Home automation Code Camp Report

SMART DORMS

Lappeenranta University of Technology
Degree Program in Pervasive Computing and Communications for Sustainable Development (PERCCOM)

Valeria Cedillo – 0528695
Askar Serikov – 0528967
Orsola Fejzo – 0528705
Daniel Schürholz – 0528954

Introduction

Smart Dorms is a fast and practical technological solution that helps students and the housing companies to adopt a more sustainable behavior. It consists of automated sensors and actuators that can be installed in any student residence, as well as remote control tools that allow the reduction of energy consumption while having a positive impact on the environment.

Objective

Energy efficiency and the environment are an important part of Smart Dorms strategy. The goal of Smart Dorms is to reduce electricity wastage, rise user awareness and engage users to improve their behavior towards a better use of electric appliances. Through an automated system, it is aimed to remotely control and monitor the flat appliances, as well as make up for people's negligence by automatically adjusting the devices' operation mode when necessary.

Vision

To automate daily tasks by transforming the daily home routine into more sustainable through practical and easy to implement solutions.

Scenario

Edwin leaves his flat early morning every day to go to university and he never checks the thermostat valve level of the radiator. After a long day in the university, he comes back home, goes to his room and turns the lights on to drop his stuff to afterwards go the kitchen to start cooking with his flat mate without turning the lights off again. After a couple of minutes cooking, the kitchen becomes overheated, so they usually open the window to cool down and ventilate the space, but they don't adjust the temperature of the radiator.

Since the students are usually in a hurry to arrive on time to class, quite often it happens that they leave the lights of the common areas on, or the lights of the rooms. Especially during spring, when there's more sunlight it's really common that while they're getting ready to go to school, they don't perceive if the light is coming from the window or if the light bulb is on.

Taking into considerations the above scenario, a cost - benefit analysis has been made, in which just one area of 71 apartments were considered, with 10 lightbulbs of 15 watts in each apartment. The amount obtained was considering an average scenario of 2 rooms apartment, assuming that students forget to turn off the lights 3 days per week for 10 hours which is the time spent at university, and assuming that in 1 week 1 hour of electricity is wasted even when people are at home.

Kourula (150 occ, 14530 m3)				
2017	**MWh**	**€**	**CO2 kg**	**% MWh wasted**
Jan-Dec	189.86	€ 3,001,686.60	141.296	9.48 %

Disclaimer: Electricity consumption includes both property electricity and household electricity, as well as electricity consumed in apartments (71 apartments approx.) whereas the electricity wastage only the lightbulbs were considered.

The results obtained showed that just the lightbulbs represent a 9.5% of CO2 emissions in a year, for one apartment complex. This, without taking into consideration yet the heating system and other facilities like laundry rooms that typically consume more energy.

User engagement

Smart Dorms has a strategy to engage the users to modify their behavior and have thought about implementing an incentive system. Currently the facilities of LOAS have saunas in each building and to make use of this service there is a monetary cost, however, it is very attractive and is quite popular among students. So Smart Dorms will take advantage of this and will be implemented an alternative method of payment for the use of saunas. The electricity consumed in each apartment will be monitored monthly, based on that, the apartment that manages to reduce the consumption will be rewarded with sauna tickets, otherwise they will have to use the current method of payment.

Idea and Motivations

Since the university residences have a fixed rent price and do not charge for the consumption of services, the students are not motivated to pay special attention to the use of them, hence, generating high costs by not making conscious use of the electric appliances.

These residences present plenty improvement opportunities through automation and the integration on IoT systems. Firstly, the lighting system is not automated, switching lights on/off is entirely manual, as a consequence, lights that have been forgotten on when out of the residence, increase the wasted energy. The same scenario can be seen in the laundry room, e.g. air driers can run more than necessary because there is no way to detect if the clothes are dry or not.

Except energy consumption, there are improvement possibilities even for water wastage and waste separation, which is currently not supervised.

Solutions and Optimizations

- Automated and remote appliances' control

Given the above scenario and motivation, we are devising a system to reduce the utilization of the appliances in an apartment when not needed. In the solution we propose, the lights turn off / on automatically depending on the presence of people in the apartment. It is worth noting though, that this scenario can be extended to any other electrical device. Every resident indicates his presence through a smart keyholder by attaching the key to the device when he enters the apartment, in that moment, the lights will be turned on for their room and the common areas, if there is not enough brightness in the apartment, which, in turn can be measured with a light sensor. When all the residents have left the apartment, the lights will be turned off automatically.
In addition to that, the residents can control the status of their electrical appliances from a Telegram app, as well as do basic actions such as turning them on / off.

- Controlled heating

Reducing the heating consumption is one of the most difficult challenges to tackle, especially in rather cold countries like Finland, but however that system can be improved. For example, by utilizing a smart thermostat and an optical sensor for the window, the heating system not only can be scheduled according to personal preferences or seasons, but also it can be programmed to turn off completely or remain at some suitable low temperature when the window of the room is open.

- Optimization of the laundry room consumption

Quite often it happens that students forget the washing machines on after finishing to use them, or even worse, there are dryer machines that can only manually programmed with a timer and cannot be turned off even after there is no need for them anymore. In addition to that, it happens that students use the laundry machines without booking first. To avoid all the above mentioned negative aspects, we are proposing a solution integrated with the Google Calendar where all the bookings are stored. The system is rather easy and powerful, the booking calendar is queried in specified repeated intervals, if there are no bookings in a specified machine it is turned on (and vice versa).
A future plan for this issue would be, for example, in the drying rooms to install humidity sensors that could somehow detect if the clothes are dry or not and eventually turn off the machine when there is no more need.

- Waste management

To avoid the problem of non-separated waste in the bins, caused both by negligence and lack of knowledge for certain products, in our future plans we are proposing a smart bin system. The smart bin consists of different compartments, one for each waste type that needs to be separated. At the moment that someone needs to dispose of waste, the latter is either scanned (if a code can be printed on it) or visually analyzed and the right bin compartment is opened to put the waste away depending on its type. This system, ideally, would make use of AI methods in order to determine the right waste type.

