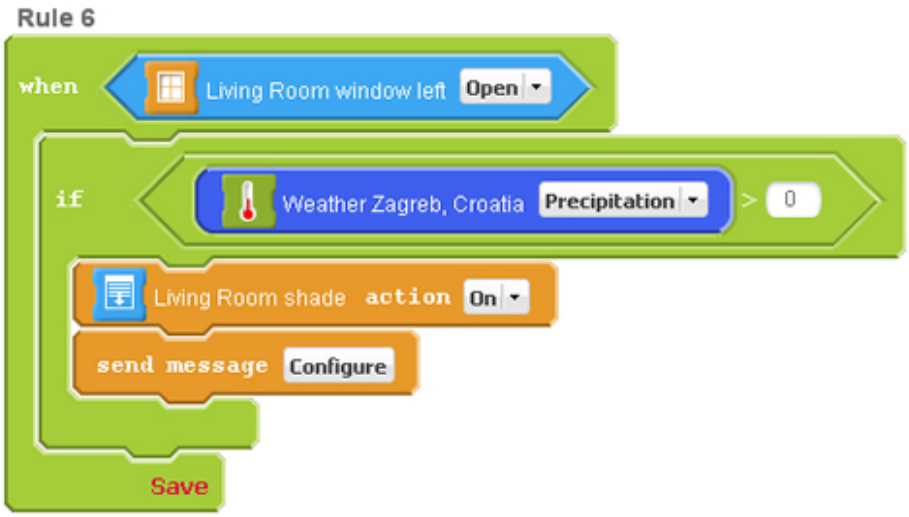

Figure 2: Sample rule from Zipato[5] home automation system[6]

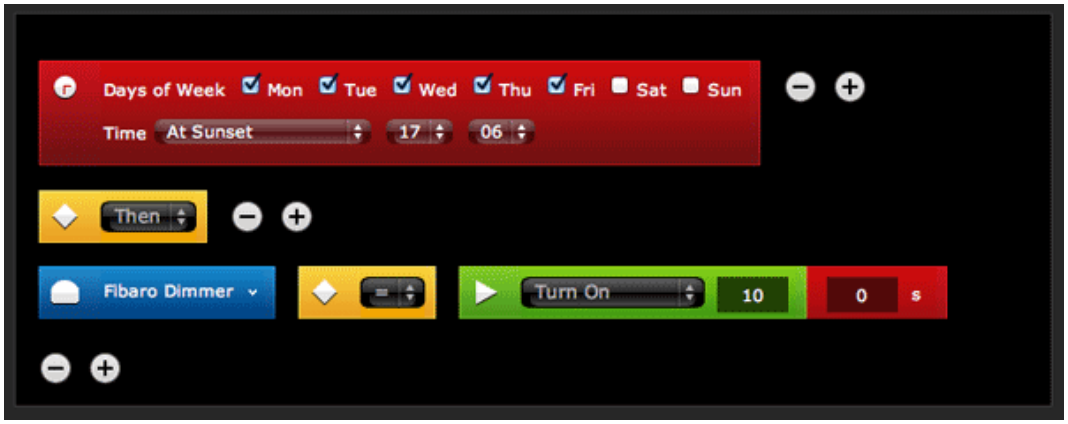

Figure 3: Rule from Fibaro[7] home automation system[8]

2 Implementation

Project has been implemented in C++ programming language as independent binary communicating with other components in micro-services architecture[2]. Following libraries have been used during project realization:

- Fast-RTPS as Real Time Publish Subscribe protocol implementation [9]
- Protocol-buffers as extensible mechanism for serializing structured data [10].

