Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science

BACHELOR PROJECT ASSIGNMENT

Student: **Tomáš Pikous**

Study programme: Open Informatics
Specialisation: Software Systems

Title of Bachelor Project: **Local Area Networks for IoT**

Guidelines:

Review and compare the main IoT technologies. Focus on the Local Area Networking technologies Physical level i.e. WiFi, 6lowpan, Lora, etc., review different topologies star, mesh network, and communication protocols such as MQTT, CoAP, REST, Thread etc.

Chose two types of reviewed technologies and design and implement a complete network solution for reading sensor and controlling actuator. For example switching on lights and reading temperature.

Assume that the local IoT HUB uses the standard open source Zetta server implementation with the Siren hypermedia type describing the local set of resources.

Implement, test the two selected combination of solutions. Measure the main parameters.

Bibliography/Sources:

Restful Web Apis , Richardson, Leonard; Amundsen, Mike; Ruby, Sam, Oreilly & Associates Inc
The Internet of Things, Samuel Greengard, The MIT Press Essential Knowledge series Paperback – March 20, 2015

Bachelor Project Supervisor: Ing. Jan Šedivý, CSc.

Valid until the end of the summer semester of academic year 2016/2017

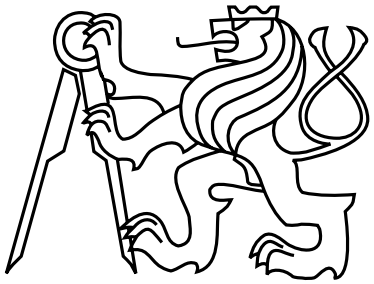
prof. Dr. Michal Pěchouček, MSC.
Head of Department

L.S.

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 10, 2016

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Bachelor's Project

Local Area Networks for Internet of Things

Tomáš Pikous

Supervisor: Ing. Jan Šedivý, CSc.

Study Programme: Open informatics, Bachelor's degree

Field of Study: Software Engineering

May 24, 2016

Aknowledgements

I would like to thank my advisor, Ing. Jan Šedivý CSc., for guidance and being helpful whenever it was necessary, and I would also like to thank my family for support throughout my whole life.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 18, 2016 .....

Abstract

Internet of Things is a largely growing topic nowadays and even bigger expansion is expected in the future. This thesis focuses on communication methods for IoT devices to collect and exchange data from sensors and to control actuators. Two wireless technologies were selected and experimentally tested in a practical use cases related to the interior lightning.

Abstrakt

Internet of Things je v současnosti velmi rychle rostoucím tématem a očekává se, že nárůst tohoto tématu se bude v budoucnosti zvyšovat. Tato práce je zaměřena na komunikační metody pro IoT zařízení, které mohou sloužit pro sběr či výměnu dat, nebo ovládání aktuátorů. Byly vybrány dvě bezdrátové technologie a experimentálně otestovány v praktických případech použití z oblasti osvětlení interiérů.

Contents

List of Figures

Chapter 1

Introduction

In the past few years, the Internet of Things [20] had become a very popular topic. It is described as a network of embedded devices that allows those devices to collect and exchange data. There are more and more IoT devices available and it is estimated, that there will be up to 40 billion devices connected by 2020 [19]. Those devices could be connected and interoperate within existing Internet infrastructure to create smart homes, offices or even cities. There are also big expectations of Industrial IoT which could possibly lead to fourth industrial revolution called Industry 4.0 [23], where cooperation of various IoT devices such as sensors and actuators, should minimize factory downtime and reduce maintenance expenses, which would lead to productivity increase.

History of IoT [30] dates back to 1980s. The idea of a network of smart devices was first suggested and then popularized by Kevin Ashton in 1999, who also invented the term *Internet of Things*. Since then the IoT made a huge step forward thanks to the miniaturization of electronics, which has lower power requirements, but also thanks to the improvement of the Internet infrastructure etc.

The Internet of Things itself is currently on top of Gartner's hype cycle for emerging technologies [5] (Figure 1.1), but it's estimated that it will reach production (plateau) phase in 5 to 10 years with potential to change the way how the society works. In cooperation with Big Data and machine learning it can automatize or optimize processes that are now inefficient, which would reduce environmental impact. Industrial IoT will be used to predict failures, optimize manufacturing processes, improve accuracy and therefore result in an economic benefit.

1.1 Motivation and goal of the thesis

There are more and more IoT devices on the market nowadays and majority of manufacturers have their own system for management of those devices, which is understandable from their point of view, but the IoT includes wide range of devices and those devices needs to be connected together to fulfill tasks mentioned before. It is almost impossible for a single manufacturer to cover that wide range of devices, which results in a natural need to standardize communication of IoT devices, in order to create systems composed of devices from different manufacturers.

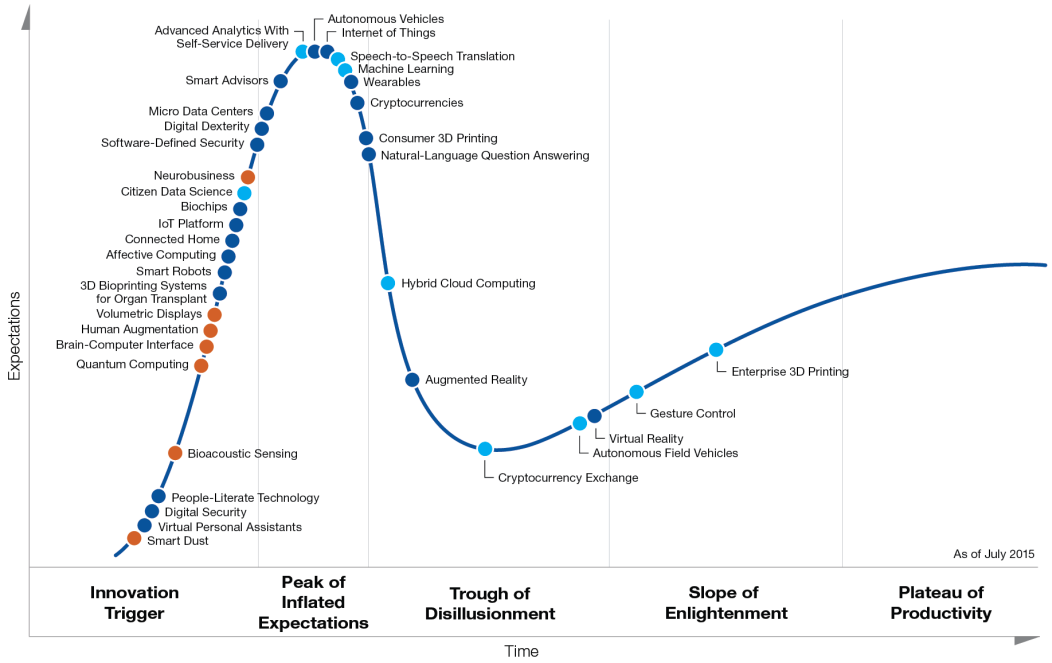


Figure 1.1: Gartner's Hype Cycle for Emerging Technologies, 2015

With increasing number of IoT devices, the efficient management of those devices will be more important. When the user brings home the new device, the system should configure itself automatically or at least with minimal effort from user. Next important feature of such system is a smart device access rights management and ownership. For example, in an office environment is not needed for all devices (e.g. security devices) to be publicly accessible, but access to the other devices such as lights or thermometers can be unrestricted.

I have worked on an IoT platform supporting described functionality. The basic architecture of our system is shown at figure 1.2. The system consists of smart devices (sensors, actuators etc.). HUBs are devices connecting sensors to the Internet. They are located at homes, offices or factories, Cloud server and clients. Clients can be smartphones or any other devices such as tablets, computers etc.

This thesis will focus on analysis of possible communication means between Hubs and smart devices in home and office environments.

There's been a great progress in wireless networks recently, especially those with low power requirements, so it can be assumed that majority of the IoT devices is going to be wireless, because wireless infrastructures are easier and cheaper to build than the wired ones, and in most cases they also offer mobility of devices without modifying the current infrastructure, which wouldn't be possible at wired networks.

We are going to describe suitable wireless technologies. We will demonstrate the technologies by implementing solutions for controlling devices such as smart lights and sensors using two of the reviewed wireless technologies.

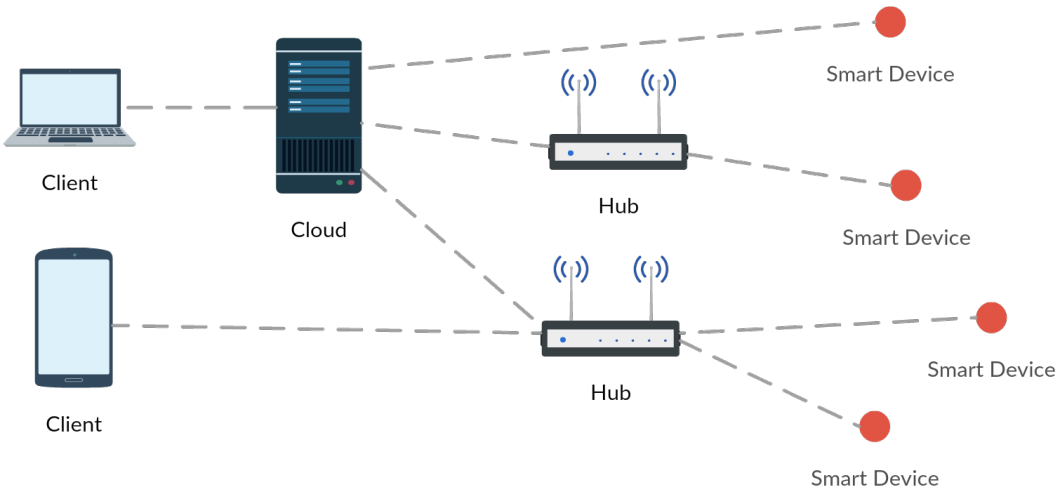


Figure 1.2: IoT system architecture

1.2 Thesis structure

Chapter **Wireless network topologies** provides quick look at the most significant wireless network topologies that can be used in our IoT system.

Packet routing in mesh topology networks is a chapter describing routing algorithms used for routing in such networks. Two major groups are described here, including their working principles.

Next chapter contains brief look at **Modifications of packet routing algorithms** including implementation-specific changes.

Description of **ISO-OSI model** is needed, because in comparison of wireless technologies, the completeness of implementation of ISO-OSI stack in available wireless technologies is an important factor, that should be taken into account.

IoT protocols is a chapter where application protocols mostly suitable for IoT devices are summarized.

Chapter **Wireless technologies** takes a look at some of the technologies that could be used in our IoT system. They are divided into two groups according to their range. Main parameters are mentioned for later comparison.

Representational state transfer API is a chapter describing modern, data-centered architecture with many benefits, that allows simple communication between server and clients.

