

MASTER THESIS

**Development of a Sensor System
for Big Data Analysis in FabLabs**

Andreas Leitner, BSc

Graz University of Technology

Institute of Innovation and Industrial Management

Head: Univ.-Prof. Dipl.-Ing. Dr. techn. Christian Ramsauer

Supervisor: Univ.-Ass. Dipl.-Ing. Matthias Helmut Friessnig

Graz, January 2017

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____

Datum

Unterschrift

Abstract

FabLabs (Fabrication Laboratories) are publicly accessible and mostly free of charge facilities where users can transform their ideas into real products by applying a set of rapid prototyping equipment such as 3D printers, CNC milling machines and laser cutters. Nowadays, there are already more than 1000 registered FabLabs all around the world and the tendency is still rising.

However, current FabLabs do have a problem. The information flow between the operators and the manufacturing machines is only single-sided, providing no feedback of the machine to outside influences or production relevant data. The aim of this thesis is to establish a bi-directional communication between the FabLab equipment and the users by the development and application of a cloud-based sensor system. Thus, machine related data is acquired by various sensors and forwarded to a cloud computing service for subsequent data processing and analytics. A smart user interface allows real-time monitoring, interaction as well as notifications on various end devices.

To achieve the objectives, the thesis is sub-divided into four essential parts. First, the theoretical background of an approach known as the Internet of Things (IoT) is introduced and the fundamentals as well as the technical functionality of Internet-connected objects is outlined. The second part deals with the creation of a concept that is required for the practical development of the sensor system. In the third part, a market research is conducted whereby sophisticated IoT capable gateway controllers are compared to each other to find an appropriate alternative for the application. The fourth part is the development and installation stage. In this part the required controller hardware and software is implemented, sensors are selected and tested, the cloud platform including the user interface is configured and the overall communication chain is established. Finally, the system including all its components is installed at a selected FabLab machine.

The overall result of this thesis is the installation of a fully working sensor system that is applied at the Ultimaker 2 - Extended 3D printer for gathering machine related data and to furthermore validate the bi-directional user interaction as well as the data monitoring.

Kurzfassung

FabLabs (Fabrication Laboratories) sind für die Öffentlichkeit zugängliche Einrichtungen bzw. Produktionsstätten, die meist unentgeltlich dazu verwendet werden können, um eigene Ideen und Entwürfe erfolgreich umzusetzen. Dazu steht eine Auswahl verschiedener Maschinen und Werkzeuge wie z.B. 3D Drucker, CNC Fräsen oder Laser Cutter zur Verfügung. Weltweit betrachtet gibt es heutzutage bereits über 1000 offiziell registrierte FabLabs, wobei die Anzahl weiterhin kontinuierlich ansteigt.

Eine Einschränkung heutiger FabLabs ist jedoch der nur einseitig vorhandene Informationsfluss ausgehend vom Anwender hin zur Maschine, wodurch die FabLab Maschinen nicht in der Lage sind auf Umwelteinflüsse oder produktionsrelevante Ereignisse selbständig zu reagieren. Ziel dieser Masterarbeit ist es diese eingeschränkte Kommunikation zu erweitern, indem ein cloudbasiertes Sensor System entwickelt und implementiert wird. Verschiedene Sensoren sollen maschinenrelevante Daten erfassen und über eine internetbasierte Verbindung in eine Cloud Plattform übertragen um sie dort zu speichern. Auf diesem Weg ist eine Weiterverarbeitung der Daten für analytische Zwecke sowie die grafische Anzeige der aktuell erfassten Sensordaten möglich und zusätzlich lässt sich eine bidirektionale Kommunikation zwischen Anwender und Maschine realisieren.

Die Arbeit gliedert sich in vier wesentliche Teilbereiche, wobei sich der erste mit dem theoretischen Hintergrund des Konzepts Internet of Things (IoT) beschäftigt, auf dem die Vernetzung des Systems mit der Cloud Plattform aufbaut. Der daran anschließende Teil widmet sich der Erstellung eines Konzepts für die spätere Implementierung des Sensor Systems. Im dritten Teil wird eine Marktrecherche hinsichtlich internetfähiger Controller (IoT Gateways) durchgeführt, um eine geeignete Variante für diese Anwendung zu finden. Der abschließende Teil behandelt die eigentliche Entwicklung und Implementierung des Systems inklusive der notwendigen Hardware und Software Elemente sowie der Konfiguration der Cloud Plattform und Herstellung der Kommunikation.

Das Ergebnis ist ein funktionsfähiges Sensor System, welches testweise am Ultimaker 2 – Extended 3D Drucker installiert wurde, um Daten aufzunehmen sowie die zweiseitige Kommunikation zwischen dem System und der Benutzerschnittstelle zu validieren.

Table of Contents

Abstract	III
Kurzfassung	III
1 Introduction	1
1.1 Motivation	2
1.2 Objectives	3
1.3 Approach	3
2 Internet of Things (IoT)	6
2.1 Definition and Characteristics of IoT	6
2.2 History & Evolution	8
2.2.1 Starting Point & Development	8
2.2.2 Internet Protocol v6 (IPv6)	9
2.2.3 Growth Forecast & Market Size	9
2.3 IoT Components & Architecture	10
2.3.1 Physical Objects & Smart Objects	11
2.3.2 Sensors	12
2.3.3 Actuators	13
2.3.4 Gateway	14
2.3.5 Cloud Computing Services	14
2.3.6 Data Transmission	16
2.4 Other Concepts beside IoT	17
2.4.1 Machine to Machine (M2M)	17
2.4.2 Industrial Internet of Things (IIoT)	18
2.5 Application Areas	19
2.5.1 Smart Grid & Energy	20
2.5.2 Smart Health & Medical Applications	21
2.5.3 Smart Homes & Buildings	22

2.6	IoT Risks & Privacy Challenges	23
3	Sensor System Concept	24
3.1	Basic Considerations	24
3.1.1	FabLab Equipment	25
3.1.2	Components	25
3.1.3	Connection & Sensor Wiring	26
3.1.4	Internet Accessibility	27
3.1.5	The Cloud Platform - IBM Bluemix	27
3.2	System Block Diagram	27
4	Market Research	29
4.1	Development Boards - Market Overview	30
4.1.1	Arduino Family	30
4.1.2	ESP 8266	33
4.1.3	Raspberry PI (RPI)	36
4.1.4	Intel® Edison & Intel® Galileo Gen. 2	38
4.1.5	Other Development Boards	39
4.2	Platform Selection Criteria	43
4.3	Value Benefit Analysis	44
4.4	Results	45
5	Sensor System Implementation	47
5.1	Functional System Overview	47
5.2	Hardware	48
5.2.1	Arduino MEGA 2560	48
5.2.2	Arduino Ethernet Shield v2	50
5.2.3	NodeMCU Development Kit (ESP8266)	51
5.2.4	Sensors	52
5.2.5	I2C Level Shifting	58
5.2.6	Wiring Diagram & Stripboard	59
5.2.7	Sensor Placement	62
5.3	Software	64
5.3.1	Arduino IDE	64
5.3.2	Network Communication	67
5.3.3	MQTT Messaging Protocol	68
5.3.4	Message Payload Format - JSON	70

5.3.5	Establishing a Connection	71
5.4	IBM Bluemix	72
5.4.1	Internet of Things Foundation	73
5.4.2	Cloudant NoSQL DB	76
5.4.3	dashDB	77
5.4.4	Node-RED	78
5.5	Sensor System Installation	81
5.5.1	IoT Gateway - Arduino MEGA 2560 & Ethernet Shield	82
5.5.2	Ultimaker 2 - Extended & Sensor Setup	83
5.5.3	NodeMCU - Compact Solution	85
5.6	User Notification & Interaction	86
5.6.1	Telegram Messenger	87
5.7	Visualization	89
5.7.1	Node-RED Dashboard UI	89
5.7.2	Local Data Visualization	92
6	Conclusion & Future Perspectives	93
	List of References	VIII
	List of Figures	XV
	List of Tables	XVI
	List of Abbreviations	XVI
	Appendix	XXI

1 Introduction

A Fabrication Laboratory (FabLab), also denoted as a Makerspace, is an open and free to use environment where users can transform their unique ideas, designs and sketches into real products with a set of easy-to-use machines and tools such as desktop CNC mills, laser cutters, vinyl cutters and 3D printers. The first and, so far, only FabLab of Graz was opened in 2014 by the Institute of Innovation and Industrial Management at Graz University of Technology located at the university campus.¹

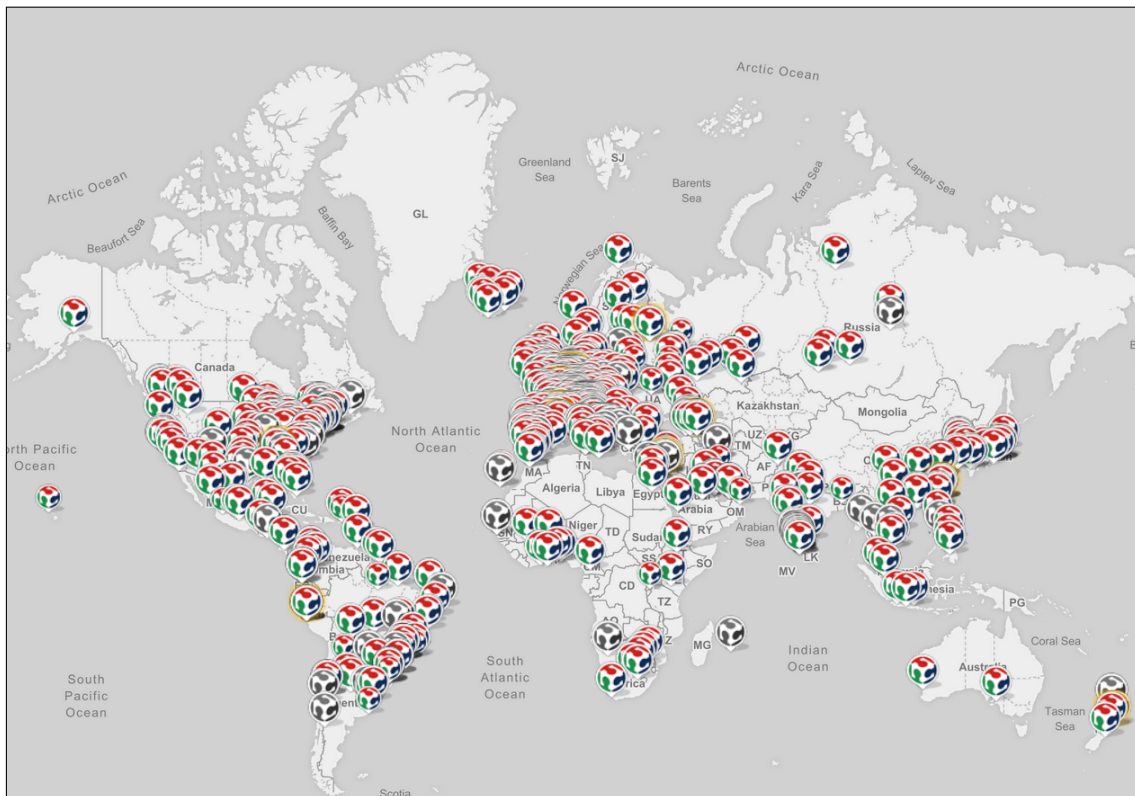


Figure 1.1: FabLabs - Overview and worldwide distribution (Jan. 2017)²

¹Cf. FRIESSNIG; BÖHM; RAMSAUER, (2015).

²Illustration extracted from FABLABS.IO, (2016).

Currently, there are more than 1000 registered FabLabs distributed in 97 countries all over the world, whereby the number of Labs is still increasing continuously. Fig. 1.1 illustrates the distribution of Makerspaces all over the globe. One must imagine that the first-ever opened FabLab introduced by the Massachusetts Institute of Technology (MIT) only started back in the year 2001³, so the rapid spread within the last 16 years is really remarkable. In Austria, there are currently eight FabLabs located in Vienna (2x), Mödling, Graz, Leoben, Wattens, Salzburg and Kirchfidisch.⁴

One essential characteristic of FabLabs is that they are accessible to the public for free and opened for a certain time at least once a week. The idea behind Makerspaces is that they share a common set of tools, machines and processes. Following this concept allows to share designs as well as the knowledge across international borders, to reproduce items in any other registered FabLab around the world by providing a similar set of machinery with the same capabilities.⁵

1.1 Motivation

FabLabs are a great and helpful prototyping establishment for makers, innovators, technologists as well as learners.⁶

Nevertheless, there is a certain drawback regarding current Makerspaces. A major problem of FabLabs is the constrained one-way interaction and one-way information flow from the user to the machine. As a result, this implicates that there is no reaction of the applied FabLab machine to outside influences such as other machines, people or environmental conditions. In case a machine would be able to send production related data for analysis purposes, the recorded data could be utilized in order to predict certain events such as service intervals, maintenance, or the replenishment of consumables. Automatic remote user notifications could contribute to receive messages about ongoing production progresses once a production is finished, or about critical errors that may occur. A machine could even prevent worst case scenarios like crashes or it performs automatically executed

³Cf. MOREL; LE ROUX, (2016), p. 2.

⁴Cf. FABLABCONNECT.COM, (2016).

⁵Cf. FABFOUNDATION.ORG, (2016b).

⁶Cf. FABFOUNDATION.ORG, (2016a).

emergency stops via predictive methods and predefined program routines. Additionally to the machine related benefits, gathered ambient data within a FabLab environment can be obtained to provide more details about the current degree of utilization and attendance during open days.

The general objective of this thesis is to overcome the one-way information flow pointed from the “user-to-computer-to machine” by establishing a bi-directional connection between the FabLab users and the machinery.

1.2 Objectives

Within the context of this thesis, it emerges out of the motivation that the efficiency and the usability of FabLabs and Makerspaces can be increased by providing the possibility of a two-way communication between users and the equipment, whereby machines are able to react automatically to predefined events, or to interact with the users via remote interfaces.

Hence, the overall objective of the thesis is to develop and implement a cloud-based sensor system within the FabLab Graz to, on the one hand, collect and store machine related data within a cloud-based platform for subsequent data analysis and analytics schemes, and on the other hand, to provide a two-way interaction between users and the machines for receiving and sending notifications as well as machine related information and the production status to miscellaneous connected devices.

1.3 Approach

In order to achieve the overall objectives of the master thesis, which are defined in section 1.2, the approach of the thesis related work is sub-divided into four main constituents. Within the structuring, the first three parts are more theoretical followed by a part that deals with the practical system implementation, as it can be seen from Fig. 1.2.

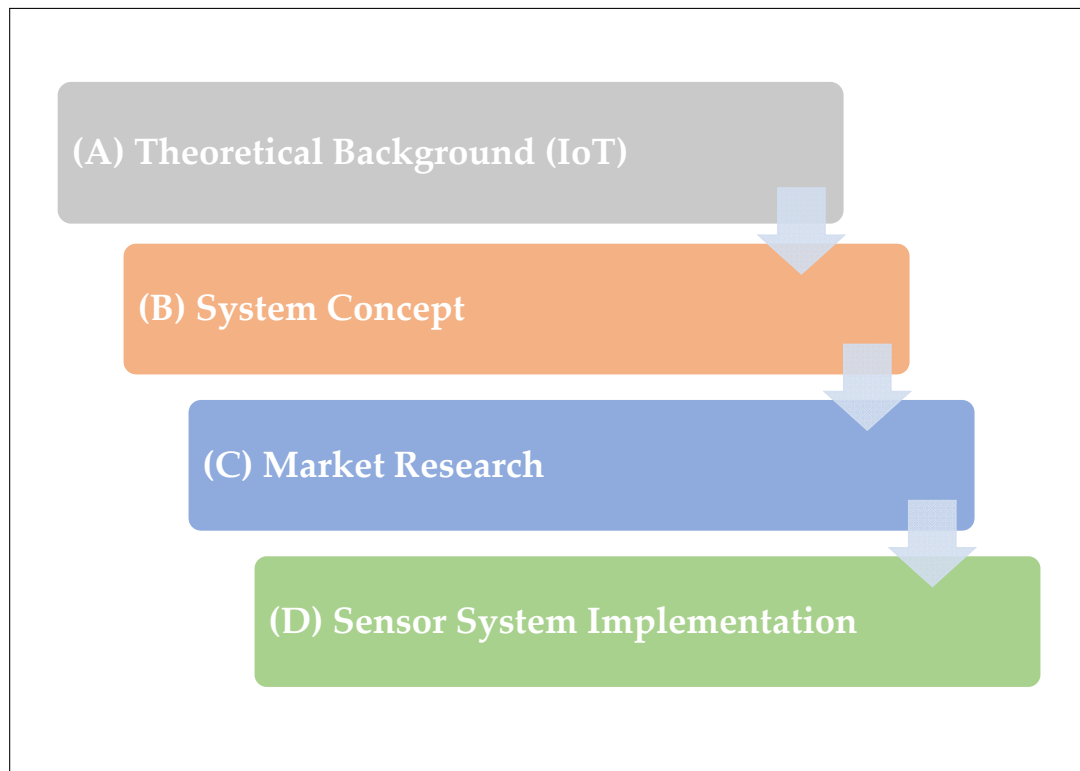


Figure 1.2: Thesis Approach

(A) Theoretical Background

The initial part of the thesis provides a theoretical overview about a fundamental concept of internet-connected objects on which this thesis is based on, introduced as the Internet of Things (IoT). It deals with the beginnings and evolution of the concept, the functionality, describes the required components and interfaces to successfully form an IoT and it outlines several use cases where it is applied nowadays.

(B) System Concept

Before it is possible to implement a sensor system, a basic concept is required where system constraints, boundary conditions and other aspects regarding the implementation have to be considered, e.g. the utilized FabLab machine or equipment that subserves for the initial installment. The output is a simplified system block diagram that is based on the fundamentals of part (A) and is furthermore used to create a detailed system overview within the subsequent implementation phase.

(C) Market Research

IoT systems include a significant component to receive, process and to forward data that is obtained from physical objects (things) in the real world through sensors. A various amount of such “gateways” with different computing architectures, communication interfaces and layouts are available on the market nowadays. Consequently, it is not straightforward to pick an appropriate controller that is automatically suitable for the desired application. In context of this part a market research is conducted to first, introduce various devices and device families and second, to oppose them regarding their technical specifications and IoT capabilities that are required to successfully utilize it.

(D) Sensor Sensor System Implementation

This part of the thesis is the centerpiece and deals with the practical implementation of the smart sensor system within the FabLab. Based on a detailed system overview that includes all the required components, interfaces and cloud applications, the realization is outlined in detail for all system parts such as the hardware, software, the installment on the machine and the configuration of the Platform as a Service (PaaS) IBM Bluemix⁷. This part also involves the visualization of gathered test data by providing a user interface for interaction with the sensor system and to receive automatic notifications on end device such as smartphones or computers.

⁷IBM Bluemix Cloud Computing, <https://www.ibm.com/cloud-computing/bluemix>, (retrieved 06/02/2017).

2 Internet of Things (IoT)

Nowadays the term IoT is widely spread and well known in various technological areas. Everything revolves about Smart Homes, Smart Energy and the possibility to control various appliances just by the use of a smartphone or other mobile devices. But what exactly is the IoT, how does it work, how did it develop that fast and most importantly, where did it originally derive from? This and more details about the concept of IoT and its evolution is outlined in the following chapter.

2.1 Definition and Characteristics of IoT

When introducing the term IoT it is obvious that it consists out of the two essential terms “Internet” and “Things”. Basically, the IoT is nothing else than physical objects called “Things” connected together via an open network that is accessible to anyone, which is known as the “Internet”.⁸

It was Kevin Ashton who was the first person to introduce the term IoT back in 1999, which will be discussed in section 2.2 that is about the history and the development of IoT in more detail. An appropriate definition for today’s IoT according to a McKinsey article is known as: *“In what’s called the Internet of Things, sensors and actuators embedded in physical objects - from roadways to pacemakers - are linked through wired and wireless networks, often using the same Internet Protocol (IP) that connects the Internet”*⁹.

According to this definition another two important components are required to successfully form an IoT, the “Sensors” and “Actuators” to either retrieve information from the real world and sending it to the Internet (sensors), or to

⁸Cf. BUYYA; DASTJERDI, (2016), p. 3.

⁹CHUI; LÖFFLER; ROBERTS, (2010), <http://www.mckinsey.com/industries/high-tech/our-insights/the-internet-of-things>, (retrieved 14/11/2016).

produce an output into the real world (actuators). So now it is reasonable that there is more about the term “Internet of Things” than just the two words “Things” and “Internet” as mentioned earlier. It is due to the fact that “Things”, which are the physical objects in our real world, simply cannot communicate without the use of sensors, actuators and an appropriate controller for further processing of the generated and the received data. The definition also indicates that the form of the physical objects is not specified and can include anything from commonplace items like a simple toaster up to objects in the high-tech or medical industry.¹⁰

Putting this all together, the IoT can be described as the sum of a few fundamental components, illustrated in Fig. 2.1 below.

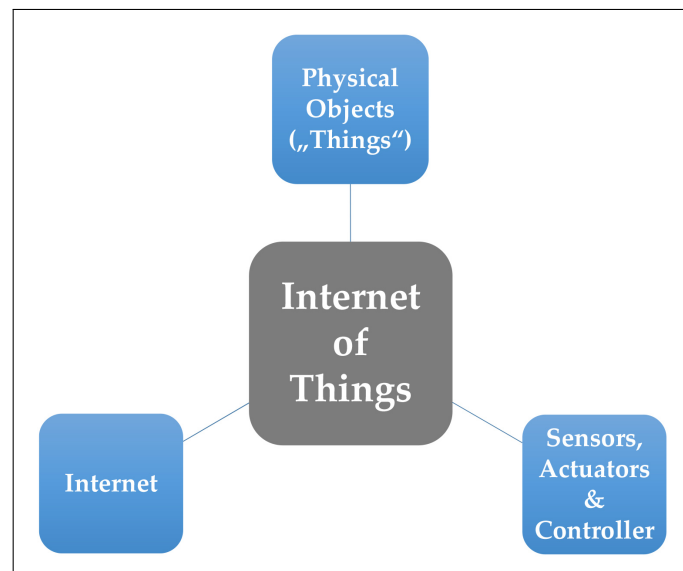


Figure 2.1: Basic Components of the IoT¹¹

The different components that are required to form an IoT will be discussed in section 2.3 in more detail. It is important to understand that the physical object (thing) does not change its form at all, although computing capabilities are being added by the installation of sensors and the connection to the Internet.¹²

Depending on the scope, the IoT can also be referred as the “Internet of Everything” (IoE), which extends the communication of physical objects with people, data and processes.¹³

¹⁰Cf. MCEWEN; CASSIMALLY, (2013), pp. 10-11.

¹¹Own illustration based on *ibid.*, p. 11.

¹²Cf. *ibid.*, p. 11.

¹³Cf. BRADLEY, (2013), p. 2.

2.2 History & Evolution

2.2.1 Starting Point & Development

As already mentioned in section 2.1, the term “Internet of Things” was coined by Kevin Ashton back in the year 1999. Ashton, co-founder of the Auto-ID Center of MIT was working on the development of Radio Frequency Identification (RFID) in context of supply chain optimization. The idea was to extend RFID uses, which are in general limited to near field applications, to broader domains. During a presentation about supply chain management and Ashtons visions to improve RFID, the term “Internet of Things” (IoT) had its first appearance and the idea of linking devices together was born.¹⁴

Although the term and the basic idea is still the same today, the concept of IoT is significantly different in 2016 than it used to be 16 years ago. Of course, this is also caused by the rapid development of technology. In 1999 the Internet was not mainstream yet and the Wireless Fidelity (WiFi) technology, which makes the connection of objects much easier and flexible, was not available at all for commercial uses. The concept Ashton was tracing at this time was not exactly the idea of assigning each device a unique identification number (IP-address) within a network, because it was technically inconceivable in terms of network availability, speeds and storage capacities. The concepts and the ideas about IoT have proceeded hand in hand with the rapid improvements regarding network technology and technology in general over the last years. New communication standards like Bluetooth, improved WiFi generations, enhanced network speeds and storage capabilities became available and all the services became much more affordable. This has increased the number of connected devices to the Internet tremendously and entirely new possibilities have been opened for IoT and its use cases.¹⁵

¹⁴Cf. BUYYA; DASTJERDI, (2016), p. 5.

¹⁵Cf. TOZZI, (2016).

2.2.2 Internet Protocol v6 (IPv6)

A key part in terms of IoT evolution was the launch of the next generation Internet Protocol Version 6 (IPv6) with increased address space back in the year 2011. Through the implementation of the IPv6 address allocation it was possible to assign a total amount of 340 sextillions ($2^{128} = 340 \cdot 10^{36}$) unique IP addresses by enhancing the address size to 128 Bit. In comparison to this, the previous standard Internet Protocol Version 4 (IPv4) only provided an address space of 32 Bit which results in a total amount of around 4.3 billion ($2^{32} = 4.3 \cdot 10^9$) addresses. Because of the fast increase of connected devices over the past years and according to different market forecasts that have predicted an enormous incline of connections in future, the limitation of IPv4 was a serious issue and the remaining addresses started to run low. This was the actual trigger to start with IPv6. With the launch of IPv6 and its large address space it became feasible to assign each physical object that is part of the IoT an unique IP-address, just like any computer or notebook that is connected to the Internet.¹⁶

2.2.3 Growth Forecast & Market Size

Over the last years a lot of companies were publishing future trends regarding the development of IoT, however, they have to be treated with caution because they show remarkable differences concerning the growth rates and it is difficult to estimate whether a forecast is feasible or not. The following Fig. 2.2 shows a forecast regarding the increase of connected devices from the year 2004 to 2018. It is noticeable that the IoT sector was setting off around the year 2010, just when IPv6 was about to start, and that its total share inclines faster than any other Internet related branch from that point.¹⁷

According to a market research that was conducted by RnRMarketResearch, the IoT market worth will rise to almost \$500 billion by the year 2019. Compared to the market value of \$44 billion in 2011, this would be the result of more than a tenfold increase. The fast increase over the last years is also associated with leading

¹⁶Cf. VERMESAN; FRIESS, (2014), pp. 226,228.

¹⁷Cf. ROSENQUIST, (2014).

tech companies like Microsoft, Google, IBM, Cisco or Intel that have started to cooperate to push the development of IoT and its market growth.¹⁸

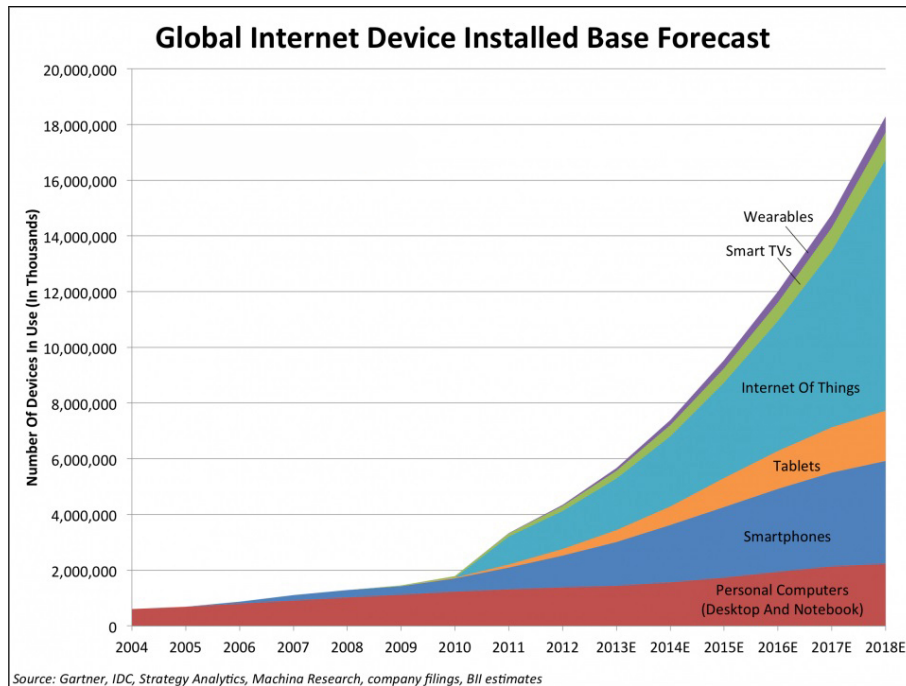


Figure 2.2: Connected Devices to the Internet - Forecast¹⁹

2.3 IoT Components & Architecture

In the following section the most important components that together form the Internet of Things will be discussed. Depending on the architecture and the type of an IoT platform the components may vary or additional components are needed. In general, IoT Systems require a cloud-based data storage within a cloud-computing service where data, gathered from sensors in the real world, is being recorded. The historical data can be analyzed afterwards within provided cloud computing services to understand a certain behavior of a machine or any other physical object. However, if only real-time data generated by the sensors is relevant, e.g. to trigger an event or an actuator based on a threshold value of a sensor, then the recording of historical data may not be necessary and thus, the implementation to the IoT platform is not reasonable. Note that the three main constituents according to Fig. 2.1 are always part of an IoT system, otherwise it would be a different concept

¹⁸Cf. BUYYA; DASTJERDI, (2016), p. 6.

¹⁹Modified illustration from ROSENQUIST, (2014)

of connected devices, such as Machine to Machine (M2M). The easiest scenario is to connect each sensor directly to the Internet via WiFi or a wired Ethernet connection, but realizing this requires a full Transmission Control Protocol/Internet Protocol (TCP/IP) stack plus a controller implemented to each sensor. Depending on the amount of data that is being collected and depending on the total number of sensors and actuators used for the application this is very inefficient, so “gateways” are used to overcome this issue.²⁰

Fig. 2.3 shows an overview of a possible IoT architecture including the key components and common communication protocols between the instances.

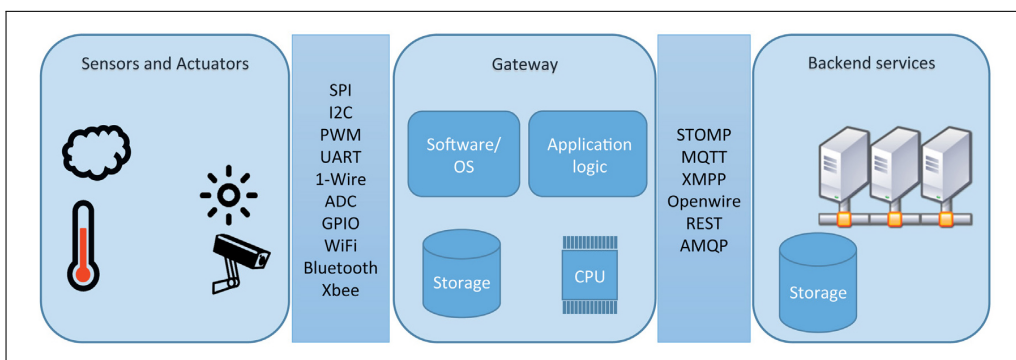


Figure 2.3: IoT Example Architecture - Key Components and Protocols²¹

2.3.1 Physical Objects & Smart Objects

A physical object, often referred as a “Thing” in context of IoT, could be any possible item in the environment of the real world no matter if it is a daily item, a machine, a vehicle or a building. The illustration according to the following Fig. 2.4 shows a classification of objects based on the size, the moveable aspect, the complexity and even if they are animate or inanimate.²²

A “Smart Object” on the other hand must not be confused with the physical object itself, although it is derived from a non-smart object and its original attributes remain the same. The essential difference is that the smart object is equipped with a form of a sensor or actuator and a controller for processing the data, therefore, it gains the ability to interact with the physical world and other smart objects.²³

²⁰Cf. BUYYA; DASTJERDI, (2016), p. 278.

²¹Illustration from *ibid.*, p. 279.

²²Cf. CHAOUCHI, (2010), p. 7.

²³Cf. VASSEUR; DUNKELS, (2010), p. 3.

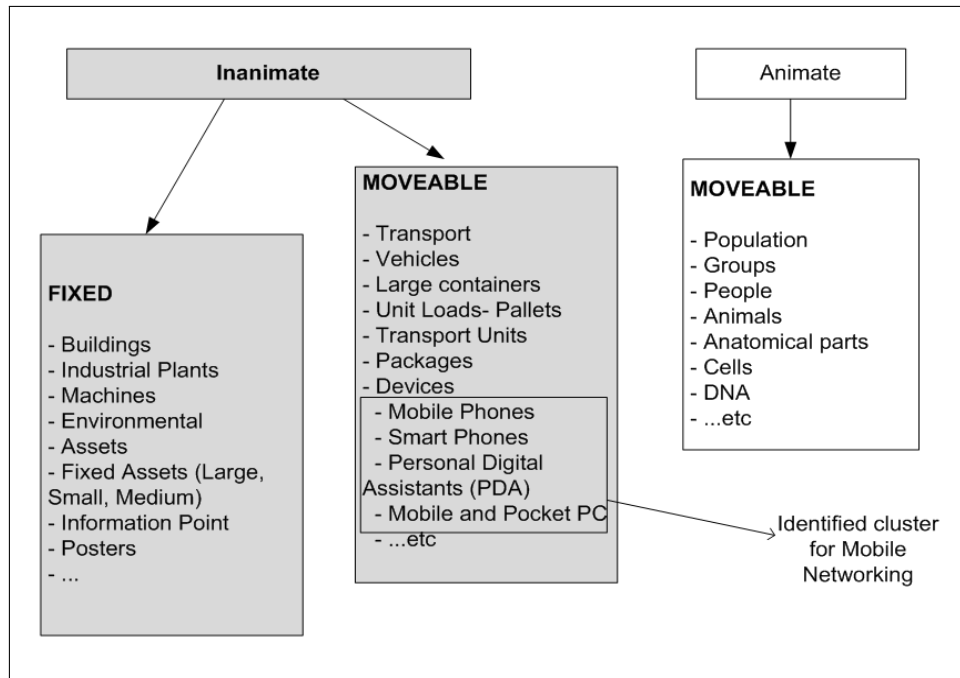


Figure 2.4: Objects “Things” of the IoT - Classification²⁴

Note that communication devices such as mobile phones or PCs are already connected objects using a wireless or wired communication. Nevertheless, in this terminology they are still considered as “Things” because IoT contributes to extend the connectivity and interconnection of these existing objects with new objects.²⁵

2.3.2 Sensors

Sensors in IoT systems are essential key components and responsible for collecting data of the physical objects or environmental factors in the real world. Basically, a sensor is a device that translates a non-electrical physical unit, e.g. the temperature of an object or the environment, into a proportional electrical output signal in the form of a voltage or current. In a physical perspective, a sensor is an energy transformer. Certain types of sensors are also referred as “detectors”, especially when it comes to applications like measuring a movement or a distance from an object.²⁶

²⁴Modified illustration from CHAOUCHI, (2010), p. 8.

²⁵Cf. *ibid.*, p. 8.