- Active shower system

This is a simple solution that aims to make people aware of the amount of water that they are consuming daily in the shower. The main idea is that when some predefined threshold of consumption is close to being reached a warning will be displayed, at that moment the resident will know exactly how much they consumed and ideally stop wasting water. To encourage people not to simply disregards the alerts, whenever they do not exceed the threshold they might gain extra points to “spend” in the saunas or the laundry room.

- Remote security control

We are also offering an enhanced security system interfaced for the user through the Telegram app. The user can manually change its preferences to “Away” for example, and in that case, he would be notified with a picture in a Telegram message if some sort of motion is detected in his room.

Investment

List of devices & prices

For apartment

Model	Quantity	Price per item	Total price
Raspberry Pi 1 512MB Model B+	1	34 EUR	34 EUR
busware CC1101-USB-Lite 868MHz CUL	1	50 EUR	50 EUR
HomeMatic Wireless Optical Door/Window Sensor	5	25 EUR	125 EUR
Homematic Brightness Sensor	2	40 EUR	80 EUR
HomeMatic Wireless Switch Actuator 4-channel, DIN-rail mount	1	100 EUR	100 EUR
Axis M1025 IP camera	1	280 EUR	280 EUR
Total per apartment			669 EUR

For laundry:

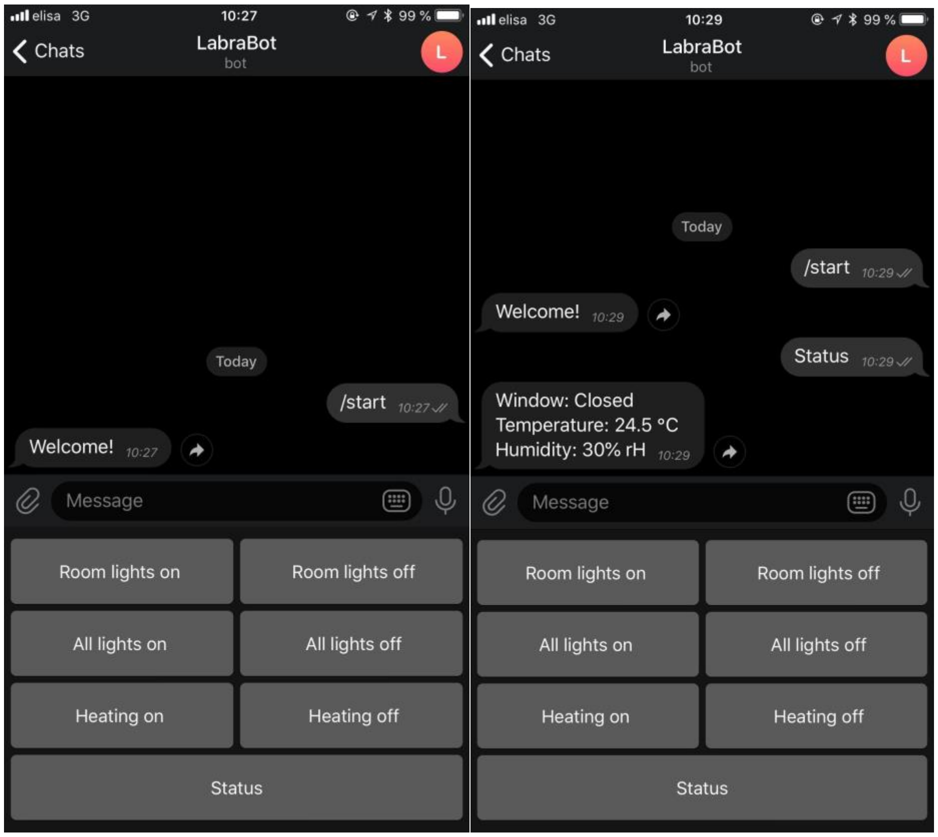
Model	Quantity	Price per item	Total price
Raspberry Pi 1 512MB Model B+	1	34 EUR	34 EUR
busware CC1101-USB-Lite 868MHz CUL	1	50 EUR	50 EUR
HomeMatic Wireless Switch Actuator 4-channel, DIN-rail mount	1	100 EUR	100 EUR
Total			184 EUR

After having obtained the costs of the devices, an estimation was made for the amount of investment required in the devices for a complex of 71 departments. Additionally, the cost of implementation was assumed, considering 3 people in a 15-day period, for this the average salary in Finland was taken into account. It is important to mention that the cost of maintenance is not included, since it is assumed that the maintenance is low and does not have a significant impact.

The total amount of investment would be **53,083 €**, which is a low quantity compared with the annual cost on electricity bill.

Features

- Accessibility
- Portability
- Scalability
- Security


elisa 3G 10:27 99 %
Chats
LabraBot
bot

Today

/start 10:27

Welcome! 10:27

Message

Room lights on | Room lights off
All lights on | All lights off
Heating on | Heating off
Status

elisa 3G 10:29 99 %
Chats
LabraBot
bot

Today

/start 10:29

Welcome! 10:29

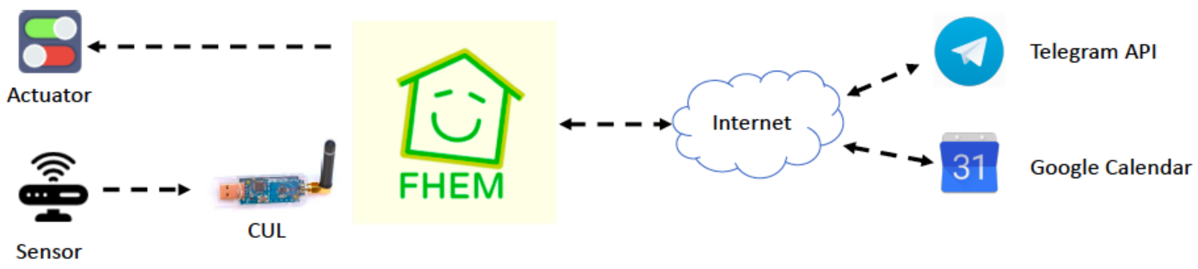
Status 10:29

Window: Closed
Temperature: 24.5 °C
Humidity: 30% rH 10:29

Message

Room lights on | Room lights off
All lights on | All lights off
Heating on | Heating off
Status