Problem solution, this chapter contains detailed description of problem and its solution.

Chapter **Experiments** is dedicated to the description of experiments performed with the implemented solution.

Chapter 2

Wireless network topologies

Network topology [28] is an arrangement of devices and their connections in network. When we think of home or office IoT network, the topology is a very important factor, because it influences the complexity of nodes and system robustness. Topologies, that we consider as suitable for an IoT systems, are described below.

2.1 Description of topologies

2.1.1 Point to point

This topology provides dedicated link between two nodes. Connection can be permanent or switched, which means that direct link is established on demand. This topology, in a network context, is best suited for temporary data sharing. It's considered to be the simplest of the topologies, but not very useful in system with higher count of nodes.

Figure 2.1: Point to point topology

2.1.2 Star

Nodes in this topology are divided into leaf nodes and central node. Each leaf node is connected to the central node, which is switching messages between them. This topology requires the central node to be quite powerful to handle requests from leaf nodes in real time, but on the other hand, leaf nodes can sleep when they're not needed. Failure of connection between one of the leaf nodes and central node doesn't affect other connections. The main disadvantage is that none of leaf nodes will be able to communicate with other leaf nodes in a case of central node failure.

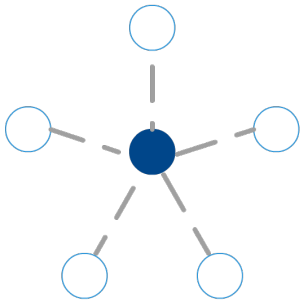


Figure 2.2: Star topology

2.1.3 Mesh

In this topology, each node is able to act as a router and relay data and therefore all nodes together cooperate in the distribution of data, which can be distributed by flooding or routing techniques. Benefit of this topology is that there can be multiple paths between two nodes, which can be used for increasing throughput or reliability. Node's ability to act as router can be disadvantage when we consider power requirements, because higher computing power is needed, which naturally leads to higher power consumption.

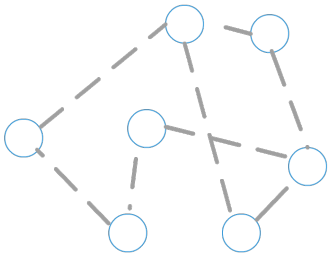


Figure 2.3: Mesh topology

2.1.4 Tree

Tree topology contains leaf nodes and router nodes. No nodes can be connected to a leaf node, but on the other hand, both types of nodes can be connected to a router node. This topology allows nodes to talk to each other and in case of router failure, just the subset of network is affected. When we think of power consumption, only the router nodes has higher power requirements, leaf nodes can be sleeping and wake up and transmit data just when it's needed. That means leaf node can be powered for example by energy harvesting methods and wake up periodically or in a case of external interrupt (alarm etc.).

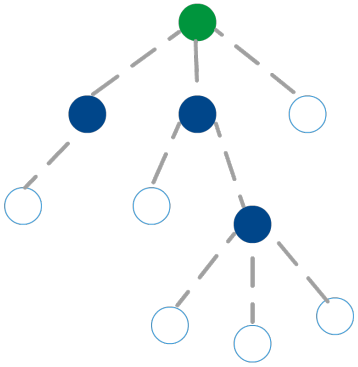


Figure 2.4: Tree topology

2.2 Conclusion

When we focus on wireless communication in our system, it can be clearly seen that there are multiple possible approaches. In each case the Hub is a central node that provides connection of IoT system to the Internet and therefore to the Cloud server.

The simplest approach is to use **star topology** with Hub as a central node, but the limited wireless range can be a disadvantage of such solution and therefore we would like to add devices, that would be cheaper than the Hub, but which could act as range extenders (relays). Those devices can be designed for different primary use than range extension, but they should have sufficient power supply, because higher computing power is needed for routing. This prevents those devices to be ultra low powered or even powered via energy harvesting methods.

This divides IoT devices into two groups according to their power requirements and thus their ability to act as wireless relays, which brings us to the **tree topology**, where the Hub would be the root node and end devices would be connected directly to the Hub or via relay node.

Last, but not least improvement would be usage of principles from **mesh topology**. This should allow bypassing unreliable connections or splitting communications between multiple paths to increase network's throughput. The only disadvantage is higher complexity of packet routing in such networks.

From observations made in the text above, we consider star, tree or mesh topologies and their combinations as most suitable for our system.

Chapter 3

Packet routing in mesh topology networks

Mesh topology itself was described in previous chapter, but the interesting part of those networks is the task how to get packets to their destinations. There are two major groups of routing protocols that are used in a networks with mesh topology [24].

3.1 Distance Vector routing protocol

This routing protocol was the original routing protocol of ARPANET. The great advantage is that every node informs just it's neighbours, so there is no need to flood network with overhead messages as in link-state based protocols. Not knowing the entire path results in a lower computation complexity. During routing, each node knows just the packet's distance from destination and direction (neighbouring node) in which the packet should be sent. Distance vector comes from the fact that each node maintains vectors of distances to all nodes in network. This routing algorithm uses Bellman–Ford or Ford–Fulkerson algorithms to compute the best paths in the network.

3.1.1 Working principle

Each node knows just its neighbours and cost of link. Cost can be calculated from network bandwidth, RSSI etc. During periodical updates, each node shares its routing table with neighbours. Node (A) that receives routing table from another node (B) then tries to update its routing table. Cost of path between nodes A and B is then added to other costs in the routing table, which was received from B. Node A updates its routing table using the modified table received from B. After this step, the new direction (neighbour) with lowest cost is selected for each destination node in network.

3.1.2 Problems

3.1.2.1 Routing by rumor

This term describes the fact that node itself cannot verify costs in its routing table, except direct one hop neighbours. It relies on information from other nodes. It is not so much a problem as it is a feature, but there are several ways how to increase the routing stability.

3.1.2.2 Count to infinity

This problem occurs in a following situation: We have three nodes connected like A-B-C. It the node A goes offline, the node B updates its routing table, because node A won't be sending any packets anymore. But when the node C, still not knowing that node A is offline, shares routing table with node B, the B will think that the node A is just two hops from C (C-B-A) and it will update its routing table, because it doesn't know that the path goes through the B itself.

The path cost to A, from the point of view of node B will be now 2+1. But the problem continues. When the node B will share routing table with C, the C is going to update cost to node A because it is reachable through B and the cost through B has increased. So new cost of travelling to the node A from node C will be 3+1. This problem continues and propagates through network until it reaches infinity, when it self corrects thanks to relaxation property of Bellman-Ford algorithm.

There are several techniques how to prevent this behavior and this problem is solved in some implementations of this routing protocol.

3.2 Link state routing protocol

This is the next protocol used for routing in the mesh notworks. When it is used, each node informs all nodes in the network about the set of its neighbours, including changes etc. We can say, that node informs the world about its neighbours by flooding the network, unlike the Distance Vector routing protocol, where each node informs just neighbouring nodes. Each node builds a map of network on its own. This map is in a form of tree shaped graph and it contains all the connections between nodes and routing tables are generated from this graph afterwards.

3.2.1 Link-state advertisements

Every single node has to determine its one hop neighbours and then periodically broadcast the "link-state" advertisements. Such advertisement can contain node ID, its neighbours and increment.

Those advertisements are flooded through the network. Recipient of such advertisement looks up the latest record it has for advertisement's source node and compares their increments. Received advertisement is thrown away if it has lower or equal increment. If the increment is higher, recipient's information about the neighbours set of the source node is updated and the advertisement is forwarded.

3.2.2 Creating a map

When the node has complete set of advertisements it will produce tree shaped graph of network using greedy algorithm (variant of Dijkstra), where the cost of hop may consist of network bandwidth, RSSI etc.

3.2.3 Routing tables

Shortest path to any node is given by nodes traversed in order to get from the destination node to the root. For every node, the best “direction” is the one hop neighbour from root on a shortest path to the destination node. This can be done while determining shortest paths, because this node giving direction is one hop before the root node (while traversing from upwards).

3.2.4 Notes

Each link between two nodes has to be agreed by both ends, unlike in distance vector routing. Routing is not based on "rumor" as in distance vector, because every node has complete set of advertisements from all other nodes in the network at the time it starts with creation of routing tables. Routing loops are possible if nodes aren’t working with exactly the same map (link-state advertisements were corrupted etc.).

Chapter 4

Modifications of packet routing algorithms

Routing algorithms described in the previous chapter were split into two main classes, but in the text below the various modifications of those algorithms which are described.

4.1 Distance Vector routing protocol

4.1.1 Destination-Sequenced Distance Vector routing (DSDV)

This modification [27] of distance vector routing algorithm uses Bellman-Ford algorithm. Except of cost, each entry contains sequence number and timeout. If the node's neighbours set changes, the sequence number is increased.

Change is then propagated through network as in normal distance vector routing, but during updates of routing tables, the value with higher increment is taken. If the increments are equal it behaves like normal DV, that means that the path with lower cost is taken. Each entry also contains timeout (or install time), which can be used for determining the nodes that weren't updated for a while, so they can be deleted.

4.1.2 Ad hoc On-Demand Distance Vector routing (AODV)

This is a demand driven protocol used for routing [26] in mobile ad hoc networks and it is used, for instance, in wireless technology ZigBee. It is a loop free protocol and also avoids "count to infinity" problem. This algorithm finds routes only if needed and it's caching them in route tables. This approach reduces complexity of routing.

4.1.3 Babel

Babel [17] is based on ideas of previously mentioned routing protocols (DSDV and AODV). It never creates a routing loops or the loop disappears as soon as one update went around the network. Babel was designed primarily for wireless networks, thus is extremely robust in presence of mobility.

4.2 Link state routing protocol

4.2.1 Optimized Link State Routing Protocol (OLSR)

In this modification [1] of link state routing protocol, some nodes in the network are chosen to be dynamic multi-point relays. This approach increases the network data throughput by creating an efficient network routing scheme.

Just a subset of neighbouring nodes relays data instead of every node acting as a relay in standard link state protocol. This technique minimises rebroadcasting and the number of control messages required to build the routing tables. Multi-point relays are elected in such way, that every node can communicate with some multi-point relay within one hop.

The network information is shared between those relays to maintain routing paths, which allows every multi-point relay to have a complete routing table while simultaneously minimizing the number of link-state advertisements that are flooded through network.

4.2.2 Open Shortest Path first (OSPF)

OSPF [25] is a open, loop-free implementation of link-state protocol concept and it is defined in RFC2328. This protocol has fast convergence rate, because changes are sent immediately and only the changes in routing tables are sent instead of the whole table. Metric used in determining path cost is an inverse of bandwidth. OSPF is also able to load balance network traffic by utilizing multiple paths with same cost.