²⁶Cf. FRADEN, (2016), pp. 2-3.

A Microcontroller Unit (MCU) cannot interpret continuous analog values directly, so the analog output signal of the sensor has to be converted into a digital form using the Analog Digital Converter (ADC) of the microcontrollers input.²⁷

The simplest form of a sensor is a pushbutton, which transforms an user input into a logic state that is interpreted digitally either as “0” or “1”. Sensors that only distinguish between a series of low and high states are called “Digital Sensors”, they do not require an additional analog-to-digital conversion just like analog sensors do.²⁸

Communication with the gateway is done via various available electronic interfaces such as ADC, Pulse Width Modulation (PWM), Inter Integrated Circuit (I2C) or General Purpose Input Output (GPIO).²⁹

The selection of a sensor for an IoT system clearly depends on the requirements and the accuracy that has to be met. Sensors have different properties like the overall accuracy, measurement range, sampling rate, power demand and they have an accuracy drift depending on environmental influences like the temperature. Thus, before picking a sensor the datasheet has to be studied carefully to ensure that it is adequate for the application.³⁰

2.3.3 Actuators

Compared to a sensor, the actuator is the exact opposite in terms of functionality. The actuators input is driven by an electrical signal which is then translated to a physical signal as an output.³¹

Good examples for actuators in the field of IoT are speakers or visual displays. A monitoring screen is able to display data points measured at the sensor site and the speaker can additionally output an acoustic alert once a defined critical value has exceeded. The output of an actuator can also be in the form of a mechanical action, like the movement of an electric motor or an electro-pneumatic cylinder.³²

²⁷Cf. FRADEN, (2016), p. 5.

²⁸Cf. WINKLER, (2014).

²⁹Cf. BUYYA; DASTJERDI, (2016), pp. 279-282.

³⁰Cf. FRADEN, (2016), pp. 7-11.

³¹Cf. *ibid.*, p. 3.

³²Cf. MCEWEN; CASSIMALLY, (2013), pp. 90-91.

2.3.4 Gateway

As already mentioned in section 2.3 it is cost-inefficient to implement the Internet ability via a TCP/IP stack to every single sensor, particularly when there is a large number of sensors used for the system. Therefore, “gateways” are used as an interface between the sensor site and the cloud-based backend service in the Internet. The gateway usually consists out of a microcontroller/microcomputer with Internet capability and the appropriate development environment/operating system. A MCU is a combination of a Central Processing Unit (CPU) with Read Only Memory (ROM), Random Access Memory (RAM) and input/output peripherals for the communication with external sensors and actuators, put together on a Printed Circuit Board (PCB). Some modern controllers are also designed as a System on Chip (SoC) solution, where all components are integrated in one single Integrated Circuit (IC). Depending on the type of the controller the Internet capability is either already on-board by the use of WiFi and Ethernet, or it has to be extended using a “Shield” which is an additional PCB that adds a full TCP/IP stack for Internet capability to the controller.³³

2.3.5 Cloud Computing Services

Cloud computing was introduced as a term that describes on-demand computing services offered by certain providers. The computing infrastructure is classified as the “cloud” where individuals can access the infrastructure, applications, services as well as storage from anywhere via the Internet³⁴

There are three fundamental instances of cloud-based services models available, the Infrastructure as a Service (IaaS), PaaS and the Software as a Service (SaaS), which are offered in different environments of providers.³⁵

Pricing of these services, if they are not free to use, is usually based on pay-per-use, subscription, or on the amount of infrastructure and virtual storage that is requested.³⁶

³³Cf. BUYYA; DASTJERDI, (2016), pp. 286-287.

³⁴Cf. BUYYA; BROBERG; GOSCINSKI, (2011), p. 3.

³⁵Cf. SRINIVASAN, (2014), p. 9.

³⁶Cf. AL-ROOMI, (2013), pp. 101-103.

Infrastructure as a Service (IaaS):

The IaaS model represents the lowest level of the model hierarchy and already provides the server infrastructure such as storage, processing power and networks. The benefit of this model is that cloud users do not need to manage and control these resources themselves, they just need to install the necessary operating system as well as the software components and can then deploy their applications. Therefore, the IaaS is also identified as the hardware-level-service.³⁷

Examples: IBM Cloud, Amazon EC2, Google Compute Engine,...

Platform as a Service (PaaS):

PaaS is a system-level-service and provides users a computing platform for developing, deploying and testing of applications directly within cloud services without the need of local development environments. Programming languages and tools that are required to build applications are already supported by the PaaS. The applications then run on the infrastructure of the provider and are presented via the Hypertext Transfer Protocol (HTTP) interface of the Internet to the end-users.³⁸

Examples: IBM Bluemix, Microsoft Azure, Google App Engine,...

Software as a Service (SaaS):

SaaS is the application-level-service and ranked on top of the cloud computing services. SaaS allows users to access the applications of the provider that are ready to use via a client interface, e.g. a web browser, without the need of installing the application locally plus they do not have to worry about managing the application and the underlying cloud infrastructure.³⁹

Examples: Google Docs, Microsoft OneDrive, Dropbox,...

³⁷Cf. SRINIVASAN, (2014), p. 10.

³⁸Cf. *ibid.*, p. 12.

³⁹Cf. *ibid.*, p. 12.

2.3.6 Data Transmission

The sensor data needs to be transmitted from the sensor installation site to the gateway and furthermore from the gateway to the cloud services via the Internet. Because an IoT platform also includes actuators that are triggered with commands coming from the gateway and respectively the backend services, the connection of the instances has to be bi-directional in order to provide the two-way communication. Examples for the low-level interfaces between a MCU or Single Board Computer (SBC) board that acts as the gateway have already been mentioned in section 2.3.2.

The communication between the gateway and the cloud service is achieved by the use of wired or wireless Internet access and an appropriate TCP/IP messaging protocol, where the data gets packed and formatted properly for the transmission via a message payload.⁴⁰

The most common messaging protocol with included TCP/IP stack is the HTTP, which is primarily known as the standard communication protocol for the World Wide Web (WWW) using a request/response model.⁴¹

However, for IoT applications protocols like Data Distribution Service (DDS), Advanced Message Queuing Protocol (AMQP), Extensible Messaging and Presence Protocol (XMPP) and in particular the Message Queue Telemetry Transport (MQTT) protocol are utilized to a wide extend, since they are better optimized for it.⁴²

Especially MQTT is beneficial, because it is a very light weighted protocol with a small message size, thus, it provides better capabilities in constrained environments compared to HTTP. This are apparently essential advantages when it comes to the connection of a large number of sensors and actuators.⁴³

The MQTT protocol is furthermore used for the communication within the practical part of this thesis where some more details about it are discussed in section 5.3.3.

⁴⁰Cf. BUYYA; DASTJERDI, (2016), p. 290.

⁴¹Cf. LAMPKIN, (2012), p. 7.

⁴²Cf. BUYYA; DASTJERDI, (2016), p. 292.

⁴³Cf. LAMPKIN, (2012), p. 4.

2.4 Other Concepts beside IoT

Besides IoT there are also two other related concepts that are relevant for applications in the industrial area and follow the idea of “connected devices”. To a certain extent they are very similar to IoT and for some parts they are overlapping, nevertheless there are some particular differences between the approaches that have to be outlined.⁴⁴

2.4.1 Machine to Machine (M2M)

M2M is, as well as IoT, a concept that allows communication of devices that are linked together by either using a wired or wireless connection. However, the essential difference to IoT is that the devices in M2M are all of the same type and that each solution is for one specific application for the use within a smaller domain. Thereby it is usually not intended to share sensor data directly via the Internet, but rather within a Local Area Network (LAN) or Wide Area Network (WAN) environment. Fig. 2.5 shows a direct comparison of the characteristics of IoT and M2M, additionally divided into different aspects.⁴⁵

Aspect	M2M	IoT
Applications and services	Point problem driven	Innovation driven
	Single application - single device	Multiple applications - multiple devices
	Communication and device centric	Information and service centric
	Asset management driven	Data and information driven
Business	Closed business operations	Open market place
	Business objective driven	Participatory community driven
	B2B	B2B, B2C
	Established value chains	Emerging ecosystems
	Consultancy and Systems Integration enabled	Open Web and as-a-Service enabled
Technology	In-house deployment	Cloud deployment
	Vertical system solution approach	Horizontal enabler approach
	Specialized device solutions	Generic commodity devices
	De facto and proprietary	Standards and open source
	Specific closed data formats and service descriptions	Open APIs and data specifications
	Closed specialized software development	Open software development
SOA enterprise integration	Open APIs and web development	

Figure 2.5: IoT vs. M2M - Differences⁴⁶

⁴⁴Cf. MOOLAYIL, (2016), pp. 12-14.

⁴⁵Cf. HÖLLER, (2014), p. 11.

⁴⁶Illustration from *ibid.*, p. 37.

A typical M2M solution is the remote monitoring of enterprise assets, where operators will be informed when and why a machine or product needs maintenance. Fig. 2.6 shows an overview of additional M2M applications including estimated market sizes of deployed M2M devices from the years 2012 and 2016.⁴⁷

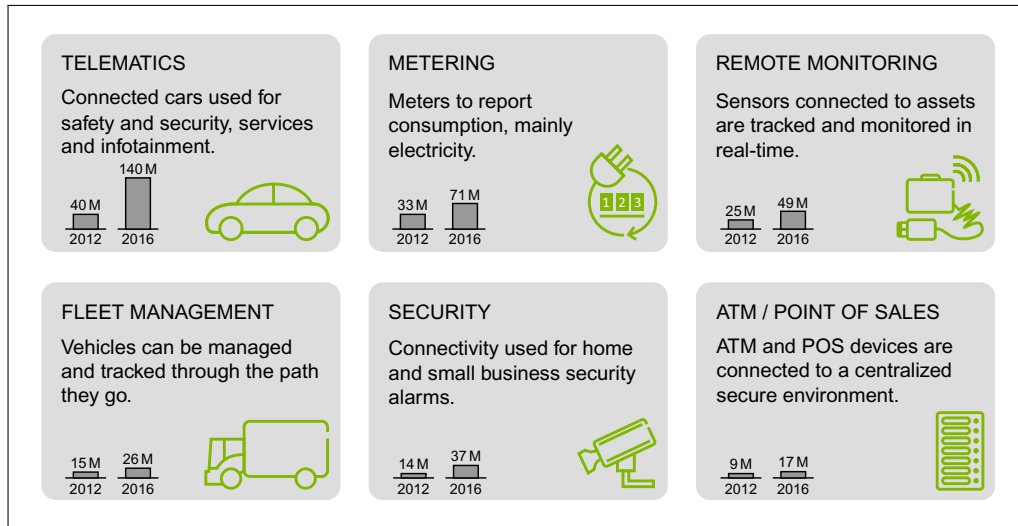


Figure 2.6: M2M Applications & Market Size⁴⁸

2.4.2 Industrial Internet of Things (IIoT)

The Industrial Internet of Things (IIoT) is not exactly another concept of IoT, but is rather to be considered as a subdomain or a special use case of IoT. Basically IIoT is the application of IoT in the industrial manufacturing section to improve the productivity.⁴⁹

Sensor technology has enhanced in recent years by reducing the size and costs of the components and so the instrumentation of devices, manufacturing machines and processes for capturing data became technically as well as financially feasible. Big amounts of data generated by the industrial equipment is stored within cloud computing services and can be used to perform advanced analytics to acquire useful information, which is known as the Big Data Analysis. This provides insights into machines and their processes to furthermore improve the procedures by predictive analytics such as machine-learning or data mining.⁵⁰

⁴⁷Cf. HÖLLER, (2014), p. 13.

⁴⁸Illustration from *ibid.*, p. 13.

⁴⁹Cf. LAMBRECHTS; SINHA, (2016), p. 13.

⁵⁰Cf. GILCHRIST, (2016), p. 4.

Other IoT cloud computing services such as the visualization of real-time machine data and the information as well as the notification of operators and users based on specific events or triggers can be implemented as well.

2.5 Application Areas

The IoT, together with M2M and other subdomains, has a wide field of applications nowadays and it is present in almost any possible scenario, from the transport and mobility sector to health monitoring up to the complete interconnection of a building and its objects as well as the integration into intelligent electrical power grids. In context of all the different applications that are illustrated in Fig. 2.7 below, the phrase “smart” does not only relate to the connection of objects as known from its definition. It is also a term used for energy-efficiency and energy saving applications and thoughts.⁵¹

The following section will discuss several areas of applications and their benefits for the users and the environment.

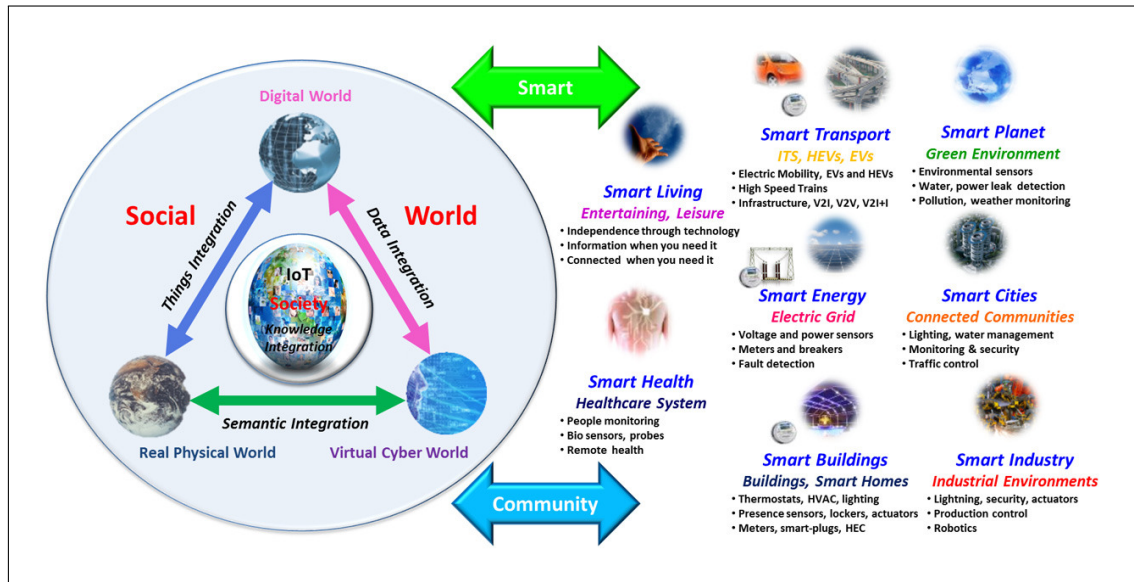


Figure 2.7: Smart Environments & Applications⁵²

⁵¹Cf. VERMESAN; FRIESS, (2014), p. 23.

⁵²Illustration from *ibid.*, p. 23.

2.5.1 Smart Grid & Energy

Nowadays there is an increasing tendency to utilize renewable energy resources to prevent climate changes caused by the further use of fossil energy resources. The idea of smart energy supply is based on the intelligent connection of various energy sources, consumers and the infrastructure to react on power fluctuations and peaks by balancing the energy using flexible and smart electrical power grids. The functionality of the “Smart Grid” is achieved by implementing the IoT concept of connected devices within a network.⁵³

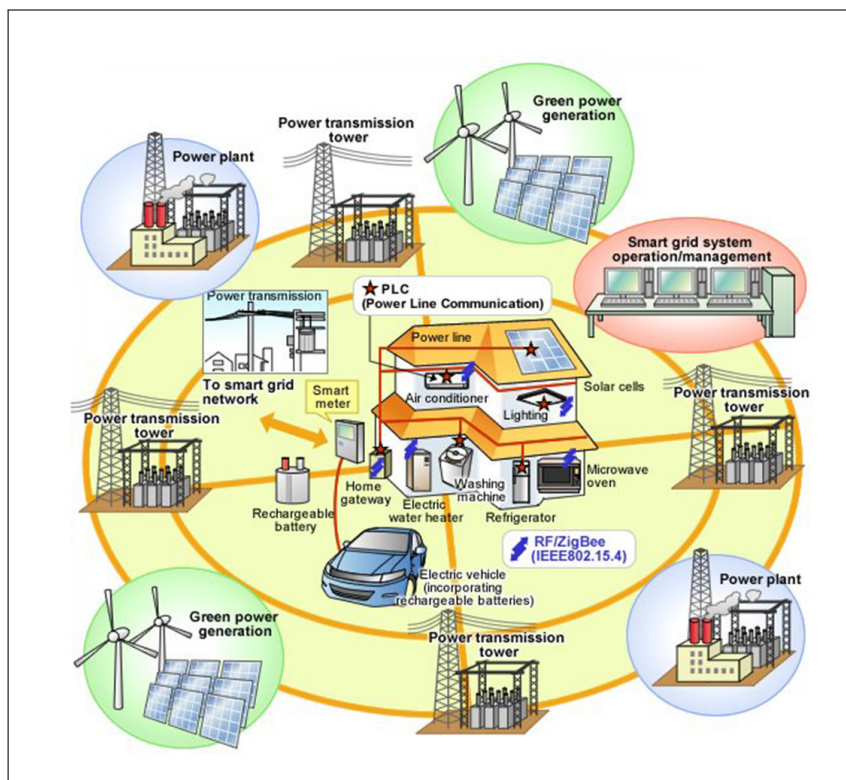


Figure 2.8: Smart Grid System - Overview⁵⁴

The implementation of such a future Smart Grid is expected to partially correspond with the Internet, where energy within the grid is managed more or less like data packets in the Internet using gateways and routers to identify the best paths for the energy flow. This concept allows energy exactly to be transferred where and when it is needed plus the monitoring of power consumption on different levels, from the single household using “Smart Meters” up to national and international levels is

⁵³Cf. VERMESAN; FRIESS, (2014), pp. 45-47.

⁵⁴Illustration from *ibid.*, p. 45.

feasible. A Smart Meter is a connected device that gives information about the real-time energy consumption to the users, thus, it is easier to identify and to eliminate energy wasting devices and to reduce the overall energy consumption.⁵⁵

2.5.2 Smart Health & Medical Applications

IoT is a big opportunity for today's healthcare management by performing real-time monitoring of people's health status and the recording of various physiological parameters like the body temperature or blood pressure using wearable, or even implemented sensors. The gathered information is provided directly to the user by the utilization of a mobile receiver (concentrator) and can be transmitted to a cloud based server via the Internet where medical staff or doctors can access the data online using appropriate backend services. The remote monitoring is helpful for the detection and early prevention of diseases as well as for physical examinations where doctors can resort to the historical records in addition to the local clinical tests. Fig. 2.9 shows an example for a cloud based patient health monitoring system.⁵⁶

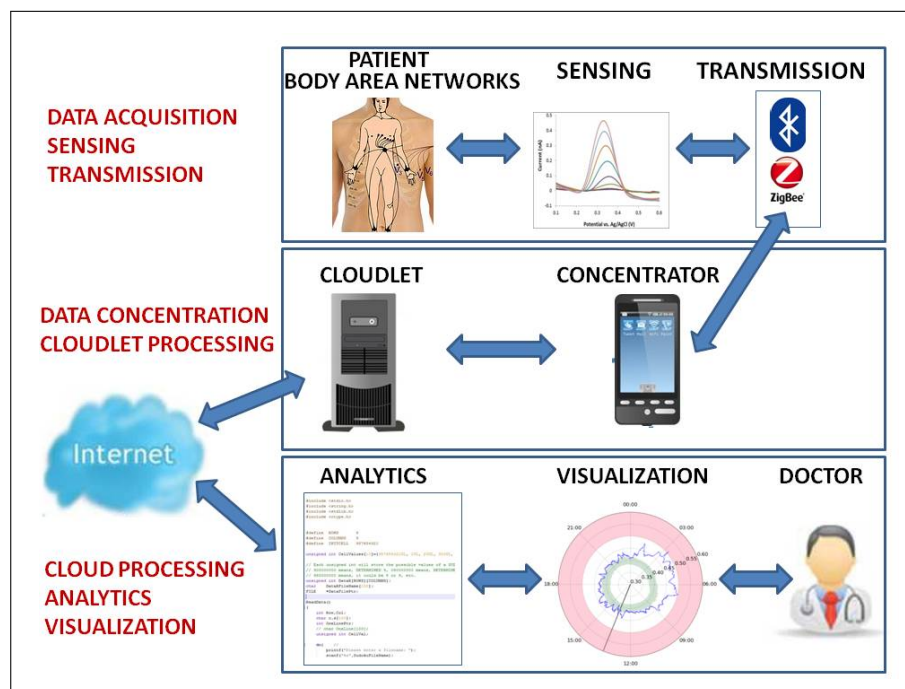


Figure 2.9: Cloud-Based Healthcare Monitoring System⁵⁷

⁵⁵Cf. VERMESAN; FRIESS, (2014), pp. 45-47.

⁵⁶Cf. HASSANALIERAGH, (2015), p. 285.

⁵⁷Illustration from *ibid.*, p. 285.

This large set of recorded data considered over a longer timespan allows medical staff to make a much better and more precise prognosis and therefore the quality of a patient's health can be improved. As a result of this disruptive technology, healthcare costs could be essentially reduced and the healthcare system could be much more efficient by improving the quality and the diagnostic speeds.⁵⁸

2.5.3 Smart Homes & Buildings

The upcoming trend of IoT and the connection of everything is also to be found in the subject of homes and buildings. A network of intelligent sensors and actuators are integrated in household facilities and devices like the heating, air conditioning, lightning or the security system. Therefore, it is possible to use a single device such as a tablet or smartphone to receive information about the state of the home, or to remote control the facilities using cloud based services and mobile backed services of different providers, e.g. the Shaspa⁵⁹ smart home system, even when not being at home.⁶⁰

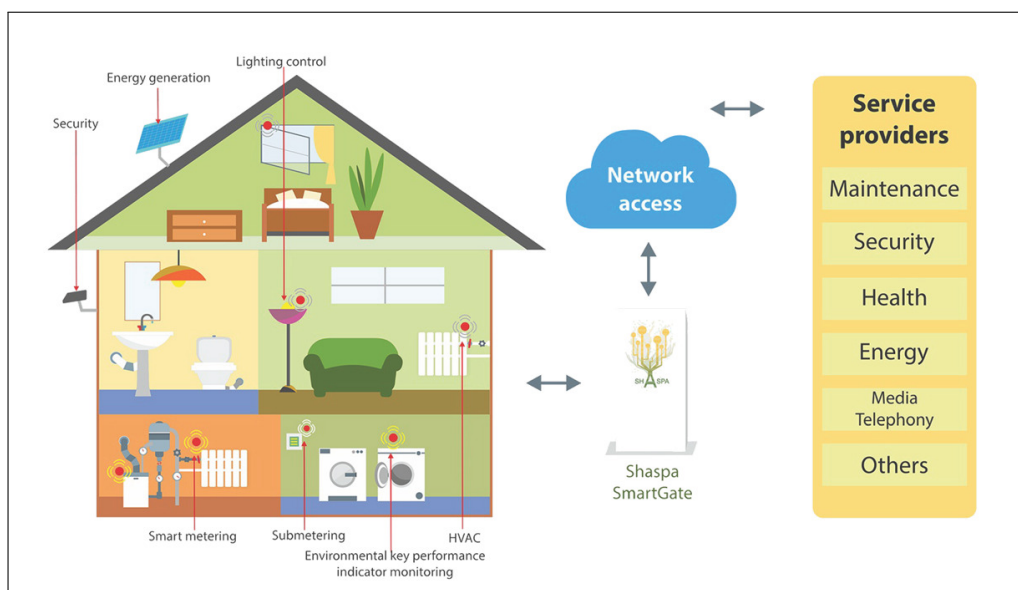


Figure 2.10: Smart Home - Example⁶¹

⁵⁸Cf. HASSANALIERAGH, (2015), p. 285.

⁵⁹Shaspa Smart Home System, <http://www.shaspa.com/smart-home/Smart-Building/2012/12/smart-home>, (retrieved 06/02/2017).

⁶⁰Cf. VERMESAN; FRIESS, (2014), p. 56.

⁶¹Illustration from *ibid.*, p. 56.

2.6 IoT Risks & Privacy Challenges

As it can be seen from the outcomes of the actual chapter, IoT has developed rapidly within the last years and it offers big potential for various application areas. Nevertheless, there are certain topics that are very controversial, particularly the privacy of the user data and the security of IoT systems. Most of all when it comes to the privacy of personal data that is gathered, e.g. in smart home systems, security is a big challenge. There are various security threads in IoT systems which are mainly to be found in the network connection. Since all the objects and things that gather sensitive user data are connected to the Internet, data is exposed to threats either through direct attacks by trying to access the cloud services and therefore the dataset, or by infiltrating malware to host devices to furthermore grand system access. It is horrific to see that for some self-managed security applications, like the use of home surveillance cameras (IP cameras), the access is sometimes not protected at all even though the user might assume. Different Internet device-scanning search engines allow to find accessible sensors and to make many devices visible for the whole world if they are not password protected. This example demonstrates how fast and easy personal information can end up in the wrong hands, and that this is a big issue to overcome for IoT and its further development in future.⁶²

⁶²Cf. LIN; BERGMANN, (2016), pp. 4-6.

3 Sensor System Concept

The following chapter deals with the considerations how a smart sensor system for the FabLab Graz machinery can be applied, by creating an appropriate concept that serves as the basis for the practical implementation and the installation of the system on the machine(s).

The foundation for it is based on section 2.3, where the theoretical background of IoT systems and its required components were discussed in detail. No restrictions were given on how to implement the system and what platforms or components have to be used, except the cloud computing backend service which was predetermined to be the platform called “IBM Bluemix”. Because Bluemix, as many other well established PaaS providers, are mostly based on a pay-per-use pricing, free trial codes were provided by IBM during and already before the start of this thesis. Therefore, it was preferred to use Bluemix as the cloud-platform service for the purpose of this thesis.

3.1 Basic Considerations

As it can be seen out of the objectives that were defined in section 1.2, a bi-directional sensor system that is capable of sending machine related data as well as sending/receiving user specific commands and notifications for manual intervention is required. Gathered sensor data should be transmitted via the Internet to the cloud service for prospective big data analytics and storage, but should also be displayed in real-time via an implemented information dashboard that is accessible from various devices like a host PC or a mobile device in order to supervise the machine activities.

3.1.1 FabLab Equipment

The FabLab in Graz provides a variety of standardized FabLab equipment such as 3D printers, laser cutters, CNC milling machines as well as an electronic workbench.⁶³

For the development of the sensor system, in order to make the machines “smart” and intelligent, it is preferable to only pick one specific machine acting as the reference device for testing, rather than the application to several machines at a time. This keeps the effort for the implementation lower and once the system is successfully working it can still be adopted to other machines by adjusting the required sensors at the installation site. The machine that is utilized for the initial installment of the sensor system is a 3D printer, more specifically the “Ultimaker 2 - Extended” that uses Fused Deposition Modeling (FDM) technology and supports an “extended” build volume compared to the standard version. This printer is frequently used in the FabLab and affords easy accessibility for the sensor assembly. Depending on the installation site and the type of the sensors additional mounting mechanisms may also be required in order to attach the sensors appropriately and save. For the installation of the sensor equipment it is also important to ensure that the accessibility to the machine, e.g. for maintenance or repair work, is not restricted due to the installed parts.

3.1.2 Components

To develop the sensor system, various components including sensors and actuators are required where the types depend on the machine for installment. For example, when installing the sensors on the 3D printer “Ultimaker 2 - Extended” which is the reference device for implementing the system, it is obvious that a parameter like the extruder temperature at the nozzle is of great interest. Thus, sensors have to be selected appropriate and of course the measurement range of each sensor has to match with the scale of the physical values that are obtained, otherwise the recorded values are incorrect. The key point of the system is the controller board, that serves as the gateway between the sensors and the cloud platform. The controller needs to be capable of processing the gathered sensor data with respect

⁶³Cf. FABFOUNDATION.ORG, (2016b).

to size and resolution and it needs to provide all the required peripherals that are needed for the sensor connection and communication. Additionally, the controller has to support network capabilities by providing a TCP/IP stack to forward the sensor data to IBM Bluemix. A market research is conducted in chapter 4, where different MCU boards and SBC solutions are compared in order to assess which controllers are most suitable for this application.

3.1.3 Connection & Sensor Wiring

For the implementation, installment and testing of the system, the 3D printer “Ultimaker 2 - Extended” is intended to be used, because this machine is frequently accessed at the FabLab opening times by students or other users, hence, it is suitable to generate a good amount of test data. For extending the system to other machines there are two possibilities. Either the same gateway, respectively the same MCU board or SBC, can be used if the machines are located fairly close, or a separate gateway for each machine needs to be implemented if the cable lengths of the sensor connection would exceed a limit where the communication gets distorted. Especially when applying sensors that make use of a bus based interface such as I2C, where multiple devices (sensors) called “slaves” participate, the maximum cable length needs to be considered as the wire capacitance that increases with the cable length, must not exceed a default limit.

The data size is also a limiting factor because not every controller is able to process a big amount of data throughput caused by a vast amount of sensors. For the connection of the sensors with the gateway, the wiring has to be reflected as well because the routing can easily become confusing when dealing with a large number of sensors and a lot of connections.

In general, the application of a gateway for each machine brings more flexibility, since every device and its gateway controller can be managed on its own and is independent from the other machines, although it is still possible to communicate among each other. However, this solution also has the disadvantage of higher costs when it comes to the connection of various machines within a FabLab. Depending on the type and capabilities of the controllers they get particularly costly.

3.1.4 Internet Accessibility

An essential point for the system consideration is the establishment of the communication chain from the sensors up to the cloud service, where data is being processed and stored. The FabLab is located within the TU Graz (TUG) campus, thus, the Internet connection has to be obtained by either using the tethered Ethernet, or the wireless TUG accessibility. Access to the TUG network is generally restricted in terms of security and therefore only possible by using appropriate user credentials in the case of WiFi, or by unlocking some devices Media Access Control (MAC) address to receive an IP address by the Dynamic Host Configuration Protocol (DHCP) server and by using the network sockets mounted within the facilities. With respect to the gateway hardware it has to be ensured that the controller and its TCP/IP stack supports this procedure in order to successfully establish a connection to forward the data.

3.1.5 The Cloud Platform - IBM Bluemix

In order to make use of IBM Bluemix that serves as the cloud computing platform, the PaaS needs to be configured properly by creating an user account for the management of the provided applications and services, like Node-RED or a NoSQL Database. As the key part of IBM IoT, Bluemix initially comes up with an “Internet of Things Platform” that basically acts as the interface for the communication with the sensor gateways by providing a suitable protocol for the data transmission. Each gateway is identified by the assignment of a unique device ID and an access token for the device registration in the cloud. The appropriate configuration of IBM Bluemix together with the necessary steps to set up the cloud service is part of the implementation chapter, where the specific realization of the sensor system on the “Ultimaker 2 - Extended” 3D printer is explained in more detail.

3.2 System Block Diagram

The following Fig. 3.1 illustrates a very basic block diagram for the implementation of the sensor system. It shows the essential components that are required to form the overall system and it indicates the communication interfaces between

the different layers. Although the exact components and communication protocols/interfaces are not declared yet, the fundamental functionality still proceeds very well out of this conceptual block diagram.

Basically, there are three different subsystems that have to be implemented which is first, the local sensor site embedded in the FabLab machine(s). Second, the appropriate gateway that is conducted as an Internet capable microcontroller or microcomputer solution and third, the cloud-based platform executed as the IBM Bluemix PaaS. Regarding the system implementation, each instance can be considered individually, where the subsequent functionality is achieved by using convenient communication interfaces and messaging protocols.

Note that the block diagram below is only applicable for the view of a single machine in the FabLab and that no specific components or communication protocols are defined at this point. It represents a demonstration of the overall system and how all the different instances are aligned, for a better insight of the component interaction.

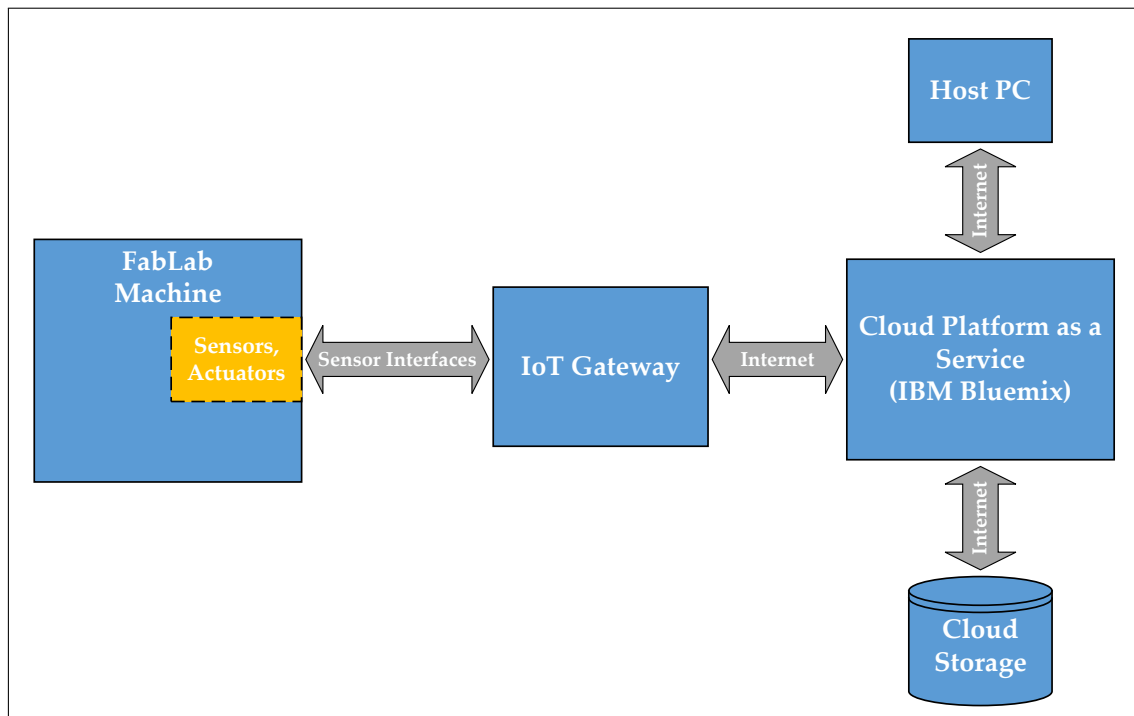


Figure 3.1: Sensor System Concept - Block Diagram⁶⁴

⁶⁴Own illustration.

4 Market Research

In the following chapter, a market research and comparison regarding IoT capable MCU boards and SBCs is conducted. As already mentioned in previous sections, the MCU board or SBC will act as the gateway between the sensor site of the IoT system and the cloud platform IBM Bluemix. Therefore, the device needs to be selected properly according to the requirements of the system and it needs to provide capabilities that IBM Bluemix can make advantage of.

