

Hochschule Karlsruhe – Technik und Wirtschaft
Fakultät für Maschinenbau und Mechatronik

**Flexible clock synchronization for a knowledge
based reasoning system.
Structure of Master thesis.**

4. Zeile

*Masterthesis (M.Sc.)

von

*Fernando Ivan Sanchez Coria

Hochschule Karlsruhe – Technik und Wirtschaft
Fakultät für Maschinenbau und Mechatronik

**Flexible clock synchronization for a knowledge based
reasoning system.
Structure of Master thesis.**

4. Zeile

*** V E R T R A U L I C H ***

*Masterthesis (M.Sc.)
Nr. *B 17/999

von

*Fernando Ivan Sanchez Coria
geb. am *29.12.1993
in *Guadalajarara, Mexico
Matr.-Nr.: *64357

Betreuer der Firma *Hochschule Karlsruhe - Technik und Wirtschaft
*1. Prof. Dr.-Ing.habil Catherina Burghart

Betreuer der Hochschule Karlsruhe
*2. Hochschule

*Karlsruhe, *27. 03. 2018 bis 27. 08. 2018

The following information is optional (!)

Satz und Herstellung:

L^AT_EX und KomaSkript / MiKTeX 2.9, TeXnicCenter 2.02

Font: Computer Modern, 11 pt

Druckdatum: 25. August 2018

Erklärung

Ich versichere hiermit wahrheitsgemäß, die Abschlussarbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles einzeln kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Falls im Rahmen einer gemeinsamen Bearbeitung eines Themas mehrere gleichlautende Fassungen abgegeben werden, ist der nachfolgende Satz einzufügen, ansonsten ist er wegzulassen:

Das Thema der vorgelegten Arbeit wurde gemeinsam mit Herrn/Frau (...) (Bachelor- und Masterarbeit Nr. (...)) bearbeitet.

Karlsruhe, den 25. August 2018

Unterschrift:

S p e r r v e r m e r k

Diese Arbeit enthält vertrauliche (oder: geheime) Informationen. Veröffentlichungen insbesondere vor dem (*31.03.2013) bedürfen der schriftlichen Genehmigung der (Organisation).

Nomenclature¹

Latin formula symbols

c_{lim}	-	Konstante des Turbulenzmodells
g_i	m/s^2	Vektor der Schwerkraft
k	m^2/s^2	turbulente kinetische Energie
l	m	Längenmaß
L	m	Characteristic length
\dot{m}	kg/s	Mass flow
p	N/m^2	Druck
R	-	Residuenmatrix
\mathbf{R}	J/(mol K)	Universelle Gaskonstante
s	J/(kg K)	spezifische Entropie
S	J/K	Entropie
S_{ij}	1/s	shear rate
t	s	Time
T	K	Temperature
u, v, w	m/s	velocity's component

Griechische Formelzeichen

α	-	General state variable
β^*	-	Konstante des Turbulenzmodells
η	kg/(m s)	dynamische Viskosität
η	W/W	Efficiency
ϑ	°C	Temperature
ν	m^2/s	kinematische Viskosität
κ, κ	-	Isentropenexponent
ξ	kg/kg	Massenanteil
ρ	kg/m^3	Dichte
σ_i	-	Konstante des Turbulenzmodells
τ_{ij}	N/m^2	Spannungstensor
φ	-	relative Feuchte
Φ	-	allgemeine Konstante des Turbulenzmodells
ω	1/s	Frequency of turbulent fluctuation

¹Example here; otherwise only (all) actually used characters are listed!
when sorting: alphabetically, first small, then large letters.

Indizes

bulk	mittlere Geschwindigkeit
i	Richtungsindex, Spezies
j	Summationsindex, Element
n	Zeitschritt
t	turbulent, total, tangential
U	Umgebung

Besondere Zeichen

d	steiles d: vollständiges Differenzial
\bar{d}	palatales d: unvollständiges Differential, alternativ δ
∂	partieller Differentialoperator
Δ	Difference
$:=$	definiert durch
\equiv	identisch
\propto	proportional
\approx	etwa
\dot{x}	Strom von x
\bar{x}	Mittelwert von x , oder molare Größe
\hat{x}	Amplitude von x
(1.1)	Gleichungsnummer, die erste Zahl gibt die Nummer des Kapitels an, die zweite Zahl ist fortlaufend im Kapitel
[12]	Nummer im Quellenverzeichnis

Dimensionslose Kennzahlen

$Re := wL/\nu$	Reynolds-Zahl
----------------	---------------

Abkürzungen²

HsKA	Hochschule Karlsruhe – Technik und Wirtschaft
IKKU	Institut für Kälte-, Klima- und Umwelttechnik
IMP	Institute of Materials and Processes
MMT	Maschinenbau und Mechatronik
R-134a	1,1,1,2-Tetrafluorethan (Kältemittel)
RKS	Redlich-Kwong-Soave

²Abkürzungen, die bereits im Duden stehen, werden nicht aufgeführt.

Inhaltsverzeichnis

Nomenclature	v
0.1 Sense of the thesis	1
0.2 Textverarbeitung	1
1 Statutory declaration	3
2 Blocking notice	5
3 Introduction	7
3.1 Motivation	7
3.2 Goals	7
3.3 Keywords	8
3.4 Summary	8
4 Related work	9
4.1 Identification of the needs	9
4.2 Evaluation and solution adopted	9
4.3 Need for establishment	10
4.4 Algorithm	10
4.4.1 Synchronization Algorithm	10
4.5 Summary	10
5 Theoretical background	11
5.1 State of the art	11
5.1.1 Knowledge representation and reasoning (KR)	11
5.1.2 Knowledge representation	11
5.1.3 Knowledge related concepts	12
5.1.4 Artificial intelligence	12
5.1.5 Knowledge-Based Systems	13
5.1.6 Knowledge representation	13
5.1.7 Clock Synchronization Techniques for Distributed Systems	13
5.1.8 Clock synchronization algorithm	13
5.1.9 A simple taxonomy	13
5.2 Differentiation with another temperature control systems	14
5.3 Knowledge representation	14
5.3.1 History	14
5.3.2 Characteristics	15
5.4 Subject-object problem	15
5.5 Intelligent Agents	15
5.6 Used techniques	16
5.7 Used techniques	16
5.7.1 Characteristics of Artificial Intelligence Problems	16

5.7.2	Characteristics of AI agents	16
5.8	Problems in knowledge representation	17
5.9	Knowledge representation	17
5.10	Expert systems (ES)	20
5.11	Expectations	20
5.12	Clock Synchronization Techniques for Distributed Systems	20
5.12.1	Benefits	21
5.12.2	Applications	21
5.12.3	The Technology	21
5.12.4	Characteristics	21
5.13	Clock Synchronization in CAN Distributed Embedded Systems	22
5.14	System model	22
5.15	Phases of clock synchronization	23
5.15.1	A simple taxonomy	24
5.16	1st phase Software detection vs. hardware detection	24
5.16.1	2nd phase Symmetric vs. asymmetric scheme	26
5.16.2	Advantages of symmetric scheme	26
5.16.3	Fixing the suitability requirements	26
5.17	The Design of a Clock Synchronization Subsystem for Parallel Embedded Systems	27
5.18	Clock Synchronization in Distributed System	29
5.19	Clocks in distributed systems (concepts and implementation)	29
5.19.1	Command ordering	30
5.19.2	Clients and leader	31
5.19.3	Executing a command	31
5.19.4	Opportunity: faster reads	31
5.19.5	What could go wrong?	31
5.20	Synchronized distributed clocks	34
5.21	Physical clock	34
5.22	problems	34
5.23	Distributed algorithms	35
5.24	Cristian's algorithm	35
5.25	Logical clocks	36
5.26	Lamport Timestands	36
5.27	'Happened before' relation	36
5.28	Lamport's algorithm corrects the clocks	37
5.28.1	Hardware Clocks	37
5.28.2	Software Clocks	37
5.28.3	Main Components of Clock Synchronization Algorithm	39
5.28.4	Algorithms classification	39
5.28.5	Network Time Protocol's Goals and Definitions	39
5.29	Environment for a Clock synchronized system	40
5.29.1	1st circuit to implement: Temperature Controller using Arduino	40
5.29.2	2nd circuit to implement: Servomotor Controller	40
5.29.3	Regular Ethernet (Beaglebone book)	41
6	Implementation	43
6.1	Intstalling compiler	43
6.1.1	Boost library	43

6.2	Socket: End point of communication	44
6.2.1	Flowchart diagram: Server	44
6.2.2	Flowchart diagram: Client	45
6.2.3	Synchronous timer	45
6.3	Communication Jetson TX2-Arduino over Serial port	57
6.4	Installing Arduino IDE compiler	58
6.5	Communication Jetson TX2-Arduino over ethernet	59
6.5.1	Servomotor control with Arduino ethernet shield	60
6.6	Temperature controller with Arduino ethernet shield	66
6.7	Clock Synchronization algorithm in Webserver	73
6.8	Clock Synchronized embedded system's functioning diagram	84
7	Experiments and results	85
7.1	Socket chat server	85
7.2	Socket with a Clock synchronization algorithm on Linux enviroment	87
7.3	Webserver with Servomotor controller	87
7.4	Webserver with Temperature controller	90
7.5	Clock synchronization algorithm on Webserver	90
8	Conclusions	95
	Bibliography	96
	Tabellenverzeichnis	97
	Abbildungsverzeichnis	101
	Stichwortverzeichnis	103
.1	Bildanhang	104

0.1 Sense of the thesis

At the end of the study, the prospective engineers and the Thesis show that they are capable to solve and edit scientific problems in a reasonable time independently. In the written elaboration of the Thesis should cover all key findings in a concise, clear and perfectly understandable for an engineer.

0.2 Textverarbeitung

MS-Office (Word), SoftmakerOffice, LibreOffice (Writer) oder doch lieber L^AT_EX?

Kurzfassung

**Flexible Uhr-Synchronization für ein wissensbasiertes
Entscheidungssystem**

1 Statutory declaration

I hereby declare that I have written the present work independently and have used no other than the specified sources and resources. Karlsruhe, 27.08.2018

Fernando Iván Sanchez Coria.

NAME SIGNATURE TITLE

Bild 1.1. Signature.