Technology



Implementation

Setup
For our system we used following devices:
- 1 HomeMatic compatible RF signals sender/receiver (CUL1)
- 1 FS20 compatible RF signals sender/receiver (CUL2)
  Both were used to enable the FHEM server to communicate with other devices

- 2 HomeMatic electric switches powering 2 light bulbs
- 1 HomeMatic optical window sensor
- 1 HomeMatic shutter contact sensor
- 1 FS20 light sensor
- IP Camera connected via LAN (Axis M1113)

During the development process, we used following FHEM modules:

- **TelegramBot**
  - We used Telegram as a remote controller that can send commands (e.g. request a live photo, control lights, set 'Away mode') and receive notifications and data (e.g. detected motion when 'Away mode' is active).
- **Calendar**
  - We integrated our system with Google Calendar to simulate power management in a laundry room based on existing bookings.
- **IPCAM**
  - This module was used to get images from the IP camera.
- **HTTPMOD**
  - This module was used to extract and parse motion detection data from the IP camera.
- **at**
  - This module was used to obtain motion detection data from the IP camera with a certain interval.
- **dummy**
  - We used a dummy device 'Away Mode' to store and read its state value (on/off).

Recommendations

Quite early in the development process, after basic experiments with electric switches and sensors, we found out that HomeMatic is faster and more reliable than FS20. Thus, we recommend using HomeMatic devices rather than their FS20 counterparts. We also highly recommend using **99_myUtils.pm** file to store all your functions/scripts as it can be edited with an IDE or a text editor of your choice and it makes your code easier to maintain and reuse. Last but not least, put all your devices and notifies inside one room (can be done by setting the 'room' attribute on each of them), it will save you time and help keep things organized.

Smart key holders, automated lights

The first feature we have developed for our system were smart key holders. The idea is that whenever tenants come to their apartments they put their keys on the key holders which have sensors and register if a tenant is inside the apartment or not. We used the optical window sensor and the shutter contact sensor to act as key holders - both have closed/open positions, which correspond to 'key is present'/'key is not present' states. This feature allows the system to know how many people are inside the apartment, so, for instance, the heating system can be adjusted based on the number. However, since we did not simulate the heating system in our solution, we used this feature to automate lights: if no one is inside - turn off lights everywhere if a tenant is not inside - turn the lights off in his/her room. Lights were imitated by electric switches with plugged-in light bulbs. We also used the FS20 light sensor to, when it is dark inside, automatically turn on lights in the hallway and corresponding room when a tenant enters the apartment and puts his/her key on the key holder. The light sensor can also be used to automate lights depending on ambient brightness even when tenants are inside, but we figured it might be annoying.

Telegram Bot

After implementing the first feature we figured that it would be convenient to control lights remotely. Thankfully, FHEM has a built-in TelegramBot module which enables fast and easy integration of the system with Telegram. The module is well-documented and all you need to do outside of FHEM is to message @BotFather on Telegram. @BotFather is a bot that will create your bot for you (yes, a bot that creates other bots, hence the name, I guess) and will generate a token that you should pass to the module. The most common scenario of the bot is to notify when a certain command is received (can be done by reading from 'msgText' field of TelegramBot module) and based on the command do certain actions. Worth noting that Telegram allows defining custom buttons for bots, so our bot shows buttons for all possible commands which eliminates the need to type them manually. Neat.
Example:

```
define sendImage notify telegramBot:msgText:\sLive\sPhoto {sendLivePhoto}
```

where {sendLivePhoto} refers to a perl function inside 99_myUtils.pm:

```
sub
sendLivePhoto {
  my $peer = ReadingsVal("telegramBot", "msgPeerId", "");
  fhem("get ipcam image");
  fhem("define a5 at +00:00:01 set telegramBot sendImage @".$peer."
\"ipcam_snapshot.jpg\"");
}
```

In this example, sendImage notify triggers sendLivePhoto function whenever "Live Photo" message is received ('\s' stands for space). The function is looking to whom it should send the image by reading the 'msgPeerId' field of TelegramBot module, then asks the IP camera to take a picture and, after 1 second (to ensure the new image is saved in the filesystem), it sends the picture to the peer.
Basically, this way we have implemented remote lighting control and the 'Status' feature for the bot that on request sends information on temperature and humidity inside the apartment (extracted, of course, from the HomeMatic temperature and humidity sensor), if the window in the tenant's room is open or not (HomeMatic optical window sensor) and whether the 'Away Mode' is on or off.

Away Mode, motion detection, live photo on demand

Since we got to use a camera (on how we made it work, read the dedicated chapter), first of all, we started implementing a photo-on-demand feature which we explained in the example above. Later we found out that the camera we used (Axis M1113) has a built-in

motion detection function (again, detailed explanation of this function can be found in the dedicated chapter). The idea is that tenants using the bot can turn on the 'Away Mode' during which the camera will detect motion and, when motion is detected, take a photo and send it to them. Worth noting that we created a dummy device to store the state of 'Away Mode' - the device had one WebCmd attribute, 'state', which can either be equal to 'on' or 'off'. It can be done with following FHEM commands:
define awayMode dummy
attr awayMode webCmd on:off
Thus, whenever you want you can read or change the state of the 'Away Mode' by manipulating the state of the dummy device.
Automated laundry
The idea of this feature is to turn the power in laundry off when there are no active or upcoming bookings. It happens quite often when tenants leave washing machines and dryers on after using them or when they use the facility without booking it. This feature will prevent both problems. For the feature to work we integrated a Google calendar that contains all bookings. It can be done by using the built-in 'Calendar' module. It accepts any .ical calendar and allows setting certain update interval for the calendar. NB: not all functions of this module work as described in the documentation, we, for instance, could not set our own date/time format even though it was mentioned in the documentation. The algorithm is simple: update the calendar every ten minutes but look only for bookings that are either active right now or will start within 15 minutes. If no bookings found, turn the power off.