Chapter 5

ISO-OSI model

ISO-OSI model [18] is a concept, released by ISO organization as a standard ISO7498. It is a general purpose paradigm for communication between two computers, which divides communication into 7 layers, where each layer uses layer below and serves to the layer above. Completeness of implementation of ISO-OSI stack needs to be taken into account while comparing wireless technologies, because different wireless technologies are implementing various subset of ISO-OSI model (some layers may not be implemented etc.).

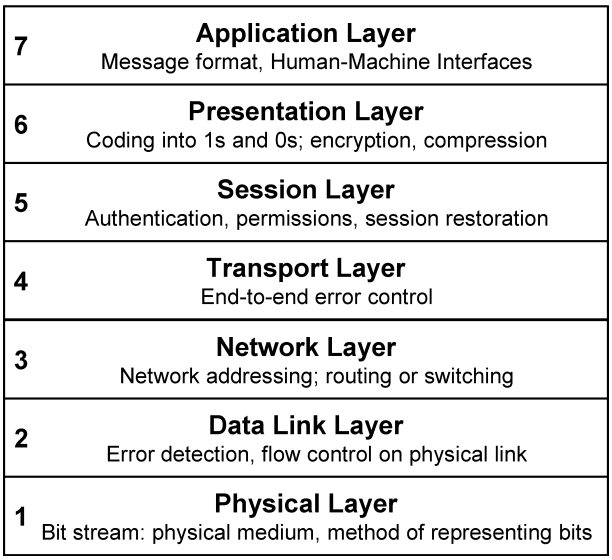


Figure 5.1: ISO-OSI stack

5.1 Description of layers

5.1.1 Physical layer

This layer specifies communication on the lowest level. It defines transmission medium, which can be metallic cable, optical cable or radio. Next properties specified in this layer are

signal characteristics, bits encoding, network topology and transmission mode (half duplex, full duplex).

5.1.2 Data link layer

Data link layer takes care of communication between two directly connected nodes, which includes error correction and flow control. Connection establishment and termination are also defined by this layer together with addressing Well-known standard IEEE 802 divides this layer into two sublayers: Media Access Control and Logical Link Control layer.

5.1.3 Network layer

This layer is responsible for packet forwarding including routing via one-hop neighbours. Network layer also manages the quality of service.

5.1.4 Transport layer

Transport layer may include features such as multiplexing, which means that data can be multiplexed using ports. When a process listens on a specific port, only packets addressed to that port will be delivered to that process.

Packets can be delivered in different order than they were sent. This may occur due to, for example, different paths of each packet during routing in lower layers. To solve that problem, the the Transport layer may include useful feature called **same order delivery**, which ensures that packets will be delivered in the same order as they were sent. Next useful features like flow control, automatic request repeat (reliability) etc. can be also implemented.

Mostly used protocols, operating on this layer are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

5.1.5 Session layer

This layer controls dialogues between computers and it is responsible for session checkpointing and recovery. In a case of connection loss, it may try to recover connection or, in other case, if the connection is not used for a long time, it can be temporarily closed. Session Layer provides synchronization points for streams and can be used for remote procedure calls.

There are several protocols designed to work on this layer, like NetBIOS, RPC, SCP etc.

5.1.6 Presentation layer

This layer has the task of transforming data into format which is used by applications. It takes care only about structure, but not about the content. For example transforming from little to big endian or text formatting EBCDIC to ASCII etc. Mentioned task also includes serialization and deserialization of objects.

Despite the fact that data encryption and decryption can be done even on Transport Layer of Application layer, it is commonly done by this layer. Representatives of protocols used on this layer are the Telnet, LPP or NDR.

5.1.7 Application layer

Application layer is an user interface used for displaying received information to the user. It specifies protocols and interfaces used by hosts. Well known protocols used in this layer are Hypertext Transfer Protocol (HTTP) or File Transfer Protocol (FTP).

5.2 Internet protocol suite (TCP/IP)

It is a computer networking model, which was developed during 1960s as part of ARPA effort to build a nationwide packet data network. TCP/IP is quite similar to OSI model, thanks to the fact that layers from ISO-OSI model were merged into four layers, thus the overall functionality is very similar.

Chapter 6

IoT protocols

In this chapter, we are going to take a look at protocols, that are frequently used for communication with Smart Devices.

6.1 Protocols description

6.1.1 Message Queue Telemetry Transport

Message Queue Telemetry Transport (MQTT) [7] is a lightweight messaging protocol developed by Andy Stanford-Clark and Arlen Nipper from Cirrus Link Solutions Company in 1999. It runs on TCP protocol and thanks to its small code footprint it can be used even on simplest devices with limited bandwidth.

MQTT uses publish subscribe pattern, that means the network is divided into broker and clients. Clients can be IoT devices which are connected to he broker. Each client can be subscribed or publish on channels called topics. The broker is responsible for delivering messages published by one client to other clients subscribed to that topic. MQTT is space and time decoupled, which means, that clients doesn't need to know anything about other clients and their states. There are many MQTT brokers available such as open-source Mosquitto or HiveMQ.

This protocol is loose coupled, therefore the modules (servers, clients) can be independent, but on the other hand it can overload network by overhead messages. MQTT also offers Quality Of Service (QoS) which is level of guaranteed message delivery. Well known application using the MQTT protocol is the Facebook Messenger.

6.1.2 Constrained Application Protocol

Constrained Application Protocol (CoAP) [29] is also aimed for use on a very simple embedded devices. It is an open protocol, defined in RFC7252[1], running on to of the UDP protocol, which allows smaller packets and lower overhead, but it is generally less reliable than TCP.

[1] https://tools.ietf.org/html/rfc7252

CoAP messages are divided divided into requests and responses. Messages can be addressed to specific device or multicasted. CoAP is able to run on IEEE 802.15.4 networks, which will be mentioned late, and messages should fit into one frame when the 6LoWPAN extension is used.

Reliabilty is a CoAP alternative to QoS in MQTT, which determines whether the message reception has to be confirmed by other end, but unlike in MQTT it doesn't care about possible message content corruption.

Usage of Constrained Application Protocol is expected on Thread network which will be mentioned later.

6.1.3 Extensible Messaging and Presence Protocol

Extensible Messaging and Presence Protocol (XMPP) is an open, community based standard which was firstly used for instant messaging (Jabber) at the beginning of its life. This protocol runs of the top of TCP and uses client-server architecture.

It allows sending messages and status retrieval. Messages are structured in XML format. XMPP doesn't offer any form of QoS itself. Quality Of Service must be realized on top of the XMPP layer.

For the needs of the Internet of Things, special release called XMPP-IoT is being developed. XMPP-IoT should become a competitor for previously mentioned protocols. It should improve provisioning, power consumption or explore possible usage of Efficient XML Interchange Format (EXI[2]).

6.1.4 Advanced Message Queuing Protocol

Advanced Message Queuing Protocol (AMQP) is also an open standard application layer protocol running on TCP, but it is not so widespread as previously mentioned protocols. AMQP was approved as an International Standard (ISO/IEC 19464) in 2014.

This protocol is pretty similar to the MQTT, because it offers publish-subscribe pattern, Quality Of Service etc. But it also offers some features, that MQTT doesn't have, like transactions etc. Which predetermines the AMQP usage mainly for enterprise deployment (e.g. It can be used between hubs and cloud on a large scale).

6.2 Conclusion

Protocols mentioned above are in different phasees of development, but the MQTT and CoAP are the most mature for immediate deployment for communication between sensors and hub. On the other hand, the AMQP seems to be suitable protocol for communication between hubs and cloud server, but it is not the aim of this thesis.

Both MQTT and CoAP are, rapidly evolving, leading protocols for Internet of Things, but the MQTT is currently more mature and stable than CoAP. It is easier to get the MQTT network up and running than same CoAP network.

[2] https://www.w3.org/TR/exi/

MQTT offers more advanced features than CoAP, which is, on the other hand, more lightweight. This is given by transport layer protocol they're using. TCP, used by MQTT, delivers reliability, but increases overhead messages and overall complexity, whether UDP, used by CoAP, has opposite properties. This results in shorter wake-up times and less overhead messages, which will, among others, reflect in battery life.

MQTT uses star topology with broker as a central node, which perfectly fits in our system. MQTT broker could be running on the Hub which would also serve as a border router to the Internet. This approach delivers single point of failure to our system, but it was already designed that way and this problem must be resolved by other methods.

Finally, we would like to conclude, that both MQTT and CoAP are reasonable choices for Smart Devices. CoAP is a simple and lightweight protocol, but MQTT is more mature and offers advanced features at the expense of higher computation requirements.

Chapter 7

Wireless technologies

This chapter is dedicated to comparison of available wireless technologies, that could be possibly used for communication between Smart Devices and Hubs in our IoT system.

7.1 Short range

Technologies with range up to hundreds meters will be described in this section.

7.1.1 WiFi

WiFi is a widespread technology, which can be found in many electronic devices. Communication via WiFi allows simple integration, because there is usually an infrastructure of access points. In other case, building new WiFi infrastructure isn't complicated. WiFi allows high data rate compared to other technologies, which is redeemed by higher power consumption. Disadvantage is lower indoor range due to usage of 2.4 GHz frequency.

7.1.2 Bluetooth Low Energy

Bluetooth Low Energy (BLE) was introduced around 2010. BLE provides lower consumption in comparison with WiFi, but also lower data throughput, which may not be a disadvantage when we consider size of data from most of sensors. Useful feature is an advertising mode, which allows device to act as "beacon" and broadcast small pieces of information without establishing the ad-hoc connection. BLE is also using 2.4 GHz frequency which means that indoor range isn't so good.

7.1.3 IEEE 802.15.4

IEEE 802.15.4 is a standard for low rate WPANs. It is basis for several wireless technologies such as ZigBee, Thread, MiWi and WirelessHART, but it can be also used with 6LoWPAN extension and standard Internet protocols. It is focused on low cost and low speed networks, which are ideal for Smart Devices.

It is able to operate on three frequency bands: 868 MHz (EU), 915MHz (USA) and 2.4 GHz (Worldwide) in majority of available solutions. Ability to operate on sub-ghz bands ensures better indoor range than 2.4GHz with same modulation and output power. IEEE 802.15.4 defines two types of nodes: full-function device (FFD) and reduced-function device (RFD). Full-function device can act like network coordinator, relay messages, but it can also behave like RFD. Pure RFD nodes are very simple and cannot relay messages and therefore can be cheap and have low power consumption. IEEE 802.15.4 covers physical and data link layers from OSI model.

7.1.3.1 6LoWPAN

Name of this concept is acronym of IPv6 over Low power Wireless personal area networks [22]. It allows to send IPv6 packets over IEEE 802.15.4 networks, which means that Internet Protocol packets can be delivered directly to the simplest sensors and actuators with limited resources. 6LoWPAN defines encapsulation and header compression. 6LoWPAN together with IPv6 fulfills the role of network layer in the OSI model.