2 Blocking notice

The present work contains internal and confidential information of the Faculty of Mechanical Engineering and Mechatronics (MMT) of the Karlsruhe University of Applied Sciences (HS-KA). The transfer of content, work in whole or in part, as well as making copies or transcripts - even in digital form - are strictly prohibited. Exceptions require the written permission of the University of Applied Sciences Karlsruhe - Technology and Economics.

3 Introduction

A fundamental criterion in the design of a robust distributed system is to provide the capability of tolerating and potentially recovering from failures that are not predictable. (failures require incorporating tolerances to various faults). The patented protocols ensure that all nodes reach the same decisions regardless of faulty components. The systems do not rely on any assumptions about the initial state of the system and do not require a central clock or externally generated pulse. The protocol, or algorithm, allows for self-stabilization of nodes within a distributed system. Distributed synchronous systems that are required to provide globally coordinated operations require each component (node) in the system to be precisely synchronized. These protocols provide distributed autonomous synchronization (i.e. no master clock signal required) and do not rely on any assumptions regarding the initial state of the system or internal status of the nodes. Over time has been the questioning of the different actions that can be programmed in a microcontroller, this represents an expenditure of time necessary, within a mechatronic system you can work with one or more microcontrollers for the design of a control plant for a factory, company or a social environment. In this research the main point of the investigation will be the functioning of different electronic circuits to perform different synchronized actions in a close loop of time.

3.1 Motivation

Clock synchronized Embedded systems are known as devices implemented in Mechatronic projects where two or more devices reach the same state of time in an approximate amount of time, they share information and they are capable of react in parallel. In this section this main topic is dependent to another different methods, structures, concepts and definitions. Software tools implemented in this project and Designed software interfaces will play a significant role in all the research about Clock Synchronization Algorithms theory previously researched.

3.2 Goals

General: Generate the synchronization of two microcontrollers with different programming actions for the control of a refrigerated sustainable environment.

- Design a temperature controller which can detect the conditions in a favorable and not favorable refrigerated environment.
- Design the servomotor controller which will perform different actions within the control plant and help to access to different conditions of temperature through opening or closing gates.

- Design a software control interface which can access the different functions of the two designed controllers.
- Design and obtain the synchronization of these three microcontrollers through the theory of programming of embedded systems.
- Data exchange between microcontrollers.

3.3 Keywords

Design, Control, Synchronization, Embedded systems, Distributed systems, Microcontrollers, Programming, structure, software, Web interface, cost-effectiveness, automation, control plant, refrigeration, sustainable.

3.4 Summary

The system is defined as a Synchronized based reasoning system (control plant) made of different sensors of which the only ones used will be the temperature controller, in addition there is one servomotor included in the plant that will be responsible for keeping the refrigeration space close / open depending of the needs. The microcontrollers included in this system will be responsible for providing the control loop by means of the included programming based on the theory of Clock synchronized embedded systems. This control plant is controlled by means of a control interface designed by software, to the user has direct contact with the elements integrated into the system. The device responsible for being the processor of all this control loop, for this 'Mechatronic system' is a Jetson TX2 that is a Embedded system (development board) based on an open source system (Linux).

4 Related work

Clock synchronized Embedded systems are known as devices implemented in Mechatronic projects where two or more devices reach the same state of time in an approximate range of time and exchange data themselves. This main topic is dependent to another different methods, structures, concepts and definitions. Software tools implemented in this project and Designed software interfaces will play a significant role in all the research about Clock Synchronization Algorithms theory previously researched.

4.1 Identification of the needs

Many times, there is a need to have varied materials or devices in refrigeration which, due to their physical or chemical structure cannot have contact for a long time in an ambient temperature, it is important to design a prototype of a refrigeration plant which will allow keep materials or specific devices in an appropriate environment. The use of one or more synchronized microcontrollers is a frequent necessity that sometimes it is required to perform different programmed actions in different devices which have a common purpose in a close loop of time, the reaction time of these microcontrollers must be often immediate, and the time is vital for the conservation of different devices and / or materials.

- Different programmed actions in a close loop of time.
- Detect optimum temperature in a closed environment.
- Have different actuators in operation to open and close doors and / or windows within a closed environment.
- An interface through software which allows us to interact with a synchronized embedded system.
- Timing the correct temporal of events.

4.2 Evaluation and solution adopted

An analysis was carried out to decide the project in which we worked during the present semester, concluding that Flexible clock synchronization would be the solution to this type of needs and many others in which time is a key factor for the actions of microcontrollers in the same mechatronic system. The project was defined as Flexible clock synchronization for a knowledge based reasoning system.

4.3 Need for establishment

Normally in different companies, environments and factories it is required that different products or devices have an adequate conservation due to the destination that they have, for this the refrigeration is an important part of the conservation since it allows to have an environment with lower temperature inside of a closed installation. These cooling areas are quite expensive and require many energy and economic resources, the proposal of this project is to have an automated and sustainable system which allows us to carry out the temperature control in a closed space. For these devices due of their low price and the effortless way to get them they will be the pillars of the design controllers implemented, they will be programmed and synchronized to work together in a close loop time.

4.4 Algorithm

A clock synchronization algorithm must assume a suitable fault model and must guarantee precision and accuracy of the clock in the desired range even in the presence of the faults. Clock synchronization algorithms can be evaluated with respect to only three basic requirements: tightness, reliability and cost-effectiveness. (where tightness refers to a combination of precision and accuracy). Every dimension of the system indicates the "quality" of the clock synchronization algorithm with respect to one of those requirements.

4.4.1 Synchronization Algorithm

IEEE 1588 is a Standard for Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. It is also known as Precision Time Protocol [PTP] and is used for time/clock synchronization in packet-based networks that support multicasting.

4.5 Summary

The existence of a global notion of time brings significant benefits for a distributed system, as it allows the nodes to agree on the time at which events occurred as well as to agree on the time to trigger actions. The mathematical models of the theory of synchronization will be a fundamental part of this project as well as the notion of the internal operation of the microcontroller, this in order to understand how the programming structure will be.

5 Theoretical background

In these times, many technologies are based on embedded systems which process necessary amounts of information and convert them into signals which through digital outputs of microcontrollers are responsible for causing a determined action. Microcontrollers are necessary for certain projects because they are very simple systems to manipulate within a mechatronic system and it's very simple to include them in a control system. The implementation of the programming in microcontrollers help us to develop projects of great complexity and to provide solutions within an electronic system in which it is required to perform a certain action at certain periods of time.

For this, a synchronized system has been proposed which can control two microcontrollers in order to obtain an almost immediate reaction inside a mechatronic system, in which the frequency of time is a differential factor, inside of these systems it is required that one or more microcontrollers work together and reach each one its state of time, this project will promote the bases of a sustainable and automated Clock synchronized embedded system.

5.1 State of the art

5.1.1 Knowledge representation and reasoning (KR)

Is the field of artificial intelligence (AI) dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in a natural language, that will make complex systems easier to design and build, These systems featured data structures for planning and decomposition.

5.1.2 Knowledge representation

1. Incorporates findings from psychology, about how humans solve problems and represent knowledge in order to design formalisms. 2. Incorporates findings from logic to automate various kinds of reasoning, such as the application of rules or the relations of sets and subsets.

Examples of knowledge representation formalisms include sem antic nets, systems architecture, frames, rules, and ontologies. Examples of reasoning engines include inference engines, theorem provers, and classifiers.

Knowledge representation makes complex software easier to define and maintain than procedural code and can be used in expert systems.

5.1.3 Knowledge related concepts

- Knowledge (and Memory).
- Knowledge: To know. Humans deal with knowledge of many kinds. We have models of the world living in. We have models of ourselves in the world. We have knowledge of society, knowledge of facts, knowledge of how to do things etc.
- Knowledge based systems: Usually refers to the systems that employ domain problem solving knowledge in some form.
- Semiotics: A symbol is something that stands for something else.

Examples: The “number” seven can be represented in many different ways. Road signs-curves, pedestrians, schools, U-turns, eating places, etc. All programming languages are semiotic systems (Language C, C++, Java etc).

5.1.4 Artificial intelligence

Artificial intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans. Some of the activities computers with artificial intelligence are designed for include:

- Speech recognition.
- Learning.
- Planning.
- Problem solving.

Research associated with artificial intelligence is highly technical and specialized. The core problems of artificial intelligence include programming computers for certain traits such as:

- Knowledge.
- Reasoning.
- Problem solving.
- Perception.
- Learning.
- Planning.
- Ability to manipulate and move objects.

5.1.5 Knowledge-Based Systems

Knowledge-Based Systems (KBS) are based on the interaction between two major components: the inference engine and the knowledge base. The overall system acts as a problem-solver in a more or less delimited application domain. The inference engine implements a reasoning model and makes use of the knowledge base to explicit solutions to problems. The knowledge base includes relevant pieces of information (e.g.: rules, object descriptions) in an application domain and declarative pieces of knowledge such as rules, object descriptions and facts, relevant to the problem domain. The importance of knowledge in KBS, is the representation of points of view (knowledge representation language) and a methodological point of view (e.g.: knowledge elicitation, man-machine interfaces).

5.1.6 Knowledge representation

It is a subarea of artificial intelligence concerned with understanding, designing, and implementing ways of representing information in computers, programs can use this information to derive information that is implied by it, converse with people in natural languages, to plan future activities and solve problems in areas that normally require human expertise.

5.1.7 Clock Synchronization Techniques for Distributed Systems

Distributed synchronous systems that are required to provide globally coordinated operations require each component (node) in the system to be precisely synchronized. These protocols provide distributed autonomous synchronization (i.e. no master clock signal required) and do not rely on any assumptions regarding the initial state of the system or internal status of the nodes. The existence of a global notion of time brings significant benefits for a distributed system, as it allows the nodes to agree on the time at which events occurred as well as to agree on the time to trigger actions.

5.1.8 Clock synchronization algorithm

A clock synchronization algorithm must assume a suitable fault model and must guarantee precision and accuracy of the clock in the desired range even in the presence of the faults.

5.1.9 A simple taxonomy

Clock synchronization algorithms can be evaluated with respect to only three basic requirements: tightness, reliability and cost-effectiveness. (where tightness refers to a combination of precision and accuracy). every dimension of the system indicates the "quality" of the clock synchronization algorithm with respect to one of those requirements.