Setting up the IP camera for live snapshots and motion detection

For our smart dorm scenario, we envisioned a security system that would let the students feel safer when leaving their belongings in their accommodation, besides, it would provide an interesting comfortability gadget. The idea was to install an IP camera in the room (with the students' consent of course) for them to be able to have live feed on what is happening in their apartment while they're away. We considered two main functionalities to offer to the students:

- Taking a live picture of the room at any point in time and on demand, and message it is using the Telegram Bot explained in the previous sections.
- Use the built-in motion detection of the camera, to sense movement in the room and notify the user by sending a live picture to their Telegram account.

For demonstrating these scenarios, we decided to use one of the Axis M1113 IP cameras available in the Living Code Lab. As mentioned before, these cameras come with some HTTP API endpoints, where the user can interact with functions that the camera offers, i.e. the motion detection endpoint or the live picture endpoint, both of which we used. In this section we will explain the process of configuring and setting up these two functionalities, given that they weren't as straightforward as we initially thought.

Finding the camera in the network

The cameras in the lab come with an IP address shown on the top of the devices. These addresses are not the real ones though, so we had to connect to the Lab network using an ethernet cable and searching for the right network address of the camera we wanted to use. To achieve this, we used the NMAP tool (https://nmap.org/), which is a free and open source utility for network discovery and security auditing. We found the address of the camera by making a scan of the connected devices on the network with the same subnet IP as our computer (on an ubuntu console):

$ *nmap -sn 172.16.16.1-255*

The output of the command showed many different addresses, four of which were IP Axis cameras. So, we disconnected the camera we wanted to use, run the same NMAP command again and noticed the missing device, which in turn was the camera we were using (i.e. 172.16.16.85).

Accessing the camera's HTTP server

After knowing the IP address, we tried to access it's HTML page, which is served on the usual HTTP 80 port. By default, the Axis cameras use Basic Authentication to protect these endpoints. The default user and password combination are:

- User: root
- Password: pass

In our case, these credentials had been modified. To access the server, we had to perform a factory reset on the camera. The procedure to accomplish this is given in their documentation (Axis M113 documentation https://www.axis.com/files/manuals/um_m1113_46768_en_1206.pdf) and is pretty straight forward:

1. Disconnect power from the product.
2. Press and hold the Control button and reconnect power.
3. Keep the Control button pressed for about 15 seconds until the Status indicator flashes amber.
4. Release the Control button. The process is complete after about 1 minute (when the Status indicator turns green). The product has been reset to the factory default settings. The default IP address is 192.168.0.90 or it has received another IP if connected to a network with a DHCP server.

After the reset, the HTTP endpoint will open the initialization page, in which one can change the user and password to a combination of your own choice. Finally, the HTML web page is ready to be used.

Setting up the image retrieval endpoint in FHEM

The Axis camera has an endpoint to retrieve an image of the current camera stream, on demand. The endpoint is (for more options on the image type review the camera documentation https://www.axis.com/files/manuals/HTTP_API_VAPIX_2.pdf):

http://<camera-ip-address>/axis-cgi/jpg/image.cgi

You can test the endpoint on a browser and set some parameters (for more info on parameters refer to https://www.axis.com/files/manuals/parameter_spec_VAPIX_2.pdf) on it, before integrating it to FHEM.

For the integration with FHEM, we used the IPCAM module. This module lets you define an IPCAM object, with the IP address of a camera, and query some of it's endpoints periodically or on demand. First add an IPCAM object with the following command:

define ipcam IPCAM 172.16.16.85

Afterwards we define some attributes, that can be entered directly on the device page in FHEM or added by using the following commands

1. Setting the basic auth credentials: *attr ipcam basicauth root:pass*
2. The delay between snapshots (if more than one is taken) set to 10 seconds after the command is issued: *attr ipcam delay 10*
3. The URI path of the snapshot endpoint: *attr ipcam path axis-cgi/jpg/image.cgi*
4. Adding the camera to our room: *attr ipcam room Labra*
5. Number of snapshots to take on request: *attr ipcam snapshots 1*
6. Location to store the taken snapshots (absolute path is recommended): *attr ipcam storage /Users/askarserikov/Documents/Projects/fhem-5.8/*

After this configuration, we can send a live snapshot every time the user requests it, only by calling the *sendLivePhoto* function we defined previously on the Telegram Bot section. This function uses the *get ipcam image* action of the ipcam object and sends it over to the user's telegram account. Finally, the last part missing to comply with our two scenarios was the motion detection feature.