7.1.3.2 Thread

Thread [14] is an open, low bandwidth standard using 6LoWPAN. It was proposed in 2014 by Thread Group as royalty-free protocol. This technology is designed for usage, where is connected up to 250-300 devices. It should have low latency (less than 100ms per typical interaction), maximum data rate of 250kbps and connections between nodes are always secured.

All devices in the network are directly addressable and because of Thread's power efficiency, even energy harvesting sensors should be possible. Network configuration and maintenance is supposed to be simple and user-friendly. This technology covers network and transport layers and it uses 6LoWPAN and IPv6 as network layer (together with distance vector and DTLS). UDP and TCP protocols can be utilized in transport layer.

Thread Group claims that network uses mesh topology and has distance vector routing protocol and consists of border routers, router eligible end devices and end devices. This seems to be more like combination of mesh and tree topologies, because devices are divided into routers and end-devices.

Border routers are gateways connecting Thread network to the Internet (Ethernet, WiFi etc.). Border routers allows accessing nodes from outside of the network and multiple border routers can be connected. Router eligible end devices are capable of routing packets but this functionality is not enabled yet. In a case of need it can become Thread router and route messages across network. Network typically has up to 32 active routers. Thread router maintains its state to all other routers and border routers using MLE messages and Trickle algorithm. Thanks to this behavior all routers have up-to-date paths and path costs information and on-demand route discovery is not needed. Path cost is determined from RSSI of received packet. One of routers is self elected Leader, which makes decisions in the network. Those decisions could be router addresses assignment or new router requests. End devices, for example sensors, aren't capable of routing messages, they communicate through

their parent router and they are are supposed to be in the sleep mode most of the time in order to save energy.

In a Thread network, none of devices represents single point of failure, all devices can be replaced in a case of malfunction without impacting ongoing communication. Each node has IPv6 address, but it is shortened within the Thread network to 16-bit address composed from node's parent router id and child id. To ensure proper function of whole network, every commercially available device must pass through certification program.

7.1.3.3 ZigBee

This IEEE 802.15.4 based standard for personal area networks is capable of data rate up to 250 kbit/s. It supports both star and mesh topologies. ZigBee [13] network must have one coordinator device, which takes care about network maintenance and controls network parameters. In a star topology, central node must be coordinator. Range can be extended by routers. Last type is end device which is only able to talk to the parent node. Messages are routed by AODV routing protocol. ZigBee implements all layers from network layer to application layer.

New specification, ZigBee IP, was released in 2013. It offers similar functionality as Thread eg. self healing mesh, secured connection, same topology and node types. Thread seems to be more mature, despite the fact, that it was announced after ZigBee IP.

7.1.4 Z-Wave

Sub-ghz based technology using FSK radios. Z-Wave [16] network uses mesh topology and may contain up to 232 nodes with max throughput 100 kbit/s. In fact it is pretty similar to previously mentioned technologies, but it is based on a proprietary design and the only one chip vendor. This technology also covers all ISO-OSI layers from network layer to application layer.

7.1.5 Insteon

Technology operating on sub-ghz bands combined with communication via building's electrical wiring [6]. Insteon was one two launch partners of Apple HomeKit. In 2015 compatibility with Amazon Echo was added and initiative of Google's Nest thermostat announced.

It uses mesh topology and maximum packet payload is 14 bytes. Average case throughput is 180 bit/s, but data rates up to 13 kbit/s can be achieved. Unfortunately, there is no direct mapping to OSI stack available.

7.2 Long range

This section is focused on description of technologies which are offering coverage of several kilometers.

7.2.1 SigFox

This is a technology developed by SigFox [12] company, which was founded in 2009 in France. It uses ultra narrow band communication in unlicensed bands 868 MHz (EU) and 915 MHz (USA) and it defines physical and partially data link layers.

Maximal SigFox data rate is 100 bps and payload size of one message is 12 bytes. According to SigFox company, number of daily sent messages is limited to 140. This limitation shouldn't affect target devices and has positive effect on battery life and it limits "active time" when the device is transmitting on ISM bands.

SigFox range should be up to 50 kilometers in rural areas and up to 10 kilometers in urban environment. SigFox infrastructure is currently being developed in several countries across Europe (France, Spain, Netherlands etc.) and United States (California). In the Czech Republic, SigFox company works with T-mobile to build an infrastructure. All data collected using this infrastructure will be available from cloud server via API.

7.2.2 LoRa

LoRa [8] is an acronym which stands for **long range radio**. This technology defines physical and partially data link layers. This open standard based technology uses sub-GHz bands which means, as was mentioned before, that range and obstacle penetration is much better.

Single gateway is able to take care of thousands nodes in range up to 15 kilometers in open areas and approximately 4 kilometers in urban areas. This can be achieved by using FSK spread spectrum modulation. Data rates ranges from 0.3 kbps to 50 kbps. LoRa uses star topology, so end-point devices are connected directly to LoRa gateway, which is connected to the Internet.

7.2.3 Weightless

Weightless [15] is a set of wireless open technology standards. There are currently three versions of wireless technologies in Weightless set and all of them are operating in sub-Ghz bands.

The first one is Weightless-N, where N stands for ultra narrow band (same principle as SigFox). Weightless-N provides 1-way communication, simple feature set, low cost, 5 km range in open areas and long battery life.

Weightless-P provides 2-way communication and full feature set. Range in open areas is around 2 kilometers. Estimated battery life is shorter than that of Weightless-N, but still long enough to last approximately 3-5 years.

Last standard is Weightless-W, which uses spread spectrum frequency hopping, variable spreading factors and time-division duplex. It offers same features as previous variant, but also extensive feature set and larger coverage (around 5 km).

7.3 Comparison

In this section, parameters of all mentioned technologies are compared in the table 7.1

Name	WiFi	BLE	Thread	ZigBee	Z-Wave	Insteon	LoRa	SigFox	Weightless
Range category	short	short	short	short	short	short	long	long	long
Distance (up to)	100m	50m	100m	100m	100m	50m	20km	50km	2km (-P) 5km (-W)
Topology	star mesh etc.	star mesh	mesh	mesh	mesh	mesh el. wiring	star	star	star
Devices count	manufacturer dependent	200	250-300	1000	232	N/A	thousands	thousands	N/A
Frequency	2.4GHz	2.4GHz	868MHz 915MHz 2.4GHz	868MHz 915MHz 2.4GHz	868MHz 915MHz	sub-ghz	433MHz 868MHz 915MHz	868MHz 915MHz	868MHz 915MHz
Typical data rate (up to)	600Mb/s (802.11n)	1Mb/s	250Kb/s	20Kb/s (sub-ghz) 250Kb/s (2.4 GHz)	100Kb/s	13Kb/s	50Kb/s	100b/s	100Kb/s (-P) 10Mb/s (-W)
Wireless module current (at 3.3V)	200mA	<15mA	N/A	<30mA	<41mA	N/A	<40mA	<65mA	N/A
Maximal payload size	up to 2KB (MSDU 802.11)	27 bytes	approx. 128 bytes	up to 100 bytes	64 bytes	128 bytes	250 bytes	12 bytes (message count limit)	10+ bytes

Table 7.1: Comparison of wireless technologies

Chapter 8

Representational state transfer API

Representational state transfer (REST) [21] is an architectural style consisting of several constraints, which was firstly introduced by Roy Fielding in year 2000. This architectural style was designed for distributed systems. It is a modern approach of communication between devices, which allows decoupling of server and client. Both Hub and Cloud server in our IoT system should provide such API for simple communication with clients.

8.1 REST Constraints

There are several constraints which are mandatory. If the API fulfills all of them, it is called RESTful API.

8.1.1 Client-server

This constraint divides devices to clients and servers. Client takes care of user interface and maintains the session state, while server handles the data storage. This allows portability and independent evolution of both components.

8.1.2 Stateless

All interaction between server and client must be stateless, which means that all of the information needed, to process the request successfully, must be included in that request. Session state is maintained entirely by client, which improves system scalability.

8.1.3 Cache

Because responses can be cached by intermediary servers, every response must define itself as cacheable or not. This can eliminate some of the interactions and therefore improve network's efficiency.

8.1.4 Uniform interface

This feature is crucial to the design of any REST service, because it simplifies and decouples the architecture, so each component can evolve independently from others. It consists of interface constrains described in the following subsections:

8.1.4.1 Identification of resources

Resource is a conceptual target of hypertext reference. Resource may be addressed by unique identifier (e.g. URI) within request from client. Representation of resource returned to the client can be different than server's internal representations. That means, for example, the data can be stored in SQL-like DB on server, but they are sent to the client as JSON.

8.1.4.2 Manipulation of resources through representations

Client can manipulate only with representations of resource, not with resources itself. Same resource can be represented to different clients in different ways (e.g. HTML for browser, JSON for client application etc.). This behavior allows representation of resource in different ways and formats, without changing its identifier.

8.1.4.3 Self-descriptive messages

Desired state of resource can be represented within request from client, whereas current state of resource may be represented within response from server. Client can suggest new representation of resource, but it is up to server to decide whether the new representation will be accepted or not.

8.1.4.4 Hypermedia as the engine of application state

Representation of resource includes links to related resources. Presence, or absence of some of those links can be important part of resource's current state. This feature allows clients to seamlessly traverse through resources.

8.1.5 Layered system

This constraint enables network-based intermediaries to be transparently deployed between server and client using uniform interface. Network-based intermediaries can be for example proxies, gateways etc. Those intermediaries are usually used for security improvements, load balancing or response caching.

8.1.6 Code-On-Demand

Code-On-Demand allows servers to temporarily transfer executable scripts to clients to extend their functionality. This allows simpler clients with less features, because they can be added later on demand. This constraint improves system extensibility after deployment, but it reduces visibility. This is why the Code-On-Demand is the only REST constraint which isn't mandatory.

Chapter 9

Problem solution

As we have already mentioned in the Introduction, we would like to select two of the reviewed wireless technologies and integrate demonstration devices based on those technologies into our IoT system. Demonstration devices will be designed for interior lightning.

Current standards like 0-10V or DALI, designed for lightning control, usually require additional electrical wiring for communication with light dimmers. Benefits resulting from introduction of wireless technologies and Internet of Things into interior lightning and building automation are huge. First important factor is cheaper infrastructure, because there is no need for building separate wiring for communication with lights as is required for existing standards.

Figure 9.1: Sample hardware

Additional savings are related to the possibility of utilizing existing electrical wiring during conversion from building without any system for light control. This is possible thanks to integration of wireless modules into lights with standard sockets (fluorescent tubes, light bulbs etc.). The larger bandwidth is also required when we think of integration of various sensors into lights. The bandwidth of communication channel is also usually higher at wireless networks than wired ones, so the integration of various sensors into lights should be easier, because of smaller bandwidth limitations.