5.2 Differentiation with another temperature control systems

There are many temperature control systems but most of them don't include synchronization of one or more microcontrollers, this will be done based on the theory of 'Clock Synchronization in CAN Distributed Embedded Systems' which promotes this type of systems and requires some theoretical understanding for its future implementation.

The mathematical models of the theory of synchronization will be a fundamental part of this project as well as the notion of the internal operation of the microcontroller, this in order to understand how the programming structure will be.

5.3 Knowledge representation

Knowledge representation is based on First Order Logic (FOL). Languages which do not have the complete formal power of FOL can still provide close to the same expressive power with a user interface that is more practical for the average developer to understand.

Randall Davis MIT outlined five distinct roles to analyze a knowledge representation. • (KR) is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting. • In what terms should I think about the world? • It is a fragmentary theory of intelligent reasoning. • It is a medium for pragmatically efficient computation.

5.3.1 History

The earliest work in computerized knowledge representation was focused on general problem solvers such as the General Problem Solver (GPS) system developed by Allen Newell and Herbert A. Simon in 1959.

Expert systems gave us the terminology still in use today where AI systems are divided into a **Knowledge Base** with facts about the world and **rules and an inference engine** that applies the rules to the knowledge base in order to answer questions and solve problems.

One of the most powerful and well known was the 1983 Knowledge Engineering Environment (KEE) from Intellicorp. KEE had a complete rule engine with forward and backward chaining. It also had a complete frame-based knowledge base with triggers, slots (data values), inheritance, and message passing. (Companies involved Symbolic, Xerox, and Texas Instruments). One of the first realizations learned from trying to make software that can function with human natural language

Along with J.C. Shaw (1922-1992), also from RAND, they had already developed a program called the Logic Theorist (LT). "It was the first program deliberately engineered to mimic the problem solving skills of a human being.

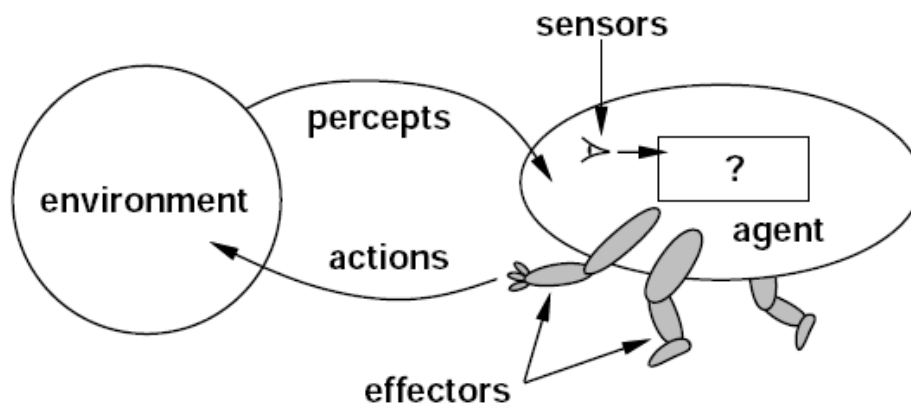


Bild 5.1. Intelligents agent's structure.

5.3.2 Characteristics

- Meta-representation. The issue of reflection in computer science. It refers to the capability of a formalism to have access to information about its own state.
- Incompleteness. additional axioms and constraints to deal with the real world as opposed to the world of mathematics. The ability to associate certainty factors with rules and conclusions. this area is known as fuzzy logic.
- Expressive adequacy. The standard that Brachman and most AI researchers used to measure expressive adequacy is usually First Order Logic (FOL).
- Reasoning efficiency. Run time efficiency of the system. The ability of the knowledge base to be updated and the reasoner to develop new inferences in a reasonable period of time.

5.4 Subject-object problem

The world as we know it. The subject-object problem, a longstanding philosophical issue, is concerned with the analysis of human experience, and arises from the premise that the world consists of objects (entities) which are perceived of otherwise presumed to exist as entities, by subjects (observers).

5.5 Intelligent Agents

One of the most common ways to understand how the structure of an embedded system is inside of a 'Mechatronic system' could be explained in next ►Bild 5.1

And how this information can be processed inside of an embedded system to have an output signal is well explained in ►Bild 5.2

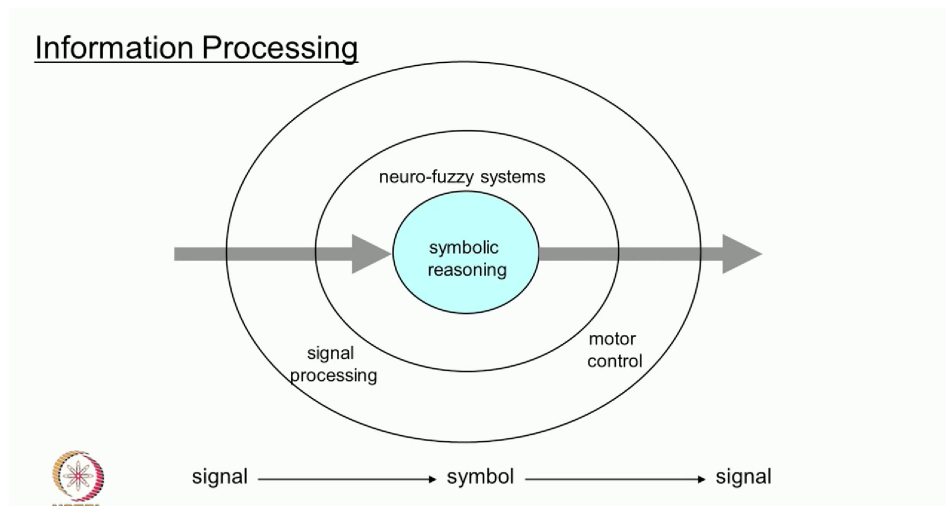


Bild 5.2. Information processing.

5.6 Used techniques

Knowledge engineering is a core part of Artificial Intelligence research. Machines can often act and react like humans only if they have abundant information relating to the world. Initiating common sense, reasoning and problem-solving power in machines is a difficult and tedious task. Machine learning is also a core part of AI. Learning without any kind of supervision requires an ability to identify patterns in streams of inputs, whereas learning with adequate supervision involves classification and numerical regressions. Mathematical analysis of machine learning algorithms and their performance is a well-defined branch of theoretical computer science often referred to as computational learning theory. It is defined artificial intelligence as an “Ability” of a machine to perform tasks that require human judgement, learning or problem-solving skills.

5.7 Used techniques

Robotic Process automation (RPA). Natural Language Generation (NLG). Natural language processing (NLP). Speech recognition. Virtual agents (advance chat bots). Machine learning.

5.7.1 Characteristics of Artificial Intelligence Problems

- Knowledge often arrives incrementally.
- Problems have multiple levels of granularity.
- Many problems are computationally intractable.

5.7.2 Characteristics of AI agents

- Agents have limited computing power.
- Agents have limited sensors.
- Agents have limited attention.
- Computational logic is fundamentally deductive.

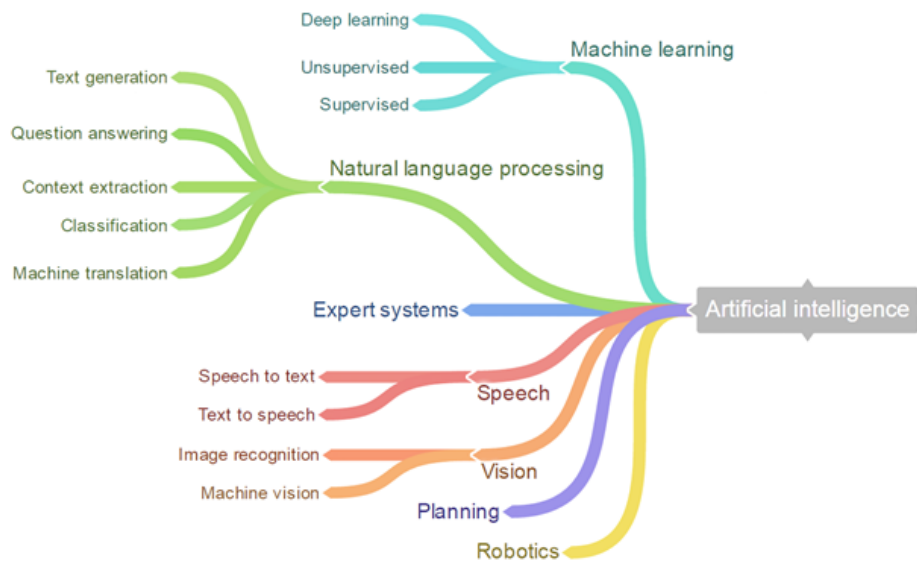


Bild 5.3. AI parts.

This kind of Artificial intelligence's knowledge is considered a 'Cognitive system' in every sense and can be explained with ►Bild 5.4

There are some examples about artificial intelligence referred to all its fields and ways of interpreting, see this in ►Bild 5.5

The Topics which are related to 'artificial intelligence' can be named as following in ►Bild 5.6, some of these fields are also related to these kinds of projects where some of these topics are applied.

5.8 Problems in knowledge representation

One of the principal unsolved problems in knowledge representation is how to provide expert systems and natural language processing systems with more world knowledge, particularly "common sense" knowledge, in order to make them more robust. AI began as a set of largely unrelated activities in areas such as game playing, theorem proving, robotics, natural language understanding, expert reasoning, machine vision, and other fields. AI programs and early expert systems used rules as their knowledge bases (KB). A KB is the body of subject knowledge that supports the performance of a "knowledge-based system, such as an expert or natural language processing system.

5.9 Knowledge representation

The Knowledge base (KB) of rule-based system consist of:

Facts (predicates).

