Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-25 (2019) 104–109

Cost Oriented Humanoid Robot communication with IoT devices via MQTT and interaction with a Smart Home HUB connected devices

Stefan Chivarov*, Peter Kopacek* and Nayden Chivarov&*****

** TU Wien, Institute for Mechanics and Mechatronics, IHRT*
Favoritenstrasse 9-11/E325 A4, A – 1040 Wien
(e-mails: chivarov@code.bg, kopacek@ihrt.tuwien.ac.at)
***Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Sofia, Bulgaria*
(e-mail: nshivarov@code.bg)
**** European Polytechnical University*

Abstract: The article describes different approaches for connecting the teen sized humanoid robot Archie with IoT devices. Main goal of the project is to develop a cost oriented autonomous humanoid robot to assist humans in daily life tasks, thus Archie robot's natural habitat will be our homes and workplaces. Methods for ROS integration of remote internet enabled sensors, actuators and other devices are described. Test with real life use cases are performed. MQTT M2M protocol have been used for connecting to internet enabled IoT devices. Smart home hubs, which connect either locally or to the cloud, IoT devices, that use the Zigbee or Z-Wave and other protocols, rather than only Wi-Fi connected devices, are disused. Finally, we have successfully implemented and tested 3 different approaches for connecting Archie robot to IoT devices, which gives the robot the ability to interact with the environment by receiving data and the ability to control the thousands of IoT devices available on the market.

Keywords: Humanoid Robots, IoT, IoRT, Industry 4.0, ROS, MQTT, openHAB

1. INTRODUCTION

The teen sized humanoid robot Archie is under development at Intelligent Handling devices and Robotics – IHRT, TU WIEN (A. Byagowi et all, 2011). Main goal of the project is to develop a cost oriented autonomous humanoid robot to assist humans in daily life tasks, thus Archie robot's natural habitat will be our homes and workplaces (D. Chikurtev, 2016).

There are currently 27 billion IoT connected devices and this number is expected to increase to 125 billion by 2030 (Compelo, 2019).

The predicted household penetration (usage) of Smart home technologies in US will reach 33.2% in 2019 (other sources predict up to 38%) and is expected to hit 53.9% by 2023. This trend is followed by the EU member states: for example, in Germany the household penetration will be 19.9% in 2019 and is expected to hit 34.5% by 2023, in France the household penetration will be 13.7% in 2019 and is expected to hit 26.2% by 2023

On the other hand, one of the nine principal technologies that make up Industry 4.0 is the Industrial Internet of Things.

This paper describes a Robot Operating System (ROS - Kinetic) (V. Ivanova, 2012) based solutions for Archie robot's direct communication with IoT devices over MQTT (Message Queuing Telemetry Transport protocol - Oasis, 2015) and for interaction with the IoT devices connected to the open source, openHAB (open Home Automation Bus - openHAB 2) Smart home hub. (openHAB) The openHAB integration gives Archie access to more than 1500 supported things. Things, based on the openHAB terminology, are smart home IoT devices from different vendors, using different means of connection and different protocols for connecting with the Smart home hub. There are over 200 technologies, APIs, and services that are supported directly in openHAB (Just to name a few: Bluetooth, Wi-Fi, Z-Wave, Zigbee, KNX, Google Home, Amazon Alexa, etc.).

SMART home technology use devices connected to the Internet of things (IoT) to automate and monitor in-home systems. Smart home technology, also often referred to as home automation or domotics (from the Latin "domus" meaning home), provides homeowners security, comfort, convenience and energy efficiency by allowing them to control smart devices, often by a smart home app on their smartphone or other networked device. A part of the internet of things (IoT), smart home systems and devices often operate together, sharing consumer usage data among themselves and automating actions based on the homeowners' preferences (https://internetofthingsagenda.techtarget.com/definition/smart-home-or-building)

Smart home automation is becoming a de-facto standard in most new home offerings.