2.1 User interface

User interface has been constructed in order to provide possibility to send commands between users and the system. As an input can be taken JSON formatted string representing payload of predefined Protocol-buffers message, which will be published to other components inside system. Presented format will used only on back-end side, which means that understanding of this format is not needed by the user. View of user interface has

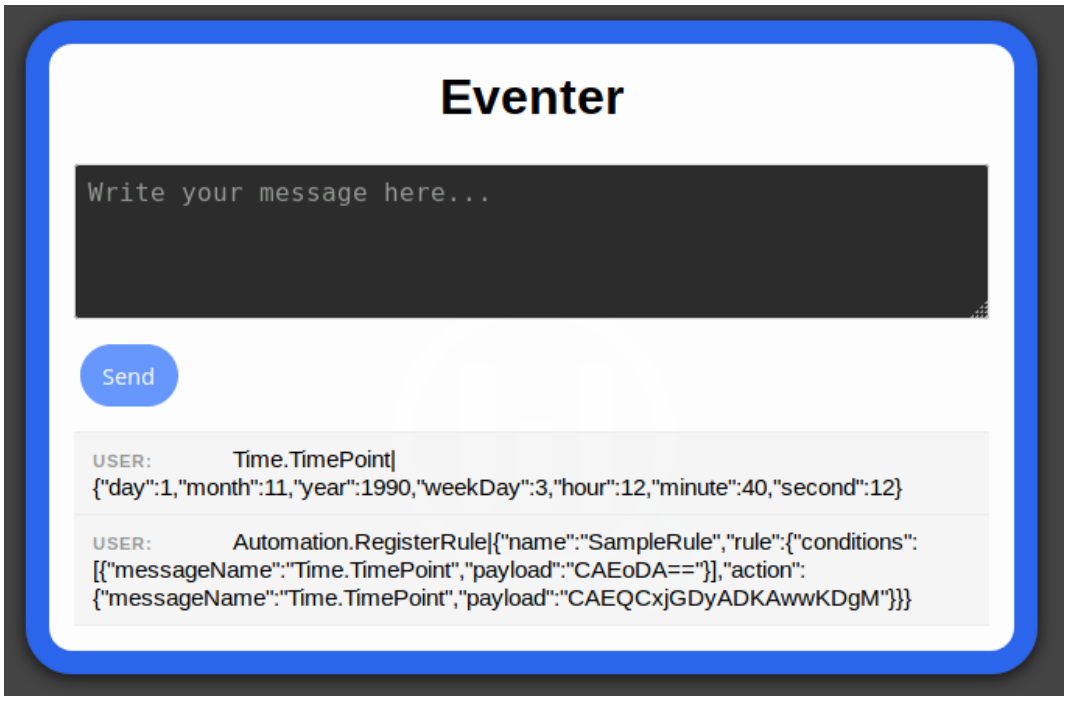

Figure 4: User interface used in the system

been shown on Figure 4. Additional informations about messages have been described in section 2.2.

2.2 Rule engine

Rule engine has been realized as component responsible for:

- Adding, storing and removing user defined rules
- Monitoring state of components requested by user
- Performing user defined actions

Logic of rule engine has been shown on Figure 6. As can be seen, logic has been realized by observation of states received from the system. Data was received with usage of RTPS protocol, where content of messages was a Protobuf serialized structure. Developed component was able to compare data payloads in order to check, if given by user content was a subset of received data, what has been shown on Figure 5. As can be seen, template have this same value of all its fields as received structure, which means that this template can be successfully matched.

3 Summary

Described method has been successfully implemented and tested in separated test environment. From received partial results have been obtained, that presented method is flexible and generic for analyzing structured data. Unfortunately, project has not been realized successfully, because one of used libraries was crashing during advanced usage. Error was not trivially solvable without additional help from library developers, suitable issue has been reported.

```
message Time
{
    uint32 day = 2;
    uint32 month = 4;
    uint32 year = 2005;
    uint32 weekDay = 6;
    uint32 hour = 21;
    uint32 minute = 37;
}
```

(a) Payload of structure

```
message Time
{
    uint32 hour = 21;
    uint32 minute = 37;
}
```