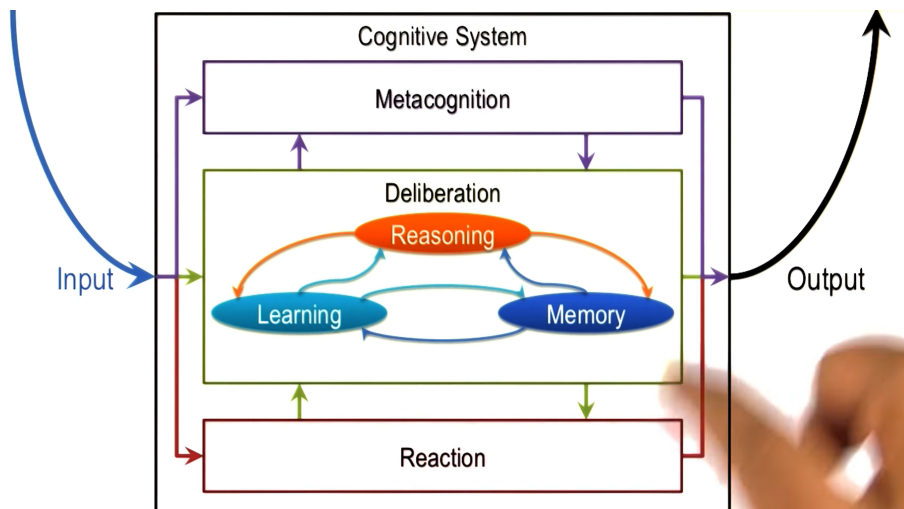


Bild 5.4. Cognitive system.

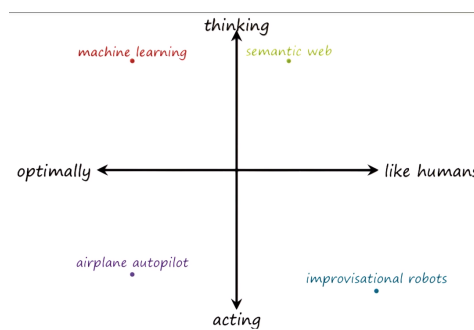


Bild 5.5. AI Quiz.

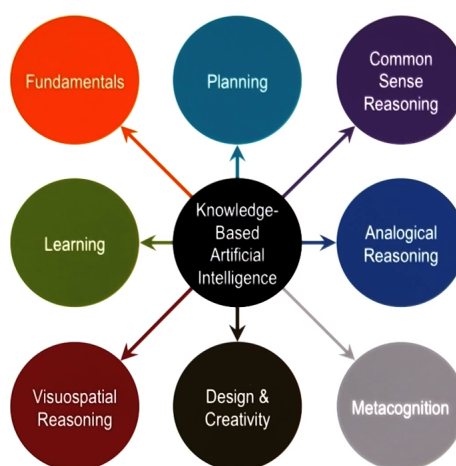


Bild 5.6. Topics KB.

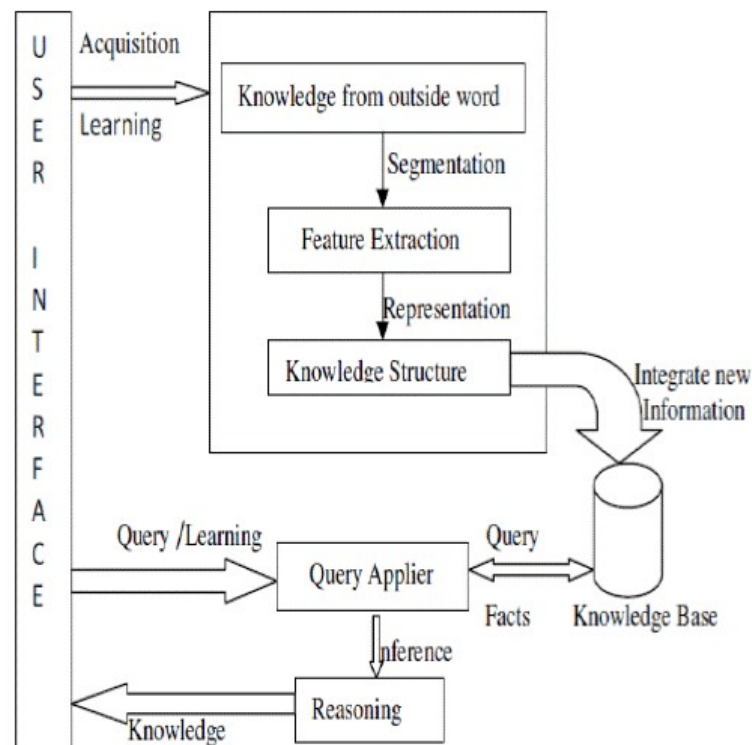


Bild 5.7. Knowledge base system model

- Declarative knowledge about the given problem.
- Statements with true or false values.
- Values can change in time and running reasoning.

Rules (conditional statements) Represent heuristics or “rules of thumb”

- Modeling human’s thinking.
- Describing expert’s knowledge (heuristics).
- Specify actions taking in a given situation.

Meta-rules (rules about how to use rules).

Knowledge base is also referred as one the responsible parts to create a ‘User interface’ architecture, see this in ►Bild 5.7 ,

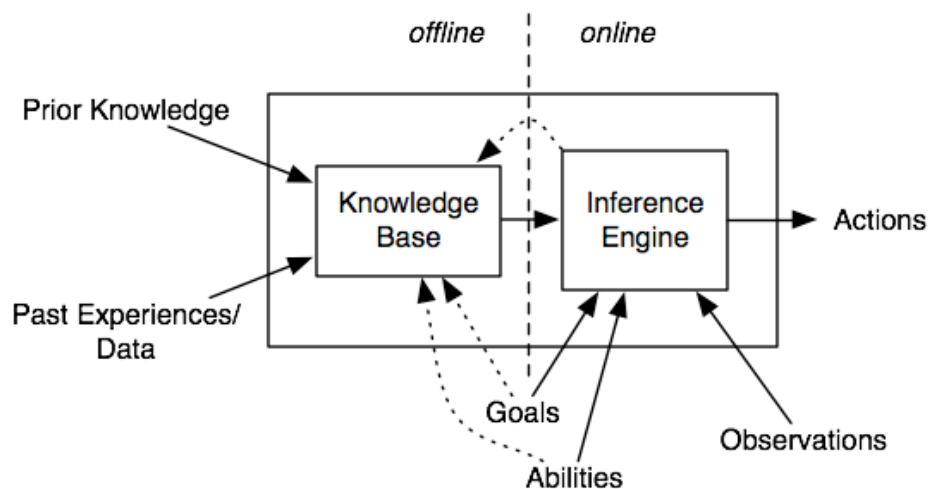


Bild 5.8. Refinement for knowledge-based agent.

5.10 Expert systems (ES)

Expert systems have been developed in many problem areas, including diagnosis, prediction, analysis, configuration, design, planning. An Expert system is characterized by the facts that a large subset of its knowledge base derives from experience and that this subset is largely responsible for the efficiency of the overall system.

The study and the design of Expert Systems may be regarded as an engineering branch in Artificial Intelligence (AI).

5.11 Expectations

Knowledge based systems results from expectations. KBS can be more versatile than traditional computer systems because their declarative knowledge can be used by the inference engine for several different purposes, including purposes that have not been explicitly anticipated when the knowledge was described. KBS can be easier to build, to modify, and to maintain because declarative knowledge can be decomposed into small stand-alone pieces (e.g. rules, frames), which can be stated independently from each other. KBS can provide more appropriate man-machine interaction because they can explain their behavior directly in terms of the pieces of knowledge which they apply.

5.12 Clock Synchronization Techniques for Distributed Systems

These protocols provide distributed autonomous synchronization (i.e. no master clock signal required) and do not rely on any assumptions regarding the initial state of the system or internal status of the nodes.

5.12.1 Benefits

- Distributed autonomous synchronization.
- Rapid convergence time.
- Fast error detection and recovery.
- Low communication overhead single bit synchronization message.
- No need to rely on assumptions about the initial state of the clocks.
- No central clock or externally generated pulse system.

5.12.2 Applications

- Embedded systems.
- Distributed process control.
- Synchronization applications.
- Inherent fault tolerance.

5.12.3 The Technology

A fundamental criterion in the design of a robust distributed system is to provide the capability of tolerating and potentially recovering from failures that are not predictable. (failures require incorporating tolerances to various faults). The patented protocols ensure that all nodes reach the same decisions regardless of faulty components.

5.12.4 Characteristics

The systems do not rely on any assumptions about the initial state of the system and do not require a central clock or externally generated pulse. The protocol, or algorithm, allows for self-stabilization of nodes within a distributed system. Moreover, only one bit is needed for communication and no central master clock is required (such as a GPS signal).

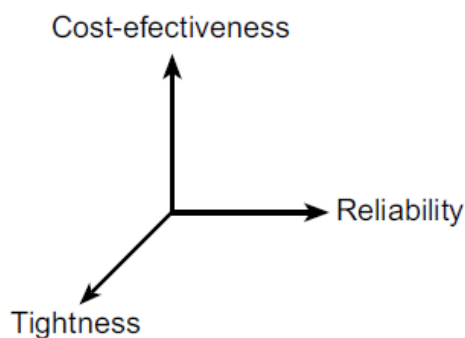


Bild 5.9. Axes of our representation system.

5.13 Clock Synchronization in CAN Distributed Embedded Systems

These systems are constrained with scarce computation and communication resources, which make the solutions available for large-scale distributed systems.

Solutions which implement fault-tolerant clock synchronization by means of complex algorithms, and massive message exchange may exceed the capacity of most low-cost processors as the available bandwidth of the network.

A solution for clock synchronization can be represented as a point within this 3-dimensional space see this in ►Bild 5.9. Previous Image reveals three most relevant properties of a clock service in low-cost distributed embedded systems.

5.14 System model

Consider a distributed system which is made up of a set of nodes that communicate through a fieldbus, see this in ►Bild 5.10.

Every node implements a local clock by means of a counter, which is monotonically incremented by the ticks of a local oscillator. The function of this counter is therefore to map non-observable real time. 't' to a **function $H_i(t)$** .

Their ticks slightly deviate from real time.

The speed at which one clock deviates from real time is called the drift

A clock ι is said to be non-faulty if it exhibits bounded drift :

$$\exists \rho : 1 - \rho \leq (H_i(t_2) - H_i(t_1)) / (t_2 - t_1) \leq 1 + \rho \quad (5.1)$$

Due to their different drifts, the clocks of a distributed system tend to disperse, unless a clock correction procedure is periodically carried out. (Known as 'clock synchronization algorithm'), it has two aims.