Setting up the motion detection endpoint in FHEM

The Axis camera has an endpoint that can be queried to receive information about the motion detection process. To use it first we have to define a motion detection group, that will handle the process by applying the detection algorithm on a window over the camera image stream itself. To define the motion detection group, we use the following url and parameters:
http://<camera-ip-address>/axis-cgi/operator/param.cgi?action=add&group=Motion&template=motion
To modify the values of the group, like window size and position, sensitivity, object size and historical data, etc. please refer to the camera's endpoints and parameters documentation. Once we added the Motion group, we can start querying the motion detection endpoint:
http://<camera-ip-address>/axis-cgi/motion/motiondata.cgi?group=0
This endpoint returns an multipart/x-mixed-replace HTTP response, that contains the motion level and the threshold for every fraction of time (we weren't able to find out the rate at which this information is updated, but seems that under one second). Because of time reasons to develop this feature, we decided to use the HTTPMOD FHEM module to read the data stream, even though a more suitable solution might be available amongst the FHEM modules. The problem with this module is that it is not well suited to read from data streams (keeping HTTP connection alive), as is the case with this motion detection endpoint of Axis cameras. So we configured the HTTPMOD module to make a query every 5 seconds to the endpoint and buffer the result, so that we can search in all the buffered data for high levels of motion that were detected. To define the HTTMOD object we used the following commands:
define motionDetect HTTPMOD http://root:pass@<camera-ip-address>/axis-cgi/motion/motiondata.cgi?group=0 5

Afterwards, we implemented a function that will look on the *buf* internal value of the HTTPMOD object:

```
sub getMotionLevel {
  my $away = ReadingsVal("awayMode", "state", "");
  if ($away eq "on") {
    my $peer = ReadingsVal("telegramBot", "msgPeerId", "");
    my $body = InternalVal("motionDetect", "buf", "");
    if ($body =~ /level=((1[5-9])|([2-9][0-9])|100)/) {
      fhem("get ipcam image");
      fhem("define a5 at +00:00:01 set telegramBot sendImage @".$peer."
\"ipcam_snapshot.jpg\"");
    }
  }
}
```

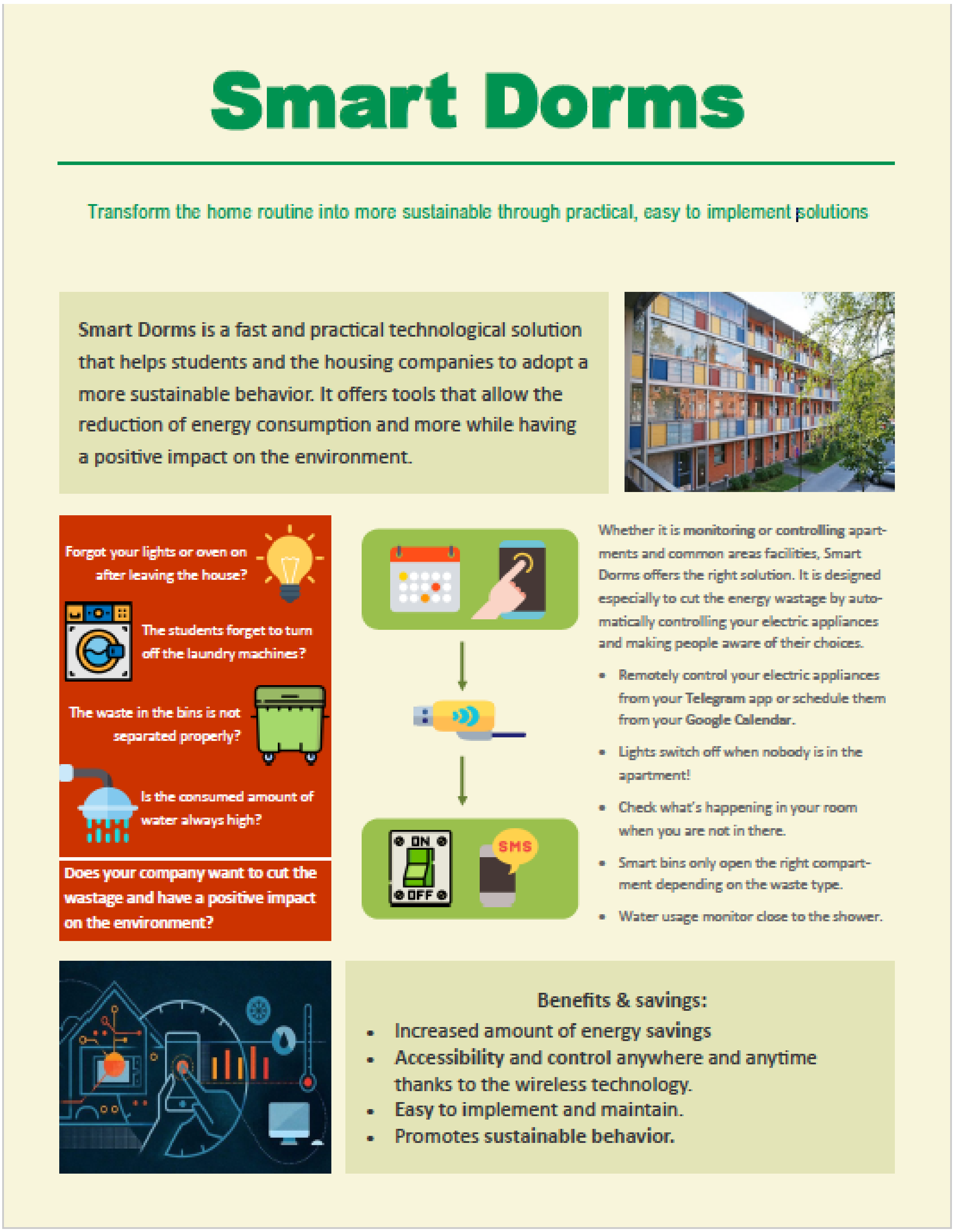
Besides getting the level of motion, the function sends a picture to the user using the Telegram Bot, only if the motion level is over 15. To change this threshold, you have to modify the regular expression in *$body =~ /level=((1[5-9])|([2-9][0-9])|100)/.* This regular expression will look over the whole buffer for a level higher to 15, meaning that motion detection will work in steps of 5 seconds. Then we defined an *at* object that would call this function every 5 seconds, to match the query rate of the HTTPMOD object. The defined object looked like this:

*define a1 at +*00:00:05 {getMotionLevel}*

To improve the motion detection procedure, a module with support for HTTP streaming could be used, so that only the live value is used at any point of time, instead of reading 5 second buffers. Nonetheless, our approach worked well for the goal of detecting movement of big objects (such as an intruder) inside the student's room.