Based on these trends, in order to really assist people in their daily tasks, a must have capability of the Archie robot is direct interaction with IoT devices and communication with the Smart Home HUB of their houses.

10.1016/j.ifacol.2019.12.455

As Archie COHR (Cost Oriented Humanoid Robot) is still in the development phase and is not fully functional, we used another robot - Robco18 (N. Chivarov et all, 2018) as a testbed for the Internet-of-Things (IoT) functionality, which will be later used on Archie. This is the power of ROS (Robot Operating System) - the source code and packages we develop, can be reused on other robots.

During our research we've put the emphasis on cost-oriented solutions for IoT connectivity and put them in to real life tests.

As the intent of this article is to share the experience we gained while experimenting with technologies, different libraries and packages and to introduce the readers to the concepts on higher level, we've simplified the excerpts of source code and configuration files for ease of understanding.

2. CONECTING SENSORS AND ACTUATORS USING ROS SERIAL OVER Wi-Fi (TCP CONNECTION)

The method described below provides ROS integration of remote internet enabled sensors, actuators and so on devices. We tested it in the following real-life use case:

At robotics exhibitions children are approaching the stands and want to interact with the robots in some way. We decided make the robot serve small candy (with its articulated arm) to the children. The feature is activated by voice request or by the press of a "Give me some candy" button.

Automated Candy dispensers are available on the market

The dispenser operation is simple: a motion-sensor actuates a motor, which rotates a screw transport auger in order to deliver the candy. In our case we need the robot (instead of the motion sensor) to activate the candy dispenser (for a pre-set time interval), after positioning the robot arm with the scoop in front of the dispenser opening.

For the task, we've chosen the ESP8266 NodeMCU v.3 microcontroller as a cheap (less than 2 €) Wi-Fi ESP-12 based module. Together with the Motor Shield Board, based on L293D driver, it is sold for about 3.30 €.

We used the rosserial ROS package (ROS)

and the rosserial_arduino Arduino Library (Michael Ferguson, Joshua Frank). rosserial is a protocol for wrapping standard ROS serialized messages and multiplexing multiple topics and services over a character device such as a serial port or network socket.

2.1 On the ROS side (Robot's computer) we start the rosserial server in TCP mode by:

```
$ rosrun rosserial_python serial_node.py tcp
```

2.2 On the ESP8266 (NodeMCU) after the code for Wi-Fi setting up the network connection, we set the rosserial server IP and port:

```
IPAddress server(192,168,1,2);

const uint16_t serverPort = 11411;
```

Then we instantiate the ROS publishers and subscribers:

```
ros::Publisher pub_feedback("/candy_dispenser/feedback", &str_msg);

ros::Publisher pub_candy_request("/candy_dispenser/candy_request", &str_msg);

ros::Subscriber<std_msgs::Bool> sub_start_motor("/candy_dispenser/start_motor", &startMotorCallback);

ros::Subscriber<std_msgs::Int16> sub_change_duriation("/candy_dispenser/motor_run_duriation", &changeDuriationCallback);

ros::Subscriber<std_msgs::Bool> sub_button_enable("/candy_dispenser/button_enable", &buttonEnableCallback);
```

In the setup() function we initialize the Node and advertise the Publishers and Subscribers

```
nh.getHardware()->setConnection(server, serverPort);

nh.initNode();

nh.advertise(pub_feedback);

nh.advertise(pub_candy_request);

nh.subscribe(sub_start_motor);

nh.subscribe(sub_change_duriation);

nh.subscribe(sub_button_enable);
```

2.3 Now our robot, by subscribing and publishing to the corresponding ROS topics, can control the motor, set the time the motor should spin the auger and also receive button pressed event and feedback for the commands sent to the dispenser controller.

3. CONECTING TO INTERNET ENABLED IoT SENSORS AND ACTUATORS OVER MQTT M2M (Machine to Machine) PROTOCOL

There are thousands of IoT and Industry 4.0 Industrial IoT devices using MQTT M2M protocol.

MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. MQTT was invented by (Andy Stanford-Clark and Arlen Nipper, 1999).

We use Mosquito MQTT broker (server) to interact with the MQTT enabled IoT devices (https://mosquitto.org/)

For this experiment, we've used Sonoff Basic Smart Switch IoT relay (https://sonoff.itead.cc/en/products/sonoff/sonoff-basic), falshed with Tasmota - alternative open source firmware for ESP8266 based devices such as iTead Sonoff, Allterco Shelly and many, many more.

(https://github.com/arendst/Sonoff-Tasmota/wiki)

3.1 On the Robot side (ROS) we use the mqtt_bridge package (http://wiki.ros.org/mqtt_bridge).

in the config file /mqtt_bridge/config/config.yaml first we set the connection to the MQTT broker and serializer / deserializer:

```
mqtt:
 client:
   protocol: 4      # MQTTv311
 account:
   username: < user_name_for_the_broker >
   password: < user_name_for_the_broker >
 connection:
   host: < MQTT_broker_IP_address >
   port: 1883
   keepalive: 60
 private_path: device/001#
 serializer: json:dumps
 deserializer: json:loads
```

Then we set the topics to be bridged in both directions RosToMqtt and MqttToRos:

```
bridge:
- factory: mqtt_bridge.bridge:RosToMqttBridge
   msg_type: std_msgs.msg:String
   topic_from: /sonoff_01/power
   topic_to: /cmnd/sonoff_01/POWER
- factory: mqtt_bridge.bridge:MqttToRosBridge
   msg_type: std_msgs.msg:String
   topic_from: /stat/sonoff_01/POWER
   topic_to: /sonoff_01/state
```

We launch the mqtt_bridge by:

```
roslaunch mqtt_bridge sonoff_control.launch
```

3.2 Now our Robot can control the relay by publishing "ON" or "OFF" string to the /sonoff_01/power ROS topic and get the state of the relay subscribing to /sonoff_01/state ROS topic.

4. CONNECT TO THE THOUSANS OF IoT DEVICES AVAILABLE, TROUGH A SMART HOME HUB

In the previous two approaches for robot interaction with IoT devices, we are limited to connecting only to IoT devices, which can either be programmed to be part of the ROS environment or connected and controlled over the internet using MQTT.

There are thousands of other home automation devices which use different communication technologies, most popular of them being Z-Wave and ZigBee. These technologies use different radios, frequencies and communication protocols, rather than Wi-Fi.

The Internet of Things (IoT) initiative connects smart home devices such as doorbells, lights, locks, security cameras, and thermostats has made it easy for everyone to install and use these gadgets in their homes, but we still need some sort of mechanism to control them (John Delaney).

A Smart Home Hub is hardware or software that connects devices on a home automation network and controls communications among them.

Smart home hubs, which connect either locally or to the cloud, are useful for internet of things (IoT) devices that use the Zigbee or Z-Wave and other protocols, rather than Wi-Fi.

Sometimes referred to as a smart home bridge, a smart home hub collects and translates various protocol communications from smart home devices. For example, if a smartphone, which does not use Zigbee to communicate, wants to "talk" with a smart lock, which only uses Zigbee and not the Wi-Fi or Bluetooth native to the smartphone, the smart home hub acts as a translator between the two.

As such, a smart home hub acts as the heart of a smart home network, tying together various devices and systems in a centralized platform (Margaret Rouse).

Many different Smart home hubs are commercially available (to name a few - Google Home Hub, Amazon Echo, Wink Hub 2, Samsung SmartThings, Vivint Smart Home).

Although these commercial hubs support multiple communication protocols and can control devices by other vendor, they are focusing on making the customer use the smart things ecosystem of devices provided by the vendor. Most of the Smart home (IoT) devices and Smart home hubs connect to (and are controlled trough) some kind of cloud service, this brings up the serious concern about privacy and not being able to locally control the devices in case of cloud or internet connection failure.