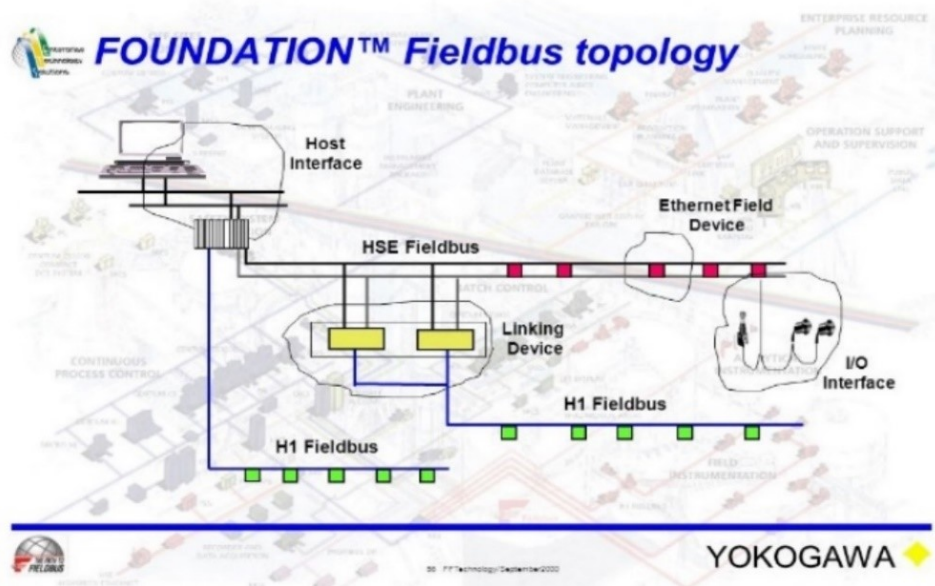


Bild 5.10. Nodes in field bus.

First, aim guaranteed that the time kept by any two *non – faulty* clocks of the system do not differ by more than a given amount δ which is called ‘the precision’:

$$\exists \delta : |H_i(t) - H_j(t)| \leq \delta \quad (5.2)$$

Second, guaranteed that the difference between any clock of the system and real time is less than a certain value α , which is called the ‘accuracy’:

$$\exists \alpha : |H_i(t) - H_j(t)| \leq \alpha \quad (5.3)$$

5.15 Phases of clock synchronization

Synchronization algorithm can be divided into three phases:

1. The first phase is detection of the resynchronization event, where the resynchronization event is the reception of a preconcerted (convenient) message. (labelled Sync message).

When a node receives this message, the node takes a sample of its clock. Due to the differences in the transmission delays, every node receives the Synchronization message at a slightly different instant.

The processing of this message must be performed before actually sampling the local clock, also requires a different time t_i at every node. This causes the samples of the clocks not to be simultaneously taken by the nodes.

1. The second phase is estimation of remote 'clock information'. at least one node (though usually more than one node) sends a message that conveys the instant, at which this node received the Sync message which was sampled in the previous phase.
1. The third phase is correction of the clock. In this phase, each node uses the samples exchanged in the previous phase to calculate the correct value of time, and then adjusts its own clock in order to adapt to this time value.

5.15.1 A simple taxonomy

In order to indicate the "quality" of a clock synchronization algorithm in terms of one requirement, the suggestion is use a mark in the range $[0,1]$, where '0' means "unsuitable" and '1' means "suitable".

The representation system must determine which absolute values are suitable or not. For this it's necessary a simple taxonomy of clock synchronization algorithms, This taxonomy categorizes clock synchronization algorithms regarding one characteristic that have stronger impact on the achievable properties:

- The way to sample the time associated to the Synchronization message and the symmetry of the second and third phase.

5.16 1st phase Software detection vs. hardware detection

1. The first phase of any clock synchronization algorithm is the detection of the synchronization event. This detection can be implemented either in software or hardware. In this phase is necessary to analyze how the chosen implementation influences the three fundamental requirements of clock synchronization.
1. In the second phase of a clock synchronization algorithm, one or more nodes spread its time view by means of message exchange, observe this in ►Bild 5.11

Concerning tightness, the choice between software or hardware has a strong impact. The latency of the detection ('t' in ►Bild 5.12) is higher in the case of software detection, and moreover it has a significantly higher variability, Therefore, the use of hardware is more recommended from the point of view of precision and accuracy.

Concerning reliability, both software and hardware solutions are similar, one potential weak point of software is that the CPU is shared with other tasks, this means that the value of 't' (time) can be increased by the influence of other tasks in the system. (producing a failure of the clock service), since a too high 't' may cause the precision and the accuracy, to be out of the desired range. (this situation can be prevented with a proper **design of the software**).

Concerning cost-effectiveness, software detection should be the preferred solution, as hardware detection requires installation of a specific circuit in every node that performs the detection of the Synchronization message. (software detection is more flexible and also has lower installation costs). Suitability is assigned as a low-cost distributed embedded system. (this requirement could be indicated with a mark in the range $[0,1]$).

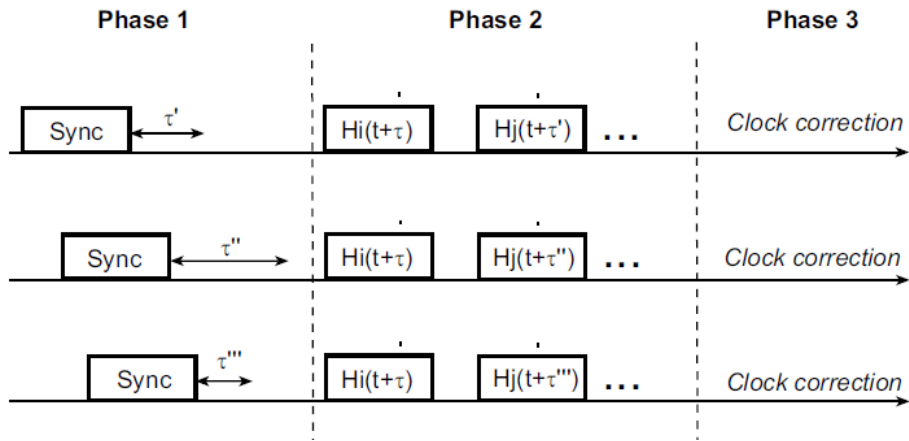


Bild 5.11. Clock synchronization phases.

	Tightness	Reliability	Cost-eff.
Sw	0	1	1
Hw	1	1	0
Sym	1	1	0
Asym	1	0	1

Bild 5.12. Suitability of some design decision.

		Tightness	Reliability	Cost-eff
Sw-sym	a_0	0	1	0
Sw-Asym	a_1	0	0	1
Hw-Sym	a_2	1	1	0
Hw-Asym	a_3	1	0	0

Bild 5.13. Table symmetric vs asymmetric.

5.16.1 2nd phase Symmetric vs. asymmetric scheme

This phase can follow a symmetric or an asymmetric scheme, depending on the number of messages exchanged.

In a symmetric scheme, every node sends a message conveying its corresponding sample of the resynchronization event. After that, each node executes the same algorithm in order to calculate the clock correction. (This algorithm takes into account the contribution of each clock in the system).

Asymmetric scheme (also called master/slave) there is one privileged node. This node is the only one which spreads its time view, so the rest have to synchronize to it.

Concerning tightness Both the symmetric and the asymmetric scheme can achieve the same values. Symmetric solutions are often considered more accurate (closely approximations of real time).

5.16.2 Advantages of symmetric scheme

To reach high reliability, the use of a symmetric scheme seems more suitable. Symmetric schemes are naturally redundant and hence favor implementation of more complete fault tolerance mechanisms.

An asymmetric scheme, reliability mainly depends on the reliability of the master clock as it represents a single point of failure of the system.

Concerning cost-effectiveness Asymmetric schemes are more recommended for low-cost distributed embedded systems, because symmetric schemes require nodes to manage a high number of messages in a short time interval (it involves most low-cost processors and low-cost fieldbuses).

5.16.3 Fixing the suitability requirements

Requirements:

- Concerning tightness, a precision and accuracy in the order of $[1, 0] \mu s$ are considered suitable.

- Considering reliability, we consider that a suitable algorithm must tolerate crashes and performance failures of the clocks, as well as inconsistent omissions of messages due to channel errors.
- Concerning cost-effectiveness, consider the use of hardware detection is too costly and therefore unsuitable.

5.17 The Design of a Clock Synchronization Subsystem for Parallel Embedded Systems

If the ordering of communication events is lost then the phenomenon of “tachyons” occurs in which a message apparently arrives before it is sent, Solutions exist in which logical clocks are kept but these involve time stamping each message.

A criterion for a portable design is that one should aim to avoid a solution involving low level manipulation of the hardware as for instance in the synchronized wave solution for transputers.

The requirements of the centralized portable design can thus be summarized as:

- A minimum of user intervention is needed in regard to the clocking mechanism.
- The design should not be dependent on any feature of the hardware such as would, for example necessity of the use of assembly language Solutions involving hardware interrupts.
- The number of synchronization messages should be minimized.
- The mechanism should not rely on a particular architecture.
- It should not be assumed that the system is already in steady state or can converge slowly to synchronization.
- The times should conform to a suitable real time reference.

In practice one requires two algorithms, an algorithm to bring the clocks on all nodes, within some minimum error range and an algorithm subsequently to maintain time against clock drift.

Local clock requirements Experimental evidence shows that crystal clocks drift linearly if the temperature regime is constant and that anyway drift is small $\prec 10^5$ so that second order terms are neglected in the following. (For runs longer than a few minutes temperature oscillations may occur for transputer clocks, in which case second order corrective terms offer one solution.) It is therefore assumed that for correct clocks the accuracy of clocks should be such that.

$$(1 - f) (t - \hat{t}) - p(t) - H(\hat{t}) \leq (1 + f) (t - \hat{t}) + p. \quad (5.4)$$

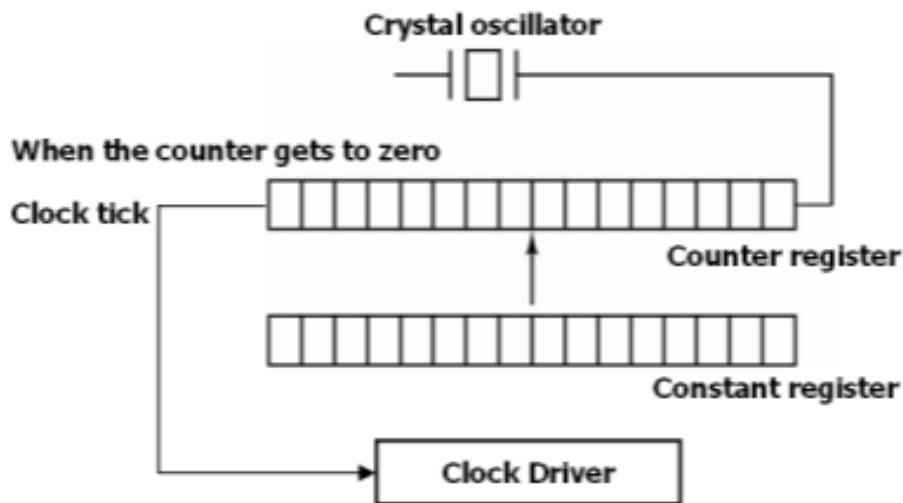


Bild 5.14. General computer clock model.