Nowadays, a broad range of controllers provided by different suppliers and developers are available on the market represented in wide price ranges and performance levels. Thus, not every controller is appropriate for IoT scenarios, although it may support network functionality via Ethernet or WiFi since certain types are very limited in program memory, working memory and in terms of interfaces for the sensor connection. The allocated memory space of a controller can be insufficient very fast, depending on the complexity of the program code, the number of sensors and the required update rate of the sensor values that are gathered from the machinery. Moreover, the measured sensor data needs to be translated into an appropriate data format within the program code to successfully transmit it to the IBM Bluemix “Internet of Things Foundation” by using a suitable messaging protocol.

Within this chapter, available controller families on the market are introduced individually and are additionally compared to each other using different properties that are essential for the selection. Subsequently, a tool called “Weighted Criteria Matrix” is utilized to rate the different characteristics based on their importance to furthermore receive a scoreboard of the evaluation, whereby the highest numerical number of the overall rating represents the most suitable alternative for the application. The benefit of this tool is the possibility to weight the relevance of all criteria individually, therefore an over-influence of specific characteristics can be avoided.

4.1 Development Boards - Market Overview

The following section provides an outline of the most popular development boards at the market, including an overview of the technical specifications for each product group. The full comparison table including all the boards is represented in the appendix (A.1 Market Overview - IoT Development Boards).

4.1.1 Arduino Family

The first Arduino board was invented in 2005 by a group of the Interaction Design Institute Ivrea (IDII) to be used by their students for building projects. At this time MCU boards were rather expensive and complicated to handle. The intention was to introduce an affordable board that is easy to use, but that still includes all the necessary interfaces that characterize a MCU such as onboard serial connection for programming the device, or other common interfaces like I2C, Serial Peripheral Interface (SPI), Universal Asynchronous Receiver Transmitter (UART), ADC and GPIOs. The Arduino project was based on open source from the very beginning, meaning that the code and the schematics of the boards are accessible to anyone for better extension or adaption of the platforms to suit the individual needs.⁶⁵

Arduino also provides a compact open source Integrated Development Environment (IDE), including a text editor for writing the code using C or C++ language. The IDE also supports other controllers like an ESP8266 by including the appropriate library files.⁶⁶

These days, lots of different Arduino boards, modules, bundles and extension called “shields” are available for different application areas. The “shield” can be racked onto the board to extend its functionality with additional features, e.g. network capability for IoT applications. Fig. 4.1 shows an overview of the current Arduino product range according to their homepage. Note that not all of these products like the MKR1000 were available at the start of this market research in context of the sensor system implementation, therefore, not all of the products could be considered in the research and the comparison.

⁶⁵Cf. MCEWEN; CASSIMALLY, (2013), pp. 100-101.

⁶⁶Cf. ARDUINO.CC, (2016c).

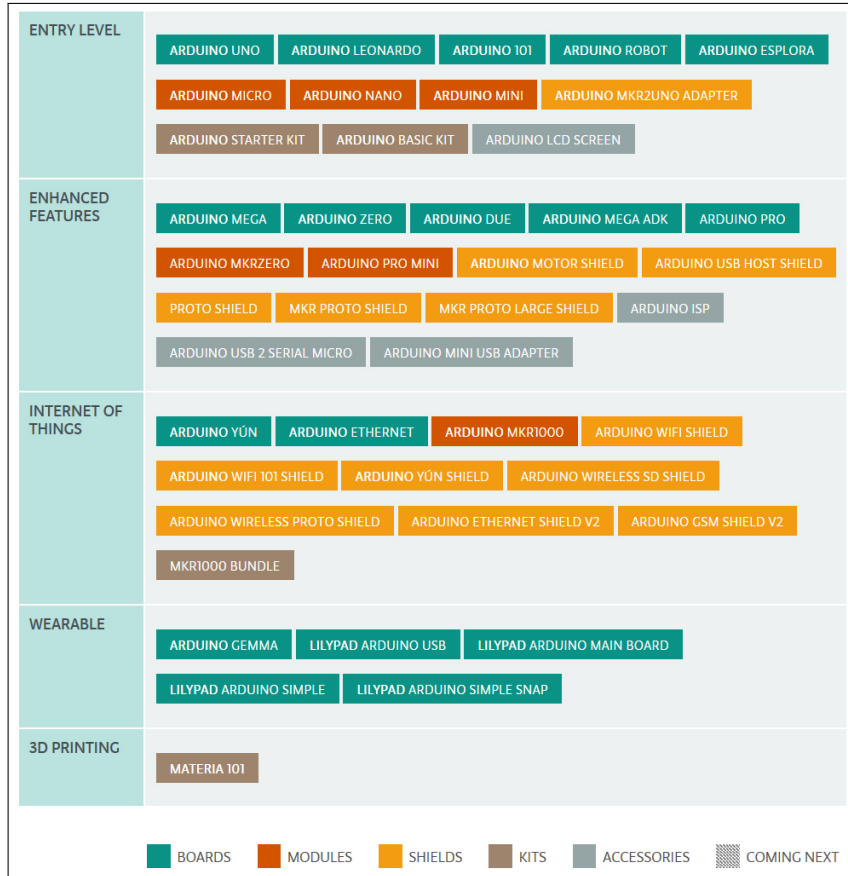


Figure 4.1: Arduino Product Range⁶⁷

As shown in Fig. 4.1 above, Arduino provides controllers within different application areas and performance levels. Low-budget entry devices like the Arduino UNO have less functionalities, are slower in terms of processing speed and they are equipped with less RAM and Flash Memory for the program storage. Devices of the “Wearable” series are not considered for the comparison, since they are not suitable at all and also the “Materia 101” which is a complete 3D printer kit is insignificant in this context.

Table 4.1 shows a tabular comparison of some Arduino based MCU boards that are potentially appropriate to serve as the IoT platform gateway, including all the detailed characteristics and technical specifications. Most of the Arduino devices are based on a type of the Atmel AVR MCUs, apart from some exceptions like the Arduino DUE that uses an Advanced RISC Machines (ARM) based MCU. There are several models, like the Arduino Yun, that already provide a built in WiFi or Ethernet module. In general, the differences between the controllers are mainly

⁶⁷Illustration from <https://www.arduino.cc/en/Main/Products>, (retrieved 12/12/2016).

to be found in the embedded MCU, the amount of GPIO Pins, RAM and Flash Memory size.

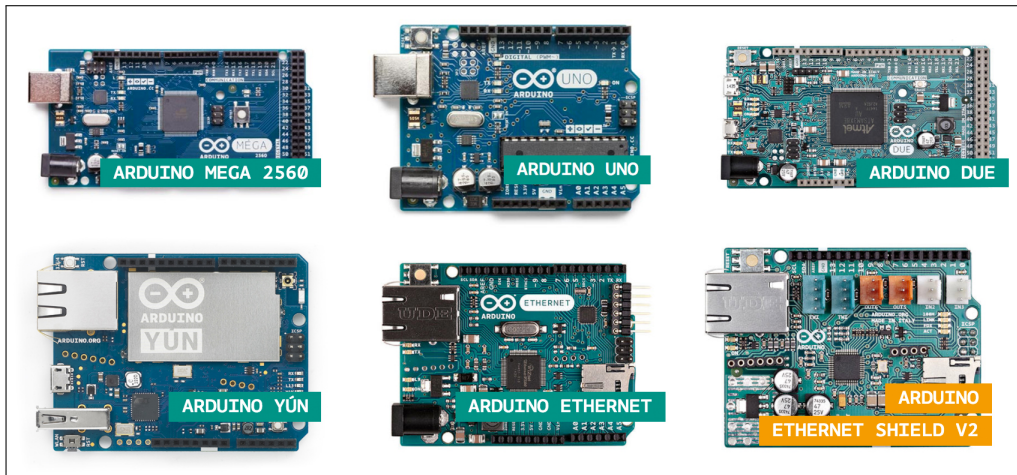


Figure 4.2: Arduino Models & Ethernet Shield (Examples)⁶⁸

At the first glance, the Arduino DUE seems to be a perfect Arduino alternative, since it comes up with a large amount of memory, a good amount of GPIOs and a fast ARM processor with enhanced features compared to an Atmega AVR. However, the issue with the DUE is that the board operates on 3.3 V only, meaning that shields or sensors that require a 5 V supply, or analog sensors that deliver a 5 V output signal, e.g. the HC-SR04 ultrasonic distance sensor, are not compatible with this board without further customization.

⁶⁸Illustrations from ARDUINO.CC, (2016b).

	ARDUINO					
	Arduino UNO	Arduino DUE (ARM)	Arduino 101	Arduino YUN	Arduino ETHERNET	Arduino MEGA
Price	\$24,95	\$49,95	\$30,00	\$74,95	\$45,95	\$59,95
Digital I/O Pins	14	54	14	20	14	54
Analog I/O Pins	6	12	6	12	6	16
Microcontroller (MCU)	ATmega328	AT91SAM3X8E	Intel Curie	ATmega32U4	ATmega328	ATmega2560
Flash Memory (MCU)	32 kB	512 kB	196 kB	32 kB	32 kB	256 kB
RAM (MCU)	2 kB	96 kB	24 kB	2.5 kB	2 kB	8 kB
EEPROM (MCU)	1 kB	none	none	1 kB	1 kB	4 kB
Clock Speed (MCU)	16 MHz	84 MHz	32 MHz	16 MHz	16 MHz	16 MHz
Microprocessor (MPU)	-	-	-	Atheros AR9331	-	-
Flash Memory (MPU)	-	-	-	16 MB	-	-
RAM (MPU)	-	-	-	64 MB	-	-
EEPROM (MPU)	-	-	-	1 kB	-	-
Clock Speed (MPU)	-	-	-	16 MHz	-	-
Input Voltage (Power Jack)	7 - 12 V	7- 12 V	7 - 12 V	5 V	7 - 12 V	7 - 12 V
Operating Voltage	5 V	3.3 V	3.3 V / 5 V	5 V	5 V	5 V
I/O Pin Current	20 mA	800 mA	20 mA	40 mA	40 mA	20 mA
Length/Width	68.6 x 53.4 mm	101.52 x 53.3 mm	68.6 x 53.4 mm	68.6 x 53.4 mm	68.6 x 53.3 mm	101.52 x 53.3 mm
Connectivity	USB Type B Power Jack	2x micro USB Power Jack	USB Type B Power Jack	USB Type B Power Jack Ethernet 10/100	6-PIN SPI Power Jack Ethernet 10/100 microSD Slot	USB Type B Power Jack
Low-level peripherals	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART, CAN	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART
Programming	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE
Real-Time Capability	Yes	Yes	Yes	Yes (MCU)	Yes	Yes
Features	-	-	Bluetooth 6-Axis accelero./gyro	MCU + MPU WiFi IEEE 802.11b/g/n	-	-

Table 4.1: Technical Specifications - Arduino Boards⁶⁹

4.1.2 ESP 8266

The ESP8266 is an integrated SoC solution invented by the Chinese company Espressif, including a WiFi chip with full TCP/IP stack as well as a 32 Bit Tensilca MCU. The chip features an Ultra Low Power (ULV) 16 Bit Reduced Instruction Set Computer (RISC) processor that clocks with a maximum speed of 160 MHz. Therefore, it is possible to provide an easy WiFi add-on for other MCU boards, or to use it even as a standalone solution for hosting its own software applications. The SoC comes up with all common interfaces like SPI, I2C, ADC and 16 GPIOs for digital and analog uses.⁷⁰

For better interface accessibility, different third-party manufacturers like AI-Thinker provide ESP8266 breakout modules for the possibility to attach external devices such as sensors or actuators. The simplest breakout is the ESP-01, a compact board

⁶⁹Technical specifications based on ARDUINO.CC, (2016b)

⁷⁰Cf. ESPRESSIF, (2016), pp. 1-3.

providing only two external GPIO pins and UART functionality. Larger development boards such as the NodeMCU illustrated in Fig. 4.3 below are using the ESP-12 module that offers more external GPIO pins for enhanced interface usage and connectivity.⁷¹

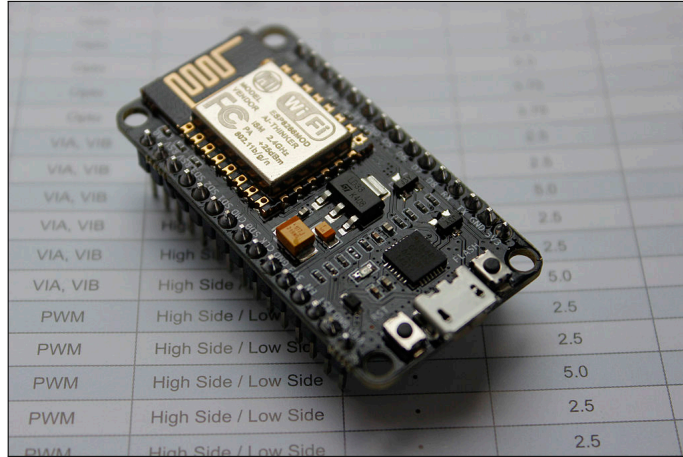


Figure 4.3: NodeMCU DevKit v1.0⁷²

The NodeMCU development kit also contains an onboard 3.3 V power regulator for USB supply of the unit and an Universal Serial Bus (USB) to serial converter (USB-TTL) for programming the device directly via micro USB cable. These implementations make it possible to use the development board in the exact way as an Arduino MCU board.⁷³

Besides the official Software Development Kit (SDK) from Expressif, there are also other open source SDKs available for the ESP8266. Especially the possibility to use the Arduino based C++ firmware makes it easy to use the SoC just like any other Arduino device within the Arduino IDE. Considering the specifications and capabilities of the board, the advantages compared to an ordinary microcontroller solution are obvious. The ESP8266 boards are much smaller and more compact compared to a common Arduino or Raspberry PI (RPI) so there is much more flexibility, for example when mounting the unit inside a machine near the sensor site. Compared to some other controllers such as the Arduino MEGA 2560, the NodeMCU SoC initially comes up with integrated WiFi and does not required

⁷¹Cf. ESP8266.COM, (2016).

⁷²Illustration from NODEMCU, (2015a).

⁷³Cf. NODEMCU.COM, (2014).

additional shields or extensions to grant network accessibility. Another big advantage is the price factor because the NodeMCU board is around 5 to 10 times cheaper compared to certain other development boards, depending on the market of purchase. Although the module is very compact and limited in interfaces, the SoC includes a remarkable size of RAM (96 kB) and Flash Memory (64 kB) which is important when dealing with a great amount of sensor data that has to be forwarded to the IBM Bluemix backend.

However, of course there are also some drawbacks compared to a larger platform. In consequence of the compact size of the ESP8266 development boards, they only provide a limited amount of GPIO pins with a maximum amount of 16 provided by the ESP-12. Another disadvantage is the missing possibility to use a cable based network communication via RJ45 just like the Arduino MEGA 2560 with Ethernet Shield solution. A tethered connection is more reliable, more interference-free and better in terms of security compared to a WiFi connection. Detailed specifications of the ESP8266 modules and development kits can be found in Table 4.2 below.

	ESP 8266		
	NodeMCU DEVKIT (ESP-12)	WiFi Module (ESP-01)	WiFi Module (ESP-201)
Price	\$12	\$6,95	\$8,95
Digital I/O Pins	13	2	
Analog I/O Pins	1	0	1
Microcontroller (MCU)	Tensilica L106 32-Bit	Tensilica L106 32-Bit	Tensilica L106 32-Bit
Flash Memory (MCU)	96 kB	96 kB	96 kB
RAM (MCU)	64 kB	64 kB	64 kB
EEPROM (MCU)	-	-	-
Clock Speed (MCU)	80 MHz	80 MHz	80 MHz
Microprocessor (MPU)	-	-	-
Flash Memory (MPU)	-	-	-
RAM (MPU)	-	-	-
EEPROM (MPU)	-	-	-
Clock Speed (MPU)	-	-	-
Input Voltage (Power Jack)	4.5 - 9 V	3.3 V	3.3 V
Operating Voltage	3 - 3.6 V	3.3 V	3.3 V
I/O Pin Current	12 mA	12 mA	12 mA
Length/Width	49 x 24.5 mm	25 x 14.5 mm	35 x 25 mm
Connectivity	micro USB	GPIO (TX, RX)	GPIO (TX, RX)
Low-level peripherals	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART
Programming	LUA, Arduino IDE	LUA, Arduino IDE	LUA, Arduino IDE
Real-Time Capability	Yes	Yes	Yes
Features	WiFi 802.11 b/g/n Deep sleep power < 10 uA	WiFi 802.11 b/g/n Deep sleep power < 10 uA	WiFi 802.11 b/g/n Deep sleep power < 10 uA

Table 4.2: Technical Specifications - ESP8266 Boards⁷⁴

⁷⁴Technical specifications based on TECHBLOG, (2015).

4.1.3 Raspberry PI (RPI)

The RPI is a small personal computer in a Single Board Computer (SBC) format. It includes USB ports, a SD card slot, an Ethernet port, a graphical interface using HDMI and a 3.5 mm audio connector for sound. Because of the compact construction it does not include on-board devices such as DVD and hard drives, nevertheless it is capable of acting like a full personal computer by connecting peripheral devices like a keyboard, mouse and a monitor. Using the SD slot, that replaces the hard drive of a conventional computer, it is possible to boot from various Linux distributions or to use a Windows 10 IoT Core Version. The RPI also supports several programming languages such as Python, C, C++, Java, Scratch and Ruby, whereby Python is the recommended language for beginners since it is rather easy to learn.⁷⁵

The first Raspberry PI model was released back in 2012, using a Broadcom BCM2835 SoC with an embedded 700 MHz ARM processor, a VideoCore IV Graphics Processing Unit (GPU) and a total amount of 256 MB Synchronous Dynamic Random Access Memory (SDRAM). Over the years, new models with better specifications were released, including the Raspberry PI 2, the Raspberry PI Zero and the latest revision, the Raspberry PI 3, in early 2016. The RPI 3 already uses 1 GB of SDRAM and has an improved Broadcom SoC (BCM2837) with a 1.2 GHz 64 Bit ARM-Cortex-A53 quad-core processor, making it a quite powerful SBC. Additionally, the RPI 3 includes integrated WiFi besides the standard Ethernet jack for easier network integration. In contrast to the traditional RPI models where the PCBs are in the dimension of a credit card, the RPI zero is designed even smaller. At a price of 5 \$ it is the low-budget entry device with similar functionalities as the first RPI Model A. Power supply for all RPI models is provided by simply using the 5 V USB bus and an appropriate USB cable, depending on the model and its USB connector. The detailed specifications of all revisions are to be found in Table 4.3 above, where all the models are compared to each other.⁷⁶

⁷⁵Cf. RASPBERRYPI.ORG, (2016b).

⁷⁶Cf. *ibid.*

	RASPBERRY PI				
	Raspberry PI 3 Model B	Raspberry PI 2 Model B	Raspberry PI Zero	Raspberry PI Model A (1/1+)	Raspberry PI Model B (1/1+)
Price	\$39,95	\$41,95	\$5,00	\$29,95	\$39,95
Digital I/O Pins	17	17	17	17 (Model 1+) 8 (Model 1)	17 (Model 1+) 8 (Model 1)
Analog I/O Pins	-	-	-	-	-
Microcontroller (MCU)	-	-	-	-	-
Flash Memory (MCU)	-	-	-	-	-
RAM (MCU)	-	-	-	-	-
EEPROM (MCU)	-	-	-	-	-
Clock Speed (MCU)	-	-	-	-	-
Microprocessor (MPU)	BCM2837 Cortex-A53	BCM2836 ARMv7	BCM2835	BCM2835	BCM2835
Flash Memory (MPU)	microSD	microSD	microSD	microSD	microSD
SDRAM (MPU)	1 GB	1 GB	512 MB	512 MB (Model 1+) 256 MB (Model 1)	512 MB
EEPROM (MPU)	-	-	-	-	-
Clock Speed (MPU)	1,2 GHz	900 MHz	1 GHz	700 MHz	700 MHz
Input Voltage (Power Jack)	5 V	5 V	5 V	5 V	5 V
Operating Voltage	5 V	5 V	5 V	5 V	5 V
I/O Pin Current	50 mA	50 mA	50 mA	50 mA	50 mA
Length/Width	85 x 56 mm	85 x 56 mm	65 x 30 mm	85 x 56 mm	85 x 56 mm
Connectivity	4x USB Type A HDMI 3.5 mm Audio Ethernet 10/100 microSD Slot micro USB WiFi	4x USB Type A HDMI 3.5 mm Audio Ethernet 10/100 microSD Slot micro USB	microSD Slot miniHDMI 2x microUSB	1x USB Type A HDMI 3.5 mm Audio	2x USB Type A HDMI Ethernet 10/100 3.5 mm Audio
Low-level peripherals	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART
Programming	Linux (Python), C, C++, Java, Scratch, Ruby	Linux (Python), C, C++, Java, Scratch,	Linux (Python), C, C++, Java, Scratch,	Linux (Python), C, C++, Java, Scratch,	Linux (Python), C, C++, Java, Scratch,
Real-Time Capability	No	No	No	No	No
Features	BCM43143 WiFi on board	-	-	-	-

Table 4.3: Technical Specifications - Raspberry PI Family⁷⁷

Originally, the RPI was developed in the UK for providing an inexpensive personal computer for educational uses, which is also capable of interacting with hardware for low-level programming. The RPI combines features of a common MCU board such as external GPIOs with the convenience of a personal computer by using a graphical operation system. Although the RPI might look like a MCU board and also shares some capabilities, it must not be confused with a MCU solution like an Arduino. The RPI, in difference to a MCU solution, is not real-time capable because of the underlying operating system and it comes with less interfaces, e.g. the missing ADC, for the connection of analog sensors. On the other hand, the RPI can handle multiple threads respectively fast compared to a MCU that is only able to process low-level tasks that are executed one after another. Therefore, it remains to be seen if the RPI is appropriate for the application of a sensor system, where multiple sensors that use different interfaces need to be connected.⁷⁸

⁷⁷Technical specifications based on RASPBERRYPI.ORG, (2016a).⁷⁸Cf. MCEWEN; CASSIMALLY, (2013), pp. 111-121.

4.1.4 Intel® Edison & Intel® Galileo Gen. 2

Intel provides, together with the Edison and the Galileo, two IoT development systems. They are based on Intel's own 32 Bit x86 Quark SoC X1000 microcontroller, which was introduced for low-power wearable devices and IoT applications.

The Intel Galileo is basically an Arduino certified SBC, which is fully based on open source and it is compatible with a wide range of Arduino Uno extension shields. In contrast to Arduino boards that use Atmel AVR MCUs, the 400 MHz clocked Intel Quark SoC applied at the Galileo is much more powerful and it provides considerably more RAM and Flash Memory than all available Arduino MCU boards. However, it does also support the Arduino IDE by using an Arduino I/O adapter that communicates with the Linux kernel of the board firmware⁷⁹

The Intel Edison comes up with an Intel Atom dual-core processor additionally to the Quark SoC X1000, for performing more powerful tasks and multiple threads. It is capable of 3D graphics acceleration and supports a full USB 3.0 stack. In difference to the Galileo, the Quark SoC that acts as the MCU is clocked down to 100 MHz. Similar to the RPI it is possible to use several Linux distributions as the operating system, but also the Arduino IDE for low-level, machine-oriented C programming can be utilized.⁸⁰

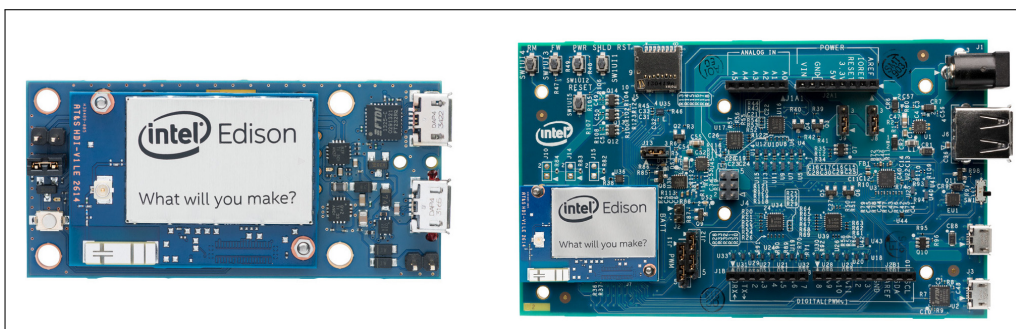


Figure 4.4: Edison Breakout Board (left) & Arduino Edison Board (right)⁸¹

Unlike the Intel Galileo that already implicates a full Arduino based MCU board, the Intel Edison is only a small module including the integrated Quark SoC and the Atom CPU. Therefore, an additional “Edison Breakout Board” has to be obtained

⁷⁹Cf. INTEL, (2014), p. 1.

⁸⁰Cf. INTEL, (2015a).

⁸¹Illustration from ibid.

in order to add microUSB connectivity that is required to gain accessibility of the module. To make use of the external interfaces and other MCU features, a larger “Edison Board” for Arduino, similar to the Galileo layout, is also available. For both solutions the Edison module is placed on the boards within the appropriate socket, as illustrated in Fig. 4.4.⁸²

	INTEL		
	Intel® Edison (Arduino Kit)	Intel® Edison (Mini Breakout Board)	Intel® Galileo Gen 2
Price	\$99,95	\$74,95	\$74,95
Digital I/O Pins	20	4	14
Analog I/O Pins	6	–	6
Microcontroller (MCU)	32 Bit Intel Quark SoC X1000 @ 100 MHz (clocked down)	32 Bit Intel Quark SoC X1000 @ 100 MHz (clocked down)	32 Bit Intel Quark SoC X1000 @ 400 MHz
Flash Memory (MCU)	–	–	8 Mb / SD card
RAM (MCU)	512 kB SRAM	512 kB SRAM	512 kB SRAM
EEPROM (MCU)	8 kB	8 kB	8 kB
Clock Speed (MCU)	100 MHz	100 MHz	400 MHz
Microprocessor (MPU)	Intel Atom Z34xx @ 500 MHz	Intel Atom Z34xx @ 500 MHz	–
Flash Memory (MPU)	1 GB	1 GB	–
RAM (MPU)	4 GB eMMC / SD card	4 GB eMMC / SD card	256 MB
EEPROM (MPU)	–	–	–
Clock Speed (MPU)	500 MHz	500 MHz	–
Input Voltage (Power Jack)	7 - 15 V	–	7 - 15 V
Operating Voltage	3.3 - 4.5 V	1.8 V	3.3V / 5.5 V
I/O Pin Current	80 mA	80 mA	80 mA
Length/Width	127 x 72 mm	61 x 29 mm	124 x 72 mm
Connectivity	2x USB Type A microUSB microSD Slot Power Jack	2x micro USB battery re-charger	USB Type A, micro USB Ethernet Power Jack microSD
Low-level peripherals	SPI, I2C, I2S, UART, PWM, ADC, PCIe	SPI, I2C, I2S, UART, PWM, ADC, PCIe	SPI, I2C, I2S, UART, PWM, ADC, PCIe
Programming	Linux Arduino IDE	Linux Arduino IDE	Linux Arduino IDE
Real-Time Capability	Yes (Via MCU)	Yes (Via MCU)	Yes
Features	WiFi Bluetooth 4.0 Arduino shield compatible	Battery Powered Supply WiFi Bluetooth 4.0	Arduino shield compatible

Table 4.4: Technical Specifications - Intel® Boards⁸³

4.1.5 Other Development Boards

Apart from the well-established controllers and SBCs with a large community such as the Arduino Family or the Raspberry PIs, a couple of other development boards are existing as well that are rather unknown. Two of them are the “BeagleBone Black” and the “LinkSprite pcDuino3”.

⁸²Cf. INTEL, (2015a).

⁸³Technical specifications based on ibid. & INTEL, (2014).

pcDuino3(B)

The pcDuino series invented by LinkSprite is a high performance, open source SBC that is able to run operation systems such as Googles Android or Ubuntu (Linux). It uses an AllWinner-A20 SoC that is based on the 1 GHz ARM Cortex A7 Dual-Core with 1 GB of Dynamic Random Access Memory (DRAM) and 4 GB internal Flash Memory, serving as the boot medium. For graphical processing, a Mali-400 GPU with hardware video processing is implemented that together with a HDMI interface supports video formats up to 1080p. Network capability is provided by the use of integrated WiFi or the Ethernet Port. Due to the specifications, the board is capable of handling multimedia and office applications just like a classic PC. The pcDuino3 is also available as the model pcDuino3(B) which includes the exact same hardware except that it supports faster Ethernet speed up to 1 Gbps instead of only 100 Mbps. This results in a larger RJ45 port that can be seen from Fig. 4.5 below. The device comes with pre-installed Ubuntu as well as the Arduino IDE to execute Arduino scripts in C/C++ language without additional modifications, although it does not provide full real-time capability.⁸⁴

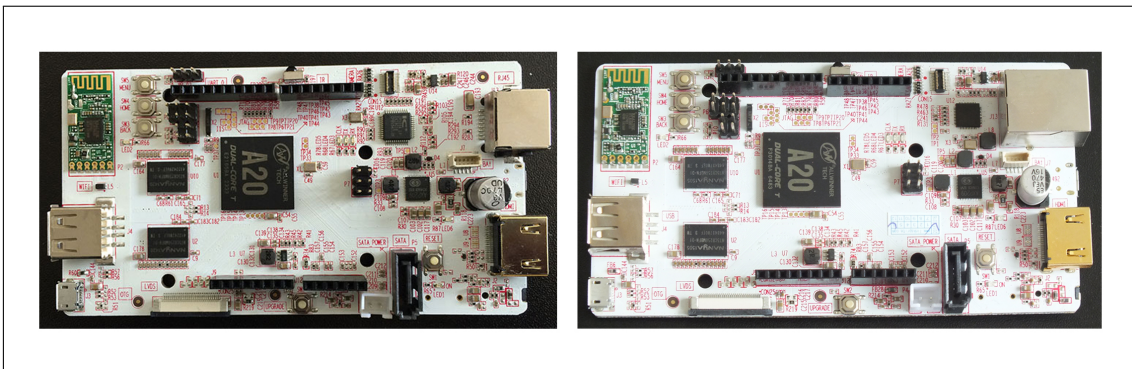


Figure 4.5: LinkSprite pcDuino3 (left) & LinkSprite pcDuino3(B) (right)⁸⁵

In contrast to many other SBCs that only provide digital I/Os, the pcDuino3 also provides 6 analog I/Os giving it better MCU characteristics, including the common interfaces I2C, SPI, UART, PWM and ADC for processing all kinds of sensors. The board also provides a battery interface for the connection and charging of standard lithium polymer batteries to use the device for mobile applications. As an extra, the pcDuino is designed for the compatibility with Arduino shields that can

⁸⁴Cf. LINKSPRITE, (2014).

⁸⁵Illustrations from *ibid.*

immediately be installed by attaching them on the pin headers of the board.⁸⁶

However, an obvious drawback of the pcDuino3 is that the output current capacity is not as high as on Arduino boards and the device is only capable of 3.3 V pin logic, thus 5 V Arduino Shields or 5 V sensors cannot be used without additional actions in order to prevent damage to the board. This makes the device rather inflexible for low-level sensor processing because sensors and actuators have to be selected according to the limited 3.3 V logic of the board.

BeagleBone Black

BeagleBone provides with its “Black” model a community-supported development SBC platform for developers, students and hobbyists that is very similar to the pcDuino3. The device is also compatible to a range of software such as Linux distributions (Debian, Ubuntu) or Android and it is possible to run the Arduino IDE on it, although it is not pre-installed like on the pcDuino.⁸⁷

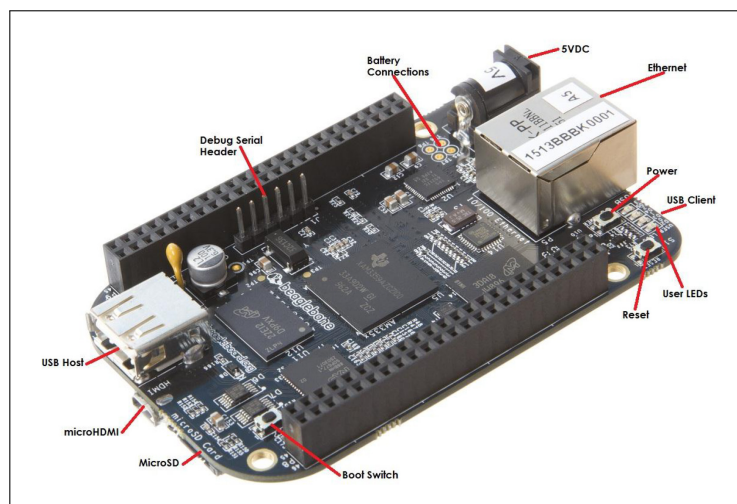


Figure 4.6: BeagleBone Black Rev. C⁸⁸

The BeagleBone Black uses a Sitara AM335x 1 GHz ARM-Cortex A8 SoC that includes 512 MB DDR3-RAM and a 4 GB on-board flash storage together with a SGX530 3D accelerator for graphical output via HDMI. In contrast to the pcDuino, the Sitara chip additionally contains two 200 MHz, 32 Bit MCUs with 12 kB RAM, called Programmable Real-time Unit (PRU), for the support of real-time processing,

⁸⁶Cf. LINKSPRITE, (2014).

⁸⁷Cf. COLEY, (2014), pp. 30-34.

⁸⁸Illustration from *ibid.*, p. 31.

which is apparently not possible under the use of an operation system such as Linux. For connectivity, the BeagleBone Black supports 10/100 base Ethernet and the board is equipped with 2x 46 pin headers for external use of the interfaces I2C, SPI, Controller Area Network (CAN), Timers and the digital as well as the 7 analog Input/Output (I/O) pins. Equivalent to the “Shields” that are used for Arduino based solutions, the Beagebone Black also supports I/O extensions known as “Capes” that are stacked onto the board by using the pin headers.⁸⁹

Note that the logic levels for all pins are such as for the pcDuino only 3.3 V, thus, the BeagleBone Black suffers from the same drawbacks regarding 5 V sensor connectivity. The following Table 4.5 shows a specification overview of the BeagleBone Black and the pcDuino3(B).