Here the open source hubs, which give you an independent local control over the connected Smart home devices, come into play (the top being openHab, Home Assistant and Domoticz).

We've based our research on openHab (short for Open Home Automation Bus), as one of the best-known home automation tools among open source enthusiasts, with a large user community and quite a number of supported devices and integrations.

OpenHAB runs on most popular platforms such as Linux, Windows and MacOS and on almost any hardware ranging from Raspberry Pis to desktop computers and server PCs. openHAB is very flexible and can support many DIY devices. openHAB 2 is developed in Java, Jetty is included as an HTTP server. Through the openHAB REST API most aspects of the openHAB system can be readily accessed by other programs (https://www.openhab.org/docs/).

With its pluggable architecture, openHAB supports more than 200 different technologies and systems and thousands of devices! (https://www.openhab.org/addons/).

For our research experiment we've connected two different IoT devices to the openHAB Smart home hub: an ESP8266 (NodeMCU) based mailbox sensor (alerting when new mail is placed in your mailbox by the postman) and a Sonoff Dual IoT switch controlling imaginary ceiling and reading lights in the living room of "Archie's home" (https://sonoff.itead.cc/en/products/sonoff/sonoff-dual).

There are few ways to configure openHAB, based on the level of knowledge of the user, the one which give us full control is via configuration files. There are three main configuration files:

openhab.cfg, <sitemap-name>.items and <sitemap-name>.sitemap

We will use Mosquito MQTT broker (server) to interact with the IoT devices connected to the openHab server, using the MQTT M2M (Machine to Machine) protocol.

In order add the MQTT functionality we have to activate the openHab MQTT binding in the Bindings manager.

openhab.cfg is the overall config file located in /etc/openhab/configurations/. In this file global config is set such as security authentication and MQTT server setting. E.g. to connect to our MQTT server we add the following:

mqtt:mosquitto.url=tcp:// <MQTT_broker_IP_address>:1883

mqtt:mosquitto.user=<user_name_for_the_broker>

mqtt:mosquitto.pwd=<password_for_the_broker>

mqtt:mosquitto.qos=2

mqtt:mosquitto.retain=true

For simplicity of explanation, we will define just few items in openHab:

Presence Item - which will report if the smartphone of the homeowner can be located on the home Wi-Fi network (by MAC address). This does not require a separate sensor, the check for presence is done using the openHAB server Wi-Fi connection.

Mailbox sensor item – this item will check the state of the mailbox sensor (programming of the mailbox sensor will be described below) and report if there is mail in the box.

Sonoff Dual Item which will allow control and report the state of the two relays of this IoT device, which will control the lights in the living room (https://sonoff.itead.cc/en/products/sonoff/sonoff-dual). The Sonoff Dual is flashed with Tasmota alternative open source firmware (https://github.com/arendst/Sonoff-Tasmota/wiki).

The user will be able to manually switch on the lights as usual, on top of that user will be able to control and view the status of this 3 Items using the openHAB WEB based user interface or the dedicated openHAB apps for Android and IOS smart phones and tablets. In the same time Archie robot will be able to get the information from the sensors and control the Sonoff Item relays trough the MQTT M2M protocol.

- Items file:

The items file defines all the 'things' you might want to view or control in openHAB. Items can be bound to 'bindings' which are optional packages that used to extend functionality of openHAB. Our use case makes use of the MQTT and HTTP bindings. The items file is located in /etc/openhab/configurations/items.