Next important improvement is the possibility of automating the building infrastructure by cooperation of various sensors. In our IoT system the smart devices are connected together by a scripts. Those scripts utilizing multiple devices brings almost unlimited possibilities. Lights could be dimmed according to ambient light level, by cooperation with illumination sensors. Other possible feature is turning on the lights just on locations where it is needed (presence of a person). Both of those examples would lead to energy savings and therefore reduce environmental impact.

First technology that we have chosen for demonstration is the WiFi. Main reason is the possibility to connect to current infrastructures that are almost everywhere. Next reason is the cost of WiFi modules, like ESP8266, which are now entering the market. The ESP8266 cost is under 2 USD, making it one of the cheapest of the reviewed technologies. The facts, that we have had previous experience with ESP8266 module and there is a large community were also the factors, that we had taken into account during selection, because it allows faster development and less bugs.

The second wireless technology is a candidate from long range radios. We have selected the LoRa as a wide spread, open source representative of long range wireless technologies. LoRa modules are more expensive than WiFi modules, but thanks to the increased range, lesser number of Hubs will be needed. For introduction to the LoRa technology, we have selected the RN2483 module, which can be controlled via UART from standard MCU.

9.1 Wireless technologies selection

Following aspects were taken into account during the selection of suitable wireless technologies:

- Cost of hardware
- Adoption speed
- Range of wireless coverage
- Bandwidth

Selected technologies weren't mentioned as competitors to be compared. They are more like two representatives from different groups of wireless technologies, which are suitable for different devices. The WiFi has higher throughput, but higher power consumption, whereas datarate of LoRa significantly lower, but the ranges are much larger. We would like to try both technologies in interior lightning devices, and explore their properties in such use case.

9.2 Hardware description

Part of our system, which is described in this thesis, consists of Hub and smart devices. Those devices are light tubes with integrated wireless modules and illumination reading sensor. Majority of custom built hardware was developed by member of our research group. Sample hardware used in our system is seen in Figure 9.1

9.2.1 Hub

Device called the Hub (Fig. 9.2) is based on Intel Edison [3]. Intel Edison is a tiny computer on module with dimensions approx. 35x25 mm. This computer is equipped with 500MHz dual-core Atom CPU, 1GB of RAM and 4GB flash memory. The module itself features WiFi and BLE connectivity. In our case the Intel Edison module is mounted on a custom made board which extends it's functionality. This board provides supply, but also contains 4-port USB hub and 868MHz proprietary radio for further experiments. LoRa connectivity is ensured by custom built USB LoRa dongle, which is the RN2483 module with FT230X USB to Serial converter.

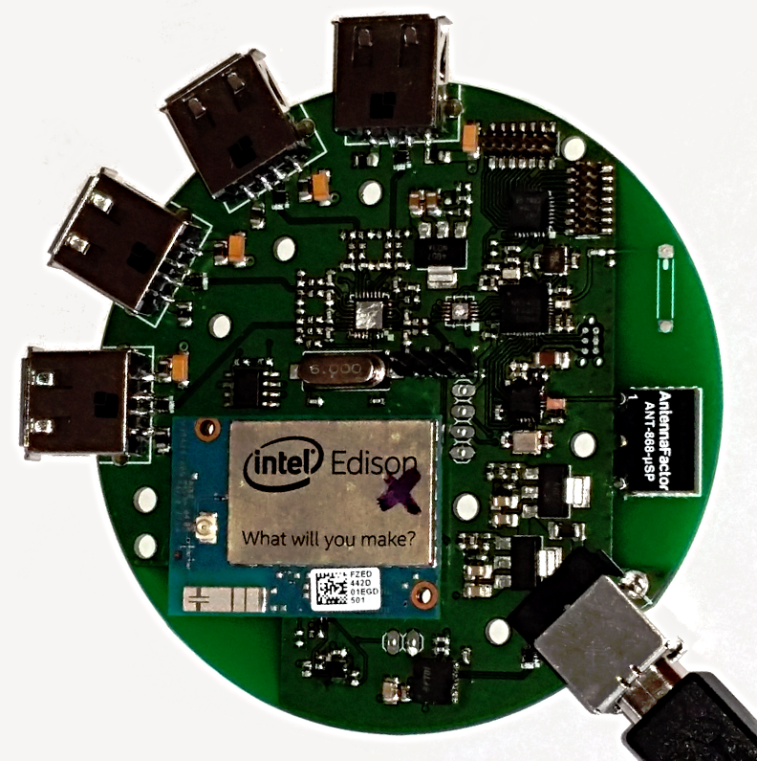


Figure 9.2: The Hub

9.2.2 Light controllers

There are two variants of wireless modules, according to used wireless technology, but both of them are powered by 3.3V and have PWM output to control the LEDs inside the light tube.

The WiFi light controller module (Fig. 9.3) contains the ESP8266 module from Chinese company Espressif Systems [4]. This module is equipped with RISC CPU from Tensilica which is capable of running on frequencies up to 160MHz. Tensilica CPU is combined with 64KB of instruction RAM and 96KB of data RAM. It has 4MB of flash memory for storing firmware and user data. Except 802.11b/g/n WiFi this module also contains GPIO, I2C, ADC, SPI, PWM etc. There are many versions of this module which differs in size or pin layout, but in our light controller module the ESP-12 version is used.

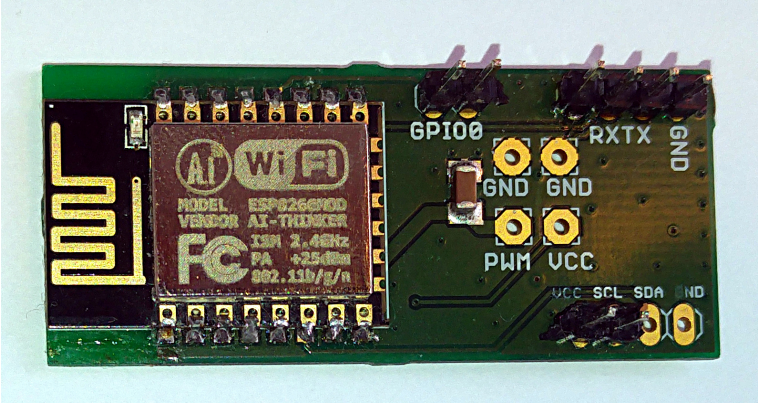


Figure 9.3: WiFi module

The other module, which uses LoRa, contains RN2483 module [11][10] and PIC18F26K22 [9] from Microchip Technology company. The RN2483 module itself consists of SX1276 long range transceiver from Semtech and PIC18LF46K22 MCU from Microchip. The PIC18F26K22 MCU is capable of running on frequency up to 64MHz and it has 1KB of data EEPROM and 64KB of program memory.

Those chips inside the RN2483 module are communicating via SPI Serial Peripheral Interface (SPI). The PIC18LF46K22 controls the SX1276, but also provides handles the commands received via UART. There is an obvious opportunity to use sole SX1276 transceiver in combination with SPI capable MCU to cut down the costs, but for the first encounter with LoRa, the RN2483 seems like wiser choice, because it has the LoRa network stack already implemented.

9.2.3 Illumination sensor

Illumination sensors are basically the same modules as light controllers. But instead of controlling the LEDs via PWM, it reads data from sensor board (Fig. 9.5). Sensor board is a small PCB with sensors like OPT3001 for reading illumination, but it also has the

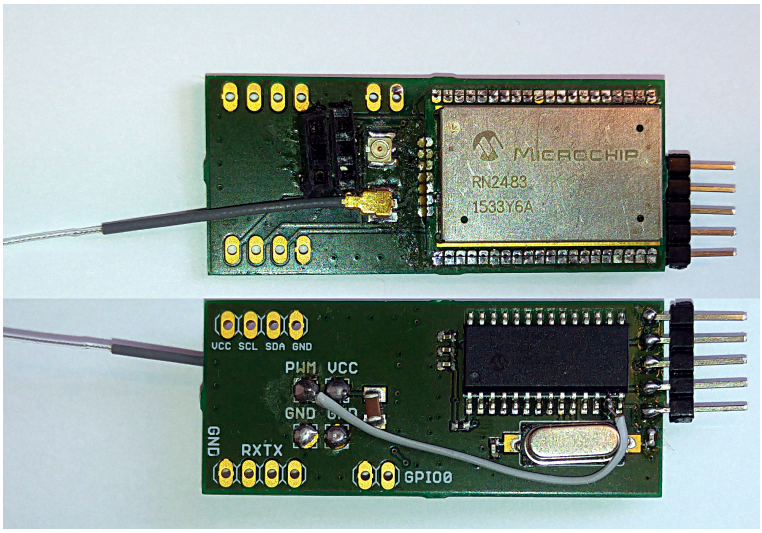


Figure 9.4: LoRa module

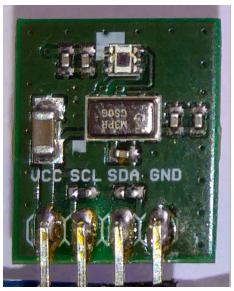


Figure 9.5: Sensor Board

MPL3115 sensor, which is able to measure temperature and atmospheric pressure. Utilization of MPL3115 is scheduled into future.

This sensor was developed as a proof of concept, because it wasn't optimized for low power consumption. In the future versions, we're planning to use energy harvesting methods for powering sensors like this one.

9.2.4 Lights

There are many versions of LED lights prototypes. Our goal is to integrate all electronic inside the tube, which could be then plugged into standard fluorescent tube socket and powered directly from power grid. During software development for wireless modules, the modified version was used for safety reasons, because there high voltage does not occur here.

This modified version (shown in Fig. 9.6) is powered by standard 12V laptop power supply. It contains 3.3V voltage regulator for wireless modules and BD911 transistor used for LED strip intensity regulation via PWM.

Figure 9.6: LED Light Prototype

9.3 Hub software

9.3.1 Linux distribution

Standard Linux distribution called Yocto, supplied by Intel, was replaced by Ubilinux from EmutexLabs, which is based on Debian 7 "Wheezy". Main reason to this was the lack of FTDI drivers in Yocto Linux. Those drivers are necessary for successful connection of USB LoRa dongle.

9.3.2 Zetta

This software is an open source, API-first IoT framework and also important part of our system. It's a basis for our further modifications, which were mentioned previously. Zetta runs in Node.js and allows users to create and connect servers at different locations into one network providing REST API to access smart devices.

It provides mapping of functions and properties of smart devices to the API. Those functions can be platform dependent (e.g. acces to bluetooth stack etc.), but once they are mapped to the API using a piece code called the Driver, they can be used by every client which doesn't need to know what does it take to perform that action.