	OTHERS	
	Beaglebone Black - Rev C	pcDuino3(B) - Dev Board
Price	\$54,95	\$59,95
Digital I/O Pins	65	14
Analog I/O Pins	7	6
Microcontroller (MCU)	2x PRU 32-Bit	-
Flash Memory (MCU)	8 kB	-
RAM (MCU)	8 kB / 12 kB (shared)	-
EEPROM (MCU)	-	-
Clock Speed (MCU)	200 MHz	-
Microprocessor (MPU)	AM3358 Cortex-A8	AllWinner A20 Cortex-A7
Flash Memory (MPU)	4 GB eMMC	4 GB / microSD
RAM (MPU)	512 MB (SDRAM)	1 GB (SDRAM)
EEPROM (MPU)	-	-
Clock Speed (MPU)	1 GHz	1 GHz
Input Voltage (Power Jack)	-	5 V
Operating Voltage	3.3 / 5 V	5 V
I/O Pin Current	-	-
Length/Width	86.44 x 54.54 mm	121 x 65 mm
Connectivity	USB Type A microHDMI Ethernet 10/100 Power Jack mirco HDMI	USB Type A HDMI Ethernet 10/100 (pcDuino3) Ethernet 10/10/1000(pcDuino3B) 3.5 mm Audio SATA socket
Low-level peripherals	SPI, I2C, UART, Timers, CAN, ADC	SPI, I2C, ADC, UART, PWM, ADC
Programming	Linux (Python)	Linux (Python) Android (Java)
Real-Time Capability	Yes (Via MCU)	No
Features		WiFi Arduino shield compatible

Table 4.5: Technical Specifications - Other Boards⁹⁰

⁸⁹Cf. COLEY, (2014), pp. 30-34.

⁹⁰Technical specifications based on ibid. & LINKSPRITE, (2014).

4.2 Platform Selection Criteria

Before starting a selection of the different IoT platforms introduced above, it is essential to clarify which criteria of the devices regarding the IoT application is important and has to be considered the most, for balancing within the criteria matrix.

The development board that serves as the gateway for this application needs to be network capable as a basic requirement, otherwise the collected sensor data cannot be forwarded to the selected cloud platform, IBM Bluemix. Also important is the amount of GPIOs and the interfaces that are integrated for connecting all the required sensors to the board. Some development boards, especially SBCs, do not provide an ADC and therefore the use of analog sensors is not possible. This is critical because certain types of sensors are only executed in an analog form due to their functionality. A key criterion is the provided program memory and the working memory (RAM) of the different development boards, where the required size depends on the amount of attached sensors and their complexity regarding additional libraries that each sensor comes up with. Additionally, the full TCP/IP stack that provides the Internet access runs on the controller too, which also demands extra memory when forwarding the sensor data to the cloud by the use of a messaging protocol. In general, memory size is only an issue for MCU based controllers because they use low-level hardware with much less power and storage compared to SBCs that are similar to personal computers with memory in the scale up to gigabytes. While SBCs provide very powerful hardware that is easily capable of processing a vast amount of sensor data and multiple tasks, MCUs have the big advantage that they are real-time capable because they do not use an operation system. MCUs execute the program code using a compiler that translates the C/C++ or assembler language into low-level machine-code that is stored on the MCUs flash memory.

Summarizing, it can be said that characteristics such as the network capability, the amount of GPIOs and interfaces, the hardware performance including flash memory and RAM as well as the real-time capability are the most important selection criteria that have to be taken into account for the balancing within the matrix. Other characteristics like the provided IDE or the documentation are

also to be considered, but they are weighted less than the key criteria mentioned above.

4.3 Value Benefit Analysis

For the implementation of the IoT sensor system on FabLab machines such as the “Ultimaker 2 - Extended” 3D printer, there are certain specifications of the introduced development platforms that are rather more important than others, as already mentioned in section 4.2 above. To take this into account, a selection tool called “Value Benefit Analysis” is utilized to weight the importance of the different selection criteria by using a scoreboard. Within this scoreboard, each selection criteria such as the amount of GPIOs, the price, or the hardware performance is rated by assigning each alternative a specific score between predefined boundaries, e.g. a score from 0 to 5. Depending on whether the alternative (development board) is well suited or not, the score for the specific criteria is either high, low or somewhere in between if it is partly suitable.

In order to distinguish the relevance of each selection criterion individually and to avoid an over-influence of certain characteristics that might be not as important as others, the score of each criteria is “balanced” as a percentage, depending on the importance compared to the other criteria. For this purpose, the criteria are placed in the columns and in the rows of a matrix, where the main diagonal remains empty. The criteria are then compared in pairs and depending on the importance to each other criterion, it receives a rating of 2 (more important than the other one), 1 (equally important), or 0 (less important). After summing up all the ratings for each criterion and dividing it by the total sum over all single sums in the matrix lines, the relevance of each criterion is received. The following Table 4.6 shows the “pairwise comparison” of the criterion pairs, in order to receive the weights for the balanced scoring.⁹¹

⁹¹Cf. SCHULZE, (2016).

	Cost Factor	GPIO Capability	Peripherals	Hardware	Speed	Features	IDE	Coding Skills	Documentation	Community (IoT)	Usability	Expandability	SUM	Weight
Cost Factor		0	1	0	1	1	2	2	2	2	2	2	15	11,4%
GPIO Capability	2		2	1	2	2	2	2	2	2	2	2	21	15,9%
Peripherals	1	0		0	1	1	2	2	2	2	2	2	15	11,4%
Hardware	2	1	2		2	2	2	2	2	2	2	2	21	15,9%
Speed	1	0	1	0		1	2	2	2	2	2	2	15	11,4%
Features	1	0	1	0	1		2	2	2	2	2	2	15	11,4%
IDE	0	0	0	0	0	0		1	1	1	1	1	5	3,8%
Coding Skills	0	0	0	0	0	0	1		1	1	1	1	5	3,8%
Documentation	0	0	0	0	0	0	1	1		1	1	1	5	3,8%
Community (IoT)	0	0	0	0	0	0	1	1	1		1	1	5	3,8%
Usability	0	0	0	0	0	0	1	1	1	1		1	5	3,8%
Expandability	0	0	0	0	0	0	1	1	1	1	1		5	3,8%
Total Score													132	100,0%

Ranking: 2... criterion is more important, 1... importance is equal, 0... criterion is less important

Table 4.6: Pairwise Comparison⁹²

4.4 Results

The following Table 4.7 shows a simplified extract that is based on the full weighted criteria matrix that is to be found in the appendix (A.2 Value Benefit Analysis - IoT Development Boards). It shows the rating and balancing of the selection criteria and how the overall score is being calculated. The result of the comparison shows that the MCU based solution Arduino MEGA 2560 has achieved the highest score, followed by the ESP8266 NodeMCU Lua development kit. Although the Arduino MEGA 2560 requires an additional “Ethernet Shield” for the TCP/IP stack and it does not provide onboard network capability such as, e.g. the Arduino YUN, it is still more suitable because of better hardware requirements and plenty of GPIOs for sensor connectivity. The required shield for the Arduino is already implied within its rating of the cost factor. Since the implementation of the sensor system during this thesis is only at the prototype stage, the cost factor for the components is not that important as it would be for extensive application in multiple FabLabs later on.

As a second alternative next to the Arduino MEGA, the ESP8266 NodeMCU Lua development kit is used as a WiFi based option. This controller follows a different approach, with the possibility to place the board closer to the sensor site or even

⁹²Own illustration.

Decision Matrix		Arduino MEGA	ESP8266 NodeMCU Lua	Raspberry PI 3	Intel Galileo Gen 2	Beaglebone Black Rev. C
Criteria	Weight					
Cost Factor	11,4%	1	5	3	3	3
GPIO Capability	15,9%	5	2	2	2	3
Peripherals	11,4%	3	3	2	3	4
Hardware	15,9%	3	3	4	3	1
Speed	11,4%	3	2	1	3	3
On-Board Communication	11,4%	1	2	3	2	3
IDE	3,8%	2	2	3	2	3
Coding Skills	3,8%	2	2	1	2	1
Documentation	3,8%	3	3	3	2	1
Community (IoT)	3,8%	4	4	3	2	2
Usability	3,8%	3	2	2	2	2
Expandability	3,8%	3	1	2	2	2
Weighted Scores	100,0%	2,83	2,69	2,51	2,50	2,53

Ranking: 0-1... bad fulfillment, 2-3...medium fulfillment, 4-5... good fulfillment

Table 4.7: Value Benefit Analysis (simplified extract)⁹³

inside machines due to its small dimensions. The much lower price compared to an Arduino MEGA including the Ethernet Shield allows to use several ESP8266 development boards for a single machine in order to compensate the lack of GPIOs. Regarding the possibility to use the same IDE (Arduino IDE) with the ESP8266 SoC, the implemented program code to establish the functionality can easily be adapted to it.

Whereas other development platforms that are based on SBCs provide more computing power compared to the introduced MCU boards, they are not really appropriate for sensor data acquisition and processing that benefits more from capabilities that are provided by MCU solutions.

⁹³Based on appendix: A.2 Value Benefit Analysis - IoT Development Boards.

5 Sensor System Implementation

The following chapter deals with the implementation of the sensor system on the Ultimaker 2 - Extended 3D printer that is based on the results of chapter 3, where the basic system concept was introduced and on chapter 4 (Market Research), where the appropriate development platforms were compared and chosen. This chapter is mainly separated into a hardware section, a software section, and a section where the establishment of the IBM Bluemix environment will be discussed in more detail. For sensor data visualization and interaction with the system, a Node-RED application running within IBM Bluemix is implemented in section 5.6 (User Interaction) that can be controlled via a Telegram Bot Application Programming Interface (API) by the simple use of a smartphone or any other device that supports the Telegram messaging service.

5.1 Functional System Overview

Based on the simple concept block diagram that was introduced in section 3.2, a full functional system diagram, including all the essential hardware as well as software components and services was realized. Therefore, the full implementation of the system for the Ultimaker 2 Extended 3D printer can be described by the following Fig. 5.1, which illustrates the full system overview including all required modules. During this chapter, the most important components and processes of the diagram are discussed in detail to understand the coherences and the overall functionality of the bi-directional, cloud-based sensor system.

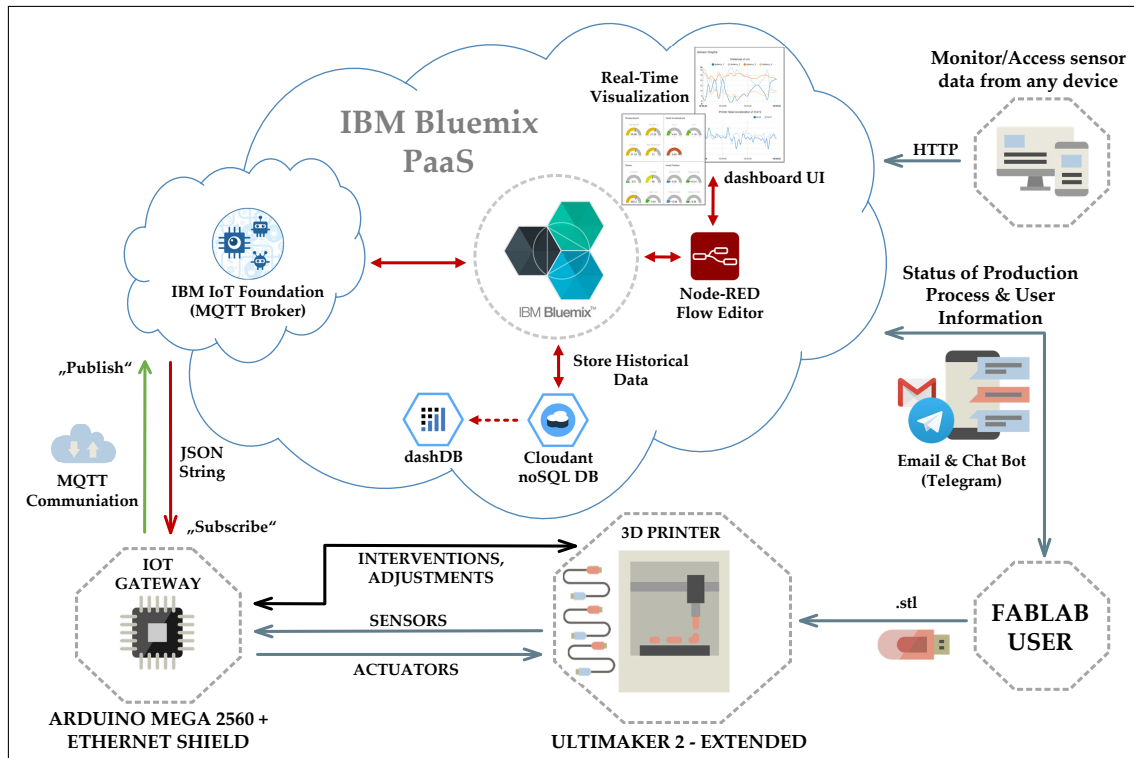


Figure 5.1: Cloud Sensor System - Full Functional Overview⁹⁴

5.2 Hardware

In order to be able to attach the sensors, placed on the 3D printer to the development board (IoT gateway), the pinout of the the hardware is of great importance. It is highly relevant for the controller software, because the peripheral functions need to be assigned to the appropriate pins manually within the code and the relevant GPIOs have to be configured properly, depending whether they are used as inputs or outputs.

5.2.1 Arduino MEGA 2560

Fig. 5.2 shows the pinout of the Arduino MEGA 2560 that acts together with the Ethernet Shield as the gateway for the communication of the Ultimaker 2 - Extended with the PaaS IBM Bluemix. The Board supports a large amount of digital as well as analog I/Os. In the center of the board, between the ATmega 2560 MCU and the reset button, the 6-pin In-Circuit Serial Programming (ICSP)

⁹⁴Own illustration.

header is required to connect Arduino shields by the internal use of the SPI bus, which is also accessible using the pins 50 to 53, according to Fig. 5.2 below.

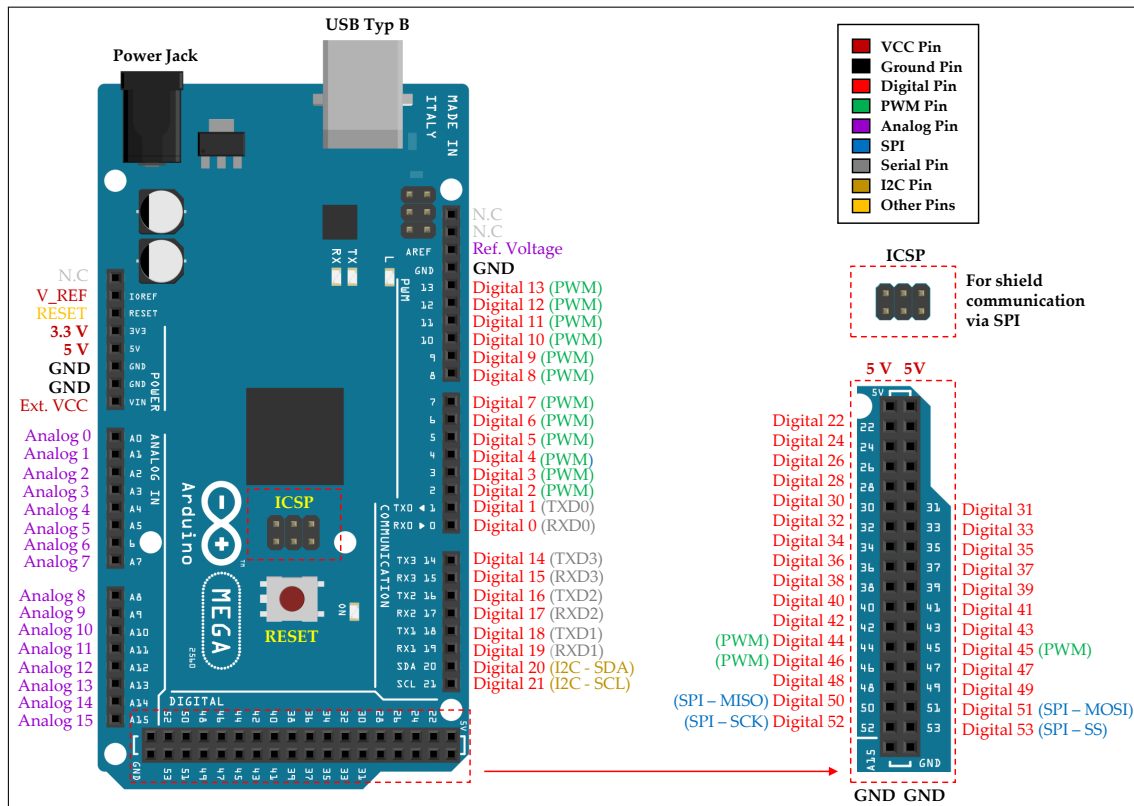


Figure 5.2: Arduino MEGA 2560 - Pin Definition⁹⁵

The Arduino also supports an external reset that can be used to perform remote resets by triggering its input. Therefore, it is possible to reset the sensor system by the use of a defined “reset signal”, that is sent from the cloud platform via the communication protocol to the controller.

The board is capable of handling 3.3 V as well as 5 V signals, respectively sensors and actuators. However, the I2C bus of the 2560 is using a 5 V tolerant logic, thus, a 3.3 V I2C device that is directly connected to the bus, is exposed to a higher voltage level that it is designed for. This does potentially work, although it is recommended to use a “logic level shifter”, that performs a voltage shift of the 5 V I2C signal to 3.3 V, in order to prevent damage or a reduced lifespan of the 3.3 V tolerant components. The process of level shifting, to use 3.3 V and 5 V I2C devices on the same bus is explained in more detail in section 5.2.5.

⁹⁵Own illustration based on ARDUINO.CC, (2016d).

5.2.2 Arduino Ethernet Shield v2

Since the Arduino Mega 2560 does not provide on-board network functionality, the Arduino Ethernet Shield is required to extend the microcontroller board with an Ethernet connection and the TCP/IP stack for network communication. The shield is currently available in version 2 that uses the Wiznet W5500 Ethernet controller, supporting network speeds up to 100 Mbps and providing an internal 32 kB buffer. As an addition, the shield also includes a micro-SD card slot that can be utilized to directly store data. The shield is simply plugged onto the Arduino Board, using the ICSP header for communication via SPI. The required TCP/IP stack for network communication is added by the use of the appropriate Ethernet library that needs to be included to the program code within the Arduino IDE. Note that the pinout of the shield is equal to the Arduino Uno, although it is possible to use it on the larger MEGA 2560 due to the same header alignment. The following Fig. 5.3 shows the pinout of the shield.⁹⁶

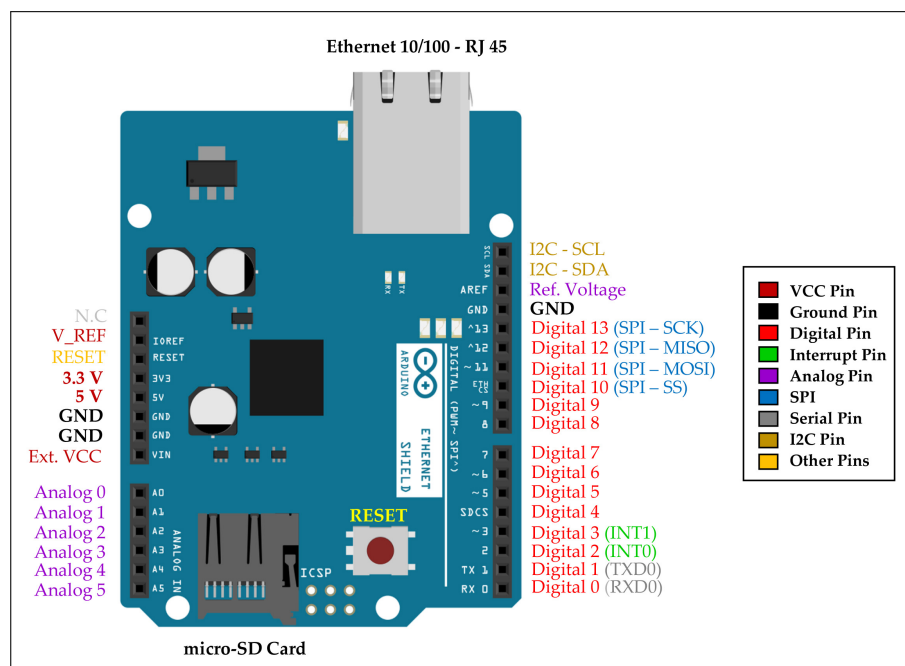


Figure 5.3: Arduino Ethernet Shield - Pin Definition⁹⁷

⁹⁶Cf. ARDUINO.CC, (2016a).

⁹⁷Own illustration.

5.2.3 NodeMCU Development Kit (ESP8266)

The NodeMCU development kit builds upon the ESP8266 SoC and is another alternative for the application as an IoT gateway by using a WiFi interface for data transmission. It is implemented additionally to the Arduino MEGA 2560 which is intended as the gateway for the Ultimaker 2 - Extended 3D printer. Advantages and features of the NodeMCU were already mentioned in section 4.1.2, where the board was introduced with all details.

For the appropriate integration of NodeMCU development boards in the Arduino IDE, it is also important to have knowledge about the actual pin mapping. In case of the NodeMCU, the access for programming the device is based on the number of the I/O index (GPIO) of the board and not on the actual pin numbers that are printed as a silkscreen onto the PCB.

The following Fig. 5.4 shows an overview of the I/O pin labels and the equivalent GPIOs. It points out that, for example, the digital pin D5 corresponds to GPIO14. This has to be taken into account within the software, otherwise the attached peripherals or sensors won't be recognized and they furthermore won't work, if they are declared wrong within the code.

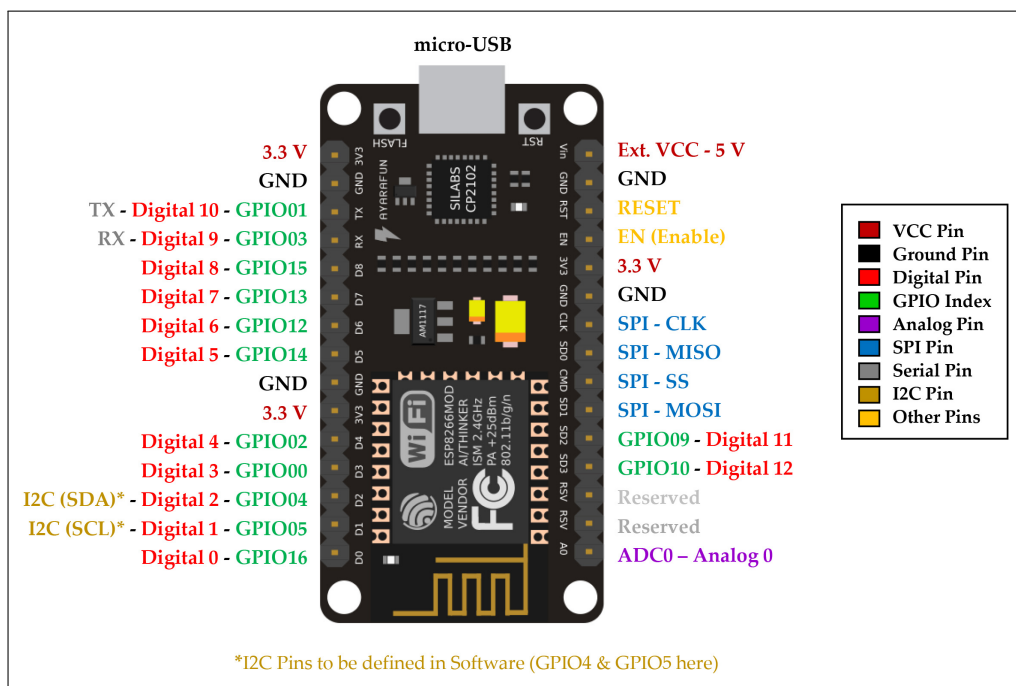


Figure 5.4: NodeMCU - Pin Definition⁹⁸

⁹⁸Own illustration based on NODEMCU, (2015b).

In contrast to the Arduino, the I2C bus is running on a 3.3 V level, whereas the board is also running on 3.3 V by the use of an internal Low Dropout Regulator (LDO) that down-regulates the 5 V input voltage provided by the USB bus. Therefore, a level conversion for the use of 5 V I2C devices is not required necessarily if the 3.3 V high level is sufficient, because it does not exceed the devices ratings. The related I2C pins are not predetermined for the NodeMCU and have to be assigned within the software. According to Fig. 5.4, the Serial Clock Line (SCL) is referred to pin D2 (GPIO4) and the Serial Data Line (SDA) to pin D1 (GPIO5). These I/Os are also used for the definition within software later on to match the pinout.

5.2.4 Sensors

The following section provides a brief overview about the sensors as well as the actuators that are connected to the Arduino MEGA 2560, used for the installation at the Ultimaker 2 Extended 3D printer as well as for the NodeMCU development kit to monitor various environmental data within the FabLab. The consideration for the first implementation of the system is to install different types of sensors for acquiring machine related data such as temperatures inside the machine at various installation points, noises caused by an ongoing production progress or the actual position and acceleration of the printer head, depending on its travel speed. Moreover, a detection of the filament presence is realized to inform the user in case that the material is running low.

ADXL345 - Accelerator

In order to be able to detect the acceleration of the printer head during a print job, a 3-axis accelerator is placed directly onto the head. For practical use of the installation at the machine only the XY-axes are relevant, since the head only travels two-dimensional and the printer table, which is attached to the Z-axis, is moving too slow to detect any considerably acceleration change. Several accelerators are suitable for the application according to Table 5.1. They mainly differ in measurement range, resolution and the communication interface. The accelerator ADXL345⁹⁹ is a well suited option and is chosen because it offers a

⁹⁹SPARKFUN, (2016b).

wide measurement range combined with a relatively low current consumption during the operation.

	Acceleration Sensors				
	ADXL335	ADXL337	ADXL345	LSM303	MMA8452Q
Price	\$14,95	\$9,95	\$17,95	\$14,95	\$9,95
Axis	3	3	3	3	3
Measurement Range	± 3g	± 3g	± 2/4/8/16 g	± 2/4/8/16 g	± 2/4/8 g
Operating Temp. Range	-40 - 85 °C	-40 - 85 °C	-40 - 85 °C	-40 - 85 °C	-40 - 85 °C
Resolution	±1%	±1%	10-13 Bit	12 Bit	12 Bit
Supply Voltage	1.8 - 3.6 V	1.8 - 3.6 V	2 - 3.6 V	3.3 - 5 V	1.6 - 3.6 V
Supply Current	350 µA	300 µA	40 µA	110 µA	6 -165 µA
Communication	analog	analog	SPI / I2C	I2C	I2C

Table 5.1: Acceleration Sensors¹⁰⁰

HC-SR04 - Range Detector

By attaching two HC-SR04¹⁰¹ range detectors at the end stop of the linear units of the Ultimaker 2, it is possible to detect the position of the head within the XY-plane. Since the overall travel length in both directions is known, it is only required to measure the distance to the head from one side each, the other spans are calculated within the Arduino code. The selection of affordable and precise range detectors that provide the required measurement range is rather limited. Consequently, the only suitable option starting at a minimum measurement range of 20 mm and providing a proper resolution is the HC-SR04 range detector, which is an ultrasonic detector based on a run-time measurement of the signal. The specifications are outlined in Table 5.2 on the next page, where also other detectors with inappropriate measurement ranges are opposed.

Temperature/Humidity/Pressure/Altitude

For the detection of physical parameters such as temperatures, humidity or the pressure, various sensor breakouts are available. Table 5.3 shows the applied sensors that are to be installed at the Ultimaker 2, at different installation points. Certain breakouts like the MPL3115A2¹⁰² and the BME280¹⁰³ are combined sensors

¹⁰⁰Technical specifications based on SPARKFUN, (2016a).

¹⁰¹SPARKFUN, (2016g).

¹⁰²SPARKFUN, (2016j).

¹⁰³SPARKFUN, (2016c).

that are capable of measuring multiple parameters, as illustrated in Table 5.3. It is convenient because it reduces the wiring effort for the installation.

	Distance Sensors				
	SRF02	SRF04	HC-SR04	GP2Y0A21YK	GP2Y0A02YK0F
Price	\$13	\$22,00	\$5,00	\$13,95	\$14,95
Type	Ultrasonic	Ultrasonic	Ultrasonic	Infrared	Infrared
Measurement Range	16 - 6000 cm	3 - 3000 cm	2 - 400 cm	10 - 80 cm	15 - 150 cm
Operating Temp. Range	-30 - 70 °C	-30 - 70 °C	-15 - 70 °C	-10 - 60 °C	-10 - 60 °C
Resolution	3 cm	3 cm	0.3 cm	n.a	n.a
Supply Voltage	5 V	5 V	5 V	4.5 - 5.5 V	4.5 - 5.5 V
Supply Current	4 mA	30 - 50 mA	15 mA	30 mA	33 mA
Communication	I2C	I2C	two-wire (digital)	analog	analog

Table 5.2: Distance Sensors¹⁰⁴

	Temperature/Humidity/Pressure/Altitude Sensors				
	MPL3115A2	BME 280	TMP 102	DHT22 (RHT03)	MAX31855K (ThermoCouple)
Price	\$13	\$19,95	\$5,95	\$9,95	\$13,95 (Thermocouple) \$14,95 (MAX31855K)
Type	Temperature Pressure Altitude	Temperature Pressure Humidity	Temperature	Temperature Humidity	High Temp.
Measurement Ranges	-40 - 85 °C (Temp.) 20 - 110 kPa (Baro.)	-40 - 65 °C (Temp.) 30 - 110 kPa (Baro.) 0 - 100% (RH)	-40 - 125 °C	-40 - 80 °C (Temp.) 0 - 100% (RH)	-200 - 700 °C
Operating Temp. Range	-40 - 85 °C	-40 - 85 °C	-40 - 125 °C	-40 - 80 °C	-200 - 700 °C
Accuracy	± 1-3 °C (Temp.) ± 0.5 kPa (Baro.)	± 0.5 °C (Temp.) ± 1 hPA (Baro.) ± 3 % (RH)	± 0.5 °C (-25 - 85 °C)	± 0.5 °C (Temp.) ± 2 % (RH)	± 2°C
Resolution	0.25 - 1.5 Pa (Baro.) 0.0625 - 0.3 m (Alt.)	0.01 °C (Temp.) 0.18 Pa (Baro.) 0.008 % (RH)	0.0625 °C	± 0.1 °C (Temp.) ± 0.1 % (RH)	0.25 °C / 14 Bit
Supply Voltage	1.6 - 3.6 V	3.3 V	1.4 - 3.6V	3.3 - 6 V	3 - 3.6 V
Supply Current	2 mA	300 - 700 µA	1- 10 µA	1 - 1.5 mA	0.9 - 1.5 mA
Communication	I2C	SPI/I2C	two-wire (I2C)	single-wire (digital)	SPI

Table 5.3: Temperature/Humidity/Pressure/Altitude Sensors¹⁰⁵

¹⁰⁴Technical specifications based on SPARKFUN, (2016k).

¹⁰⁵Technical specifications based on SPARKFUN, (2016m) & SPARKFUN, (2016e).

Type-K Thermocouple

An important parameter is the actual temperature of the nozzle, mounted within the extrusion head. The nozzle is heated from 180 °C up to 260 °C¹⁰⁶ and required to melt the filament used for production. Depending on the filament material, an inappropriate nozzle temperature can easily lead to a nozzle clog that interrupts and/or corrupts the ongoing production with the possibility to damage the extruder in the worst case. Because of the high temperature it is not possible to attach an ordinary temperature sensor as introduced above, due to their limited measurement ranges. Therefore, a Thermocouple Type-K¹⁰⁷ wire together with the MAX31855K¹⁰⁸ digitizer for data conversion is utilized, which supports temperature readings up to 700 °C at a resolution of ± 2 °C, as it can be seen out of the previous Table 5.3. The communication between the digitizer breakout and the MCU board is realized by using the SPI bus.

TSL2561 - Luminosity Sensor

The TSL2561 breakout includes a sophisticated, integrated light sensor that allows to measure the visible light intensity in the range of 0.1 to 40 klx. Thus, the TSL2561 is able to detect changes of the ambient lightning, e.g. someone enters the room and turns on the light, or the lightning inside the machine caused by the installed LED stripes. The sensor requires a 3.3 V supply voltage and is capable of I2C communication.¹⁰⁹

Electret Microphone

To detect a variety of sounds inside the Ultimaker 2 caused by an ongoing production progress or to monitor surrounding sounds within the FabLab, electret microphone breakouts are utilized to detect various noise influences. The microphone includes a 60x pre-amplifier for the ADC of the Arduino or the NodeMCU so that even very slight noises that are not within close proximity, are detectable.

¹⁰⁶<https://ultimaker.com/en/products/ultimaker-2-plus/specifications>, (retrieved 06/02/2017).

¹⁰⁷SPARKFUN, (2016n).

¹⁰⁸SPARKFUN, (2016i).

¹⁰⁹SPARKFUN, (2016p).

Depending on the loudness, the output delivers a proportional voltage signal that represents a certain noise level.¹¹⁰

Air Quality Sensor

Air quality sensors are suitable for sensing different gas concentrations within the air, e.g. smoke, methane, carbon monoxide or other flammable gases. Therefore, different types of “MQ” sensors are available, whereby all providing different gas sensing capabilities. The breakout uses a built-in heater together with an electro-chemical sensor and requires a rather complex calibration using the “load-resistor” for a precise detection of gas concentrations. Nevertheless, the sensor is also applicable without exact calibration in form of a tendency, since the analog output value will increase in case the gas concentration is rising or vice versa. Additionally, the heater requires a “burn-in” time of 12-24 hours to make the analog sensor readings more consistent. For the installation, the MQ-135 and MQ-9 air quality sensors are tested in order to detect possible emissions inside the Ultimaker 2 caused by the heated Polylactic Acid (PLA) and Acrylonitrile Butadiene Styrene (ABS) thermoplastics, used as the filament material.¹¹¹

Photo Interrupter

A photo interrupter is composed of an infrared emitter (sender) at one side, and of an infrared detector at the other side of the gate. The detector simply identifies if there is a present object between the emitter and the detector of the uprights breaking the infrared beam. Depending on whether the gate is clear or blocked by an object, the output signal is either HIGH or LOW. By utilizing such a photo interrupter at the Ultimaker 2, it is possible to detect the filament presence at the machine by leading the filament through the photo interrupter’s detection area before the filament enters the material feeder. In fact that the machine is not capable of detecting the run out of filament during a production, it will continue the programmed print job even without material. As a result, the production is incomplete and the machine still keeps on running without any purpose. Therefore, the photo interrupter detects empty filament as soon as the interrupter gate is clear again, as a consequence that no filament is pulled into the feeder anymore. A

¹¹⁰SPARKFUN, (2016d).

¹¹¹ARDUINO.CC, (2016e).

detailed overview of the intended installation at the back of the machine can be found in section 5.2.7.¹¹²

SSD1306 OLED Display

The SSD1306 is an Organic Light Emitting Diode (OLED) display, serving as an actuator to output measured sensor values. It consists of 128x64 pixels at a screen size of 0.96". Although the display is rather small, due to its good resolution and integrated lightning of the OLED technology compared to LCD screens, it is possible to visualize multiple sensor values at a time. Communication with the controller is done via the I2C interface which is already 5 V tolerant and does not require additional level shifting. See section 5.7.2 for more information, where the display is installed at the machine to locally visualize sensor data.¹¹³

¹¹²SPARKFUN, (2016f).

¹¹³ADAFRUIT, (2016).