Where f is the maximum clock frequency drift, t is a notional reference time (real time) such as Universal Time Coordinated (UTC) 'p' is the discrete clock precision, $H(t)$ is the reading of a clock at time t and $(t\hat{t}\hat{1}) > p$. Every computer needs a timer mechanism to keep track of current time and also for various accounting purposes such as calculating the time spent by a process in CPU utilization, disk I/O, (user can be charged properly).

In a distributed system, an application may have processes that concurrently run on multiple nodes of the system. For correct results require clocks of the nodes are synchronized one each other. Nowadays, time synchronization has been a compulsory thing as distributed processing and network operations are generalized. A network time server obtains, keeps accurate and precise time by synchronizing its local clock to a standard reference time source and distributes time information through a standard time synchronization protocol.

General Computer Clock Model A computer clock usually consists of three components:

- A quartz crystal that oscillates at a well-defined frequency.
- A counter register used to keep track of the oscillations of the quartz crystal.
- A constant register. The constant register is used to store a constant value that is decided based on the frequency of oscillation of the quartz crystal.

That is, the value in the counter register is decremented by 1 for each oscillation of the quartz crystal. When the value of the counter register becomes zero, an interrupt is generated, and its value is re-initialized to the value in the constant register. Each interrupt is called a clock tick. ►Bild 5.14 shows a general computer clock model.

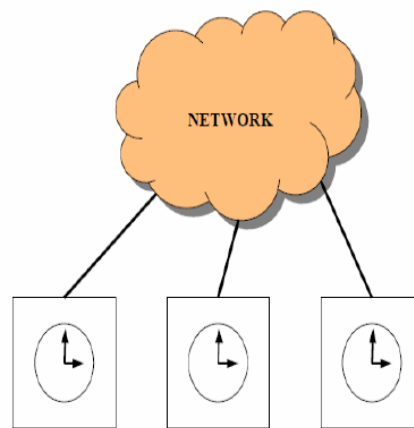


Bild 5.15. Communicate using network.

5.18 Clock Synchronization in Distributed System

Ethernet based communication protocols have recently gained more importance. The IEEE 1588 standard is the most widely used synchronization algorithm for Ethernet based communication protocols. The IEEE 1588 standard uses the centralized synchronization method, observe this in ►Bild 5.15. In this case, the malfunction of a specific node that sends the master clock can affect the performance of synchronization of every node on the network. The low synchronization performance of the nodes can cause malfunctions of the synchronous control system (in this project a master clock is not implemented as reference).

In distributed control systems, a common notion of time of all nodes is fundamental for control, because it allows a consistent system view of dynamic environments and supports meaningful exchange of time-related data.

5.19 Clocks in distributed systems (concepts and implementation)

Motivation for using clocks Two algorithms:

- Time-based leader leases.
- Shared memory using clocks.

Why consider algorithms that use clocks?

- By making stronger assumptions about the system we can get better efficiency / performance.
- The stronger system model adds some (limited) assumptions about synchrony.

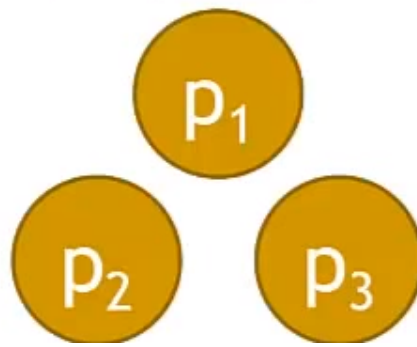


Bild 5.16. Number of replicas.

Old state	Command	Result	New state
{}	<u>Write(x,1)</u>	OK	{x=1}
{x=1}	Write(y,0)	OK	{x=1,y=0}
{x=1,y=0}	Read(x)	1	{x=1,y=0}
{x=1,y=0}	CAS(y, <u>0</u> ,1)	<u>0</u>	{x=1,y=1}
{x=1,y=1}	CAS(y,0,1)	1	{x=1,y=1}
{x=1,y=1}	Read(y)	1	{x=1,y=1}
{x=1,y=1}	Write(y,0)	OK	{x=1,y=0}

Bild 5.17. Commands of paxos.

In this slightly stronger model we cannot solve problems that are harder than what can be solved in the asynchronous model.

Background.

To implement a key-value store using RSM.

Supporting the following commands.

$Read(k), Write(k, v), CAS(k, v_{exp}, v_{new})$

- CAS writes vnew if old value is vexp, returns old value.
- Needs RSM to do CAS (shared Mem. Is too weak).

Service runs on $N=3$ replicas, $\Pi_r = \{P_1, P_2, P_3\}$, observe this in ►Bild 5.16, each one acting as proposer, acceptor, learner.

5.19.1 Command ordering

Paxos guarantees that all replicas execute commands in same order, observe this in ►Bild 5.17.

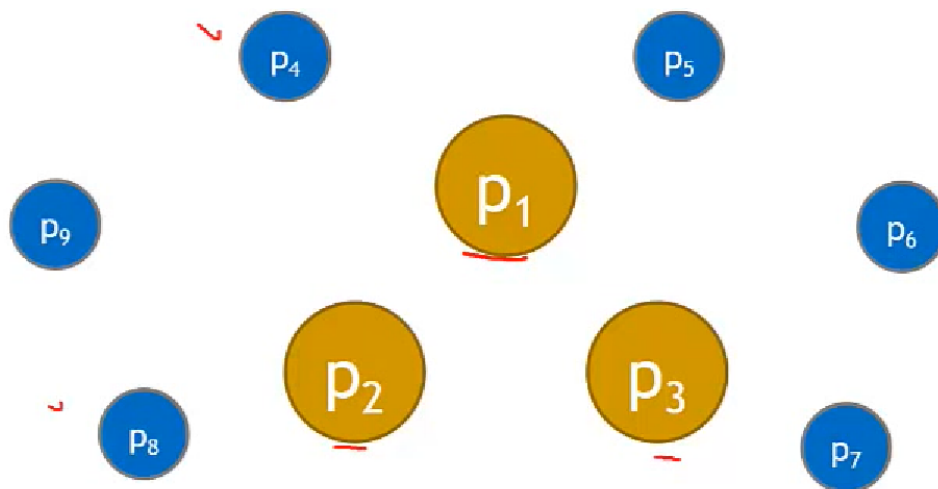


Bild 5.18. Clients and leader.

5.19.2 Clients and leader

Can have any number of clients $\Pi_c = P_4, \dots$. Assume network is stable and P_1 is leader (has started the highest round), observe this in ►Bild 5.18.

5.19.3 Executing a command

P_1 . proposes a command using Paxos, which sends Accept messages (2) to replicas (using previously prepared round number), observe this in ►Bild 5.19.

After P_1 gets AcceptAck messages from a majority, the command order is chosen and P_1 sends Decide messages (4), observe this process in ►Bild 5.20.

P_1 executes the command using the state of the state machine and sends response (4) with result of the operation to P_4 , observe this in ►Bild 5.21.

5.19.4 Opportunity: faster reads

Assume that the operation requested by P_4 is a read operation, $C = read(x)$.

P_1 stores the entire state, so can P_1 read the state variable x and respond immediately, observe this in ►Bild 5.22.

5.19.5 What could go wrong?

- A network split partition P_1 away from P_2 and P_3 .
- P_2 is elected leader but P_1 never hears about this.

Observe the previous problems in ►Bild 5.23.

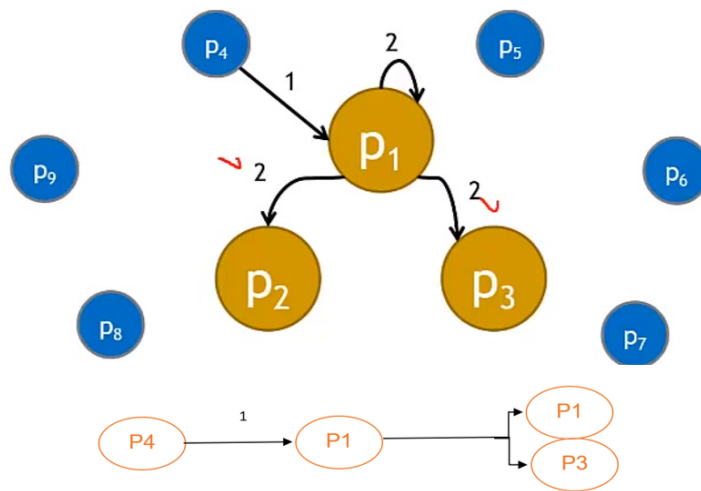


Bild 5.19. Executing a command.

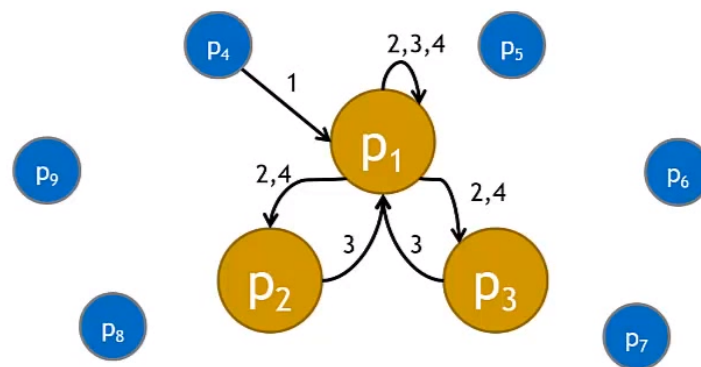


Bild 5.20. Message exchange.