Scout is a script which takes care of communication using certain technology. So there can be, for example, bluetooth scout, LoRa scout, MQTT scout and so on. Device discovery is also performed by scouts. Once the scout for given technology receives message from device it does not know it can create new instance of device, using driver according to the device type. Otherwise it passes messages to already existing instances of devices.

Next Zetta feature is the possibility of connecting multiple devices into scrips. Those scripts are called apps. Zetta app allows to observe streams of properties of multiple devices and using logic contained in the script control the actuators.

The process of connecting two Zetta servers is called linking. There can be two Hubs linked together or Hub to the cloud. Big advantage of linking two computers together is the API tunnelling, which means that network composed of linked Zetta servers provide API which is joined from APIs of all connected servers.

Server resources are represented within Siren hypermedia type via API. Siren is an open source Hypermedia type for representing entities. Initial implementation is the JSON Siren, but other implemetations, like XML Siren are expected. The most important entities for device description are class, properties and actions. As the name of those entities suggests, the class defines device type, device state and all readings from sensors are in the properties entity and available actions, which can be easily called using the HTTP POST request, for given device are described within actions entity.

9.3.2.1 Zetta modifications

Following features are needed for our IoT system. Some of them were already implemented and some of them are planned.

- Access Control List
- OAuth 2.0 autenthication
- embedded database for data logging (MongoDB)
- addition of new devices using smartphone (e.g. QR code)

9.3.3 MQTT broker and client

MQTT was selected as an application layer protocol for communication with WiFi modules, so there is an obvious need for MQTT broker and client on the Hub.

The Mosquitto is an open source MQTT broker, which is running on our Hub. Wireless modules are connected to this broker, as well as the client inside Zetta MQTT scout, which serves as a bridge from MQTT to Zetta.

9.4 WiFi module

9.4.1 NodeMCU

NodeMCU[1] is an open-source firmware for ESP8266 system-on-chip. It provides APIs for built-in hardware and multiple libraries for various devices. There is a big community around this firmware, providing frequent updates and fast troubleshooting. It is also based on Espressif SDK, which means, that newest features are always supported.

The ESPlorer[2] IDE is a developer tool which was used to program our ESP8266 based modules. Scripts are interpreted using Lua interpreter and they are saved in a built-in flash memory using SPIFFS[3]-based file system.

[1] https://github.com/nodemcu/nodemcu-firmware/tree/dev
[2] https://github.com/4refr0nt/ESPlorer
[3] https://github.com/pellepl/spiffs

The usage of MQTT protocol was supported by existence of mature MQTT client included in NodeMCU, this fact allowed us to use event-driven programming. Other approaches of development for ESP8266 have been also tested. Next, but not the last one, possibility how to program ESP8266 is to use C programming language and Espressif libraries. This approach was abandoned due to errors in MQTT library written in C and overall slower development.

9.4.2 Program description

Workflow of the software for light controller is shown at the figure 9.7. After initialization, it tries to connect to the WiFi AP. If it was successful it attempts to connect to the MQTT broker on the Hub. Then the it periodically transmits the state and increment in JSON format to the Hub. When the change of light intensity is needed, the Hub transmits the new desired power level, also in JSON, which is received by the light controller and after check for errors, the new power level is set. There is a transition implemented on the side of the light controller to make the change between two power levels smooth.

Software for the illumination sensor is simpler, because there is no need to control an actuator. Initialization and WiFi AP connection phases are same as those at light controller, but after that the sensor performs periodic readings of illumination value from OPT3001 sensor via I2C bus. Results of those sensor readings are transmitted to the broker together with the increment.

9.5 LoRa module

As was mentioned before, our wireless LoRa module consists of RN2483 transceiver and PIC microcontroller. LoRa transceiver has its software already preloaded from manufacturer, so programming for our LoRa module contained only programming for PIC18F26K22 MCU.

Integrated development environment MPLAB X from Microchip was used for development for PIC18F26K22 MCU. The RN2483 transceiver is controlled via UART using commands[4]. UART library for PIC was modified to extend its functionality to buffer incoming messages.

LoRa networks usually consists of end devices and gateways. In an optimal case, the LoRa gateway should be included in our Hub, but due to the lack of the existing LoRa network infrastructure and high cost of LoRa gateway, we needed to come up with an alternative solution.

To solve problem mentioned above, we have decided to disable the MAC layer, which allowed us to send packets directly between RN2483 modules. Unfortunately, this approach disables error correction methods etc. which are working at MAC layer. Next disadvantage is the lask of possibility to simultaneously receive multiple packets. This approach allows introduction to the LoRa technology, while keeping low cost, but in a case of system with higher number of LoRa devices the LoRa gateway will be still needed.

Activity diagram for LoRa light controller is shown on figure 9.8. At the beginning of the program, the LoRa module initializes GPIO, PWM and UART. Listening for packets is

[4] http://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf

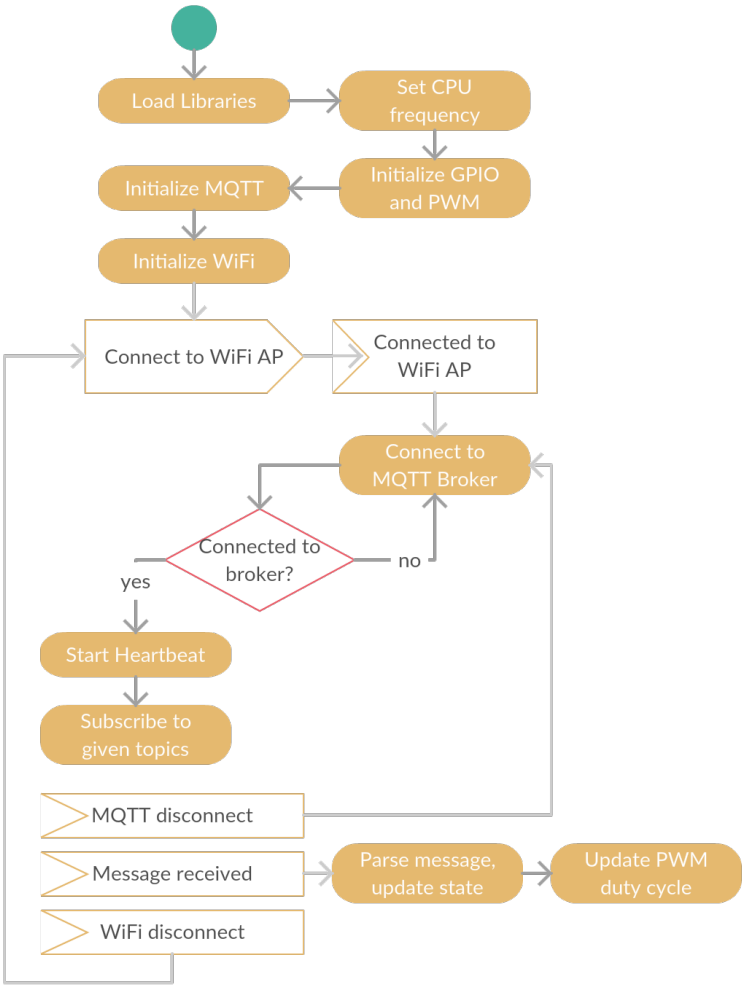


Figure 9.7: WiFi light module diagram

started immediately after the initialization is done. First of the possibilities that can happen next is the reception of the new packet. The other one is the receive watchdog timeout, which means that no packet was received during receive window. Next step, after the packet is received, is the CRC and UUID matching. If the packet CRC matches as well as the UUID contained in the packet, the new power level is set. This is followed by an immediate heartbeat packet transmission to confirm the new power level to the Hub. Aside from this task, the LoRa module also periodically transmits heartbeat packets to inform the Hub its state.

Software for previously mentioned LoRa illumination sensor is also simpler, because there is no need to receive packets. When the sensor is fully initialized, it starts sending packets with illumination value, which was read by the OPT3001 sensor.

There weren't any major problems during implementation. Module ESP8266 has a big community support which makes the development easier. That means more libraries available

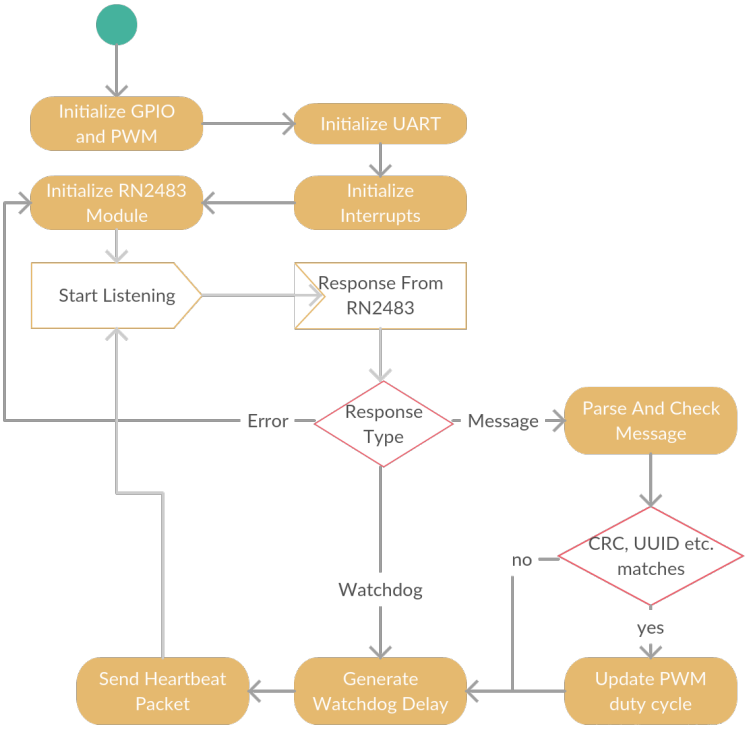


Figure 9.8: LoRa light module diagram

or more computing power. The LoRa module is, on the other hand, simplistic with less overheat, which could lead to optimized solutions with lower power consumption. The usage of Intel Edison inside the Hub wasn't a bad choice, because we did not encounter any restrictions.

Chapter 10

Experiments

10.1 Light control

First and obvious task to be done, was to control the LED lights via API from client. In our experiment, the client was smartphone and desktop computer. The task of controlling the lights is very simple, thanks to the client web application provided by Zetta. This functionality was mandatory and the correct functioning was expected from the very beginning. We've chosen to measure round trip delay and wireless range as parameters of implemented solution.

10.1.1 Round trip delay time

Round trip delay time, also known as ping time, is the time between sending a request and receiving a response. In our system, we have divided round trip delay into smaller sections using checkpoints. Division of the message route and checkpoints locations are shown in Fig 10.1.