Inbound configurations allow us to receive MQTT messages into an openHAB item. Every item is allowed to have multiple inbound (or outbound) configurations. The structure of the inbound MQTT configuration string is:

Item myItem
{mqtt="<direction>[<broker>:<topic>:<type>:<transformer>:<regex_filter>],
<direction>[<broker>:<topic>:<type>:<transformation>],
..."}

Below you can see the structure of the outbound MQTT configuration string. Outbound configurations allow us to publish (send) an MQTT message to the MQTT broker when an item receives a command or state update, and other MQTT clients that are subscribed to the given topic on the same broker, like Arduino devices for example, will receive those messages.

```
Item itemName {
mqtt="<direction>[<broker>:<topic>:<type>:<trigger>:<transformation>]" }
```

The in inbound and outbound MQTT configuration string can be combined.

Excerpt of our archieshome.items file defining the 3 items described above looks like this:

```
……

Group gGF "Ground Floor" <groundfloor> ["GroundFloor"]

Group GF_Living "Ground Floor Living Room" <groundfloor> ["GroundFloorLivingRoom"]

Group Garden "Garden" <garden> ["Garden"]

Group Presence "Presence" <presence> ["Presence"]

Group ROS (All)

String ROS_Status "ROS [%s]"

/* Lights */

Switch Light_GF_Living_Ceiling "Ceiling light" (gGF, GF_Living, Lights ,ROS) ["Light"] { mqtt=">[broker:cmnd/sonoff_01/POWER1:command:*:default],
<[broker:stat/sonoff_01/POWER1:state:default]" }

Switch Light_GF_Living_Reading "Reading light" (gGF, GF_Living, Lights ,ROS) ["Light"] { mqtt=">[broker:cmnd/sonoff_01/POWER2:command:*:default],
<[broker:stat/sonoff_01/POWER2:state:default]" }

/* Presence */

Switch Presence_OwnersSmartPhone "Owners SmartPhone presence detected" (ROS) <presence> { channel="network:pingdevice:1d2fc7d4:online" }

/* Mailbox */

Switch MailboxSensor "Got mail?: [%s]" (Garden, ROS) {mqtt="<[broker:stat/WiFi_Mailbox_sensor/:state:default]"}

…..
```

- Sitemap

Sitemaps are used to create elements of a user interface for making openHAB items accessible to the front-end interface. A simplified sitemap file for Archie's Smart Home looks like this:

```
sitemap ArchiesHome label=" Archie's home"

{

        Frame label="Living Room Lights"

    {

        Switch item= Light_GF_Living_Ceiling

        Switch item= Light_GF_Living_Reading

    }

Frame label="Is there mail in the mailbox?"

    {

        Text item=MailboxSensor

    }

    Frame label="Is the owner at home?"

    {

        Text item=Presence_OwnersSmartPhone

    }

}
```

On the ROS side Archie robot's computer:

The power of openHab is that it can connect not only devices using the Internet (Wi-Fi or wired), but also devices using different radio frequencies and protocols like Z-Wave, Zigbee, Bluetooth and so on.

Hopefully, there is a package available which allows ROS to communicate with the openHab server via Internet/MQTT and in that way to access all of the devices connected to the openHab server, regardless of the protocol and the type of physical connection used between the device and openHab. This allows a ROS robot to connect to a vast variety of IoT devices such as motion detectors, Z-Wave, Zigbee, Bluetooth devices, lighting, door locks, etc.

The iot_bridge package provides a bi-directional bridge between ROS and the OpenHAB Home Automation system (M Herbert). After installing the iot_bridge package, we should update the host address and port to match the openHAB server in iot_bridge/config/items.yaml

Then we need to make some changes in the openHAB configuration files (this is already done in the files mentioned above).

We have to create a ROS group: Group ROS (All) and add the ROS group to each item, that should send status updates to ROS. We can send commands from ROS to any openHAB item regardless of what group it is in. We also need to add ROS_Status (String ROS_Status "ROS [%s]") to the openHAB's item file.