GitHub repository of our project: https://github.com/askarserikov/smartdorms

Poster



Smart Dorms

Transform the home routine into more sustainable through practical, easy to implement solutions

Smart Dorms is a fast and practical technological solution that helps students and the housing companies to adopt a more sustainable behavior. It offers tools that allow the reduction of energy consumption and more while having a positive impact on the environment.

Forgot your lights or oven on after leaving the house?

The students forget to turn off the laundry machines?

The waste in the bins is not separated properly?

Is the consumed amount of water always high?

Does your company want to cut the wastage and have a positive impact on the environment?

Whether it is monitoring or controlling apartments and common areas facilities, Smart Dorms offers the right solution. It is designed especially to cut the energy wastage by automatically controlling your electric appliances and making people aware of their choices.

- Remotely control your electric appliances from your Telegram app or schedule them from your Google Calendar.
- Lights switch off when nobody is in the apartment!
- Check what's happening in your room when you are not in there.
- Smart bins only open the right compartment depending on the waste type.
- Water usage monitor close to the shower.

Benefits & savings:

- Increased amount of energy savings
- Accessibility and control anywhere and anytime thanks to the wireless technology.
- Easy to implement and maintain.
- Promotes sustainable behavior.

Protocol: LonWorks - LonTalk

1. Introduction

In this section it is given a description of the LonWorks technology, was well as LonTalk, its communication protocol.

LonWorks is a networking platform used in the building automation domain created by the Echelon Corporation in 1988. It is built over a protocol created by the same company for networking devices over media such as twisted pair, powerlines, fiber optics, RF. It is estimated that the number of devices connected with LonWorks technology is around 90 million. LonWorks is a standard technology for many of the global standards' organizations like IEEE, ANSI, SEMI etc. Over 300-member companies are part of the LonMark Interoperability Association, reflecting its strength in the automation field.

The term LonWroks technology refers to the Neuron chips from multiple vendors, the LonTalk protocol, the various physical media which connects the devices, the connectivity devices (routers, PC interface cards), network management tools and all the various products built around the platform.

2. The Neuron Chip

The Neuron Chip is the core of almost all LonWorks based devices and it contains the whole LonTalk protocol stack. Every cheap has a unique 48-bit physical Neuron ID (managed by Echelon) and they consist of:

- three CPU-s
  - Media Access Processor – responsible for transmitting messages on the network and check the messages' correctness
  - Network Processor – responsible for packet routing, destination addressing, end-to-end ACK-s, retries etc.
  - The application processor – responsible for running the user's custom application
- Memory: RAM, ROM, EEPROM. In some Neuron Chip types the ROM is already preprogrammed with the LonTalk protocol, the operating system, I/O libraries etc..
- 11 I/O pins
- Communications port
- Firmware
- Operating system

In order to connect to the network each Neuron chip is equipped with a 5-pin configurable transceiver, additionally it has 11 I/O pins to be connected with external devices and an I/O library firmware.

3. LonTalk

a. Introduction

The LonTalk protocol is designed for communication in control networks. Features of these networks are the short message sizes, the low cost per node, the availability over multiple communication media, low bandwidth, low maintenance, multi-vendor equipment and low support costs. The LonTalk protocol is based on the 7-layer OSI model. In order to be supported over multiple transmission media the protocol supports different transmission speeds up to 1.25 Mbps.

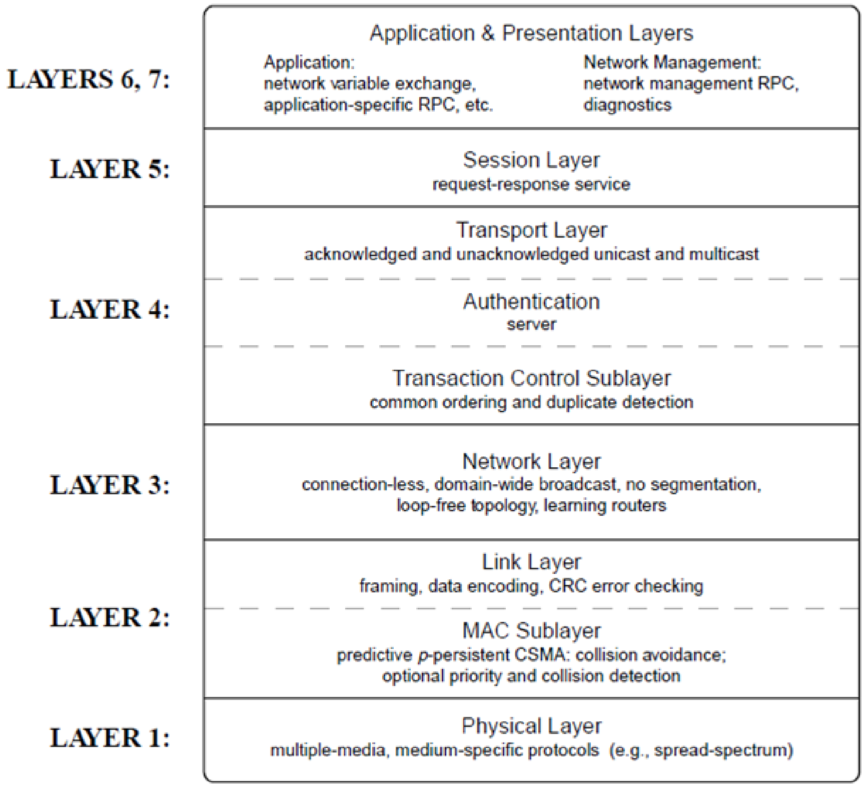

LAYERS 6, 7:
Application & Presentation Layers
Application: network variable exchange, application-specific RPC, etc.
Network Management: network management RPC, diagnostics
LAYER 5:
Session Layer
request-response service
LAYER 4:
Transport Layer
acknowledged and unacknowledged unicast and multicast
Authentication
server
Transaction Control Sublayer
common ordering and duplicate detection
LAYER 3:
Network Layer
connection-less, domain-wide broadcast, no segmentation, loop-free topology, learning routers
LAYER 2:
Link Layer
framing, data encoding, CRC error checking
MAC Sublayer
predictive p-persistent CSMA: collision avoidance; optional priority and collision detection
LAYER 1:
Physical Layer
multiple-media, medium-specific protocols (e.g., spread-spectrum)