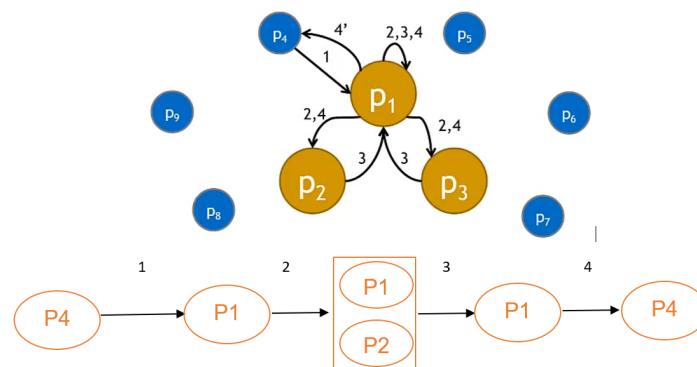


Bild 5.21. Synchronization order.

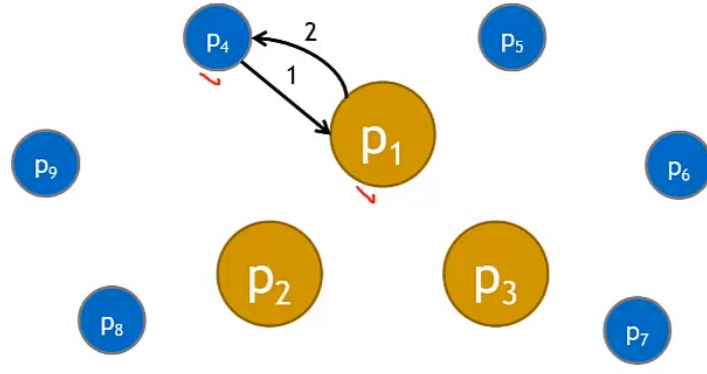


Bild 5.22. Faster reads.

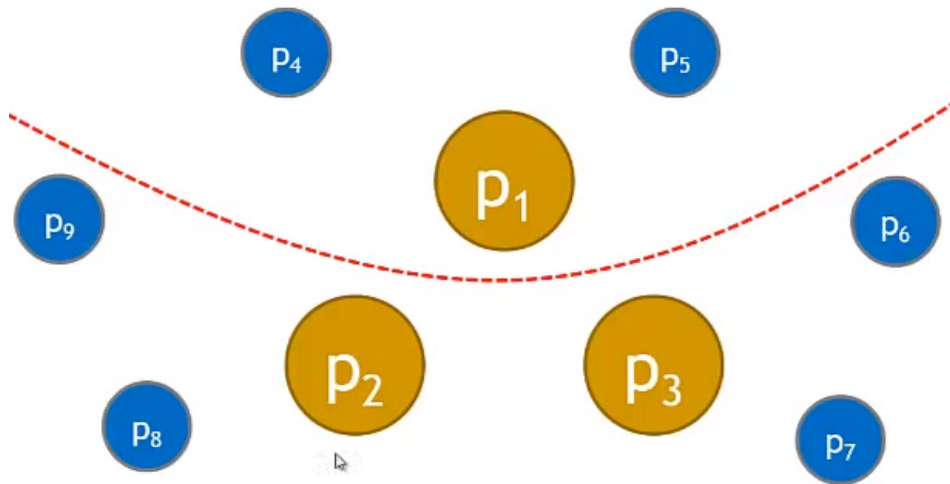


Bild 5.23. Partition.

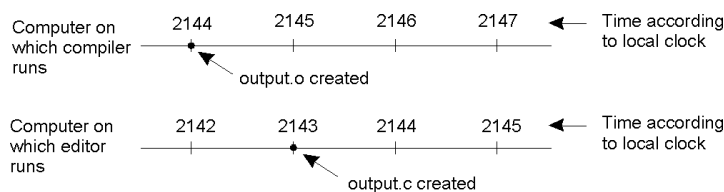


Bild 5.24. Lack of global time.

5.20 Synchronized distributed clocks

Need. Calculating the duration of activities: if such an activity starts on one processor and finishes on another (example: transmitting a message), calculating the duration needs clocks to be synchronized. Lack of global Time in 'Ds' It is impossible to guarantee that physical clocks run at the same frequency. Lack of global time can cause problems; observe this in ►Bild 5.24

When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.

5.21 Physical clock

- Every computer is equipped with CMOS circuit these are electronic devices that count oscillations occurring in a crystal.
- Also called timer, usually a quartz crystal, oscillating at a well-defined frequency.
- Timer is associated with two registers: A counter and a holding register, counter decreasing one at each oscillation.
- When a counter reach zero, an interrupt is generated, this is the clock tick.
- Clock tick have a frequency of 60-100 ticks per seconds.

5.22 problems

1. Crystal cannot be tuned perfectly. Temperature and other external factors can also influence their frequency. Clock drift: the computer clock differs from the real time.
1. Two crystals are never identical. Clock skew: the computer clocks on different processors of the distributed systems show different time.

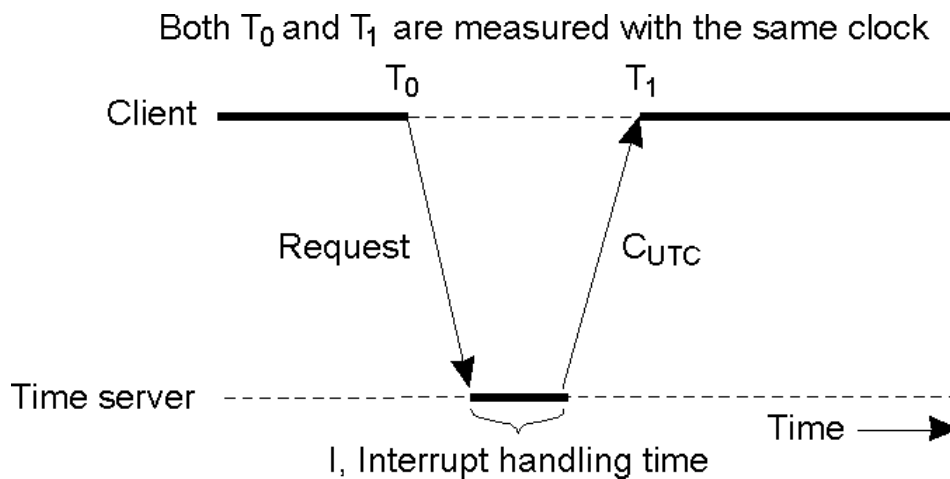


Bild 5.25. Interrupt handling time

5.23 Distributed algorithms

- There is no particular time server.
- The processors periodically reach agreement on clock value by averaging the time of neighbor's clock and its local clock.
- This can be used if no UTC receiver exists (no external synchronization is needed). Only internal synchronization is performed.
- Processes can run on different machines and no global clock to judge which event happens first.

5.24 Cristian's algorithm

- The simplest algorithm for setting time, it issues a remote procedure call to time server and obtain the time.
- A machine sends a request to time server in " $d/2$ " seconds, where $d = \max$ difference between a clock and UTC (Coordinated Universal Time).
- The time server sends a reply with current UTC when receives the request.
- The machine measures the time delay between time server sending the message and machine receiving it. Then it issues the measure to adjust the clock.

Both T_0 and T_1 are measured with the same clock.

The best estimate of message propagation time = $(T_0 + T_1)/2$. The new time can be set to the time returned by server plus time that elapsed since server generated the timestamp.

$$T_{new} = T_{server} + (T_0 + T_1)/2 \quad (5.5)$$

5.25 Logical clocks

Assume no central time clocks.

- Each system maintains its own local clock.
- No total ordering of events.
- Allow to get global ordering on events.

Assign sequence numbers to message. All cooperating processes can agree on order of event.

5.26 Lamport Timestands

It is used to provide a partial ordering of events with minimal overhead.

- It is used to synchronize the logical clock.
- It follows some simple rules:

A process increments its counter before each event in that process i.e. Clock must tick once between every two events. When a process sends a message, it includes its timestamp with the message. On receiving a message, the receiver process sets its counter to be the maximum of the message counter and increments its own counter.

5.27 'Happened before' relation

a 'Happened Before' b: $a \rightarrow b$

1. If a and b are events in the same process, and a comes before b, then $a \rightarrow b$.

1. If a: message sent.

b: receipt of the same message. then $a \rightarrow b$.

1. Transitive: If $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$.

1. Two distinct events 'a' and 'b' are said to be concurrent

If $a - / \rightarrow b$ and $b - / \rightarrow a$. This all process is shown in ►Bild 5.26

Three processes, each with its own clock. These clocks run at different rates.

For two events a and b in same process p1:

$$C(b) = C(a) + 1 \tag{5.6}$$

If a is sending process and b is receiving process of pi and pj then,

$$C_j(b) = \max((C_i(a) + 1), C_j(b)) \tag{5.7}$$

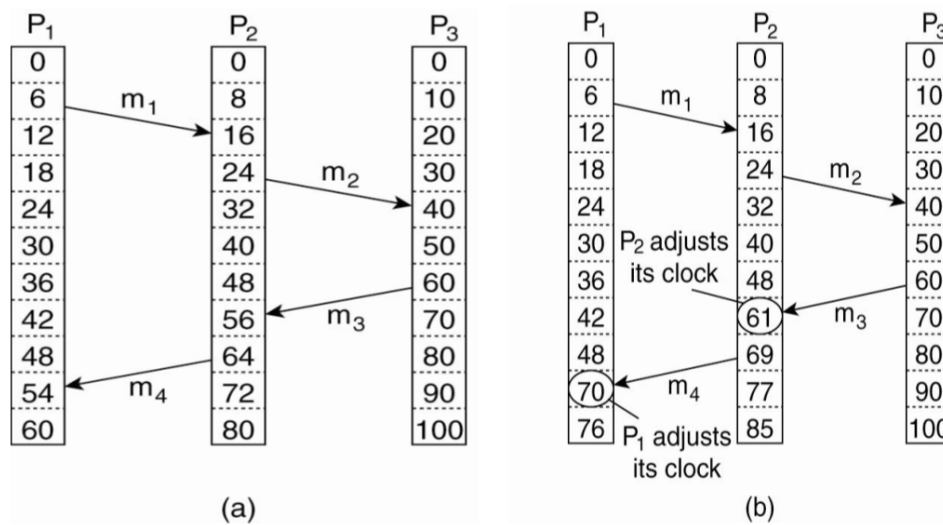


Bild 5.26. Relation for clock adjusting

5.28 Lamport's algorithm corrects the clocks

According to clock Synchronization in Distributed Systems

1. At any time, the values of all the nonfaulty clocks must be approximately equal (*within_{max}*).

There is a small bound on the amount by which a nonfaulty process clock is changed.