5.2.5 I2C Level Shifting

A lot of the sensors that have been introduced in the previous section make use of the I2C bus. Due to the fact that not all sensors and actuators are available with the same logic levels, 3.3 V devices and 5 V devices have to be utilized together using the same 5 V I2C bus of the Arduino MEGA 2560. This might be critical, as already mentioned in previous sections, because a 3.3 V device is principally not designed to work on a 5 V voltage level. Although it does work for some cases, the lifespan of the devices can get reduced or they might even get damaged. To prevent this issue, a so called “Logic Level Shifter” is applied which converts the 5 V I2C logic level to a lower 3.3 V level. The following Fig. 5.5 shows the component, including some colored highlights for further explanation.

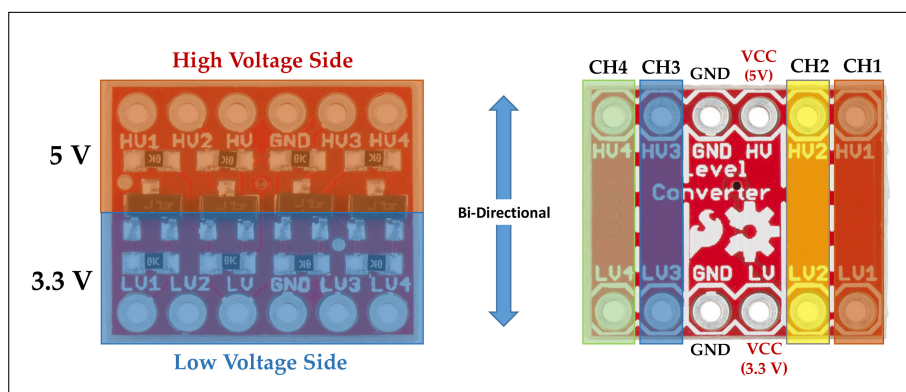


Figure 5.5: 4-Channel I2C Level Shifter¹¹⁴

Basically, it is a 4-channel converter which means that it is possible to transform four different logic levels independently at a time. The high-level side, in this case 5 V on the top is marked red, whereas the low-level side (3.3 V) on the bottom is marked blue. The right illustration, presenting the opposite side of the level converter, additionally shows the four channels plus the pins that are required for ground and voltage supply on each side. Note that this converter works bi-directional, thus, it ensures that the two-way I2C communication is able to work properly.¹¹⁵

For the use of the level shifter, the I2C data signals SDA and the clock signal SCL need to be connected to the high side (HV1, HV2, HV3 or HV4). As a consequence,

¹¹⁴Modified illustrations from SPARKFUN, (2016h).

¹¹⁵Cf. *ibid.*

the I2C lines of the 3.3 V sensors and devices need to be connected to the corresponding low-level pins (LV1, LV2, LV3 and LV4), where the voltage level of the bus is then reduced to 3.3 V. To validate the level conversion and the functionality of the bus on 3.3 V, the measurement of a random Arduino I2C scanner signal using a Tektronix TDS 2024C Digital Oscilloscope was performed. The result is illustrated in Fig. 5.6 below, where the clock line SCL is measured using channel 2 (light blue) and the data line SDA by the use of channel 1 (yellow). Out of the adjusted voltage divisions of both channels (CH1 - 5 V, CH2 - 2 V) for the screen follows that the voltage level is on 3.3 V, at a measured standard clock speed of 100 kHz.

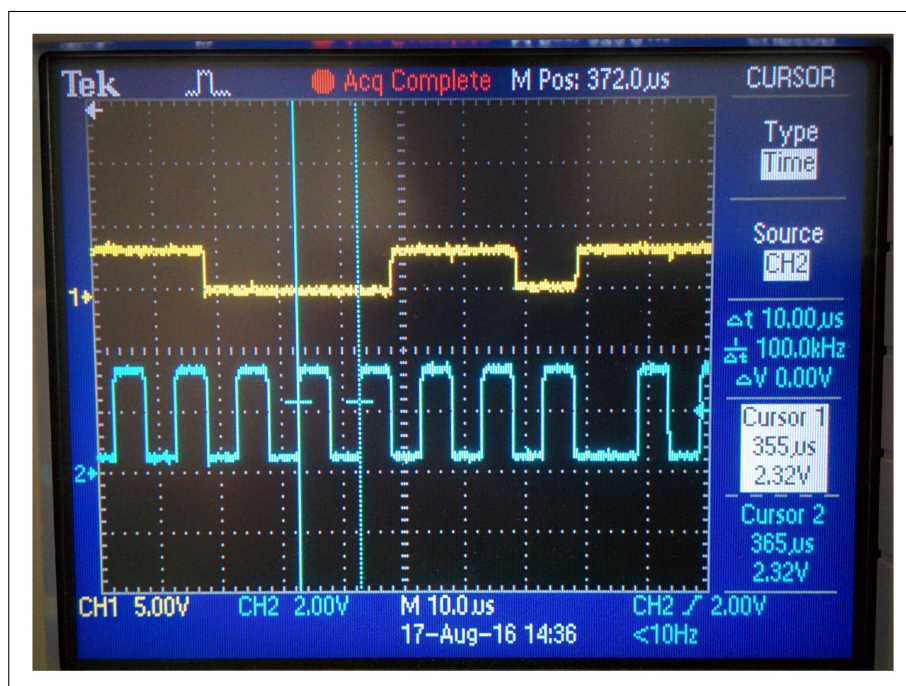


Figure 5.6: I2C - 3.3 V Communication Test¹¹⁶

5.2.6 Wiring Diagram & Stripboard

Due to a great amount of wires and cables for all the sensor connections at the Ultimaker 2 Extended 3D printer, a wiring diagram was prepared as an auxiliary tool for the installment that is performed at the machine and the Arduino later on. It is helpful to keep track of all the traces and it is useful for debugging in case of errors. The tool which has been utilized for creating the schematic is called

¹¹⁶Own illustration, Tektronix TDS 2024C measurement.

Fritzing. It is an open source project including a great library of development boards as well as sensors, actuators and other components of various suppliers. In comparison to more professional schematic editors such as Eagle, it is very easy to handle in context of stripboards and it provides an authentic look of the schematic by using all virtual models of the components.¹¹⁷

Fig. 5.7 below shows the schematic of all the intended sensors and components, connected to the Arduino MEGA 2560 and the Ethernet Shield for the first installment at the machine. The figure also shows the full connection of the I2C level shifter with its attached 3.3 V I2C devices that was discussed in the previous section.

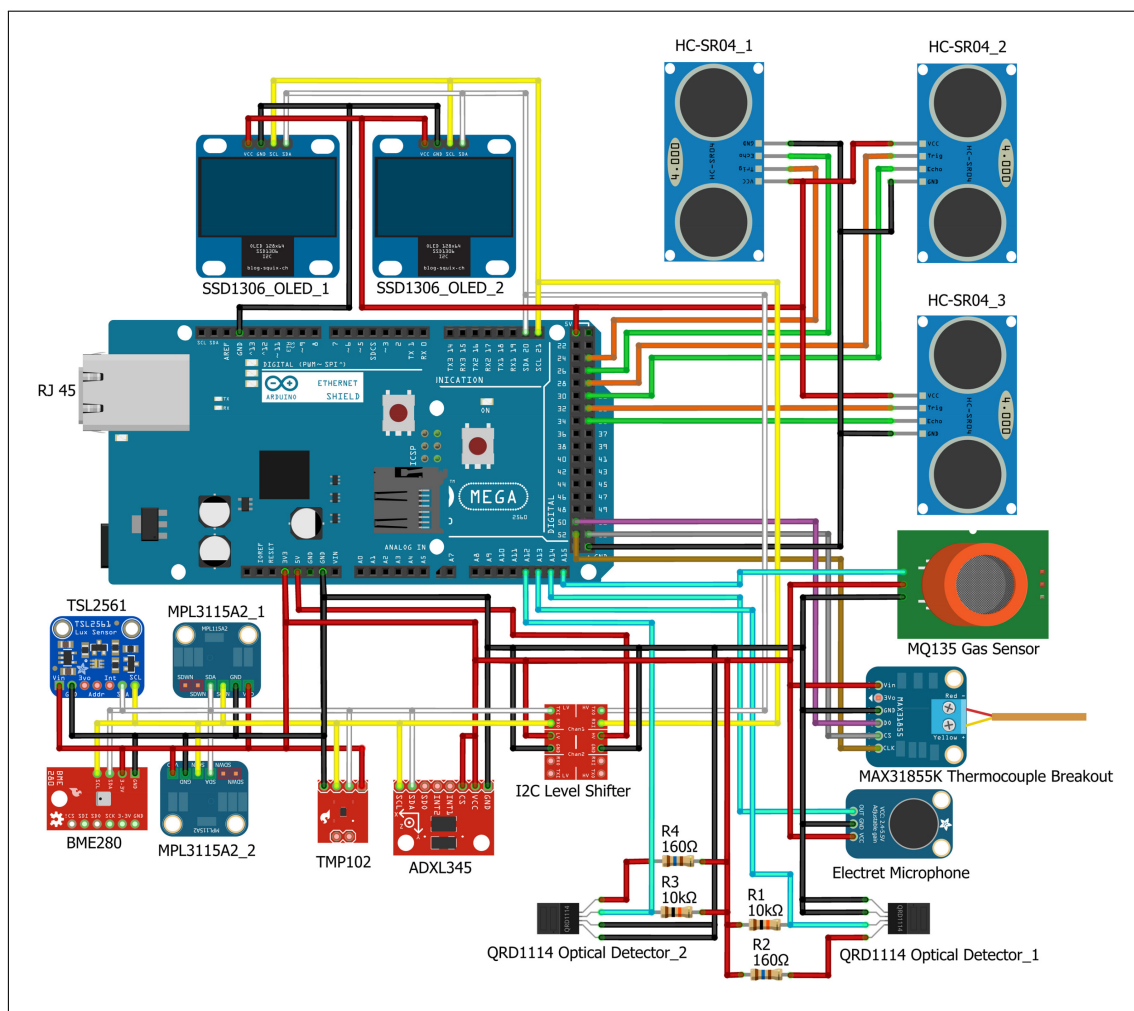


Figure 5.7: Sensor Wiring Diagram¹¹⁸

Because each sensor, depending on the interface comes up with several electrical

¹¹⁷<http://fritzing.org/home/>, (retrieved 06/02/2017).

¹¹⁸Own illustration.

connections such as supply, ground, signal and clock lines, the total amount of wires that need to be attached to the Arduino MEGA 2560 and the Ethernet Shield is quite a lot. Wiring all the cables and lines of each sensor directly to the controller would end up in a total mess, moreover, it is not even possible because of insufficient supply and ground output pins of the controller. Therefore, a stripboard is deployed to arrange the sensor connectors suitable on it and to route all the required electrical connections at the bottom layer of the board. The following Fig. 5.8 shows a sketch of the stripboard together with the Arduino and the Ethernet Shield on top. For sensor connection, common pin headers and cables are used to be flexible regarding the connectivity and attachments.

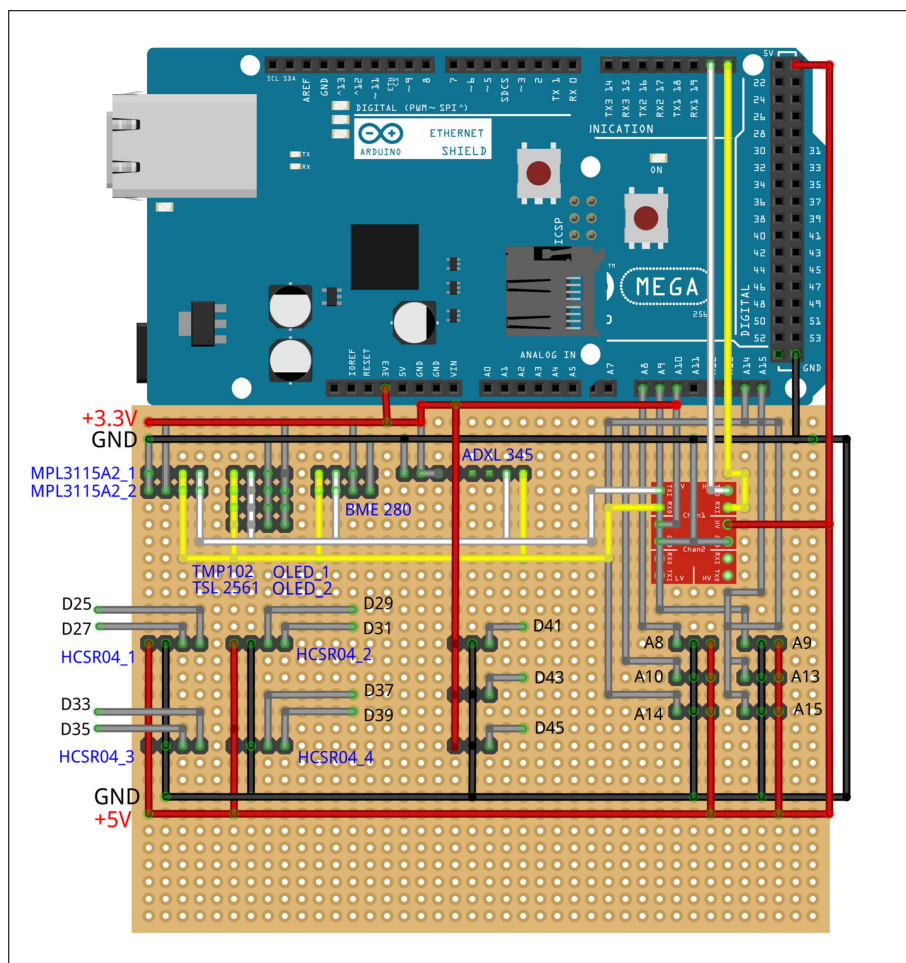


Figure 5.8: Stripboard - Sensor Connections¹¹⁹

After the installation of the sensor and component connectors at the stripboard and the Arduino MEGA 2560, the whole unit shall be placed under the machine by the use of an appropriate housing that still provides access from the outside for

¹¹⁹Own illustration.

the external power supply and the RJ45 network connection of the Arduino.

5.2.7 Sensor Placement

The intended sensors for the installation at the Ultimaker 2, which were introduced in section 5.2.4, are placed inside the machine at different installation points. For some sensors such as the range detectors at the printer head or the photo interrupter for detecting an empty filament, additional mounting supports are required since it is not possible to position them in a suitable way without additional parts by the use of screws or double-sided adhesive tape. The mountings are designed within the CAD tool Solid Work¹²⁰ and manufactured using the Ultimaker 2 3D printer.

The following Fig. 5.9 illustrates the mounting mechanism for the HC-SR04 ultrasonic range detectors and the additional reflector that is placed onto the printer head for better reflection of the ultrasonic signal.

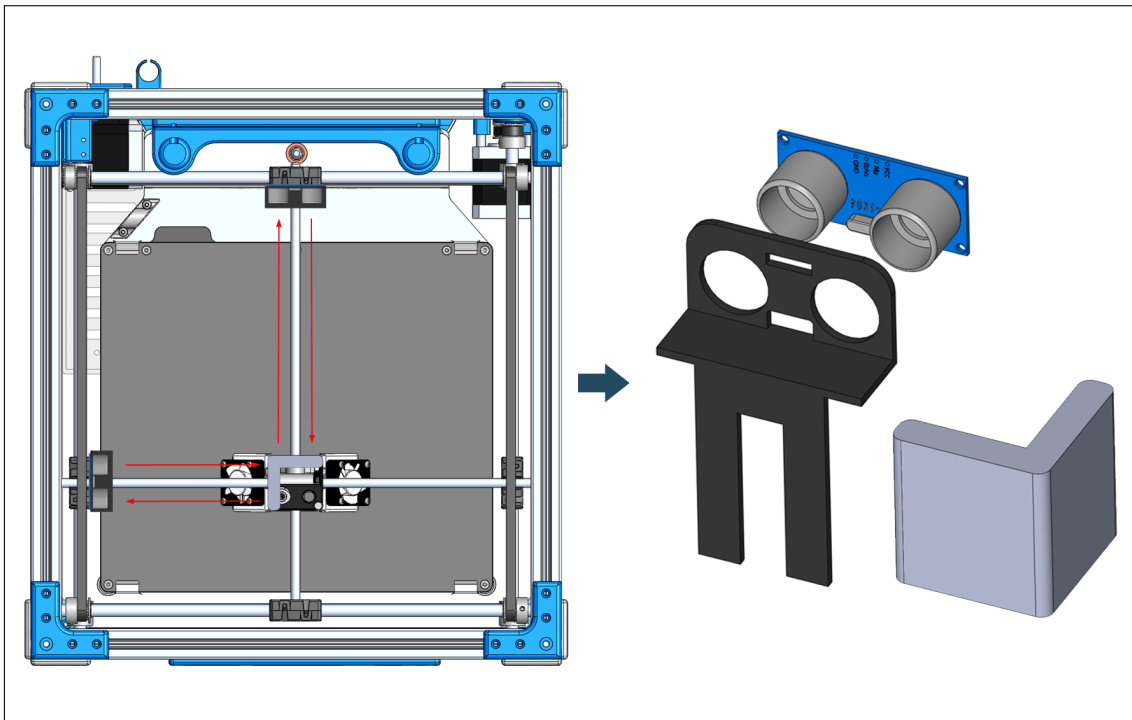


Figure 5.9: Sensor Mounting - Installed Mounting (left); HC-SR04 Mounting (middle); Reflector (right)¹²¹

¹²⁰<http://www.solidworks.com/>, (retrieved 06/02/2017).

¹²¹Own illustration.

The completed assembly of the mountings including the sensors is to be found in section 5.5.2

Another mounting support is required for the photo interrupter that is intended to detect a filament run-out by continuous monitoring of the material between the photo emitter and the receiver of the interrupter. Since the filament needs to run perpendicular to the interrupter's gap before it leads into the filament feeder of the machine, a mounting mechanism according to the following Fig. 5.10 was constructed and produced with the Ultimaker 2.

Once the gap of the interrupter is clear, meaning no filament is supplied any longer from the material spool, the output signal of the sensor changes from 5 V to 0 V or opposite, depending on the signal logic that is implemented within the software. Thereby it is possible to inform the user beforehand once the material is running low.

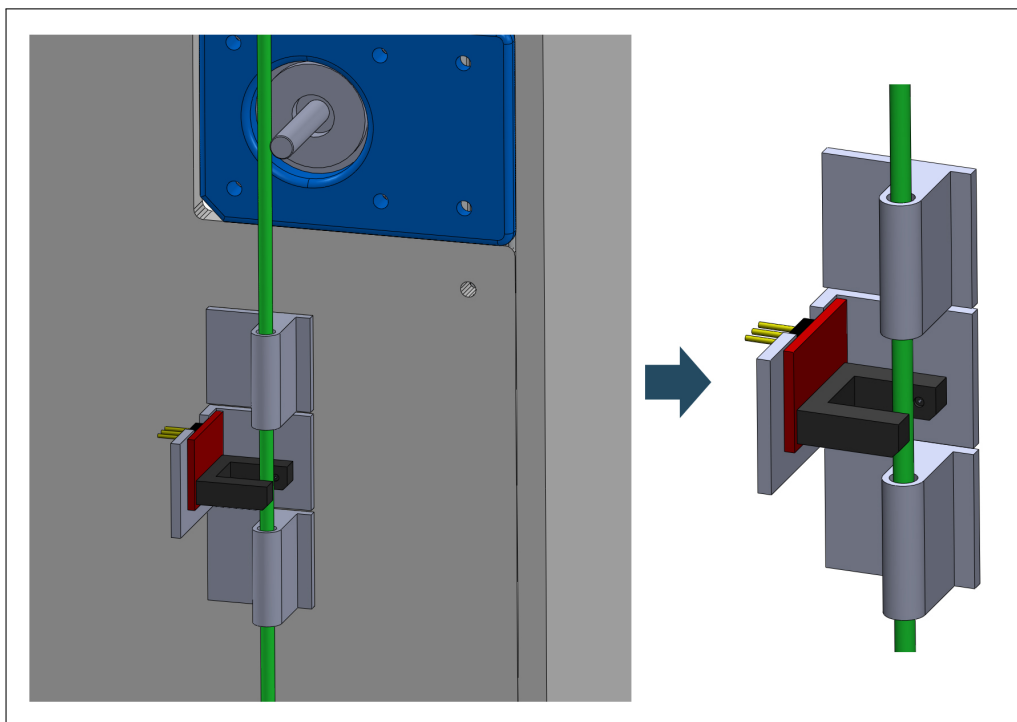


Figure 5.10: Photo Interrupter - Installed Mounting (left); Detailed View (right)¹²²

¹²²Own illustration.

5.3 Software

The controller software that needs to be implemented for the IoT gateway is an essential part of the system implementation process, since it is responsible for controlling, managing and forwarding the received data that is gathered by the sensors. Within the software all the attached peripherals need to be configured properly according to their connected GPIOs. Furthermore, the application of the TCP/IP stack, which is required to establish a connection with IBM Bluemix via the Internet, is also part of the software.

The following sections discuss how the software for the controller and the messaging protocol that is required to establish a bi-directional communication between the IoT gateway and the PaaS IBM Bluemix is implemented and how to set up the development environment properly.

5.3.1 Arduino IDE

Arduino provides its own development environment to connect Arduino boards, but also third-party hardware such as the NodeMCU development kit for programming the connected devices and to communicate with them. The IDE contains a text editor for writing the program code in C/C++ language, a console to output compiling information and errors, a “avr-gcc” compiler to translate the C code into machine instructions and a serial monitor to visualize serial inputs as well as outputs. The whole IDE is kept rather simple compared to other development tools, hence it does not provide additional features such as a debugger for troubleshooting. By the use of the Arduino IDE, program files that are written are called “sketches” and are saved with an .ino file extension. Within the IDE it is also possible to split the code into multiple files to keep a better overview.¹²³

Fig. 5.11 shows an overview of the Arduino IDE with the opened program code for the Arduino MEGA 2560 and the Ethernet Shield that are together used as the sensor gateway for the Ultimaker 2 Extended 3D printer. As it can be seen out of the figure several tabs are created to manage certain program parts such as the different sensors separately.

¹²³ ARDUINO.CC, (2016c).

```

1 /*-----*/
2 /*Arduino MEGA 2560_1 used for Ultimaker 2 - Extended @ FabLab Graz */
3 /*Leitner A. - 2016 */
4 /*-----*/
5 #include <SPI.h>
6 #include <Wire.h>
7 #include <Ethernet2.h> // Ethernet Shield v2
8 #include <PubSubClient.h>
9 #include <IPStack.h>
10 #include <Countdown.h>
11 #include <MQTTClient.h>
12 #include <Ultrasonic.h>
13 #include <TCN75A.h>
14 #include <Adafruit_Sensor.h>
15 #include <Adafruit_TSL2561_U.h>
16 #include "SparkFunMPL3115A2.h"
17 #include "SparkFunBME280.h"
18 #include <SparkFunMAX31855k.h>
19 #include <Adafruit_ADXL345_U.h>
20 #include <Adafruit_GFX.h>
21 #include <Adafruit_SSD1306.h>
22 #include "MQ135.h"
23
24 //-----Ethernet Shield v2-----
25 byte mac[] = {0x90, 0xA2, 0xDA, 0x10, 0xA2, 0x79}; // Only needed for Ethernet Shield 2 --> This is the MAC on the bottom
26 char macstr[] = "90a2da10a279";
27
28 //----- Bluemix & Server Credentials -----
29 #define ORG "wwettl"
30 #define DEVICE_TYPE "ArduinoMega2560"
31 #define DEVICE_ID "ArduinoMega2560_1"
32 #define TOKEN "asdf1234"
33
34 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
35 //char topic[] = "iot-2/evt/status/fmt/json";
36 char topic[] = "iot-2/evt/sensor/fmt/json";
37 char topic_subscribe[] = "iot-2/cmd/+ /fmt/json";

```

Figure 5.11: Arduino IDE - Overview¹²⁴

In order to upload the code to the attached device, the appropriate controller needs to be selected out of the board manager. The Arduino IDE also supports third-party hardware such as the NodeMCU development board by applying the required “Board Manager URL” in the preferences so that the IDE is able to find the desired controller within its boards manager.

The following Fig. 5.12 shows the preferences menu where the URL for for additional ESP8266 boards (marked red) has to be added to provide the installation of the ESP8266 Library. Within the preferences it is also possible to change the sketchbook location where all the program files are being stored. In context of this project, the sketchbook was assigned to a cloud based location to synchronize the progress when writing and testing code on different computers.

¹²⁴Own illustration.

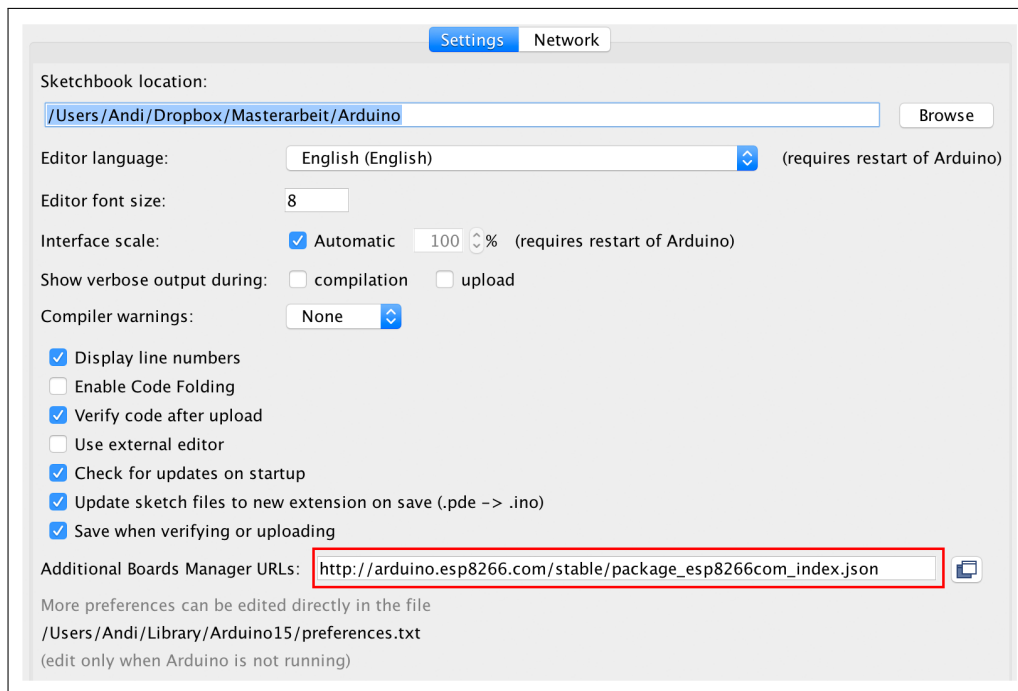


Figure 5.12: Arduino IDE - Preferences¹²⁵

As shown in Fig. 5.13 below, the ESP8266 board library that includes several development boards and modules is successfully listed in the boards manager after adding the URL in the preferences above.

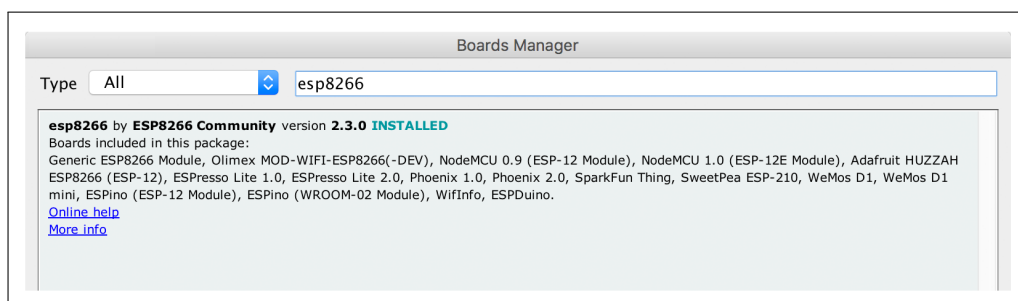


Figure 5.13: Board Manager - ESP8266 Board support¹²⁶

Once the library is installed, the additional ESP8266 boards can be found in the “Tools” menu (ESP8266 Modules), as illustrated in Fig. 5.14. Consequently, the NodeMCU Development Board can be programmed just like any other regular Arduino board using the Arduino IDE.

¹²⁵Own illustration.

¹²⁶Own illustration.

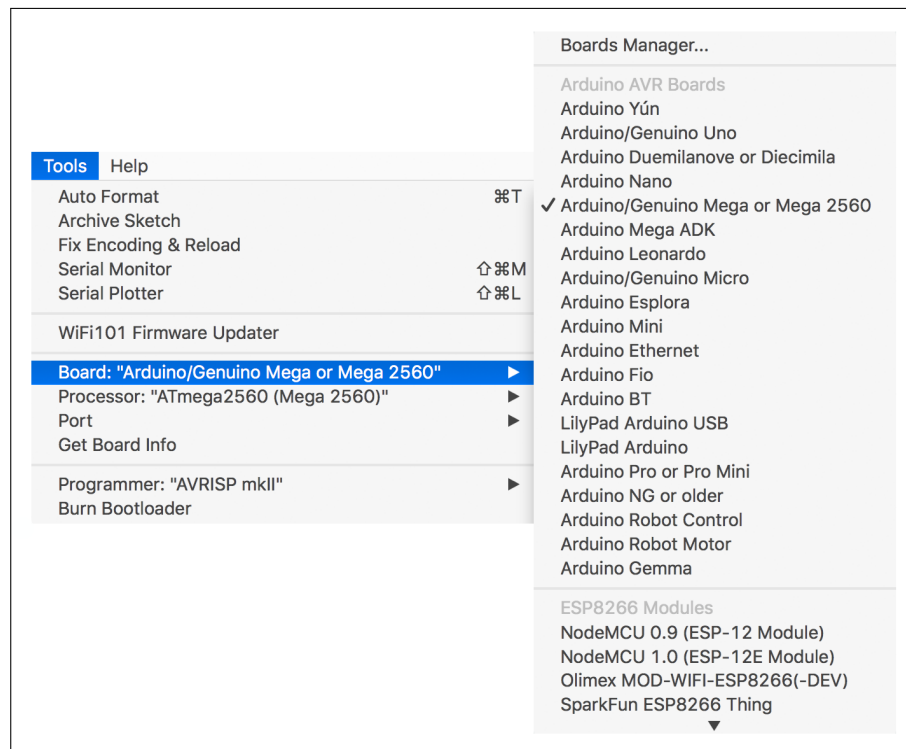


Figure 5.14: Arduino IDE - Board Selection¹²⁷

5.3.2 Network Communication

Depending on whether the Arduino MEGA 2560 including the Ethernet Shield or the NodeMCU development kit is applied, the network communication is either performed wired (Ethernet), or wireless (WiFi). However, the utilized communication form does not affect the actual program code, except the configuration of the TCP/IP stack since different libraries are required for the devices.

For the wired solution using the Ethernet Shield to integrate the TCP/IP stack, the library needs to know the MAC address of the shield in order to receive an IP address by the DHCP server of the network router. The MAC is usually to be found on a sticker that is attached on the bottom of the shield.

Using the NodeMCU on the other hand, the registration within a wireless network is done via the Service Set Identifier (SSID) and the appropriate password, whereby the device obtains the IP from the DHCP server. In case no DHCP server is available, then the IPs have to be assigned manually for both solutions. The differences of the configuration within the program code are opposed in Fig. 5.15.

¹²⁷Own illustration.

```

18 #include <ESP8266WiFi.h>                               #include <Ethernet2.h>
19
20 //----- Bluemix & Server Credentials //----- Bluemix & Server Credentials -----
21 const char* ssid = "SSID";                               byte mac[] = {0x90, 0xA2, 0xDA, 0x10, 0xA2, 0x79}; //
22 const char* password = "ssid_password";                 char macstr[] = "90a2da10a279";
23

```

Figure 5.15: Network Configuration - Arduino (right) & NodeMCU (left)¹²⁸

5.3.3 MQTT Messaging Protocol

MQTT is a very simple and lightweight messaging protocol on top of the TCP/IP protocol that is based on a bi-directional publish/subscribe architecture, supporting up to thousands of remote clients. It is especially well-suited for M2M or IoT applications, because it is optimized for sensors, actuators and remote devices and it minimizes the network bandwidth at a high level of reliability by choosing between three Quality of Service (QoS) levels. Therefore, the MQTT protocol is ideal for the use in constrained environments where the bandwidth is low, a high latency is occurring and remote devices with limited processing capabilities and memory (microcontrollers) are participating.¹²⁹

Publish/Subscribe Pattern

As mentioned above, the communication by the use of MQTT is based on a “Publish” and “Subscribe” concept, in contrast to the traditional “Client - Server” connection that uses a direct point-to-point communication. The device or gateway that wants to provide or share information is called the “Publisher”, whereas the application or device that consumes the shared information is known as the “Subscriber”. The big advantage of this approach is that the publisher and the subscriber do not need to know anything from each other since there is a third component, the MQTT broker that is located in between for distributing all the messages.¹³⁰

When a publisher sends a message, e.g. gathered sensor data for this application, the message subject needs to be classified by the use of a “Topic”. A topic is required to define the content of a message, so the broker is able to forward the message according to those subscribers that are “Subscribed” to the same topic.

¹²⁸Own illustration.

¹²⁹Cf. LAMPKIN, (2012), pp. 4-5.

¹³⁰Cf. *ibid.*, p. 26.

Therefore, only those subscribers that “Listen” to the same topic will receive the information. While a point-to-point communication, such as HTTP, knows the destination of a message packet due to a specific destination address, the distribution of the data utilizing MQTT is only based on the “Topic” name.¹³¹

Fig. 5.16 shows an example of the publish/subscribe procedure, based on the sensor connection with the Arduino MEGA 2560 and the IBM Bluemix Internet of Things Foundation that acts as the MQTT broker. Additionally, it is also possible to use external, open source MQTT brokers such as “Mosquitto”¹³². However, since the broker is already integrated into IBM Bluemix there is no need for the use of an external one.