Figure 1 - LonTalk protocol stack

b. Addressing

Addressing is of a hierarchic structure, with three basic address types:

[Domain, Subnet, Node]
[Domain, Subnet, Neuron_ID]
[Domain, Group, Member]

The domain and subnet components are used in routing, whereas the last component is rather a name than an address since it doesn't change when the node is moved. The subnet and group fields are 8 bits and the node field is 7 bits, this gives ~2^15 nodes per LonTalk domain.

The **domain** identifier is the first component of the addressing hierarchy, its field size is usually 48 bits, but there might be also used smaller domain identifiers in certain circumstances when there are limitations. A domain is a virtual network that serves as the unit of management and administration, its communications are limited within the domain itself.

A **subnet** is a subset of the domain which consists of nodes for which there is no need of routing to communicate amongst them. Subnets are a routing abstraction for a channel, however one or more subnets can be mapped into a channel if they are connected by repeaters or bridges.

The **node identifier** identifies a logical node within a subnet. A physical node may belong at most to two subnets which, in turn, are in different domains.

Group addressing allows the one-to-many communication and supports the concept of network variables used in the LonTalk application protocol.

The 48-bit Neuron_ID is unique worldwide and is set at the time of node manufacture. An unconfigured LonTalk node has no assigned address except its 48-bit Neuron_ID. These unconfigured nodes receive packets from all domains looking for and responding to any packet containing their Neuron_ID.

c. Network access

LonTalk uses Predictive p-persistent CSMA method to access the network which is a collision avoidance technique that randomizes channels access using knowledge of the expected channel load. A node that wishes to transmit always accesses the channel with a random delay. To avoid throughput degradation under high load the randomization window is proportional to the channel *backlog*.
LonTalk also supports optional priority for channel basis. The number of priority slots per channel ranges from 0 t 127. Not all the priority slots are mandatory to be utilized, a channel decides on a message by message basis whether to use the assigned priority slot.
When collision detection is enabled the MAC sublayer acts as the following:

1. If a collision is detected after two successive attempts to transmit a priority packet in the node's priority slot, the next transmission of a priority packet will not use the priority slot, but a normal non-priority slot.
2. Whenever a collision is detected by a transmitting node, that node increments the estimate of the channel backlog by 1.
3. Whenever a collision is detected on 255 successive attempts to transmit a packet, the packet is discarded.

d. The link layer

The **link layer** provides subnet-wide, ordered unacknowledged packed delivery with error detection, but no error recovery. A corrupted frame is discarded as soon as its CRC check fails. The CRC is generated using the polynomial $x^{16} + x^{12} + x^{5} + 1$.

e. The network layer

The protocol supports a variety of topologies in order to meet the application's requirements. Within a single channel the topology can be a bus, ring, star or free as depicted in the picture. The protocol supports physical repeaters, store and forward repeaters and bridges.