Now we can launch the IoT Bridge node:

```
roslaunch iot_bridge iot.launch
```

When there is an item (belonging to the ROS group) state change in openHab, we receive this on /iot_updates topic. We can view this in the terminal by

```
rostopic echo /iot_updates
```

For example, a when presence of the owner's smartphone (as defined in the openHab items config file above) is detected/triggered in OpenHAB. The OpenHAB bridge will publish the following to the iot_updates topic in ROS:

message type: diagnostic_msgs/KeyValue,

key: Presence_OwnersSmartPhone,

value: 1 (string).

Our robot can control devices connected to openHab by publishing to /iot_command topic, in Pythonit looks like:

pub = rospy.Publisher('iot_command', KeyValue, queue_size=10)

pub.publish(item_name, item_value)

So, in order for our Robot to switch the celling lamp on we need to publish:

pub.publish("Light_GF_Living_Ceiling", "ON")

And to turn off we need to publish:

pub.publish("Light_GF_Living_Ceiling", "OFF")

5. CONCLUSIONS

We've successfully implemented and tested 3 different approaches for connecting Archie robot to IoT devices. This gives the robot the ability to interact with the environment by receiving data and (potentially) controlling the thousands of IoT devices available on the market.

An emerging trend in robotics is integrating robots with the Internet as a form of Internet-of-Things (IoT). This allows robots to be controlled and monitored through the Internet and is a must for cloud robotics applications.

We already have experience using RoboWebTools (http://robotwebtools.org/), but we still need to test and evaluate other approaches like Roslink (Koubaa, Anis, Alajlan, Maram & Qureshi, Basit, 2017). ROSLink is a new protocol to integrate Robot Operating System (ROS) enabled-robots with the IoT.

ACKNOWLEDGEMENT

The research presented in this paper was supported partially by the Bulgarian National Science Fund under the contract № KP-06-M27/1–04.12.2019 "Research and Development of Innovative, Intelligent Information and Communication Technologies for Control of Service Robots".

REFERENCES

A. Byagowi, P. Kopacek, J. Baltes, "Archie": A Cost Oriented Humanoid Robot, Proceedings of the 18th IFAC World Congress Milano (Italy), 978-3-902661-93-7/11/$20.00 © 2011 IFAC

D. Chikurtev (2016), Indoor Navigation for Service Mobile Robots Using Robot Operating System (ROS), PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, Vol 67, Sofia, , p. 61-70, ISSN 0204-9848.

Compelo (2019): https://compelo.com/smart-device-trends-2019/

Statista, (2019): https://www.statista.com/outlook/279/109/smart-home/united-states

Ivanova et all, Application of Meta-Operating System Ros for Control of Service Mobile Robots, ADP 2012, p. 247-254, ISSN 1310-3946.

Oasis (2015): "MQTT 3.1.1 specification". OASIS. December 10, 2015. Retrieved April 25, 2017.

openHAB, 2019 by the openHAB Community and the openHAB Foundation e.V., https://www.openhab.org/

Chivarov, N., Chikurtev, D., Markov, E., Chivarov, S., Kopacek, Cost Oriented Tele-Controlled Service Robot for Increasing the Quality of Life of Elderly and Disabled - ROBCO 18, P., Volume 51, Issue 30, 2018, Pages 192-197, ISSN: 24058963

ROS, http://wiki.ros.org/

Arduino, https://www.arduinolibraries.info/libraries/rosserial-arduino-library

Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999. (http://mqtt.org/faq)

Contributing Editor for PCMag, John Delaney, https://www.pcmag.com/article/327457/what-is-a-smart-home-hub-and-do-you-need-one

Margaret Rouse https://internetofthingsagenda.techtarget.com/definition/smart-home-hub-home-automation-hub

M Herbert http://wiki.ros.org/action/show/iot_bridge?action=show&redirect=openhab_bridge]

Koubaa, Anis & Alajlan, Maram & Qureshi, Basit. (2017). ROSLink: Bridging ROS with the Internet-of-Things for cloud robotics. 10.1007/978-3-319-54927-9_8.