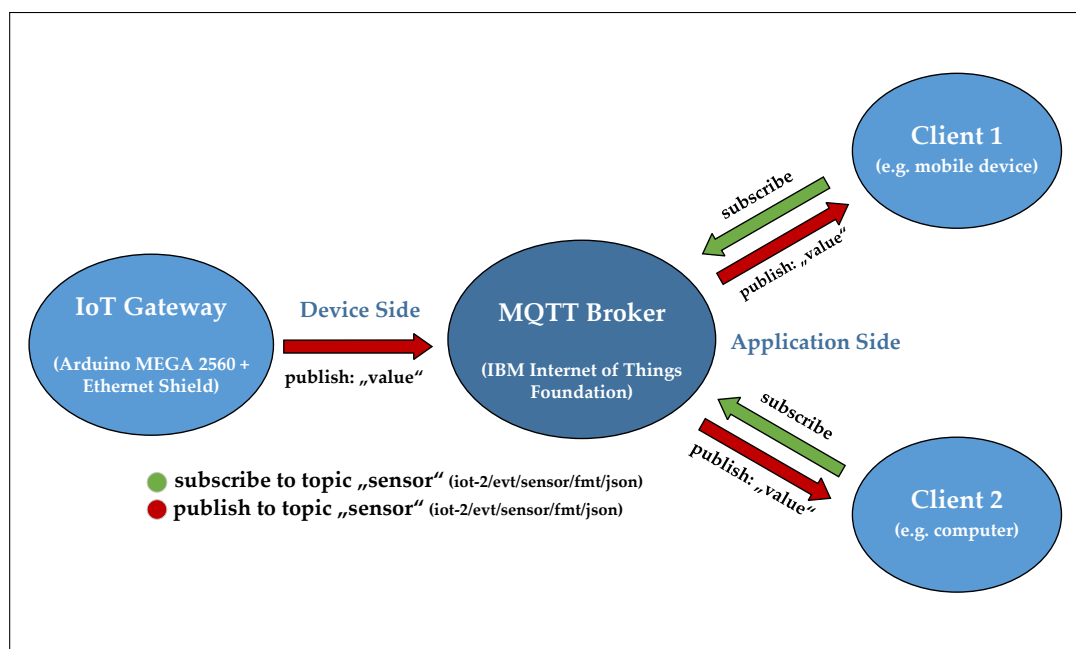


Figure 5.16: MQTT - Publish/Subscribe¹³³

The following Fig. 5.17 shows the configuration of the MQTT topics, the access point for the broker and the IBM Bluemix credentials of the programmed Arduino 2560 MEGA microcontroller code in order to establish the connection with the “Internet of Things Foundation” that is discussed in section 5.4.1.

There are two topics available, one for “Publishing” the gathered sensor data to the broker that will distribute the data within IBM Bluemix and its applications

¹³¹Cf. LAMPKIN, (2012), p. 27.

¹³²<https://mosquitto.org/>, (retrieved 06/02/2017).

¹³³Own illustration, based on Cf. LAMPKIN, (2012), p. 26.

and the other one for “Subscribing” the gateway device (Arduino MEGA 2560) to a Node-RED¹³⁴ application, where users can send specific commands via the Telegram Messenger¹³⁵ back to Node-RED and the gateway to furthermore interact with it. The possibilities of the user interaction with the system will be discussed in section 5.6 in more detail.

```
28 //----- Bluemix & Server Credentials -----
29 #define ORG "wwett1"
30 #define DEVICE_TYPE "ArduinoMega2560"
31 #define DEVICE_ID "ArduinoMega2560_1"
32 #define TOKEN "asdf1234"
33
34 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
35 char topic_publish[] = "iot-2/evt/sensor/fmt/json";
36 char topic_subscribe[] = "iot-2/cmd/+ /fmt/json";
37 char authMethod[] = "use-token-auth";
38 char token[] = TOKEN;
39 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
40
```

Figure 5.17: MQTT - Arduino Code Configuration¹³⁶

5.3.4 Message Payload Format - JSON

IBM Bluemix requires a special message payload format, containing the data that is published to, or subscribed from the MQTT broker. The utilized standard format for the transmission is JavaScript Object Notation (JSON), which is a data-interchange format to transmit data objects with the advantage that it can be read and used by any program language.¹³⁷

It supports basic data types such as numbers, strings and arrays. The message payload requests a notation according to the example illustrated in Fig. 5.18, where two sensor values are transmitted using the JSON string. Each object contains a set of names or value strings written with double quotes and is delimited with curly braces. The additional backslashes in the string are required to escape the quotation marks within a text string, which is needed for proper JSON formatting.

¹³⁴<https://nodered.org/>, (retrieved 06/02/2017).

¹³⁵<https://telegram.org/>, (retrieved 06/02/2017).

¹³⁶Own illustration.

¹³⁷Cf. VASSEUR; DUNKELS, (2010), pp. 94-95.

Note that the IBM Bluemix MQTT broker also requires a top-level property “d”, at the very beginning of the payload string, otherwise the messages won’t be recognized within the Internet of Things Foundation and no data can be transmitted. The length of the payload is also limited by Bluemix to a total size of 131 072 Bytes, where all message payloads that exceed this size will not be accepted by the broker. As a result, the connection to the gateway is disconnected and has to be established again with reduced message size.¹³⁸

```
4 String payload = "{\"d\":{\"myName\":\"ArduinoMega2560_1\"},";
5   payload += "\"T2_BME280\"";
6   payload += "\"";
7   payload += BME280_TempC; // BME280 Temperature Value
8   payload += "\",\"";
9   payload += "\"Pressure\"";
10  payload += "\"";
11  payload += pressure_1; // MPL3115A2 Pressure Value
12  payload += "\"";
13  payload += "}}";
```

Figure 5.18: JSON Message Format¹³⁹

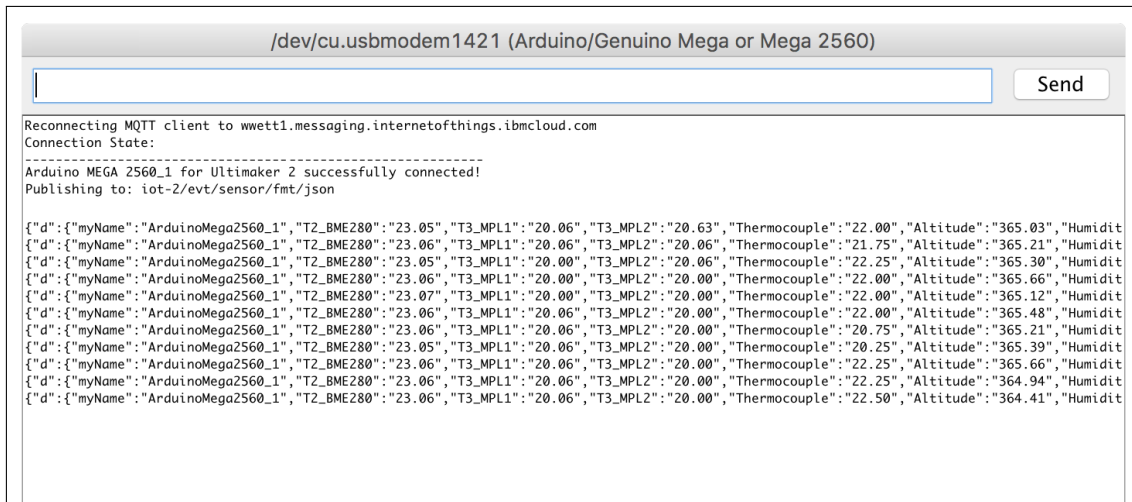
5.3.5 Establishing a Connection

As soon as the program code is successfully uploaded to the controller and the IoT gateway is connected to an Internet capable network either by Ethernet with the Arduino and the shield, or by WiFi when using the NodeMCU development kit, the program routine tries to connect to the MQTT broker and furthermore starts to publish the gathered sensor data. The following Fig. 5.19 shows the serial monitor of the Arduino IDE where the connection process and some published JSON strings, containing the sensor values, are illustrated.

As already mentioned in previous sections, it is also possible to use an external reset pin of the Arduino and the NodeMCU to soft-reset the gateway hardware by the use of a MQTT subscribe. This is achieved with Node-RED where a user specific command, e.g. the string “reset”, is sent back to the controller that is subscribed to the topic in Node-RED. The received command then triggers a digital output pin which is connected to the reset pin via a resistor.

¹³⁸Cf. IBM, (2016).

¹³⁹Own illustration.



```

/dev/cu.usbmodem1421 (Arduino/Genuino Mega or Mega 2560)
Send

Reconnecting MQTT client to wuett1.messaging.internetofthings.ibmcloud.com
Connection State:
-----
Arduino MEGA 2560_1 for Ultimaker 2 successfully connected!
Publishing to: iot-2/evt/sensor/fmt/json

{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.05","T3_MPL1":"20.06","T3_MPL2":"20.63","Thermocouple":"22.00","Altitude":"365.03","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.06","T3_MPL2":"20.06","Thermocouple":"21.75","Altitude":"365.21","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.05","T3_MPL1":"20.00","T3_MPL2":"20.06","Thermocouple":"22.25","Altitude":"365.30","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.00","T3_MPL2":"20.00","Thermocouple":"22.00","Altitude":"365.66","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.07","T3_MPL1":"20.00","T3_MPL2":"20.00","Thermocouple":"22.00","Altitude":"365.12","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.06","T3_MPL2":"20.00","Thermocouple":"22.00","Altitude":"365.48","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.06","T3_MPL2":"20.00","Thermocouple":"20.75","Altitude":"365.21","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.05","T3_MPL1":"20.06","T3_MPL2":"20.00","Thermocouple":"20.25","Altitude":"365.39","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.06","T3_MPL2":"20.00","Thermocouple":"22.25","Altitude":"365.66","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.06","T3_MPL2":"20.00","Thermocouple":"22.25","Altitude":"364.94","Humidit
{"d":{"myName":"ArduinoMega2560_1","T2_BME280":"23.06","T3_MPL1":"20.06","T3_MPL2":"20.00","Thermocouple":"22.50","Altitude":"364.41","Humidit

```

Figure 5.19: Serial Monitor - Connection & Data Publishing¹⁴⁰

Thus, after the implementation of a mobile user interaction with Telegram that is described during section 5.6, it is possible to reset the system from any device that is connected to the Internet if required.

5.4 IBM Bluemix

Bluemix is an open cloud platform that allows developers and end-users to build, run and deploy applications directly within the cloud. It supports own IBM software and services as well as applications from other business partners. The Bluemix cloud platform is based on PaaS with additional Backend as a Service (BaaS) capabilities and it supports various programming languages for developing web and mobile applications such as iOS or Android. By providing a wide range of services that are ready-to-use, it is possible to either directly use the prebuilt services via “Boilerplates”, or to customize them by adding additional functionalities. As a developer, the interaction with the Bluemix infrastructure is either possible by the use of a browser-based interface, or by using the “Cloud Foundry” command-line-interface (CF) which allows to deploy locally developed applications at Bluemix.¹⁴¹

As usual for PaaS providers, Bluemix is also based on a pay-as-you-go scheme, where users only get a bill for the services that have been chosen and on the amount

¹⁴⁰Own illustration.

¹⁴¹Cf. STIFANI, (2015), pp. 2-4.

of infrastructure that is obtained. The following Fig. 5.20 shows an overview of all the available service categories in the Bluemix environment, whereby each category contains multiple services that can be executed.

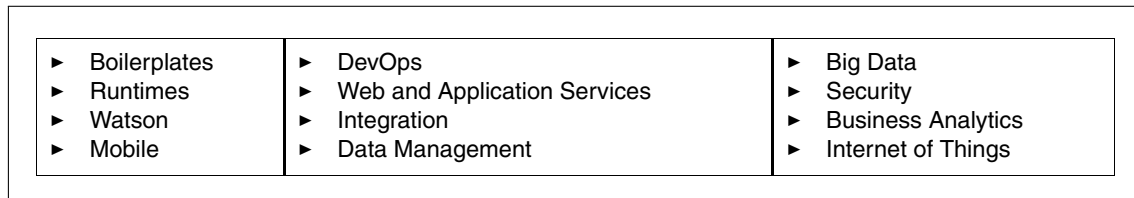


Figure 5.20: IBM Bluemix - Categories of Services¹⁴²

In context of implementing the sensor system, the “Boilerplates” named “Internet of Things Foundation Starter”, “Node-RED Starter” as well as the “Data Management” service “Cloudant NoSQL DB” and the “Big Data” service “IBM dashDB” are essential for establishing the communication with the IoT gateway via MQTT and to furthermore process, store and monitor the real-time and historical data.

Fig. 5.21 shows the application and service dashboard of the IBM Bluemix user environment. The green circles below the Node-RED applications indicate whether the app is running or not. The created services, including the setup and configuration is discussed in the following sections.

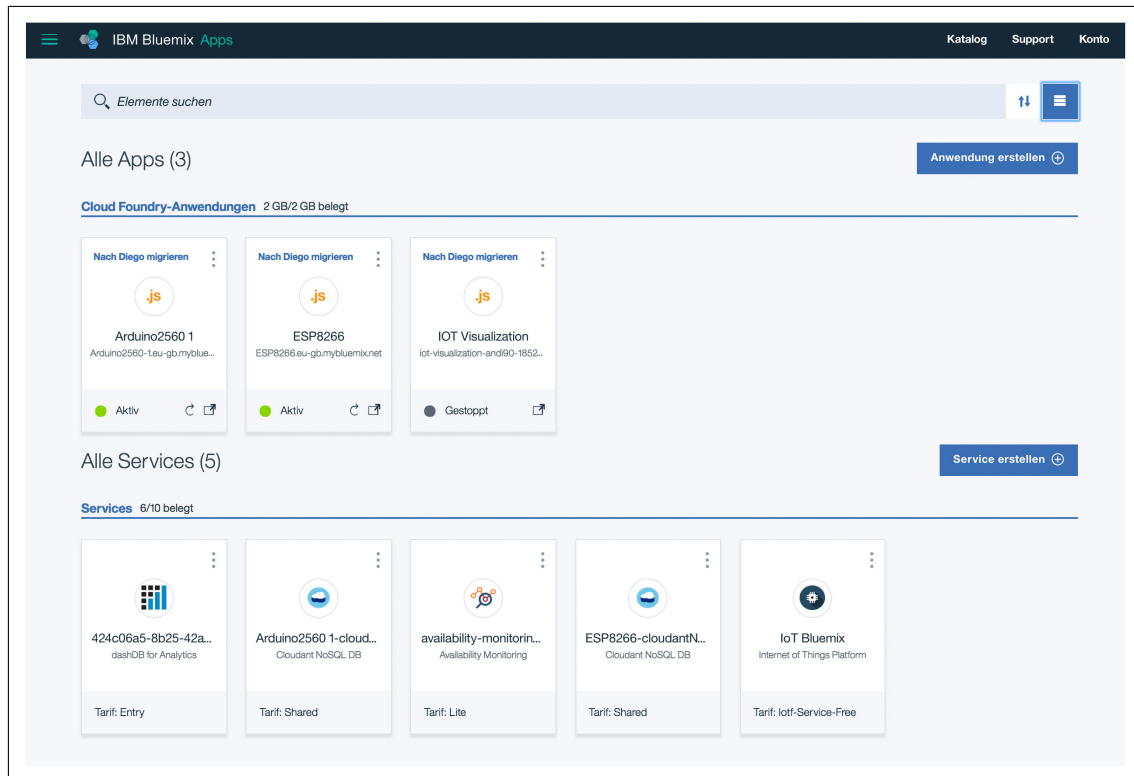
5.4.1 Internet of Things Foundation

The Internet of Things Foundation is the key service for establishing a communication between IBM Bluemix and the sensor gateway, either to retrieve sensor data from the gateway, or to send user specific commands back to the gateway via the Internet by using the lightweight MQTT protocol and acting as the “MQTT Broker”¹⁴³

In order to establish a connection with an IoT gateway, which in this case is the Arduino Mega 2560 or the ESP8266 NodeMCU, the device needs to be registered within the Internet of Things Foundation. Therefore, each connected gateway or device essentially needs to be assigned a “Device Class”, a unique “Device ID” and an authentication token that is either generated or self-defined, for security

¹⁴²Illustration from STIFANI, (2015), p. 2.

¹⁴³<https://console.ng.bluemix.net/docs/services/IoT/index.html>, (retrieved 06/02/2017).

Figure 5.21: IBM Bluemix - Dashboard¹⁴⁴

and privacy reasons. All this information is related to the “Organization ID” of the user account where the “Internet of Things Foundation” service is running. The “Device Class” is required for the connection of multiple devices of a different type, e.g. the ESP8266 NodeMCU and the Arduino MEGA 2560. It is left to the user to enter additional information, such as the manufacturer and the controller model, a serial number, the installation site, or a useful comment for each device.

Of course, the registration information for each device also needs to be configured within the Arduino program that runs on the controller so that the gateway is able to successfully connect to the Bluemix MQTT broker to publish and subscribe data. Fig. 5.22 shows the process of registering a device, with all the parameters listed.

After the registration of the IoT gateways, the devices show up in the overview of the main screen. A symbol also indicates whether there is an active connection of a gateway or not. As illustrated in the following Fig. 5.23, the registered Arduino MEGA 2560 is currently connected to the Internet of Things Foundation and is

¹⁴⁴Own illustration.

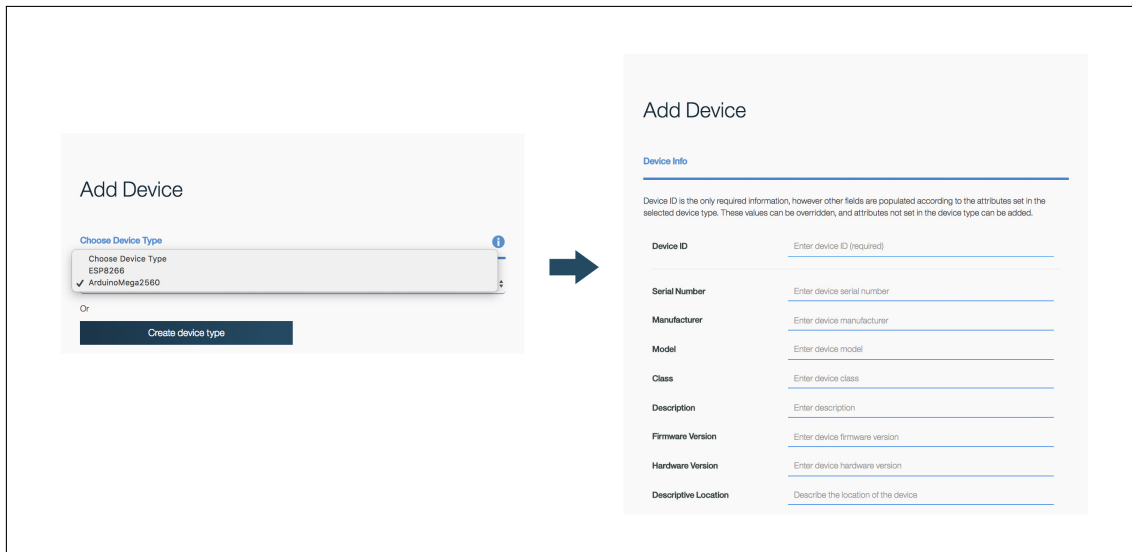


Figure 5.22: Internet of Things Foundation - Registering a Device¹⁴⁵

transmitting data to the broker. The overview also displays the additional device parameters that were entered during the registration process of the gateways.

Device ID	Device Type	Class ID	Date Added	Location
ESP8266_01	ESP8266	Device	Jun 13, 2016 4:53:34 PM	FabLab Graz
ArduinoMega2560_1	ArduinoMega2560	Device	Jul 30, 2016 6:35:05 PM	FabLab Graz
ESP8266_02	ESP8266	Device	Sep 1, 2016 8:22:33 PM	FabLab Graz
ESP8266_03	ESP8266	Device	Oct 11, 2016 8:48:57 PM	FabLab Graz

Figure 5.23: Internet of Things Foundation - Device Overview¹⁴⁶

A detailed overview of the received sensor values that are shaped in a JSON format is available by clicking on the connected device. It lists all data points that are transmitted, including the timestamp for recording the information in a database or to visualize it by the use of a real-time graph.

¹⁴⁵Own illustration.

¹⁴⁶Own illustration.

The following Fig. 5.24 below illustrates an example of some received data points at a specific timestamp that is transferred via MQTT as a JSON string.

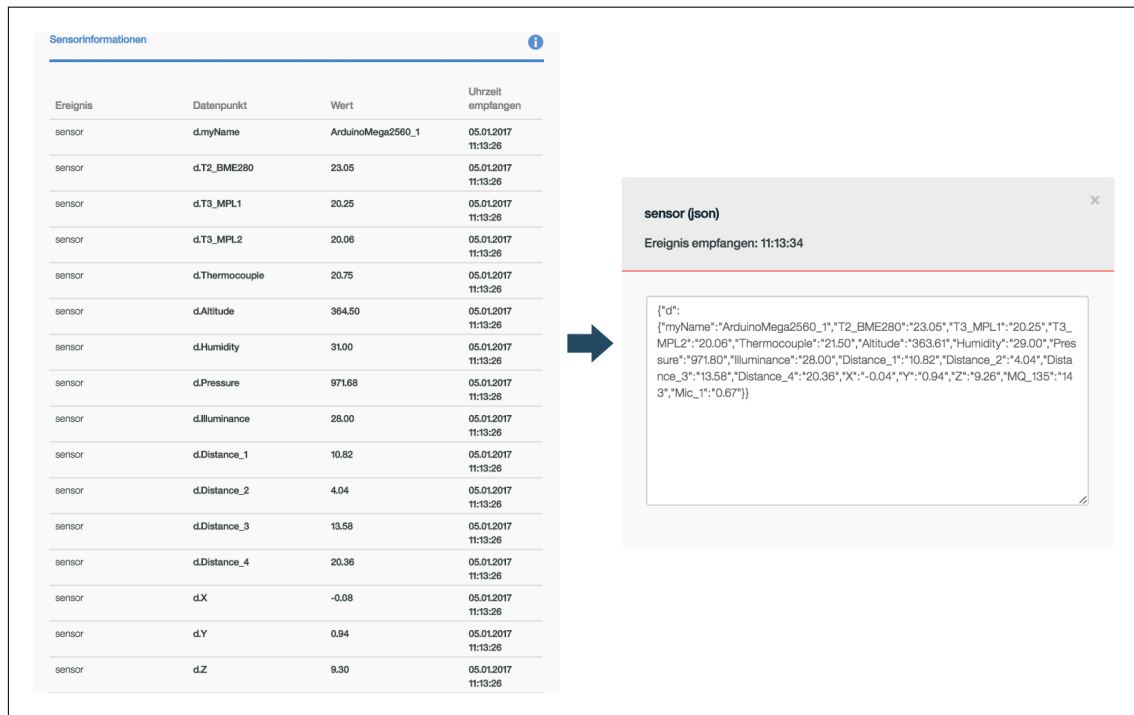


Figure 5.24: Internet of Things Foundation - Received Sensor Values¹⁴⁷

5.4.2 Clouant NoSQL DB

In order to store the historical sensor data that is gathered during the use of the Ultimaker 2 Extended 3D printer, a free-to-use NoSQL data layer, based on JSON documents is provided within Bluemix. The service is accessible through a HTTP interface and can be easily linked with the “Internet of Things Foundation”. Through this implementation, the transferred sensor data is automatically archived in the database by the use of monthly changing containers, as shown in the following Fig. 5.25.¹⁴⁸

¹⁴⁷Own illustration.

¹⁴⁸Cf. STIFANI, (2015), p. 48.

Name	Size	# of Docs	Actions
_warehouse	1.3 MB	6	[Refresh] [Lock] [Delete]
file_out	162.5 KB	169	[Refresh] [Lock] [Delete]
iotp_wwett1_default_2016-08	92.3 MB	198632	[Refresh] [Lock] [Delete]
iotp_wwett1_default_2016-09	341.7 MB	687291	[Refresh] [Lock] [Delete]
iotp_wwett1_default_2016-10	55.4 MB	121567	[Refresh] [Lock] [Delete]
iotp_wwett1_default_2016-11	132.8 MB	207238	[Refresh] [Lock] [Delete]
iotp_wwett1_default_2016-12	328.1 MB	507937	[Refresh] [Lock] [Delete]
iotp_wwett1_default_2017-01	2.8 MB	5414	[Refresh] [Lock] [Delete]

Figure 5.25: Cloudant NoSQL DB - Historical Sensor Data¹⁴⁹

5.4.3 dashDB

dashDB is a data warehousing tool to analyze historical data with built-in advanced analytics like data mining and predictive analytics. It is also possible to download a customized data set with adjusted columns and rows, as a .CSV file, for subsequent processing by the use of appropriate development tools such as Microsoft Excel¹⁵⁰ or MATLAB¹⁵¹. The warehousing tool is easy to connect to the other services such as the Cloudant NoSQL DB where the historical data is stored. Therefore, it is possible to transfer the data sets to dashDB and to start different analytics.¹⁵²

During this setup, an application of analytics tools for the gathered sensor data was not implemented yet since it is only test data that is recorded to validate the functionality of the sensor system. However, the subsequent Fig. 5.26 shows an example of a data set in dashDB which was retrieved from the Cloudant NoSQL DB.

¹⁴⁹Own illustration.

¹⁵⁰<https://products.office.com/de-at/excel>, (retrieved 06/02/2017).

¹⁵¹<https://www.mathworks.com/>, (retrieved 06/02/2017).

¹⁵²Cf. STIFANI, (2015), p. 56.

IBM dashDB™ 5% 424c06a5-8b25-42a6-Cloudant_1117842151404235 andreas.leitner@student.

Home
Tables
Load >
Run SQL
Analytics >
Monitor >
Settings >
Connect
Downloads
Help >

Create, drop, and work with tables Quick to
For existing tables, you can view details, browse data, and export data. [Learn more](#)

Add Table Delete Table Schema DASH105713 Table Name IOTP_WWETT1_DEFAULT_2017-01

Table Definition Browse Data

Click a row to see its details.

Maximum number of rows to retrieve: 1000 Apply

DEVICE ID	TIMESTAMP	DATA_D_X	DATA_D_Y	DATA_D_T2_BME280	DATA_D_DI_STANCE_1	DATA_D_DI_STANCE_2	DATA_D_DI_STANCE_3	DATA_D_DI_STANCE_4	DATA_D_HU_MIDITY
ArduinoMeg a2560_1	2017-01-05 T10:20:49.854+01:00	0.27	2.67	23.12	4.89	3.98	19.51	20.42	28.00
ArduinoMeg a2560_1	2017-01-05 T10:31:55.516+01:00	0.12	1.29	23.04	6.35	3.66	18.05	20.74	29.00
ArduinoMeg	2017-01-05	0.16	2.04	23.12	4.89	3.57	19.51	20.83	27.00

Figure 5.26: IBM dashDB - Sensor Data Table¹⁵³

5.4.4 Node-RED

The IBM Emerging Technology organization developed the open source platform Node-RED back in 2013 and introduced it in early 2015. Node-RED is a visual programming and wiring tool that makes it easier to connect, program and use the components of the Internet of Things. The naming relates to the fact that the platform was implemented using “Node.js”, which is an open-source JavaScript runtime development environment.¹⁵⁴

The basic idea of Node-RED was to make programming easier and accessible for everyone, also for people without good programming knowledge. For this reason, predefined code blocks called “Nodes” are provided and can be dragged into the Node-RED editor. Wiring them up in different ways makes up a “Flow” that can finally be deployed to build the application. For more complex tasks, “Function Nodes” can be added to use customized code blocks for adding user specific functionalities to the “Flows”.¹⁵⁵

¹⁵³Own illustration.

¹⁵⁴HEIDLOFF, (2015).

¹⁵⁵<https://nodered.org/>, (retrieved 18/12/2016).

Additionally, there is also an available database where users can contribute their own nodes and make them available for public. By importing those “Nodes” with the Node Package Manager (NPM) to the Node-RED application, other users can then easily reuse them for their own tasks. The process of importing new nodes to the application will be discussed in more detail in the next section.

Although Node-RED is already included within IBM Bluemix and can be set up really quickly by using a “Starter Platform”, it is also possible to run it locally and to deploy it as a stand-alone Node.js application.

Once a Node-Red application has been prepared using the “Node-RED Starter” Boilerplate in the IBM Bluemix environment, the source code can be edited afterwards in a created GIT repository¹⁵⁶ for the application. This is necessary when the user wants to add new custom nodes, or to update nodes of the application. See the following section for further details.

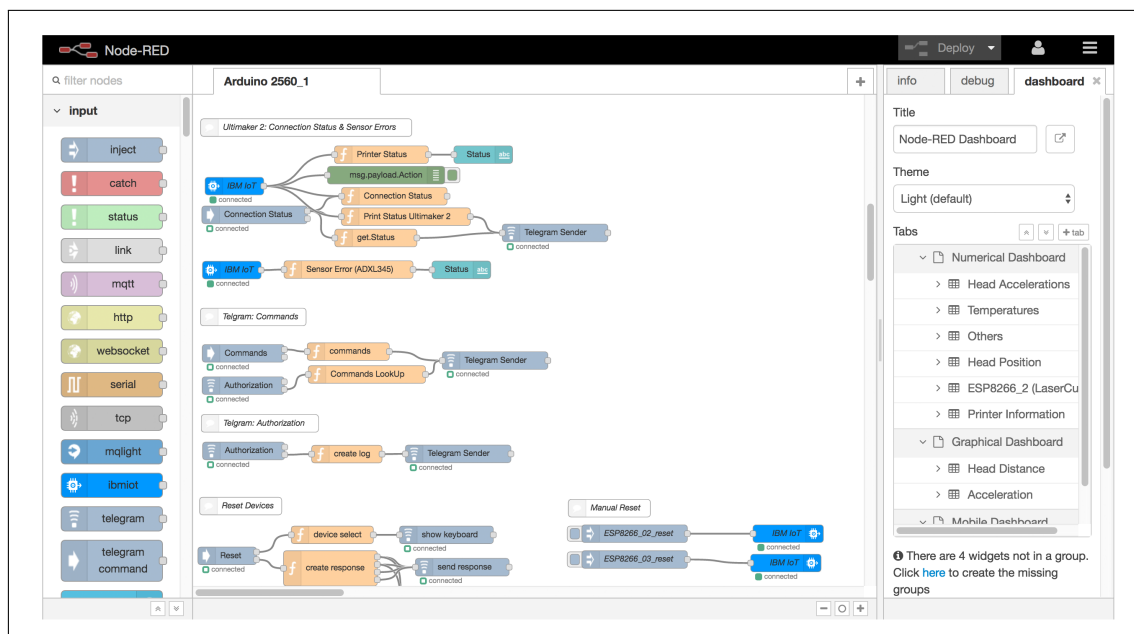


Figure 5.27: Node-RED - Flow Editor¹⁵⁷

Importing Nodes

It is possible to import new nodes to the Node-RED application by using the NPM. When using Node-RED locally without Bluemix, new nodes can be added by

¹⁵⁶<https://git-scm.com/>, (retrieved 06/02/2016).

¹⁵⁷Own illustration.

using the “npm install <npm-packagename>” command within the command line. The <npm-packagename> has to match with the corresponding name on the Node-RED flow-database¹⁵⁸ in order that the package manager is able to find and install the required node.

It is also possible to add nodes to the application directly within Bluemix by editing the source code of the Node-RED application in the IBM Bluemix DevOps Services. The nodes have to be added to the “package.json” file, again using the same package name as provided in the library. The left illustration of the following Fig. 5.28 shows the “package.json” file including manually added NPM nodes such as the Telegram Bot API as well as the Node-RED Dashboard User Interface (UI) for graphical visualization of the sensor data.¹⁵⁹

In case custom nodes need to be added that are not part of the Node-RED library introduced above, the import can also be done manually by uploading the “nodename.html” and the “nodename.js” file to a subfolder of the “nodes” folder within the applications structure. The naming of this subfolder has to match with the package name in the package.json file, just like the node in the Node-RED library. The left illustration of Fig. 5.28 shows the application structure including the manually added Telegram Bot API that is required for the user interaction discussed in section 5.6.¹⁶⁰

After performing changes to the code, the application has to be rebuilt and restarted by using the “Build & Deploy” command. It may take a little time until the application is running up and is ready to be started from the IBM Bluemix application dashboard again.

¹⁵⁸Node-RED Library, <http://flows.nodered.org/>, (retrieved 18/12/2016).

¹⁵⁹NODE-RED, (2016).

¹⁶⁰Cf. BISON, (2016), pp. 7-12.

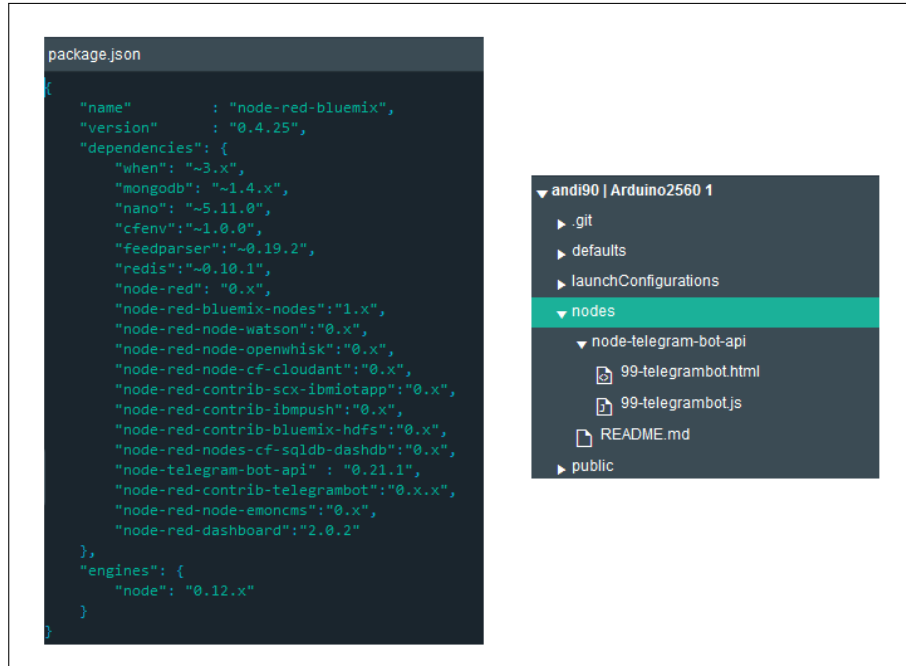


Figure 5.28: Node-RED - package.json File (left); Manually Added Nodes (right)¹⁶¹

Updating

The process of updating existing nodes via NPM to a new version is quiet similar. The “package.json” file that contains all the manually added nodes also includes the current version number for every installed node at the end of each line, as it can be seen from Fig. 5.28 above. This information does not have to be changed necessarily, however, it is still recommended for keeping the overview of the installed versions. After the changes are done, the execution of another “Build & Deploy”, as already described earlier, is required to update the added nodes to the latest version, including the Node-RED editor itself.¹⁶²

5.5 Sensor System Installation

Based on the insights regarding the previous sections of this chapter where the pinout, the wiring diagram and the sensors including the placement at the Ultimaker 2 Extended 3D printer was described, the following section outlines the finished installation of the sensor system at the machine.

¹⁶¹Own illustration.

¹⁶²NODE-RED, (2016).

5.5.1 IoT Gateway - Arduino MEGA 2560 & Ethernet Shield

The following Fig. 5.29 shows the IoT gateway for the communication with the sensors that are installed at the Ultimaker 2, including the Arduino MEGA 2560 as well as the Ethernet Shield v2 on top. The wiring and the placement of all the pin headers for connecting the sensor cables is based on the wiring diagram and stripboard layout of section 5.2.6. Because the routing of all the connections and traces is performed at the bottom layer of the stripboard, it is rather manageable to keep track of all the connected sensors.