(b) Requested template

Figure 5: Template comparison with payload structure

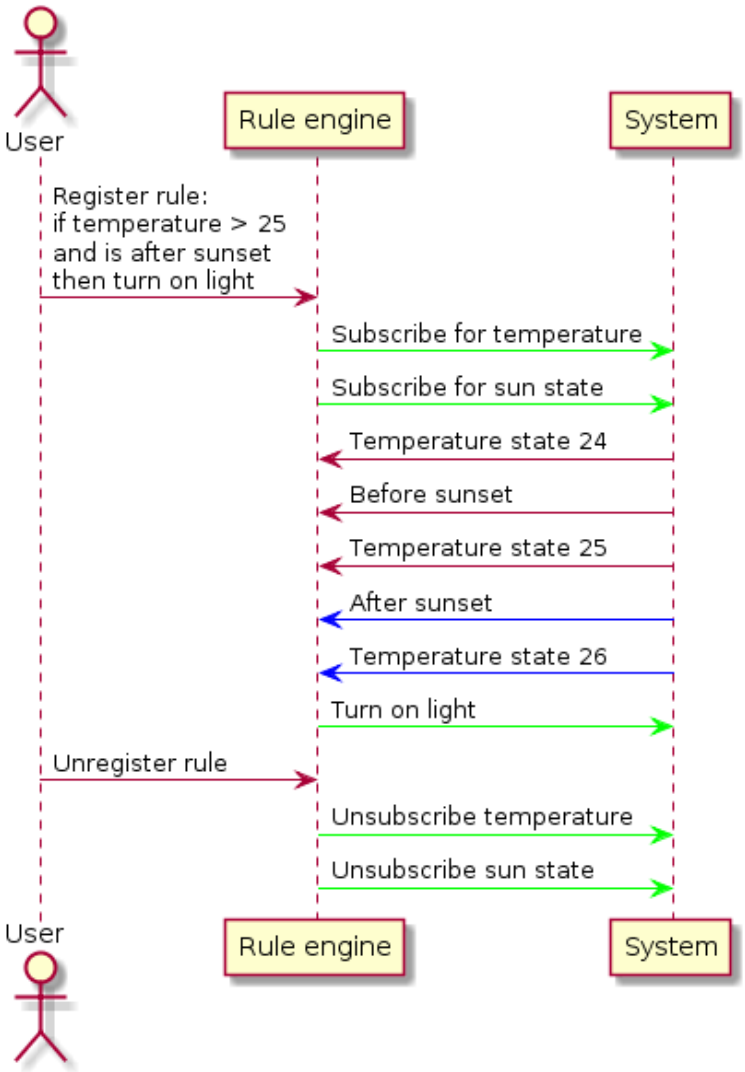

Figure 6: Logic realized by rule engine

References

[1] T. Bartos, "Home automation system based on event driven architecture," 2016.

[2] M. Fowler, *Domain-Specific Languages*, ser. Addison-Wesley Signature Series (Fowler). Pearson Education, 2010. [Online]. Available: https://books.google.pl/books?id=ri1muolw_YwC

[3] Rule-based machine learning. [Online]. Available: https://en.wikipedia.org/wiki/Rule-based_machine_learning

[4] PlauntUML. [Online]. Available: http://plantuml.com/

[5] Zipato system. [Online]. Available: https://www.zipato.com/

[6] Zipato system rule. [Online]. Available: https://www.homecontrols.com/homecontrols/products/media/AA_Manufacturers/ZP-Zipato/Zipato_Programming_02.jpg

[7] Fibaro system. [Online]. Available: https://www.fibaro.com/pl/

[8] Fibaro rule. [Online]. Available: http://fibarozwaveshop.com/wp-content/uploads/2014/11/fibaro-adding-scene.png

[9] eProsima Fast RTPS. [Online]. Available: http://www.eprosima.com/index.php/products-all/eprosima-fast-rtps

[10] Protocol Buffers — Google Developers. [Online]. Available: https://developers.google.com/protocol-buffers/