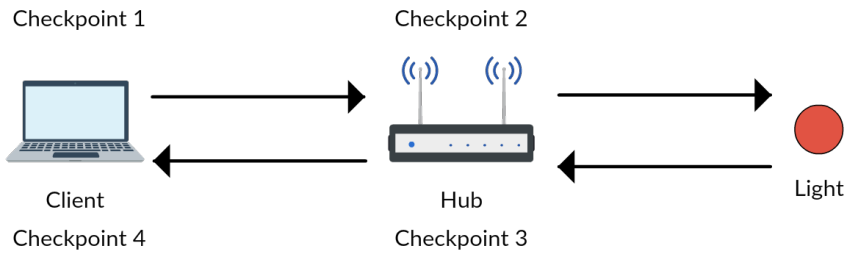


Figure 10.1: Round trip

There are three types of time delays we have measured. The first one is the **total time**, which means time delay between Checkpoint 1 and Checkpoint 4. Total time delay is an indicator how long the client will have to wait for the device to perform specified action,

Wireless Technology	Client To Hub Time	Device Response Time	Total Time
LoRa	86 ms	154 ms	240 ms
WiFi	84 ms	71 ms	155 ms

Table 10.1: Averaged round trip delay times

including request acknowledge. Next important parameter is the delay between Checkpoint 2 and Checkpoint 3, which describes response time of smart device. This time delay, also called **device response time**, includes not only the time required by given technology to transmit request from the Hub to the device and acknowledge back to the Hub, but also the time required by our wireless modules to perform specified action.

The last one is the time delay between Checkpoint 1 and Checkpoint 2 together with time delay between Checkpoint 3 and Checkpoint 4. This time delay, also **Hub to Client time**, is a parameter belonging to the network between the Client and the Hub and in our test case, it should be similar for both technologies, because the same WiFi network is was.

This experiment was conducted in office environment, where the Android smartphone, figuring as client, was connected to the Hub via local WiFi network. Smart devices, the LED lights were connected directly to the Hub, in case of LoRa, or using local WiFi network. The client was controlling smart devices via API provided by the Hub. Measuring scripts were installed on the smartphone and the Hub. There were 200 requests measured for each technology during this experiment and the averaged delay times are shown in the Table 10.1.

10.1.2 Wireless range

Purpose of this experiment is to experimentally estimate the maximum coverage in indoor environments, because it is the primary environment of the eventual deployment. Results of this experiment can be useful for determining infrastructure of IoT system, eg. if there will be need for wireless repeaters for given technology etc.

The maximal distance between smart device and the Hub was measured and as was mentioned before, the measurements were carried out in indoor areas and therefore the maximal theoretical range of given technology can be reduced by presence of commonly occurring obstacles in such environment (walls, furniture etc.).

We expect better obstacle penetration is expected from LoRa technology, because it runs on 868MHz, unlike WiFi running on 2.4GHZ. Next important factor contributing to the expected maximal range and obstacle penetration at LoRa technology is the usage of LoRa modulation, which, on the other hand, causes long transmission times.

WiFi modules was set to maximal transmit power. This is good for maximal coverage estimate, but in real life deployments with larger number of devices it can cause jamming of wireless bands and therefore negatively reflect on reliability of wireless connection. The LoRa module had following settings: Spread spectrum SF7, transmit power -1dBm, bandwidth 500KHz and coding rate 4/8. This is the setting to minimize time-on-air for LoRa packet.

Maximal range is the maximal range where the tested wireless technology was able to successfully perform 100 consecutive actions without losing connection to the Hub. Maximum measured distances are shown in Table 10.2. Measurements were performed in an office environment with obstacles in the line of sight (several walls, furniture etc.).

Wireless technology	Indoor range
WiFi	22 m
LoRa	65 m

Table 10.2: Max distances

Indoor range of ESP8266 also heavily depends on used WiFi access point, which was in our case the Turris. Range achieved with ESP8266 should be satisfactory for regular house coverage. Measurements of LoRa range proved that this technology has better obstacle penetration, but the coverage could be extended by increasing the output power, but this would shift the border of covered area outside the building we were measuring in. This is an expected result because Microchip claims that range in urban areas should be up to 5 kilometers. The only limitation in our system was the unavailability of LoRa gateway, thus the LoRa dongle was busy with retransmissions (e.g invalid CRC etc.) most of the time and control of larger number of devices would be complicated.

10.2 Intensity-based light regulation

This experiment is focused on intelligent power management of LED lights, because there is no need to have the light switched on the whole day, when there is the Sun shining through windows. To solve this problem, we would like to design solution for maintaining the constant illumination level in the interior. In other words, we would like to compensate changes of illumination coming from other sources, to maintain illumination level at user predefined value. Main advantages of this solution are the possibility of energy savings and ease of use for end user, who doesn't need to take care about controlling the lights.

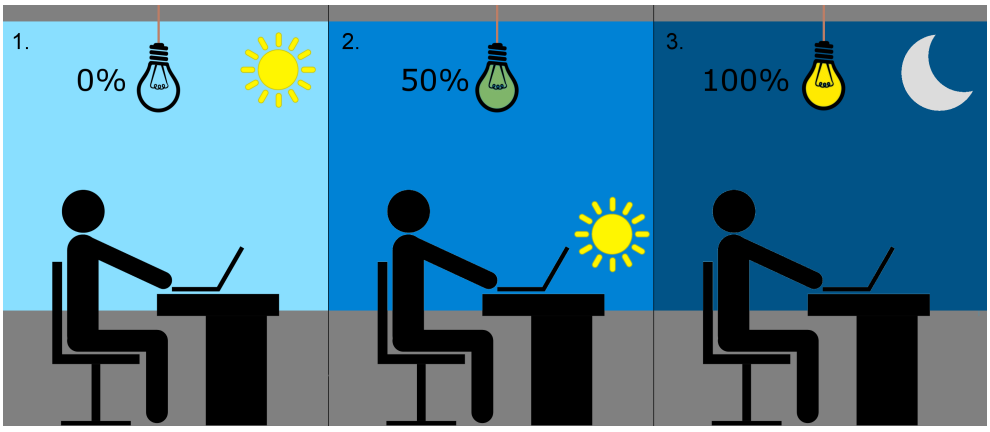


Figure 10.2: Ambient light intensity control

Our system for intensity regulation consists of wireless illumination sensor and LED light and they are both connected to the Hub. Desired illumination level can be set from the client using API and then the regulation loop starts. The sensor is reading illumination level and sending it to the Hub, where the regulation loop compares received value with target

illumination level and changes the power level of LED light to compensate the difference in a shortest possible time.

10.2.1 PID controller

Proportional-integral-derivative controller (PID) is a control loop feedback mechanism, which was used for maintaining the the constant illumination level. It constantly calculates error, which is the difference between target and current illumination levels and tries to minimize this error by modification of control variable, which is, in our case, the LED light power level.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{10.1}$$

Modification of control variable is calculated by equation (10.1), where the K_p, K_i and K_d are the coefficients (gains) for proportional, integral and derivative components.

Proportional response depends only on difference between target and current sensor reading. So if the error term has a magnitude of 10, the proportional gain of 0.5 would produce response of 5. Increasing the proportional will increase response speed, but when the proportional gain is too large, the process variable will begin to oscillate and may even oscillate out of control.

The integral component sums the error over time, so that even a small error will cause the integral component to increase slowly, unless the error is zero. Integral term serves as a kind of short-term memory. Problem called integral windup can occur, when there is a large change in target value. Integral component accumulates a significant error during the rise and then causes excessive overshooting, because it takes some time until the integral component error is reduced.

The derivative response is proportional to the rate of change of the process variable and it's task is to predict the future values.

The PID controllers are extensive topic, that could devote many pages, but for a description of our experiment the information mentioned above should be sufficient.

Controller implementation used for experiment was the pid-controller[1] and the manual tuning method was used. We have finished tuning with following coefficients.

$$K_p = 0.43; K_i = 0.12; K_d = 0.01; \tag{10.2}$$

10.3 Proximity-based light regulation

Goal of this experiment is to control power level of LED light according to the distance between that light and the user. This behavior would bring more comfort to users, but as in previous case, it could also save energy.

This experiment is a precursor to a more sophisticated way how to regulate lights, because in this case, we are doing localization of a person just in one dimension (distance from the

[1] https://github.com/wilberforce/pid-controller

light), but in future versions, we would like to be able to control multiple lights, according to indoor location of users, to shine just where it is necessary. In a case of movement of person, we would like to make smooth transitions between neighboring lights.

Use case (Fig. 10.3) of this experiment is following: User has some kind of identification device and approaches his desk in an office environment. The light above his desk will be automatically increasing intensity until the user reaches his desk. At this moment the light will be at its maximum power level. When the user leaves the desk, the light automatically decreases its intensity until it's completely off.

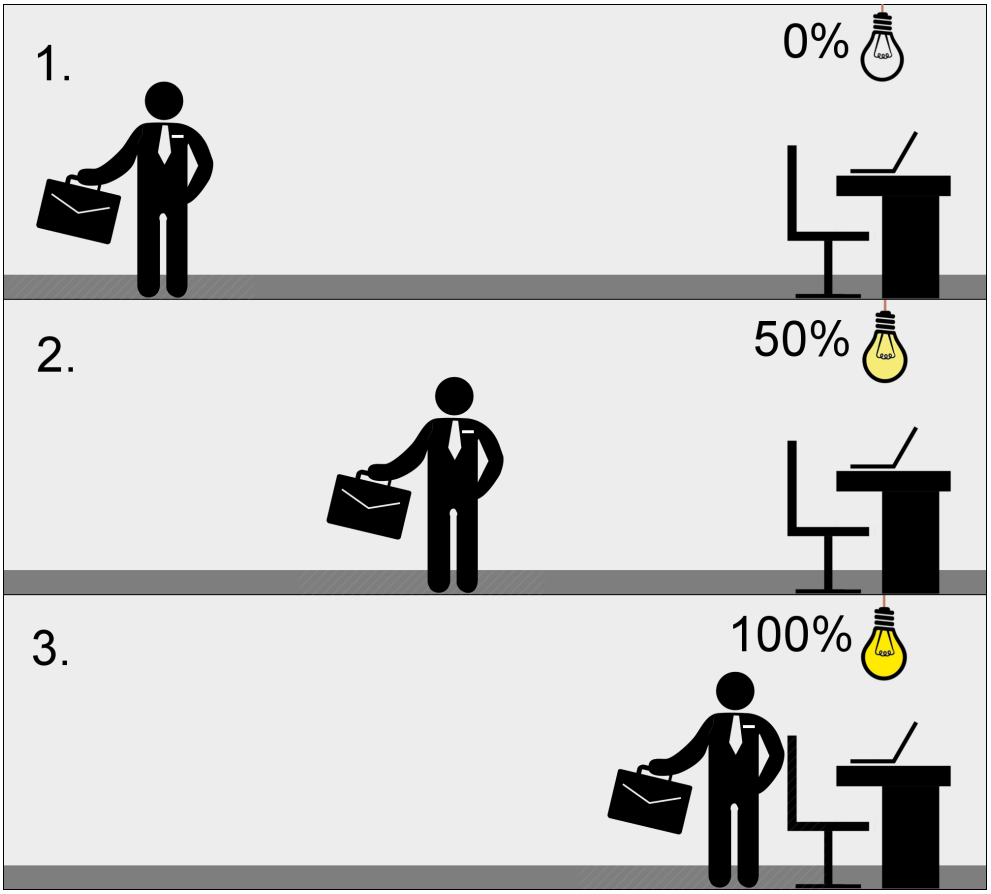


Figure 10.3: Proximity control

Our experiment setup has consisted of the Hub, smartphone as the identification device, wireless LED light and the Beacon from company Estimote. The smartphone and LED light were connected to the Hub using WiFi.