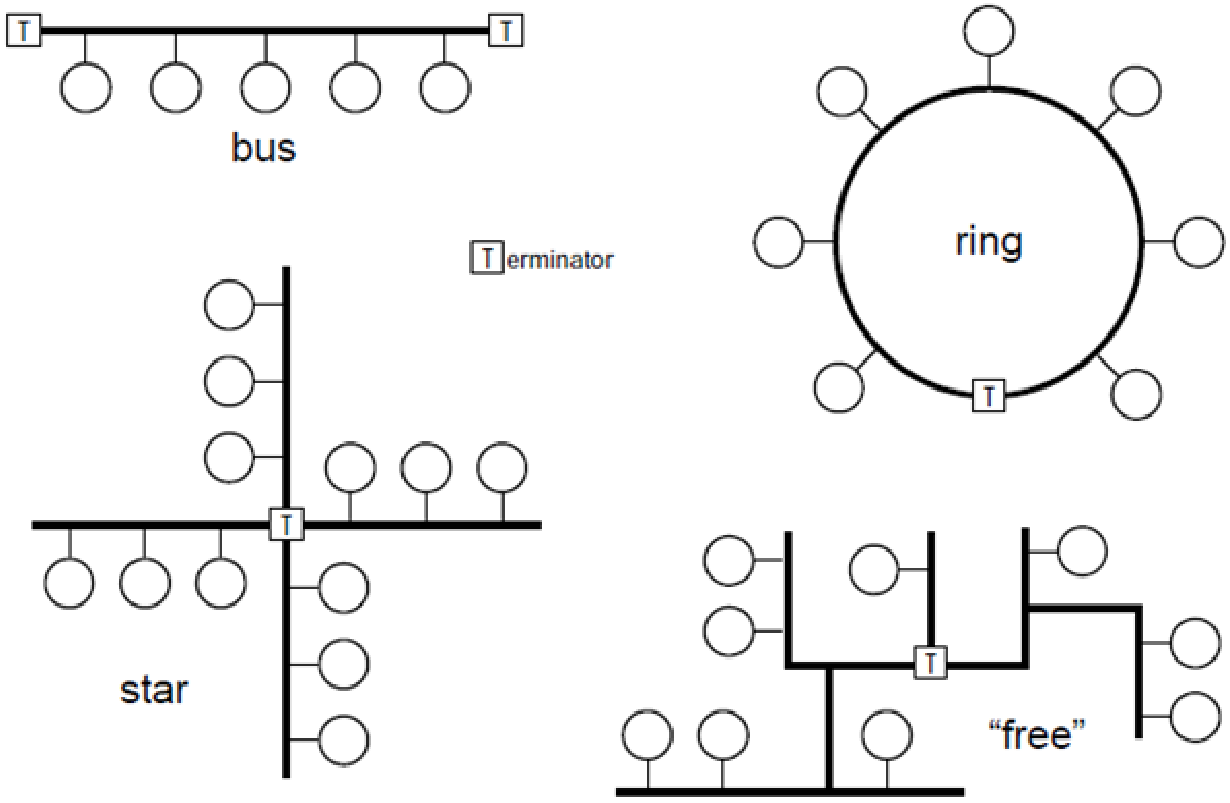


Figure 2 - LonTalk possible network topologies

The Network Packet Data Unit (NPDU) carries and envelops either a Transport Protocol Data Unit, Session Protocol Data Unit, AuthPDU or Application Protocol Data Unit.

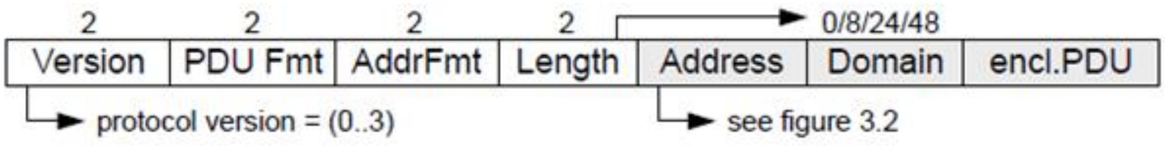


Figure 3 – Network Protocol Data Unit structure

f. Transport layer data units and types

The transport layer works with the following message types:

- Acknowledged Message – used for the first transmission of a message, it must be acknowledged by all addressed recipients
- Unacknowledged-repeated Message
- Unacknowledged Message
- Message Reminder – facilitates the selective soliciting of acknowledgements for multicast transactions

g. SNVT-s

LonTalk aims to provide interoperability among devices of different types and vendors. When an application is created in the Application Layer of the protocol stack the network variables are defined and they are later on shared by multiple nodes. By only allowing links between inputs and outputs of the same type, network variables enforce an object-oriented approach to product development. This greatly simplifies the process of developing and managing distributed systems. Whenever a node program writes a new value into one of its output variables, the new value is propagated across the network to all nodes with input network variables connected to that output network variable. This action is handled by the protocol within the Neuron Chip.

Specifically, LonWorks achieves the interoperability by using Standard Network Variable Types (SNVT). Echelon maintains a growing list of over 100 SNVTs for nearly all physical measurement types including the type of variable such as integer or floating point. For example, a SNVT for continuous level is defined as SNVT_lev_contin. If a network input variable and a network output variable are defined with the same SNVT when the application is created, they can be connected together in the network through a process called binding. The unit of a SNVT cannot be changed, additionally each SNVT contains self-documentation strings.

Variable Type	**Units**
Temperature	Degrees Celsius
Relative Humidity	Percent
Switch State	Boolean
Device State	Boolean
Day of Week	Enumerated List (Mon-Sun)
Real Time	MM,DD,YYYY
Elapsed Time	Second, Milliseconds, Days or Hours
Even Count	Counts

Table 1 - SNVT examples

4. Usage

The LonWorks technology is widely used in the home automation domain, industrial transportation, public utility control networks. Applications in the home automation domain vary from lighting and energy control systems to heating / ventilation / air-conditioning systems. Being a proprietary technology signifies high reliability and support, but, higher costs as well. This technology is hardware specific and still very far from achieving interconnectivity with Windows OS-s.

Learnings

Reflections of what we learned on the code camp

- Presentation of the product

Despite presenting a fully functional prototype that employs the latest technology to achieve inter-communication between home and personal devices, companies are not driven just by how "cool" a solution sounds. Instead, they require concrete figures on how much the solution will affect *their profit*. We learned that estimations of the costs of deployment, maintenance and the return on the investment are equally crucial to have the proposal considered.

FHEM

The FHEM platform is a very useful tool, as it allows the user to integrate different devices with different platforms into one system, thus tackling the lack of home automation technology standards and agreements. Another great characteristic of this platform is that it is an open source tool. This means that anyone can change even core modules to their own liking and needs; besides contributing directly to the evolution and development of the package. This feature has led to the existence of many extension modules of FHEM, that integrate other external services, through their APIs, or add functionalities to the platform. After getting acquainted with the system internals, it is quite easy to benefit from the flexibility that it offers and create a simple functioning environment with the desired features.

But not everything is smooth and neat on the platform's learning curve. First there is Perl. This old regular expression oriented and functional programming language (PL) lacks the user friendliness of other interpreted languages like Python or the speed and reliability of C++. Not many people have coded before in Perl, so it requires a small learning process and afterwards, it won't become your favorite PL, for sure. Another unpopular characteristic feature of FHEM was the lack of good documentation, at least in English (given that there are mainly German articles for it on the internet). This occurs given the open source nature of the platform and it is shared with most other projects that are developed by the work of independent volunteering contributors. Another stressful characteristic of FHEM is the issues that may arise during the pairing of some devices. In our case this happened mainly with some HomeMatic equipment, that wouldn't react to the pairing procedure for hours, just to start functioning without a problem on the next day. This may have occurred because of the number of active devices in the Home Automation Camp classroom, but the feedback on why the pairing was not working was insufficient or non-existing.

On a more subjective opinion, we felt that, for being an open source tool, FHEM is actually a great initiative. In less than 3 days of coding we managed to develop a working prototype that involved 10 devices, with 2 different RF protocols (HomeMatic and FS20). Even the integration of external services, such as Telegram and Google Calendar, as the integration of an IP network camera was done easily enough. Definitely this platform has the potential of being used to integrate Home Automation technologies and leverage the problem of the lack of integration capabilities of all the devices' brands.