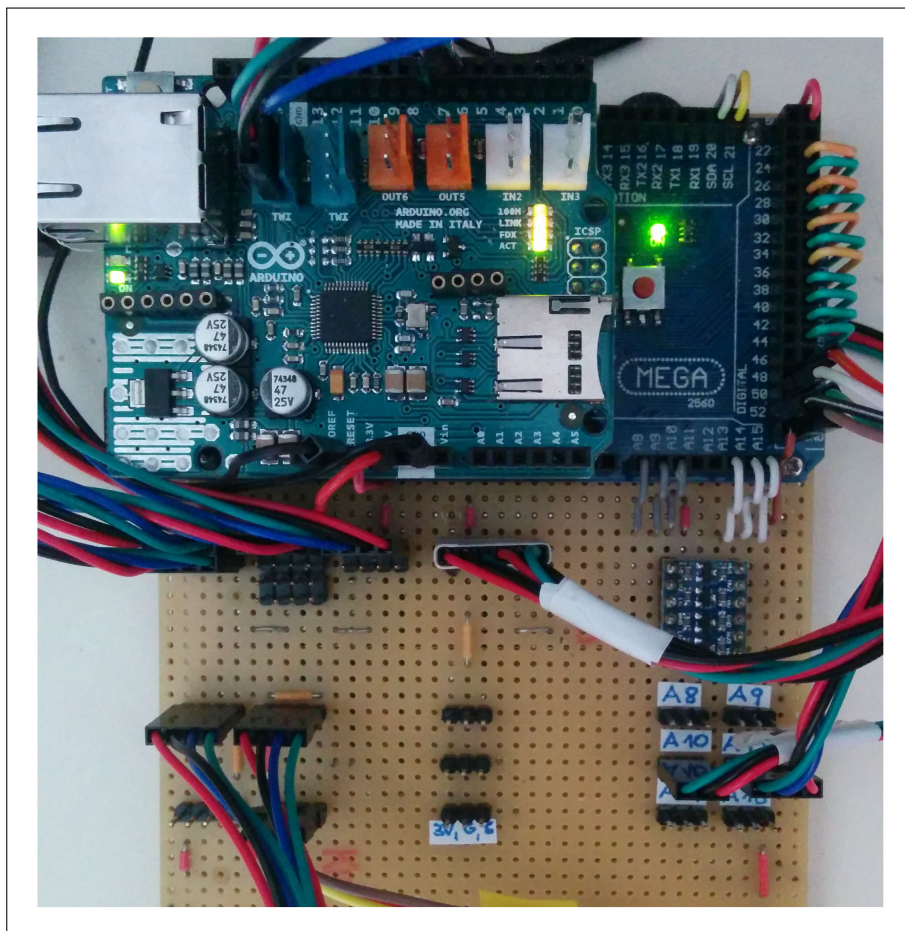


Figure 5.29: Sensor Installation - Arduino MEGA 2560, Ethernet Shield and Sensor Stripboard¹⁶³

¹⁶³Own illustration.

5.5.2 Ultimaker 2 - Extended & Sensor Setup

The finished installation of the introduced sensors in section 5.2.4 at the Ultimaker 2 Extended 3D printer is illustrated in the following example figures. Fig. 5.30 shows the installed HC-SR04 range detectors, including the printed mounting mechanisms based on the CAD models that were introduced in section 5.2.7.

As it can be seen out of the figures, the detectors “float” above the machines frame by attaching the mountings directly at the driving units, still providing enough space for the printer head to hit its end positions and the limit switches.

The illustration also shows the ADXL345 accelerator that was mounted directly onto the reflection shield for the range detectors at the printer head. Therefore, it is possible to detach the accelerator and the reflector simultaneously for providing easy maintaining access at the machine.

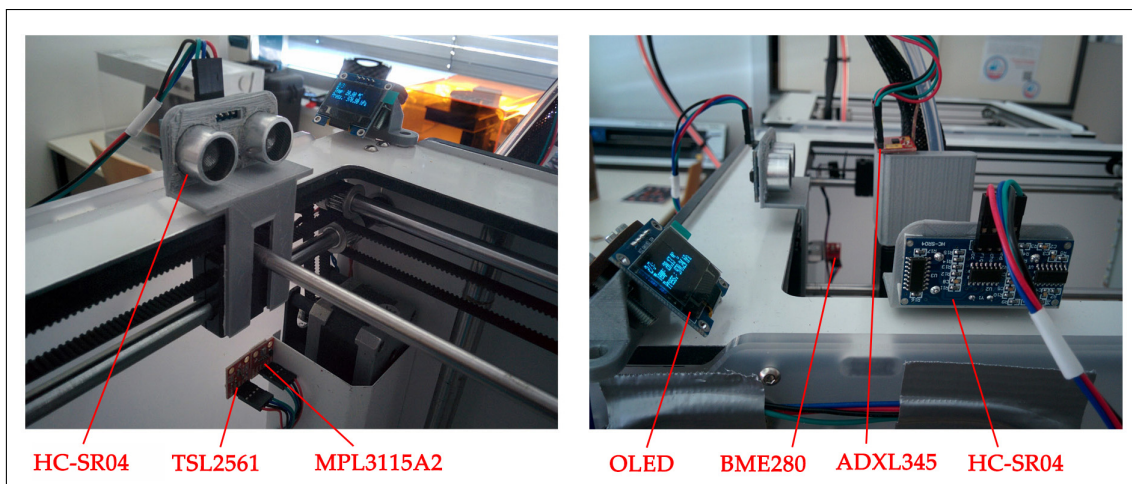


Figure 5.30: Sensor Installation - Ultimaker 2 (1)¹⁶⁴

Fig. 5.31 shows the top view of the installed sensor, where additional sensors such as the electret microphone or the MQ-135 air quality sensor are noticeable as well.

¹⁶⁴Own illustration.

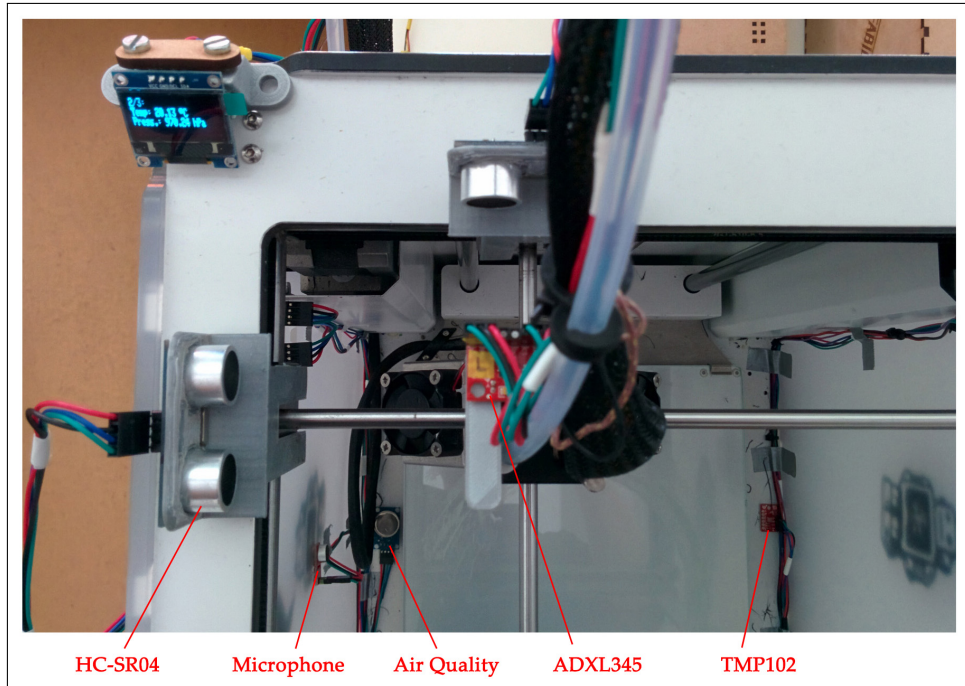


Figure 5.31: Sensor Installation - Ultimaker 2 (2)¹⁶⁵

The subsequent Fig. 5.32 shows an applied limit switch to ensure that the sensor system is only active in case that the printer head of the Ultimaker 2 drives out of its defined home position when starting a production progress.

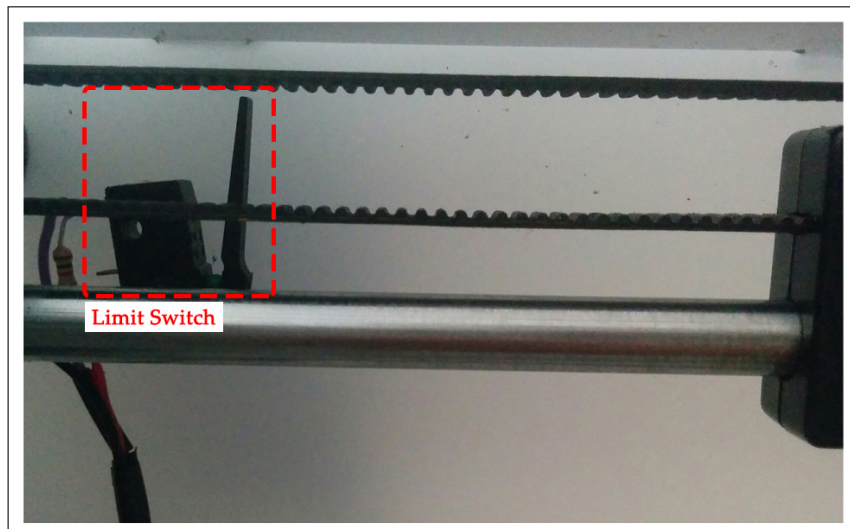


Figure 5.32: Sensor Installation - Limit Switch)¹⁶⁶

¹⁶⁵Own illustration.

¹⁶⁶Own illustration.

5.5.3 NodeMCU - Compact Solution

The NodeMCU is a compact alternative to the Arduino MEGA 2560 gateway, based on WiFi communication. As already mentioned in previous sections, due to the compact dimensions of the NodeMCU development board it is possible to install it closer to the sensor site or directly inside a machine if required. However, during the development of the sensor system in context of this thesis, the NodeMCU was mainly used to gather environmental parameters such as the ambient temperature, air pressure and humidity inside the FabLab and it was utilized for testing various sensors that were intended for the installation at the Ultimaker 2. Since the usability and the flexibility for testing is easier by the use of a WiFi connection instead of a tethered one, which is required for the Arduino and the Ethernet Shield, the NodeMCU development kit is more convenient for testing purposes.

To use the NodeMCU as an universal application for several installation purposes, a customized case was applied to place the controller inside of it on a small stripboard. For sensor communication the available analog as well as the I2C interface is accessible through pin headers, as it can be seen from the following Fig. 5.33.

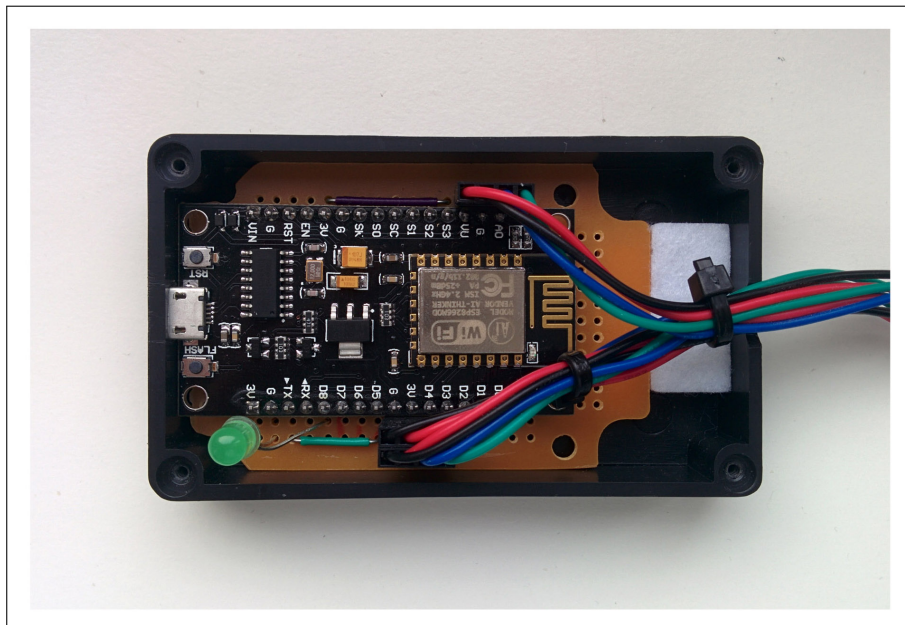


Figure 5.33: NodeMCU - Housing (1)¹⁶⁷

Additionally, a status LED is installed and connected to a digital GPIO to receive an

¹⁶⁷Own illustration.

optical notification whether the communication with IBM Bluemix or, respectively, the publishing of sensor data to the “Internet of Things Foundation” MQTT broker is successful or not.

Fig. 5.34 shows the connected NodeMCU inside the housing. The small hole is applied to perform a hard reset by triggering a small push button in case the controller software crashes.



Figure 5.34: NodeMCU - Housing (2)¹⁶⁸

5.6 User Notification & Interaction

By the use of the bi-directional MQTT messaging protocol to either transfer sensor data to IBM Bluemix or to receive user commands from Bluemix, it is possible to implement an user interaction with the sensor system and the machine. IBM Bluemix supports a lot of applications that can be controlled by the use of the Node-RED editor, which is the interface between the Bluemix IoT Foundation and applications outside the Bluemix environment such as various messaging services or Push applications.¹⁶⁹

Because of the bi-directional communication it is not only possible to provide monitoring of the current printer state, but it also allows notifications and remote

¹⁶⁸Own illustration.

¹⁶⁹TANG, (2015).

intervention of the system if necessary, or demanded. To provide a solution that is as convenient as possible, an interaction by the use of mobile devices is preferred. The fact that a smartphone is connected to the Internet continuously nowadays and that it is usually always carried with the user makes it a perfect alternative for notification, or interaction purposes. In general, it is possible to interact with the system by the use of any type of device, just like it is illustrated in the functional system diagram in section 5.1.

5.6.1 Telegram Messenger

Telegram is an easy to use messaging application available for multiple platforms such as Windows, MacOS, Android and iOS. In contrast to other messengers, Telegram supports a powerful API for building and integrating customized Telegram clients as well as an additional API to create programs that make use of Telegram messages as an interface, which are called Bots. This is also the reason why it was preferred over other messengers¹⁷⁰

The Telegram API can be integrated into Node-RED, as discussed in section 5.4.4, to implement a set of Telegram “Flows” into the Node-RED Editor for sending and receiving data or information via IBM Bluemix. The Bot acts as a HTTP-interface between the Node-RED code on the server and the user to automatically receive information based on specific events, or to send commands to the Bot within the messenger. The instruction set can be created directly within the messenger, whereas it has to match with the command names created in Node-RED in order to provide the required functionality.

During the set-up, the Bot receives an unique authorization token for making it identifiable within Node-RED. Once the Bot is created it can be invited to a chat group where multiple users can attend. Thereby, it is possible that several FabLab users can control the Bot and use its interfaces. The chat record is represented in a way that it is verifiable at any time which commands have been entered and by whom. For security reasons, the Bot can be protected in Node-RED by unlocking the “Chat-ID” of a user, so that the communication and interaction with the Bot and furthermore with the sensor system is only allowed for authorized users. Any

¹⁷⁰TELEGRAM.ORG, (2017).

attempt of controlling the Bot with an unauthorized Telegram account is recorded and the log is immediately sent to the administrator of the Bot in Telegram.

Fig. 5.35 shows the available commands of the implemented Telegram Bot, performed on an Android device. The right part of the illustration shows an example of the automated notifications, which are sent by the Bot as soon as a print job is started, or when the production is finished and the printer has returned to its home position.

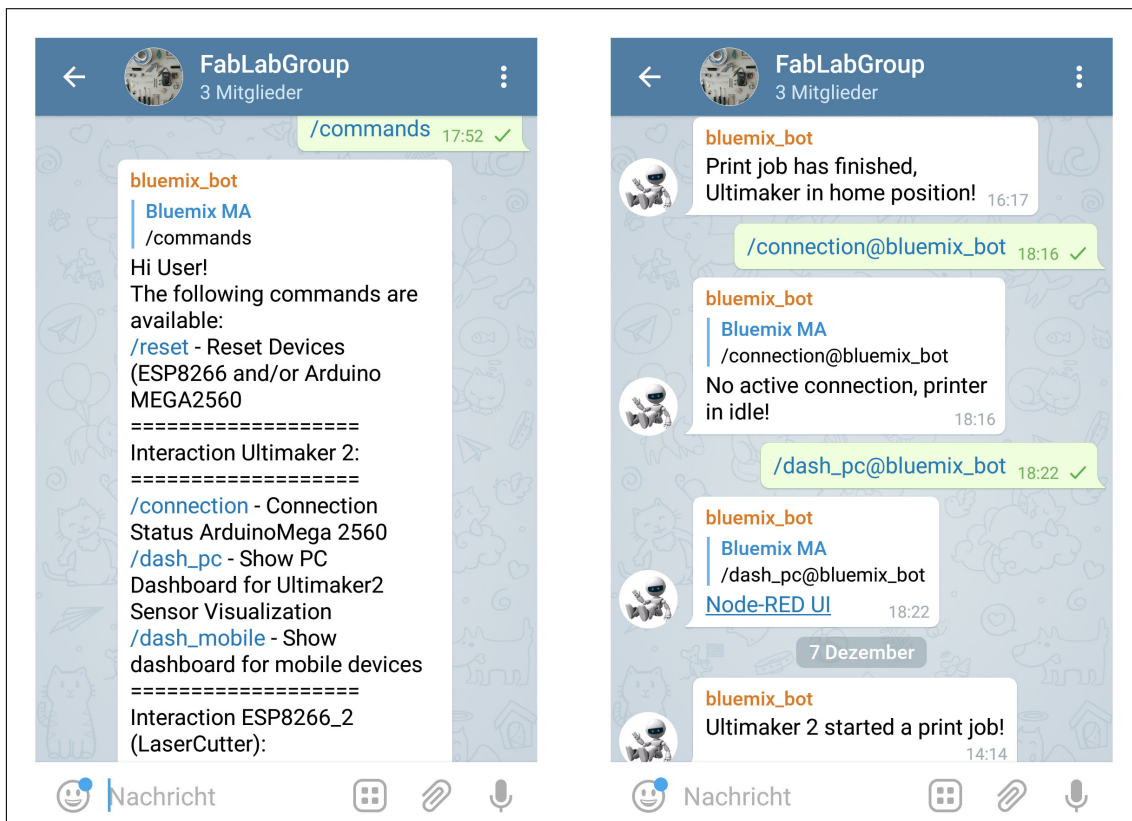


Figure 5.35: Telegram Bot - Command Overview¹⁷¹

By the use of Telegram it is also possible to reset a gateway device, e.g. if a sensor is disconnected accidentally during maintenance work on a machine. Depending on the sensor type and its interface, the communication with the controller can still be interrupted, although the header has been connected again.

In that case, a reset of the controller and the subsequent re-connection with the network is the only possible intervention. Fig. 5.36 shows the process of resetting a connected device within the Telegram messenger.

¹⁷¹Own illustration.

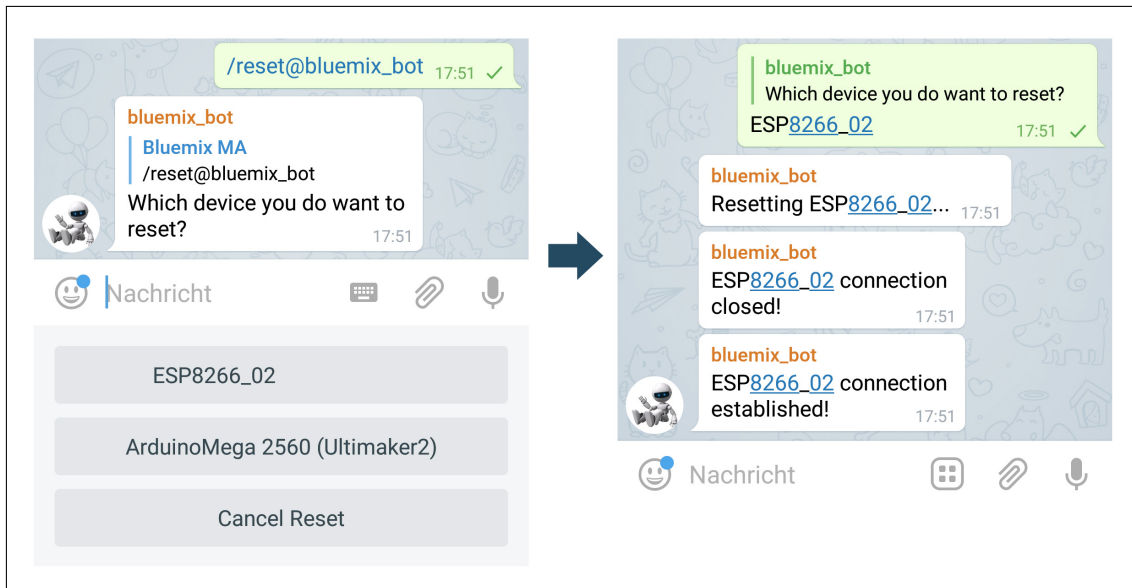


Figure 5.36: Telegram Bot - Command Overview¹⁷²

5.7 Visualization

Within the scope of introducing IBM Bluemix, section 5.4.2 and 5.4.3 have already pointed out how to store the measured sensor values within a database and how to access the historical data afterwards for analysis purposes. However, for monitoring and supervision it is also of great interest to visualize current absolute values as well as the real-time data over time. Although it is possible to visualize the published sensor data directly within the “Internet of Things Foundation” of Bluemix, this solution is rather inappropriate because it is required to be logged into the Bluemix environment, thus, it is not possible to access the charts from outside using a HTTP interface. To overcome this issue, Node-RED provides a package to create adjustable dashboards to visualize sensor graphs or data that will be described in more detail within this section.

5.7.1 Node-RED Dashboard UI

Node-RED provides a convenient user interface for creating IoT dashboards and to visualize real-time data via different types of diagrams. The dashboard is customizable regarding its dimensions, depending on which device is going to be

¹⁷²Own illustration.

used for monitoring. Whereas desktop computers or notebooks provide a larger display for visualization, a mobile device such as a tablet or a smartphone is restricted in display size due to its smaller dimensions.¹⁷³

Although it is still possible to display a large dashboard on a small screen by scrolling, it is better to provide two different visualization schemes for both, computer and mobile devices. The dashboard user interface does not only allow to display the published sensor data, but it is also possible to display information messages for the users, depending on specific events or sensor values.

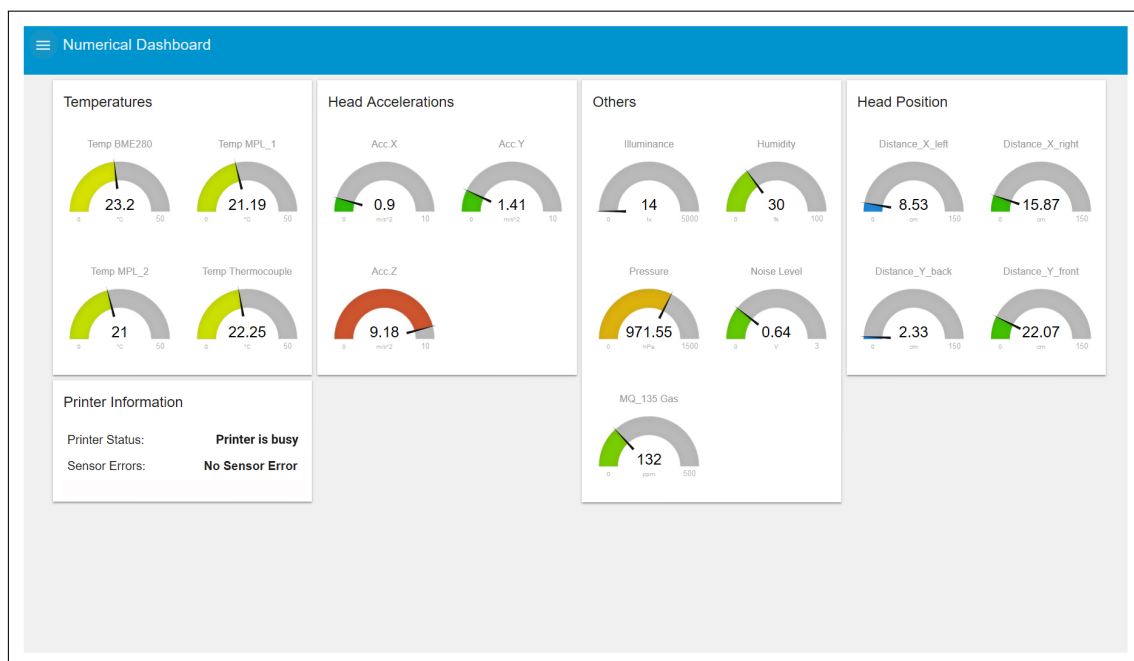


Figure 5.37: Ultimaker 2 Extended - Sensor Visualization¹⁷⁴

Fig. 5.37 above shows the example of a numerical dashboard, optimized for desktop devices to provide an overview of the real-time sensor values, measured at the Ultimaker 2 Extended 3D printer. The dashboard also displays additional information such as the printer state and if there are sensor errors in case a sensor is disconnected accidentally, e.g. during maintenance of the machine. Within the dashboard it is also possible to display information messages depending on a specific event such as the printer status or the connection status of the IoT gateway.

As mentioned earlier, the dashboard can be adjusted in its size in order to fit on

¹⁷³<https://github.com/node-red/node-red-dashboard>, (retrieved 06/02/2017).

¹⁷⁴Own illustration.

screens of mobile devices. With the use of the developed Telegram Bot acting as an interface between Node-RED and the user, it is possible to integrate the dashboard view into the Telegram Chat Group. The command list of Telegram, illustrated in Fig. 5.35 of section 5.6.1, includes separated commands for the visualization of the dashboard. By using the “/dash_mobile” command, Telegram opens the mobile dashboard via the provided HTTP interface of Node-RED.

The following Fig. 5.38 shows the mobile view of Ultimaker 2 printer head data (left figure) and environmental parameters inside the FabLab Graz, measured with the NodeMCU Development Kit (right figure).

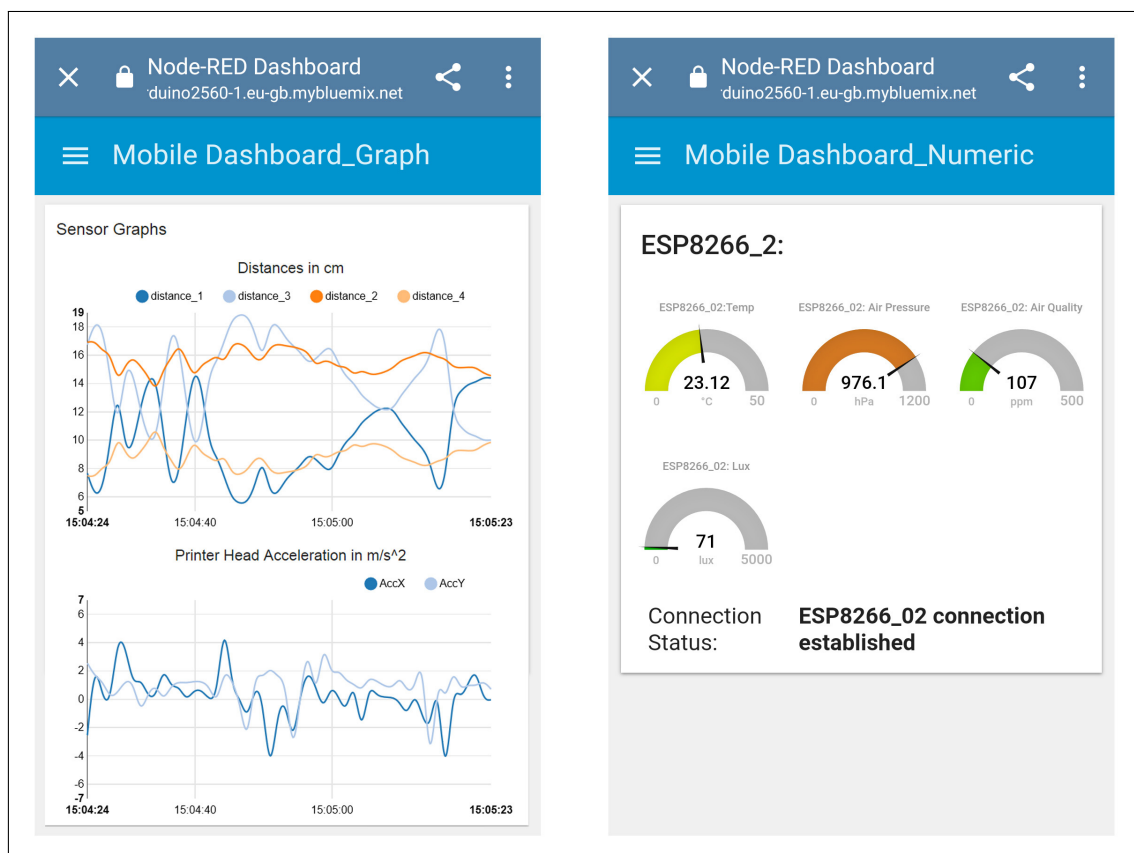


Figure 5.38: Sensor Visualization - Mobile View¹⁷⁵

¹⁷⁵Own illustration.

5.7.2 Local Data Visualization

Besides the Node-RED visualization dashboard and the remote notifications in Telegram, a local information system using small, 0.96 " OLED displays (SSD1306) with I2C interfaces, is additionally implemented.

The display shows the acquired real-time sensor data directly at the machine site for providing an overview without the need of an additional device. In context of the first implementation of the system only one display is installed, whereby the different measurement values can be monitored by toggling between different screens using a push button that is connected to a digital input of the controller.

Fig. 5.39 illustrates the three screens of the display, visualizing sensor data of the 3-axis accelerator attached at the printer head as well as various other parameters such as temperatures, the humidity or the light intensity inside the machine, gathered from the sensors that were introduced in section 5.2.4.

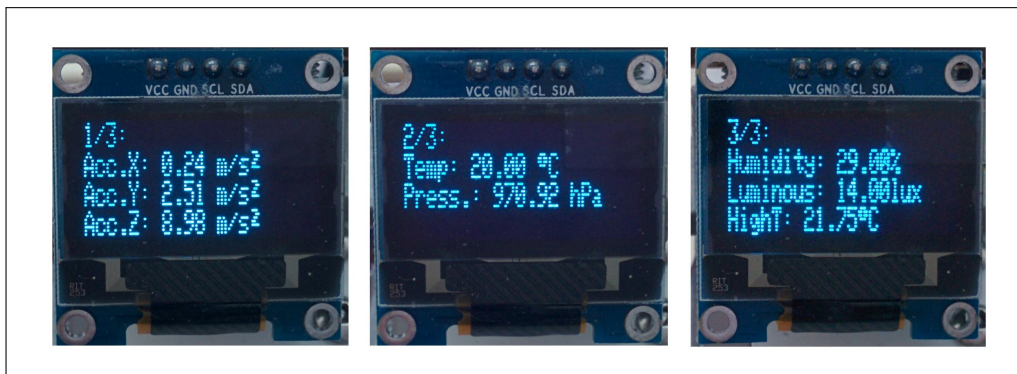


Figure 5.39: OLED - Notification Display; Ultimaker 2 - Extended¹⁷⁶

¹⁷⁶Own illustration.

6 Conclusion & Future Perspectives

This thesis provides a concept to improve a major drawback of current FabLabs or Makerspaces. Most FabLabs are free to use facilities where users can realize their own ideas and concepts by the use of various production machines and tools such as 3D printers, laser cutters or CNC milling machines. Due to the lack of a bi-directional communication between the operator and the machines within a FabLab, it is not possible to receive notifications about specific events and states of machines and it is unfeasible that a machine is able to react automatically to certain incidents, or to prevent worst case scenarios such as machine crashes.

The objective of this master thesis is the initial development of a smart, cloud-based sensor system for the application within FabLabs to extend the present one-way user-to-machine communication flow. A “smart” sensor system which is connected to the Internet and installed at the machine site is acquired to gather machine related as well as environmental data by the use of a vast amount of sensors and actuators. The data is processed by a microcontroller to store production-relevant information as well as environmental parameters within the PaaS cloud service IBM Bluemix.

Since the connection between the sensor system and the cloud platform is bi-directional, machine users are informed about specific events via notifications messages and visualization charts and they are allowed to intervene into ongoing production processes if required. For the first implementation and testing of the developed sensor system in context of this thesis, the Ultimaker 2 Extended 3D Printer, being part of the machine park in the FabLab Graz, was intended because it provides easy accessibility for the sensor installment and it is frequently used at the FabLab opening times in order to record test data during ordinary operation.

The information flow of the sensor system, starting at the low-level controller interface where all the sensor data is gathered and furthermore transmitted to the

cloud service IBM Bluemix, is based on the approach of the Internet of Things (IoT). IoT applications have experienced a rapid development since the first-ever definition of the term back in 1999 until recent years, where the amount of connected devices to the Internet started to escalate. The technological leap regarding new electronic data processing hardware, faster and affordable Internet access from multiple devices using WiFi, as well as the introduction of cloud computing technologies and services have contributed significantly to the growth of connected devices. As a consequence of the massive increase, it was essential to launch the new generation Internet Protocol Version 6 (IPv6) for providing additional address space within the Internet.

An IoT system requires essential components for collecting data of physical objects, or environmental factors in the real world to process the data and to furthermore store the data in an appropriate format within a cloud service. Thus, the consideration of a basic system concept was carried out in chapter 3 that also forms the foundation of the subsequent implementation by acquiring specific components.

Related to the sensor system concept that was outlined in chapter 3, the selection of an appropriate gateway device that was required for the connection of all the utilized sensors as well as for further data processing and forwarding of the sensor data to the cloud platform IBM Bluemix, was essential. Consequently, the IoT gateway, being the centerpiece of the entire system, is of particular significance. Several network capable gateway devices are available on the market, however, not every device is suitable for processing a vast amount of sensor data in terms of hardware interfaces and resources.

In order to provide a comprehensive comparison of current controllers and to pick the most suitable alternatives for the application, a value benefit analysis was utilized in chapter 4 in context of a market research. The matrix considers multiple aspects and device specifications that are important to meet the requirements of the sensor system. To prevent an over influence of certain factors and vice versa, the scoring of each utilized criteria was balanced within the scoreboard. As a result of the comparison, two MCU boards with different network interfaces (Ethernet and WiFi) were utilized for the application as the sensor gateway for providing two different communication approaches.