First idea was to calculate the distance between user and LED light using the received signal strength indicator (RSSI) of the WiFi access point created by the LED light with ESP8266 module. This approach has proved to be unsuitable for this experiment, because the RSSI of an AP from ESP8266 fluctuated a lot, and the WiFi networks scan time of used smartphone (LG G4) was not fast enough to achieve desired user experience.

This problem led us to the use of the Estimote Beacon [2] for proximity measurement. Beacon is periodically transmitting Bluetooth Low Energy advertising packets, that are received by smartphone. This approach has yielded quicker response times and more accurate measurement of the BLE packet's RSSI and therefore distance.

Beacon located near the LED light (on the desk etc.), was transmitting advertisement packets every 100ms with broadcasting power -30dBm. Those packets were captured by smartphone and were send, together with user identity (email account), to the Hub. Zetta app, running on the Hub, was processing incoming messages and dynamically controlling the lights paired with given user.

This application could be combined with application from previous experiment to achieve maximal power savings. There is a need for illumination sensor, which would be able to transmit BLE packets to simulate role of the Estimote Beacon in this experiment. This system would turn on the light only in case of a presence of user, but also control the power level of this light to maintain predefined illumination level.

Chapter 11

Conclusion

This chapter summarizes the achievement of the objectives of the thesis and analyzes the possible shortcomings of the implemented solution. The next section describes the possible future development of this project.

11.1 Evaluation of the objectives achievement

Our task was to describe and compare main wireless technologies available for Internet of Things networks, including network topologies and communication protocols. At the beginning, we have made an introduction to the different network topologies a methods of packet routing in mesh topology networks. Then we've reviewed application protocols suitable for application in IoT networks. Next chapters were dedicated to the description and comparison of the most important wireless technologies and the description of RESTful APIs.

We have chosen two of reviewed wireless technologies, designed and successfully implemented solution for controlling smart devices using those wireless technologies. In the experimental part, we have tested our solution and used it to realize two experiments for automation in indoor lightning. First experiment dealt with regulation of illumination level in indoor areas and the second concerned the regulation of light based on proximity of user.

We think that the goals outlined in the beginning of this work have been fulfilled and parameters of implemented solutions are satisfactory.

11.2 Future steps

There are many ways how to extend this project in the future. First important experiment should be measuring influence of larger number of smart devices onto communication (band jamming, packet retransmission etc.). Next possible modification of our IoT system is to hook up more types of smart devices, either already existing or custom built, to extend automation possibilities. We're planning devices such as PIR sensor, thermometer etc.

Next major topic is an indoor localization. This system, together with smart devices (eg. BLE beacons) or smartphones, could be used to collect and process data about location of

users. Indoor localization would probably require the use of machine learning method to achieve usable accuracy, but then the user location could be used for many different tasks. From indoor navigation or already mentioned automation (eg. lights regulation) to security purposes (proximity-based unlock of computer etc.)

Bibliography

[1] RFC 3626: Optimized Link State Routing Protocol (OLSR). Technical report, United States, 2003.

[2] Estimote Beacons Webpage. Available from: <http://estimote.com/>.

[3] Intel Edison Compute Module Hardware Guide. Available from: <http://download.intel.com/support/edison/sb/edisonmodule_hg_331189004.pdf>.

[4] Espressif Systems - Wi-Fi and Bluetooth chipsets and solutions. Available from: <http://espressif.com/>.

[5] Hype cycle for emerging technologies, 2015. *Gartner, August*. 2015. Available from: <http://www.gartner.com/newsroom/id/3114217>.

[6] Insteon Webpage. Available from: <http://www.insteon.com/>.

[7] ISO/IEC 20922. Available from: <http://www.iso.org/iso/catalogue_detail.htm?csnumber=69466>.

[8] LoRa Alliance Webpage. Available from: <https://www.lora-alliance.org/>.

[9] Microchip PIC18F26K22 datasheet. Available from: <http://ww1.microchip.com/downloads/en/DeviceDoc/41412F.pdf>.

[10] Microchip RN2483 Command Reference, . Available from: <http://ww1.microchip.com/downloads/en/devicedoc/40001784b.pdf>.

[11] Microchip RN2483 Datasheet, . Available from: <http://ww1.microchip.com/downloads/en/devicedoc/50002346a.pdf>.

[12] SIGFOX - The Global Communications Service Provider for the Internet of Things (IoT). Available from: <http://www.sigfox.com/>.

[13] The ZigBee Alliance Webpage. Available from: <http://www.zigbee.org/>.

[14] Thread Group Webpage. Available from: <https://www.threadgroup.org/>.

[15] Weightless - Setting the Standard for IoT. Available from: <http://www.weightless.org/>.

[16] Z-Wave Home control | Z-Wave Home control. Available from: <http://www.z-wave.com/>.

[17] CHROBOCZEK, J. RFC6126: The babel routing protocol. 2011. Available from: <https://tools.ietf.org/html/rfc6126>.

[18] DAY, J. The (Un)Revised OSI Reference Model. *SIGCOMM Comput. Commun. Rev.* October 1995, 25, 5, s. 39–55. ISSN 0146-4833. doi: 10.1145/216701.216704. Available from: <http://doi.acm.org/10.1145/216701.216704>.

[19] EVANS, D. The internet of things. *How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)*. 2011.

[20] FEKI, M. A. et al. The Internet of Things: The Next Technological Revolution. *Computer*. February 2013, 46, 2, s. 24–25. ISSN 0018-9162. doi: 10.1109/MC.2013.63. Available from: <http://dx.doi.org/10.1109/MC.2013.63>.

[21] FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[22] KUSHALNAGAR, N. – MONTENEGRO, G. – SCHUMACHER, C. Rfc 4919: Ipv6 over low-power wireless personal area networks (6lowpans): overview. *Assumptions, Problem Statement, and Goals*. 2007.

[23] LASI, H. et al. Industry 4.0. *Business & Information Systems Engineering*. 2014, 6, 4, s. 239.

[24] MALKIN, G. S. – STEENSTRUP, M. E. Routing in Communications Networks. Hertfordshire, UK, UK: Prentice Hall International (UK) Ltd., 1995. Distance-vector Routing, s. 83–98. Available from: <http://dl.acm.org/citation.cfm?id=214690.214693>. ISBN 0-13-010752-2.

[25] MOY, J. RFC2328: OSPF Version 2. Technical report, United States, 1998. Available from: <https://www.ietf.org/rfc/rfc2328.txt>.

[26] PERKINS, C. – BELDING-ROYER, E. – DAS, S. Ad Hoc On-Demand Distance Vector (AODV) Routing. Technical report, United States, 2003.

[27] PERKINS, C. E. – BHAGWAT, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM Comput. Commun. Rev.* October 1994, 24, 4, s. 234–244. ISSN 0146-4833. doi: 10.1145/190809.190336. Available from: <http://doi.acm.org/10.1145/190809.190336>.

[28] SHARMA, D. – VERMA, S. – SHARMA, K. Network Topologies in Wireless Sensor Networks: A Review 1. 2013.

[29] SHELBY, Z. – HARTKE, K. – BORMANN, C. The Constrained Application Protocol (CoAP)(RFC 7252), 2014.

[30] SURESH, P. et al. A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In *Science Engineering and Management Research (ICSEMR), 2014 International Conference on*, s. 1–8. IEEE, 2014.

Appendix A

Nomenclature

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

ADC Analog-to-Digital Converter

AMQP Advanced Message Queuing Protocol

AODV Ad hoc On-Demand Distance Vector

API Application Programming Interface

ARPANET Advanced Research Projects Agency Network

ASCII American Standard Code for Information Interchange

BLE Bluetooth Low Energy

CoAP Constrained Application Protocol

CRC Cyclic Redundancy Check

DALI Digital Addressable Lighting Interface

DSDV Destination-Sequenced Distance Vector

DTLS Datagram Transport Layer Security

DV Distance Vector

EBCDIC Extended Binary Coded Decimal Interchange Code

EEPROM Electrically Erasable Programmable Read-Only Memory

FSK Frequency-Shift Keying

FTP File Transfer Protocol

GPIO General-Purpose Input/Output

HTTP Hypertext Transfer Protocol

I2C Inter-Integrated Circuit

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

IPv6 Internet Protocol version 6

ISM band Industrial, Scientific and Medical band

ISO International Organization for Standardization

JSON JavaScript Object Notation

LoRa Long Range Radio

LPP Lightweight Presentation Protocol

MAC Media Access Control

MQTT Message Queue Telemetry Transport

MSDU MAC Service Data Unit

NDR Network Data Representation

NetBIOS Network Basic Input Output System

OLSR Optimized Link State

OSI Open Systems Interconnection

OSPF Open Shortest Path first

PCB Printed Circuit Board

PID controller Proportional Integral Derivative controller

PWM Pulse-Width Modulation

QoS Quality of Service

QR code Quick Response code

REST Representational State Transfer

RFC Request for Comments

RISC Reduced Instruction Set Computing

RPC Remote procedure call

RSSI Received Signal Strength Indication

SCP Secure Copy

SPI Serial Peripheral Interface

TCP Transmission Control Protocol

UART Universal Asynchronous Receiver/Transmitter

UDP User Datagram Protocol

UUID Universally Unique Identifier

WiFi Wireless Fidelity

WPAN Wireless Personal Area Network

XML Extensible Markup Language

XMPP Extensible Messaging and Presence Protocol

Appendix B

Content of the attached CD

Filename	Description
./ESP8266-light	Source files for WiFi light controller and illumination sensor
./LoRaPIC18	MPLAB-X IDE project for LoRa light controller
./LoRaPICSensor	MPLAB-X IDE project for LoRa illumination sensor
./zetta2.0	Zetta2.0 source files incl. scouts, drivers, apps
./thesis	LaTeX thesis source files
BP-Tomas-Pikous-2016.pdf	Bachelor project documentation

Table B.1: Content of the attached CD