The practical implementation of the sensor system that is carried out within chapter 5 outlines the selection of the applied sensors at the Ultimaker 2 as well as the

wiring and installation interventions at the machine. The software development for the microcontroller and several communication tests to establish a connection between the IoT gateway and the IBM Bluemix “Internet of Things Foundation” were already performed before the actual system installation to ensure a full functionality. Testing of both communication directions to either receive data from the sensors, or to send user commands to the sensor system, e.g. resetting, was an essential part of the implementation process. In context of this, an HTTP user interface based on the Node-RED programming editor together with a messaging application Bot was implemented to furthermore receive machine related information directly to a mobile device or a computer. It allows to monitor real-time machine data on different dashboards and it supports the possibility to send notifications about an ongoing production progress or sensors errors to the messenger application.

Since the controller software was implemented in a way that it is applicable on both gateway controllers independent from the network interface, it is possible to connect and mix multiple gateways of different types to IBM Bluemix in order to interchange data between different machines. Thereby, is possible to implement a complete connectivity of all devices within a FabLab to extend the implemented bi-directional user-to-machine communication also to an intelligent machine-to-machine communication (M2M).

However, the practical implementation of this thesis also provides a good foundation for further advances regarding predictive analytics or machine learning where machines can automatically react and adjust themselves based on a certain tendency or trend of a greater amount of gathered data (Big Data Analytics). A consequence of this approach would be to directly connect the IoT gateway with the machines controller and its software, which is not realized at this stage. Therefore, it is not yet possible to directly interfere with active machine parts, such as the controlling of the printer head or the drive units.

Another possible future enhancement regarding the sensor system is the installation and the connection of the sensors and hardware at the machine. Although it was already ensured to keep the wiring effort at the controller as low as possible, it is still a prototypical application with provisional attached sensors and mountings. The stripboard that was utilized to place all the sensor pin headers within close proximity to the MCU board may be substituted with a professional circuit board

by the preparation of a PCB layout, to improve the overview and to build it more compact.

Also for other application areas next to FabLabs and Makerspaces, the concept of cloud computing based data monitoring and the application of IoT based sensor systems for smart manufacturing, including connected machines or industrial robots with embedded sensors, is a big opportunity for the future. By applying such systems it is possible to enhance the efficiency of production progresses and to reduce errors in the complete production chain, whereas multiple machines and processes are involved. Furthermore, also the safety within industrial environments can be increased by applying smart sensor systems that are capable of advanced warnings, in order to reduce the risk of damages or dangerous conditions.

Overall, the result of this thesis is a fully working implementation of an initial cloud-based sensor system applied on the Ultimaker 2 Extend 3D printer within the FabLab Graz. The sensor system supports a bi-directional communication between the machine and the user as well as an automated storage process of the gathered sensor data in a structured database for historical purposes. The additionally implemented user interface allows easy communication with the system to either receive notifications, or to visualize machine related data. The achievements during this thesis provide a solid foundation for further development and expansion of the system in future.

List of References

- ADAFRUIT (2016), *Monochrome 0.96" 128x64 OLED graphic display*. (retrieved 2017-06-02). URL: <https://www.adafruit.com/product/326> (cit. on p. 57).
- ARDUINO.CC (2016a), *Arduino Ethernet Shield V2*. (retrieved 2017-04-01). URL: <https://www.arduino.cc/en/Main/ArduinoEthernetShield> (cit. on p. 50).
- (2016b), *Arduino Product Range*. (retrieved 2016-19-12). URL: <https://www.arduino.cc/en/Main/Products> (cit. on pp. 32, 33).
- (2016c), *Arduino Software (IDE)*. (retrieved 2016-19-12). URL: <https://www.arduino.cc/en/Guide/Environment> (cit. on pp. 30, 64).
- (2016d), *ATmega2560-Arduino Pin Mapping*. (retrieved 2017-04-01). URL: <https://www.arduino.cc/en/Hacking/PinMapping2560> (cit. on p. 49).
- (2016e), *MQ Gas sensors*. (retrieved 2017-10-01). URL: <http://playground.arduino.cc/Main/MQGasSensors#Introduction> (cit. on p. 56).
- BISON, J. (2016), *Custom Nodes in Node-RED*. (retrieved 2017-06-02). URL: <https://github.com/jeancarl/node-red-labs/blob/master/node-red-custom-nodes/node-red-custom-nodes.pdf> (cit. on p. 80).
- BRADLEY, J. et al. (2013), *Internet of Everything (IoE) Value Index*. White Paper. (retrieved 2016-30-11). URL: http://internetofeverything.cisco.com/sites/default/files/docs/en/ioe-value-index_Whitepaper.pdf (cit. on p. 7).
- BUYA, R.; BROBERG, J.; GOSCINSKI, A. (2011), *Cloud Computing: Principles and Paradigms*. First. Wiley. ISBN: 99780470887998 (cit. on p. 14).
- BUYA, R.; DASTJERDI, A. V. (2016), *Internet of Things - Principles and Paradigms*. First. Elsevier Ltd. ISBN: 9780128053959 (cit. on pp. 6, 8, 10, 11, 13, 14, 16).
- CHAOUCHI, H. (2010), *The Internet of Things - Connecting Objects*. First. Wiley. ISBN: 9781848211407 (cit. on pp. 11, 12).
- CHASE, J. (2013), *The Evolution of the Internet of Things*. (retrieved 2016-14-11). URL: <http://www.ti.com/lit/ml/swrb028/swrb028.pdf>.

- CHAUHAN, C. (2015), *Internet of Things - Evolution Over Time*. (retrieved 2016-14-11). URL: <https://www.linkedin.com/pulse/internet-things-evolution-over-time-chhavi-chauhan>.
- CHUI, M.; LÖFFLER, M.; ROBERTS, R. (2010), "The Internet of Things." In: *McKinsey Quarterly* (2010, March). (retrieved 2016-14-11). URL: <http://www.mckinsey.com/industries/high-tech/our-insights/the-internet-of-things> (cit. on p. 6).
- COLEY, G. (2014), *BeagleBone Black System Reference Manual*. (retrieved 2016-19-12). URL: https://github.com/CircuitCo/BeagleBone-Black/blob/master/BBB_SRM.pdf (cit. on pp. 41, 42).
- ESP8266.COM (2016), *ESP8266 Modules Family*. (retrieved 2016-20-12). URL: <http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family> (cit. on p. 34).
- ESPRESSIF (2016), *ESP8266EX Datasheet*. (retrieved 2016-20-12). URL: https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf (cit. on p. 33).
- FABFOUNDATION.ORG (2016a), *What Is A Fab Lab?* (retrieved 2017-19-01). URL: <https://fabfoundation.org/index.php/what-is-a-fab-lab/index.html> (cit. on p. 2).
- (2016b), *What Qualifies As A Fab Lab?* (retrieved 2017-19-01). URL: <https://fabfoundation.org/index.php/what-qualifies-as-a-fab-lab/index.html> (cit. on pp. 2, 25).
- FABLABCONNECT.COM (2016), *FabLabs Overview Worldwide*. (retrieved 2017-19-01). URL: <http://www.fablabconnect.com/1000-fab-labs-97-countries/> (cit. on p. 2).
- FABLABS.IO (2016), *FabLabs Overview Worldwide*. (retrieved 2017-19-01). URL: <https://www.fablabs.io/map> (cit. on p. 1).
- FRADEN, J. (2016), *Handbook of Modern Sensors - Physics, Design, and Applications*. Fifth. Springer. ISBN: 9783319193021 (cit. on pp. 12, 13).
- FRIESSNIG, M.; BÖHM, T; RAMSAUER, C. (2015), *Making FabLabs smart through sensors and Big Data analysis*. i-Know 2015 (cit. on p. 1).
- GILCHRIST, A. (2016), *Industry 4.0 - The Industrial Internet of Things*. First. Apress. ISBN: 9781484220467 (cit. on p. 18).
- GUBBI, J. et al. (2013), *Internet of Things (Iot): A vision, architectural elements, and future*. *Future Generation Computer Systems* 2013;29(7):1645-60. (retrieved

- 2016-14-11). URL: <http://www.buyya.com/papers/Internet-of-Things-Vision-Future2013.pdf>.
- HASSANALIERAGH, M. et al. (2015), *Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-based Processing: Opportunities and Challenges*. 2015 IEEE International Conference on Services Computing, pp. 285–292, DOI: 10.1109/SCC.2015.47. (retrieved 2016-06-12). URL: http://www.ece.rochester.edu/~gsharma/papers/Moeen_HealthMonitor_SCC2015.pdf (cit. on pp. 21, 22).
- HEIDLOFF, N. (2015), *What is Node-RED? How can it be used for the Internet of Things?* (retrieved 2017-05-01). URL: <http://heidloff.net/article/21.01.2015081841NHEAL8.htm> (cit. on p. 78).
- HÖLLER, J. et al. (2014), *From Machine-to-Machine to the Internet of Things*. First. Elsevier Ltd. ISBN: 9780124076846 (cit. on pp. 17, 18).
- IBM (2016), *MQTT messaging*. (retrieved 2017-10-01). URL: <https://console.ng.bluemix.net/docs/services/IoT/reference/mqtt/index.html> (cit. on p. 71).
- INTEL (2015a), *Intel Edison Development Platform, Product Brief*. (retrieved 2016-18-12). URL: http://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison_pb_331179002.pdf (cit. on pp. 38, 39).
- (2014), *Intel Galileo Gen 2 Development Platform*. (retrieved 2016-18-12). URL: http://www.intel.com/content/dam/support/us/en/documents/galileo/sb/intelgalileogen2prodbrief_330736_003.pdf (cit. on pp. 38, 39).
- (2015b), *Intel Quark SoC X1000 Series, Product Brief*. (retrieved 2016-18-12). URL: <http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/intel-quark-product-brief-v3.pdf>.
- LAMBRECHTS, J.; SINHA, S. (2016), *Microsensing Networks for Sustainable Cities*. First. Springer. ISBN: 9783319283579 (cit. on p. 18).
- LAMPKIN, V. et al. (2012), *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM Redbooks. (retrieved 2016-14-11). URL: <http://www.redbooks.ibm.com/redbooks/pdfs/sg248054.pdf> (cit. on pp. 16, 68, 69).
- LEA R., Calderon R. (2016), “Node-RED: Lecture 1 – A brief introduction to Node-RED.” In: *Sense Tecnic Systems Inc.* (retrieved 2017-05-01). URL: <http://noderedguide.com/nr-lecture-1/>.
- LIN, H.; BERGMANN, N. (2016), *IoT Privacy and Security Challenges for Smart Home Environments*. MDPI AG, Information 2016, 7(3), 44; doi:10.3390/info7030044.

- (retrieved 2016-06-12). URL: <http://www.mdpi.com/2078-2489/7/3/44> (cit. on p. 23).
- LINKSPRITE (2014), *pcDuino3 - Overview*. (retrieved 2016-19-12). URL: <http://www.linksprite.com/linksprite-pcduino3/> (cit. on pp. 40–42).
- MCEWEN, M.; CASSIMALLY, M. (2013), *Designing the Internet of Things*. First. Wiley. ISBN: 9781118430620 (cit. on pp. 7, 13, 30, 37).
- MOOLAYIL, J. (2016), *Smarter Decisions – The Intersection of Internet of Things and Decision Science*. First. Packt Publishing. ISBN: 9781785884191 (cit. on p. 17).
- MOREL, L.; LE ROUX, S. (2016), *Fab Labs: Innovative User*. First. Wiley. ISBN: 9781848218727 (cit. on p. 2).
- NODEMCU (2015a), *NodeMCU DEVKIT 1.0*. (retrieved 2016-14-11). URL: https://github.com/nodemcu/nodemcu-devkit-v1.0/blob/master/Documents/NodeMCU_DEVKIT_1.0.jpg (cit. on p. 34).
- (2015b), *NodeMCU Pin Definition*. (retrieved 2016-19-12). URL: <https://github.com/nodemcu/nodemcu-devkit-v1.0/blob/master/Documents/NODEMCU-DEVKIT-V1.0-INSTRUCTION-EN.pdf> (cit. on p. 51).
- NODEMCU.COM (2014), *ESP8266 Modules Family*. (retrieved 2016-20-12). URL: http://nodemcu.com/index_en.html (cit. on p. 34).
- NODE-RED (2016), *Documentation*. (retrieved 2017-05-01). URL: <https://nodered.org/docs/> (cit. on pp. 80, 81).
- PÉREZ HERNÁNDEZ, M.; REIFF-MARGANIEC, S. (2014), *Classifying Smart Objects using Capabilities*. Tech. rep. (retrieved 2016-30-11). University of Leicester, Department of Computer Science. URL: <http://www.cs.le.ac.uk/people/srm13/publications/smartcomp14.pdf>.
- RASPBERRYPI.ORG (2016a), *Product Range*. (retrieved 2016-19-12). URL: <https://www.raspberrypi.org/products> (cit. on p. 37).
- (2016b), *Raspberry PI - FAQs*. (retrieved 2016-20-12). URL: <https://www.raspberrypi.org/help/faqs/#intro> (cit. on p. 36).
- AL-ROOMI, M. et al. (2013), *Cloud Computing Pricing Models: A Survey*. International Journal of Grid and Distributed Computing Vol.6, No.5 (2013), pp.93-106. (retrieved 2016-30-11). URL: http://www.sersc.org/journals/IJGDC/vol6_no5/9.pdf (cit. on p. 14).
- ROSENQUIST, M. (2014), “Cyber Security is Not Prepared for the Growth of Internet Connected Devices.” In: *IT Peer Network*. (retrieved 2016-28-11). URL: <https://itpeernetwork.intel.com/cyber-security-is-not-prepared-for-the-growth-of-internet-connected-devices/> (cit. on pp. 9, 10).

- SCHULZE, L. (2016), *Value Benefit Analysis*. (retrieved 2017-06-02). URL: https://gc21.giz.de/ibt/en/opt/site/ilt/ibt/regionalportale/sadc/inhalt/logistics/module_03/61_value_benefit_analysis.html (cit. on p. 44).
- SPARKFUN (2016a), *Accelerometers*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/categories/80> (cit. on p. 53).
- (2016b), *ADXL345*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/9836> (cit. on p. 52).
- (2016c), *BME280*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/13676> (cit. on p. 53).
- (2016d), *Electret Microphone Breakout*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/12758> (cit. on p. 56).
- (2016e), *Environmental Sensors*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/categories/152> (cit. on p. 54).
- (2016f), *GP1A57HRJ00F*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/9299> (cit. on p. 57).
- (2016g), *HC-SR04*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/13959> (cit. on p. 53).
- (2016h), *Logic Level Converter - Bi-Directional*. (retrieved 2017-04-01). URL: <https://www.sparkfun.com/products/12009> (cit. on p. 58).
- (2016i), *MAX31855K*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/13266> (cit. on p. 55).
- (2016j), *MPL3115A2*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/11084> (cit. on p. 53).
- (2016k), *Proximity Sensors*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/categories/84> (cit. on p. 54).
- (2016l), *QRD1114*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/246>.
- (2016m), *Temperature Sensors*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/categories/82> (cit. on p. 54).
- (2016n), *Thermocouple Type-K - Glass Braid Insulated (Bare Wire)*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/251> (cit. on p. 55).
- (2016o), *TMP102*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/11931>.
- (2016p), *TSL2561*. (retrieved 2017-10-01). URL: <https://www.sparkfun.com/products/12055> (cit. on p. 55).

- SRINIVASAN, S. (2014), *Security, Trust, and Regulatory Aspects of Cloud Computing in Business Environments*. First. IGI Global. ISBN: 9781466657885 (cit. on pp. 14, 15).
- STIFANI, R. (2015), *IBM Bluemix: The Cloud Platform for Creating and Delivering Applications*. IBM Redbooks. (retrieved 2017-06-01). URL: <http://www.redbooks.ibm.com/redpapers/pdfs/redp5242.pdf> (cit. on pp. 72, 73, 76, 77).
- TANG, C. (2015), *Explore MQTT and the Internet of Things service on IBM Bluemix*. (retrieved 2017-08-02). URL: <https://www.ibm.com/developerworks/cloud/library/cl-mqtt-bluemix-iot-node-red-app/> (cit. on p. 86).
- TECHBLOG, Squix (2015), *ESP8266 module comparison*. (retrieved 2017-10-01). URL: <https://blog.squix.org/2015/03/esp8266-module-comparison-esp-01-esp-05.html> (cit. on p. 35).
- TELEGRAM.ORG (2017), *Telegram API*. (retrieved 2017-07-01). URL: <https://core.telegram.org/api> (cit. on p. 87).
- TOZZI, C. (2016), *IoT Past and Present: The History of IoT, and Where It's Headed Today*. (retrieved 2016-14-11). URL: <http://talkincloud.com/cloud-computing/iot-past-and-present-history-iot-and-where-its-headed-today?page=1> (cit. on p. 8).
- VASSEUR, J.P; DUNKELS, A. (2010), *Interconnecting Smart Objects with IP - The Next Internet*. First. Elsevier. ISBN: 9780123751652 (cit. on pp. 11, 70).
- VERMESAN, O.; FRIESS, P. (2014), *Internet of Things - From Research and Innovation to Market Deployment*. First. River Publishers. ISBN: 9788793102941 (cit. on pp. 9, 19–22).
- WINKLER, F. (2014), *AD32600 - Arduino Workshop 02*. (retrieved 2017-21-11). URL: http://web.ics.purdue.edu/~fwinkler/AD32600_F14/AD32600_Arduino_workshop_02.pdf (cit. on p. 13).

List of Figures

1.1	FabLabs - Overview and worldwide distribution (Jan. 2017)	1
1.2	Thesis Approach	4
2.1	Basic Components of the IoT	7
2.2	Connected Devices to the Internet - Forecast	10
2.3	IoT Example Architecture - Key Components and Protocols	11
2.4	Objects “Things” of the IoT - Classification	12
2.5	IoT vs. M2M - Differences	17
2.6	M2M Applications & Market Size	18
2.7	Smart Environments & Applications	19
2.8	Smart Grid System - Overview	20
2.9	Cloud-Based Healthcare Monitoring System	21
2.10	Smart Home - Example	22
3.1	Sensor System Concept - Block Diagram	28
4.1	Arduino Product Range	31
4.2	Arduino Models & Ethernet Shield (Examples)	32
4.3	NodeMCU DevKit v1.0	34
4.4	Edison Breakout Board (left) & Arduino Edison Board (right)	38
4.5	LinkSprite pcDuino3 (left) & LinkSprite pcDuino3(B) (right)	40
4.6	BeagleBone Black Rev. C	41
5.1	Cloud Sensor System - Full Functional Overview	48
5.2	Arduino MEGA 2560 - Pin Definition	49
5.3	Arduino Ethernet Shield - Pin Definition	50
5.4	NodeMCU - Pin Definition	51
5.5	4-Channel I2C Level Shifter	58
5.6	I2C - 3.3 V Communication Test	59
5.7	Sensor Wiring Diagram	60

5.8	Stripboard - Sensor Connection	61
5.9	Sensor Mounting - Installed Mounting (left); HC-SR04 Mounting (middle); Reflector (right)	62
5.10	Photo Interrupter Mounting - Installed Mounting (left); Detailed View (right)	63
5.11	Arduino IDE - Overview	65
5.12	Arduino IDE - Preferences	66
5.13	Board Manager - ESP8266 Board support	66
5.14	Arduino IDE - Board Selection	67
5.15	Network Configuration - Arduino (right) & NodeMCU (left)	68
5.16	MQTT - Publish/Subscribe	69
5.17	MQTT - Arduino Code Configuration	70
5.18	JSON Message Format	71
5.19	Serial Monitor - Connection & Data Publishing	72
5.20	IBM Bluemix - Categories of Services	73
5.21	IBM Bluemix - Dashboard	74
5.22	Internet of Things Foundation - Registering a Device	75
5.23	Internet of Things Foundation - Device Overview	75
5.24	Internet of Things Foundation - Received Sensor Values	76
5.25	Cloudant NoSQL DB - Historical Sensor Data	77
5.26	IBM dashDB - Sensor Data Table	78
5.27	Node-RED - Flow Editor file	79
5.28	Node-RED - package.json File (left); Manually Added Nodes (right)	81
5.29	Sensor Installation - Arduino MEGA 2560, Ethernet Shield and Sensor Stripboard	82
5.30	Sensor Installation - Ultimaker 2 (1)	83
5.31	Sensor Installation - Ultimaker 2 (2)	84
5.32	Sensor Installation - Limit Switch)	84
5.33	NodeMCU - Housing (1)	85
5.34	NodeMCU - Housing (2	86
5.35	Telegram Bot - Command Overview	88
5.36	Telegram Bot - Command Overview	89
5.37	Ultimaker 2 Extended - Sensor Visualization	90
5.38	Sensor Visualization - Mobile View	91
5.39	OLED - Notification Display; Ultimaker 2 - Extended	92

List of Tables

- 4.1 Technical Specifications - Arduino Boards 33
- 4.2 Technical Specifications - ESP8266 Boards 35
- 4.3 Technical Specifications - Raspberry PI Family 37
- 4.4 Technical Specifications - Intel[®] Boards 39
- 4.5 Technical Specifications - Other Boards 42
- 4.6 Pairwise Comparison 45
- 4.7 Value Benefit Analysis (simplified extract) 46

- 5.1 Acceleration Sensors 53
- 5.2 Distance Sensors 54
- 5.3 Temperature/Humidity/Pressure/Altitude Sensors 54

List of Abbreviations

FabLab	Fabrication Laboratory
FDM	Fused Deposition Modeling
PLA	Polylactic Acid
ABS	Acrylonitrile Butadiene Styrene
IoT	Internet of Things
IoE	Internet of Everything
IIoT	Industrial Internet of Things
M2M	Machine to Machine
MIT	Massachusetts Institute of Technology
IDII	Interaction Design Institute Ivrea
LAN	Local Area Network
WAN	Wide Area Network
MAC	Media Access Control
DHCP	Dynamic Host Configuration Protocol
SSID	Service Set Identifier
TUG	TU Graz
I2C	Inter Integrated Circuit
SDA	Serial Data Line
SCL	Serial Clock Line

CNC	Computer Numeric Control
SQL	Structured Query Language
BaaS	Backend as a Service
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
NoSQL	No-Structured Query Language
HTTP	Hypertext Transfer Protocol
MQTT	Message Queue Telemetry Transport
QoS	Quality of Service
XMPP	Extensible Messaging and Presence Protocol
AMQP	Advanced Message Queuing Protocol
DDS	Data Distribution Service
WWW	World Wide Web
I/O	Input/Output
GPIO	General Purpose Input Output
JSON	JavaScript Object Notation
NPM	Node Package Manager
ROM	Read Only Memory
RAM	Random Access Memory
DRAM	Dynamic Random Access Memory
SDRAM	Synchronous Dynamic Random Access Memory
SoC	System on Chip
MCU	Microcontroller Unit

RPI	Raspberry PI
ARM	Advanced RISC Machines
SBC	Single Board Computer
PRU	Programmable Real-time Unit
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IC	Integrated Circuit
ICSP	In-Circuit Serial Programming
LDO	Low Dropout Regulator
PCB	Printed Circuit Board
OLED	Organic Light Emitting Diode
ULV	Ultra Low Power
RISC	Reduced Instruction Set Computer
SPI	Serial Peripheral Interface
ADC	Analog Digital Converter
UART	Universal Asynchronous Receiver Transmitter
CAN	Controller Area Network
PWM	Pulse Width Modulation
IDE	Integrated Development Environment
USB	Universal Serial Bus
TTL	Transistor Transistor Logic
SDK	Software Development Kit
API	Application Programming Interface
WiFi	Wireless Fidelity

RFID	Radio Frequency Identification
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	User Interface



















Appendix

A.1 Market Overview - IoT Dev. Boards

A.2 Value Benefit Analysis - IoT Dev. Boards

A.3 Node-RED Editor - Flow Overview

A.1 Market Overview - IoT Development Boards

	OVERVIEW: IoT BOARDS																		
	ESP 8266			ARDUINO						RASPBERRY PI					OTHERS		INTEL		
																			
NodeMCU DEVKIT (ESP-12)	WiFi Module (ESP-201)	WiFi Module (ESP-01)	Arduino UNO	Arduino DUE (ARM)	Arduino 101	Arduino YUN	Arduino ETHERNET	Arduino MEGA	Raspberry PI 3 Model B	Raspberry PI 2 Model B	Raspberry PI Zero	Raspberry PI Model A (1/1+)	Raspberry PI Model B (1/1+)	Beaglebone Black Rev C	pcDuino3(B) Dev Board	Intel® Edison (Arduino Kit)	Intel® Edison (Mini Breakout Board)	Intel® Galileo Gen 2	
Price	\$12	\$8,95	\$6,95	\$24,95	\$49,95	\$30,00	\$74,95	\$45,95	\$59,95	\$39,95	\$41,95	\$5,00	\$29,95	\$39,95	\$54,95	\$59,95	\$99,95	\$74,95	\$74,95
Digital I/O Pins	13	4	2	14	54	14	20	14	54	17	17	17	17 (Model 1+) 8 (Model 1)	17 (Model 1+) 8 (Model 1)	65	14	20	4	14
Analog I/O Pins	1	1	0	6	12	6	12	6	16	–	–	–	–	–	7	6	6	–	6
Microcontroller (MCU)	Tensilica L106 32-Bit	Tensilica L106 32-Bit	Tensilica L106 32-Bit	ATmega328	AT91SAM3X8E	Intel Curie	ATmega32U4	ATmega328	ATmega2560	–	–	–	–	–	2x PRU 32-Bit	–	32 Bit Intel Quark SoC X1000 @ 100 MHz (clocked down)	32 Bit Intel Quark SoC X1000 @ 100 MHz (clocked down)	32 Bit Intel Quark SoC X1000 @ 400 MHz
Flash Memory (MCU)	96 kB	96 kB	96 kB	32 kB	512 kB	196 kB	32 kB	32 kB	256 kB	–	–	–	–	–	8 kB	–	–	–	8 Mb / SD card
RAM (MCU)	64 kB	64 kB	64 kB	2 kB	96 kB	24 kB	2.5 kB	2 kB	8 kB	–	–	–	–	–	8 kB / 12 kB (shared)	–	512 kB SRAM	512 kB SRAM	512 kB SRAM
EEPROM (MCU)	–	–	–	1 kB	none	none	1 kB	1 kB	4 kB	–	–	–	–	–	–	–	8 kB	8 kB	8 kB
Clock Speed (MCU)	80 MHz	80 MHz	80 MHz	16 MHz	84 MHz	32 MHz	16 MHz	16 MHz	16 MHz	–	–	–	–	–	200 MHz	–	100 MHz	100 MHz	400 MHz
Microprocessor (MPU)	–	–	–	–	–	–	Atheros AR9331	–	–	BCM2837 Cortex-A53	BCM2836 ARMv7	BCM2835	BCM2835	BCM2835	AM3358 Cortex-A8	AllWinner A20 Cortex-A7	Intel Atom Z34xx @ 500 MHz	Intel Atom Z34xx @ 500 MHz	–
Flash Memory (MPU)	–	–	–	–	–	–	16 MB	–	–	microSD	microSD	microSD	microSD	microSD	4 GB eMMC	4 GB / microSD	1 GB	1 GB	–
RAM (MPU)	–	–	–	–	–	–	64 MB	–	–	1 GB (SDRAM)	1 GB (SDRAM)	512 MB (SDRAM)	512 MB (Model 1+) 256 MB (Model 1)	512 MB (SDRAM)	512 MB (SDRAM)	1 GB (SDRAM)	4 GB eMMC / SD card	4 GB eMMC / SD card	256 MB
EEPROM (MPU)	–	–	–	–	–	–	1 kB	–	–	–	–	–	–	–	–	–	–	–	–
Clock Speed (MPU)	–	–	–	–	–	–	16 MHz	–	–	1,2 GHz	900 MHz	1 GHz	700 MHz	700 MHz	1 GHz	1 GHz	500 MHz	500 MHz	–
Input Voltage (Power Jack)	4.5 - 9 V	3.3 V	3.3 V	7 - 12 V	7 - 12 V	7 - 12 V	5 V	7 - 12 V	7 - 12 V	5 V	5 V	5 V	5 V	5 V	–	5 V	7 - 15 V	–	7 - 15 V
Operating Voltage	3 - 3.6 V	3.3 V	3.3 V	5 V	3.3 V	3.3 V / 5 V	5 V	5 V	5 V	5 V	5 V	5 V	5 V	5 V	3.3 / 5 V	5 V	3.3 - 4.5 V	1.8 V	3.3V / 5.5 V
I/O Pin Current	12 mA	12 mA	12 mA	20 mA	800 mA	20 mA	40 mA	40 mA	20 mA	50 mA	50 mA	50 mA	50 mA	50 mA	–	–	80 mA	80 mA	80 mA
Length/Width	49 x 24.5 mm	35 x 25 mm	25 x 14.5 mm	68.6 x 53.4 mm	101.52 x 53.3 mm	68.6 x 53.4 mm	68.6 x 53.4 mm	68.6 x 53.3 mm	101.52 x 53.3 mm	85 x 56 mm	85 x 56 mm	65 x 30 mm	85 x 56 mm	85 x 56 mm	86.44 x 54.54 mm	121 x 65 mm	127 x 72 mm	61 x 29 mm	124 x 72 mm
Connectivity	micro USB	GPIO (TX, RX)	GPIO (TX, RX)	USB Type B Power Jack	2x micro USB Power Jack	USB Type B Power Jack	USB Type B Power Jack Ethernet 10/100	6-PIN SPI Power Jack Ethernet 10/100 microSD Slot	USB Type B Power Jack	4x USB Type A HDMI 3.5 mm Audio Ethernet 10/100 microSD Slot micro USB WiFi	4x USB Type A HDMI 3.5 mm Audio Ethernet 10/100 microSD Slot micro USB	microSD Slot miniHDMI 2x microUSB	1x USB Type A HDMI 3.5 mm Audio	2x USB Type A HDMI Ethernet 10/100 3.5 mm Audio	USB Type A microHDMI Ethernet 10/100 Power Jack microHDMI	USB Type A HDMI Ethernet 10/100/1000 3.5 mm Audio SATA socket	2x USB Type A microUSB microSD Slot Power Jack	2x micro USB battery re-charger	USB Type A, micro USB Ethernet Power Jack microSD
Low-level peripherals	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART, CAN	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, ADC, PWM, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, I2S, UART	SPI, I2C, UART, Timers, CAN, ADC	SPI, I2C, ADC, UART, PWM, ADC	SPI, I2C, I2S, UART, PWM, ADC, PCIe	SPI, I2C, I2S, UART, PWM, ADC, PCIe	SPI, I2C, I2S, UART, PWM, ADC, PCIe
Programming	LUA, Arduino IDE	LUA, Arduino IDE	LUA, Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Arduino IDE	Linux (Python), C, C++, Java, Scratch, Ruby	Linux (Python), C, C++, Java, Scratch, Ruby	Linux (Python), C, C++, Java, Scratch, Ruby	Linux (Python), C, C++, Java, Scratch, Ruby	Linux (Python), C, C++, Java, Scratch, Ruby	Linux (Python)	Linux (Python) Android (Java)	Linux Arduino IDE	Linux Arduino IDE	Linux Arduino IDE
Real-Time Capability	Yes	Yes	Yes	Yes	Yes	Yes	Yes (MCU)	Yes	Yes	No	No	No	No	No	Yes (Via MCU)	No	Yes (Via MCU)	Yes (Via MCU)	Yes
Features	WiFi 802.11 b/g/n Deep sleep power < 10 uA	WiFi 802.11 b/g/n Deep sleep power < 10 uA	WiFi 802.11 b/g/n Deep sleep power < 10 uA	–	–	Bluetooth 6-Axis accelero./gyro	MCU + MPU WiFi IEE 802.11b/g/n	–	–	BCM43143 WiFi on board	–	–	–	–	–	WiFi Arduino shield compatible	WiFi Bluetooth 4.0 Arduino shield compatible	Battery Powered Supply WiFi Bluetooth 4.0	Arduino shield compatible

A.2 Value Benefit Analysis - IoT Development Boards

Weighted Decision Matrix - IoT Sensor System															
Decision Matrix			Arduino UNO	Arduino 101	Arduino MEGA	Arduino ETHERNET	Arduino YUN	NodeMCU DEVKIT (ESP-12)	WiFi Module (ESP-01)	Raspberry PI 2 Model B	Raspberry PI 3 Model B	Intel® Galileo Gen 2	Intel® Edison (Arduino Kit)	pcDuino3(B) Dev. Board	Beaglebone Black Rev. C
Criteria	Explanation	Weight													
Cost Factor	Initial costs for the development platform	11,4%	3	3	1	3	3	5	5	3	3	3	2	3	3
GPIO Capability (digital + analog)	Amount of digital and analog input/output ports that the hardware provides	15,9%	2	2	5	3	3	2	1	2	2	2	3	2	3
Peripherals (low-level)	Provided protocols such as I2C, Two-wire, SPI or ADC	11,4%	3	3	3	3	3	3	3	2	2	3	3	3	4
Hardware	Hardware architecture and performance (MCU/MPU), flash/program memory size	15,9%	1	3	3	1	1	3	3	4	4	3	4	4	1
Speed (low-level tasks)	Speed regarding low level tasks (reading analog/digital inputs) & real-time capability	11,4%	3	3	3	3	3	2	2	1	1	3	2	1	3
Features (WiFi, Ethernet, Bluetooth)	Integrated network capability (WiFi, Ethernet) on-board, otherwise additional extension shields are required	11,4%	1	1	1	2	3	2	2	2	3	2	2	2	2
Integrated Development Environment (IDE)	e.g. Arduino IDE, Python	3,8%	2	2	2	2	2	2	2	3	3	2	2	4	3
Coding Skills	Needed skills (e.g. python, C/C++, Linux skills needed for RPI) for programming/using the device	3,8%	2	2	2	2	2	2	2	1	1	2	2	1	1
Documentation	Availability, amount and quality of existing documentation	3,8%	3	3	3	3	3	3	3	3	3	2	2	1	1
Community (IoT)	How good is the support/documentation for integration into IBM Bluemix? (Receipts, FAQs)	3,8%	4	4	4	4	4	4	4	3	3	2	2	1	1
Usability	How convenient is the implementation of sensors/actuators (e.g. availability of additional software libraries)	3,8%	3	3	3	3	3	2	2	2	2	2	2	2	2
Expandability	e.g. breadboards, prototyping plates availability, sensors and additional equipment	3,8%	3	3	3	3	3	1	1	2	2	2	2	2	1
Weighted Scores		100,0%	2,26	2,58	2,83	2,53	2,64	2,69	2,53	2,39	2,51	2,50	2,59	2,39	2,34

A.3 Node-RED Editor - Flow Overview