5.28.1 Hardware Clocks

Hardware clock: $H(t)$

$$\text{Rate : } f(t) = dH(t)/d(t) \quad (5.8)$$

$$\text{Drift : } p(t) = f(t) - 1 \quad (5.9)$$

Observe this expressions in ►Bild 5.27:

5.28.2 Software Clocks

Hardware clocks in general are not synchronized: $H_p(t)$, $H_q(t)$ (is not bounded) Software clocks are used instead: $S_p(t) = H_p(t) + ap(t)$

$ap(t)$: Continuous and Discrete Software Clocks. Observe the difference between them in ►Bild 5.28:

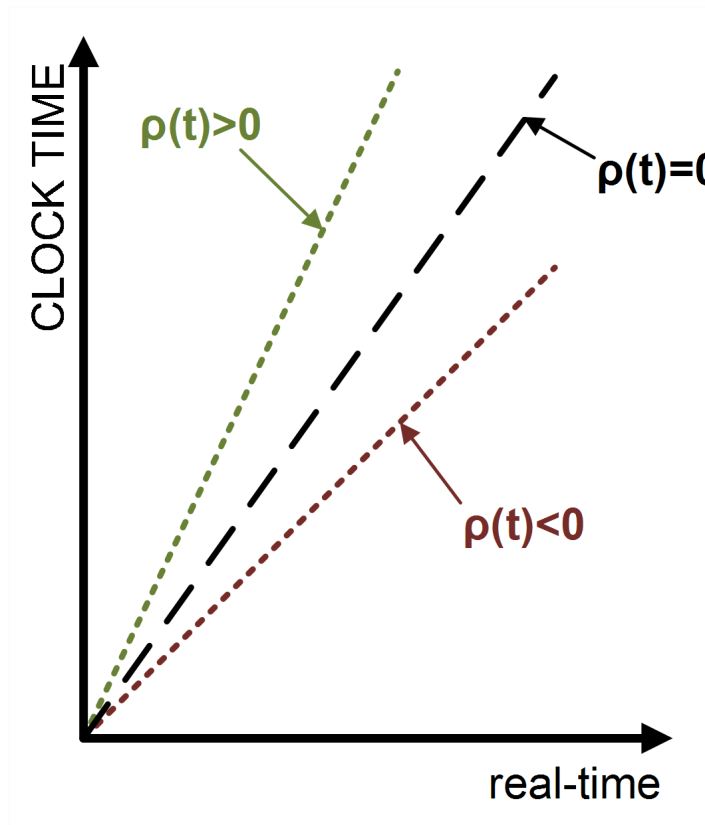


Bild 5.27. Calculus of optimum time

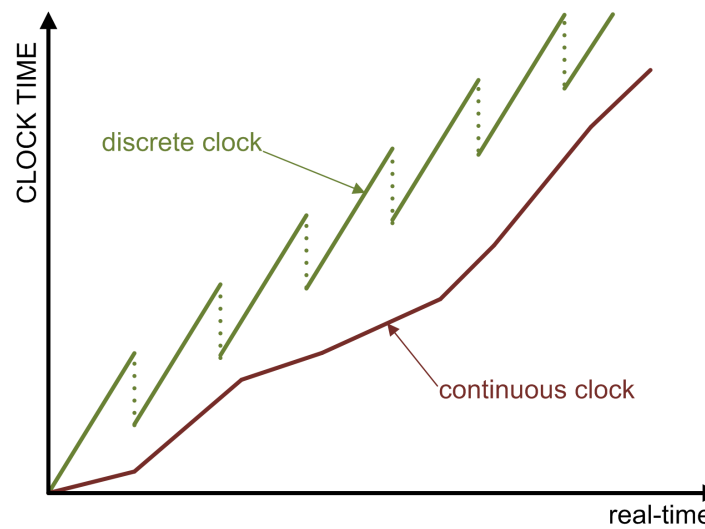


Bild 5.28. Difference between discrete clock and continuous clock

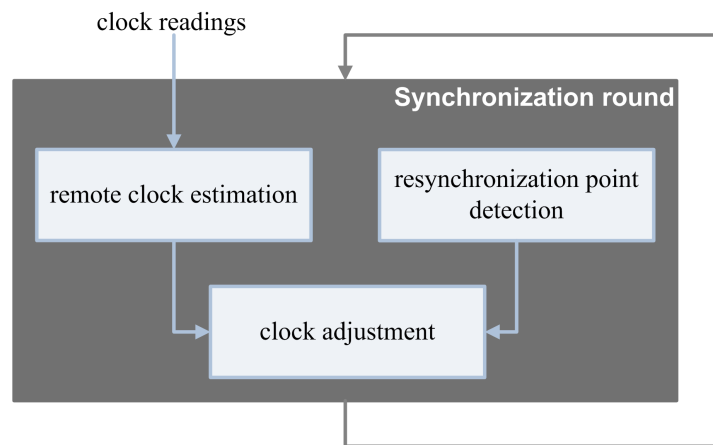


Bild 5.29. Components of synchronization algorithm

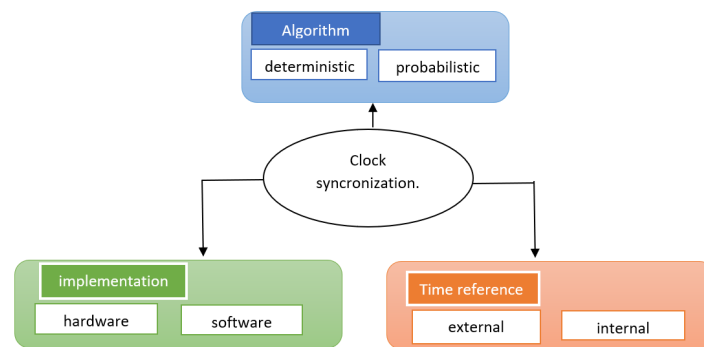


Bild 5.30. Algorithms classification

5.28.3 Main Components of Clock Synchronization Algorithm

Observe them in ►Bild 5.29:

5.28.4 Algorithms classification

Observe them in ►Bild 5.30:

5.28.5 Network Time Protocol's Goals and Definitions

Goal: accurate and precise time on a statistical basis with acceptable network overheads and instabilities in a large, diverse interconnected system.

$$\text{Offset} : |H_p(t) - H_q(t)| \quad (5.10)$$

$$\text{Skew} : |dH_p(t)/d(t) - dH_q(t)/d(t)| \quad (5.11)$$

Clock Synchronization: time synchronization: bounding offset frequency synchronization: bounding skew

5.29 Environment for a Clock synchronized system

One of the most common problems in industry is Refrigeration for different types of electronic devices, mechanical devices, machines, food or even medical resources. Most of this care is using air conditioner devices or freezers which must be continuously working, this is not economically affordable. This device will try to solve this problem using a Clock synchronized mechanism to be more efficient in time and in electric service costs. This system will give the optimal temperature and humidity levels for a cooling environment.

5.29.1 1st circuit to implement: Temperature Controller using Arduino

An AC (Air Conditioner) which was once considered to be a luxury item. Over time it has been implemented in food refrigeration systems, device's protection and objects which are quite expensive to maintain in operation and to provide maintenance. This produce a high electricity consumption. In this project the purpose is to make a small Automatic Temperature Controller that could minimize the electricity consumption by varying the AC temperature automatically based on the Rooms temperature. By Varying the set temperature periodically, it allows to make the Air conditioning work with lower temperature values for a long time and making it consume less power. **For this task it's necessary to include a microcontroller (Arduino ATMEGA 2560).**

Situation • Change the Air Conditioner's set temperature to different values during different periods of time. to automate this process, it is implemented a Temperature sensor (DHT11). which reads the present room's temperature and based on that value, it will send commands through a microcontroller programmed with the temperature detection. The Peltier is implemented to react to these commands and it will give a warm or a cold temperature depending on the desired environment.

Pre-requisites Make sure you have an Arduino Mega and any other version of Arduino, since the code size is heavy. Also check if you have already installed the following Arduino libraries.

- DHT11 Sensor Library for Temperature sensor.

5.29.2 2nd circuit to implement: Servomotor Controller

It is possible to connect small servo motors directly to an Arduino to control the shaft position very precisely. Because servo motors use feedback to determine the position of the shaft, you can control that position very precisely. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors are small, and because they have built-in circuitry to control their movement, they can be connected directly to an Arduino, (5-12V). Most servo motors have the following three connections.

ADVANTAGES	DISADVANTAGES
You have full control over IP address settings and dynamic/static IP settings.	You might need administrative control or knowledge of the network infrastructure.
You can connect and interconnect many BBBs to a single network (including wireless devices).	The BBB needs a source of power (which can be a mains-powered adapter).
The BBB can connect to the Internet without a desktop computer being powered on.	The setup is more complex for beginners if the network structure is complex.

Bild 5.31. Advantages and disadvantages of Network switch.

- Black/Brown ground wire.
- Red power wire (around 5V).
- Yellow or White PWM wire.

Hardware Required

- 1 x TowerPro SG90 servo motor.
- 1 x Arduino board Atmega 2560. (It produces more output voltage than others).
- 3 x jumper wires.

5.29.3 Regular Ethernet (Beaglebone book)

By “regular” Ethernet, I mean connecting the BBB to a network in the same way that you would connect your desktop computer using a wired connection. For the home user and power user, regular Ethernet is probably the best solution for networking and connecting to the BBB. The main issue is the complexity of the network—if you understand your network configuration and have access to the router settings, then this is by far the best configuration. If your network router is distant from your desktop computer, you can use a small network switch, which can be purchased for as little as 10~20 (euros). Alternatively, you could purchase a wireless access point with integrated multiport router, observe advantages and disadvantages in ►Bild 5.31.

6 Implementation

6.1 Installing compiler

In order to make in order programming's structure and run it into the TX2 Jetson, it is necessary to download a well know compiler which provide the system's environment a good performance, in this project instead of use the compiler that Ubuntu 16.04 has by default, the decision of download a better compiler was taken and the compiler selected is 'Qt creator 4.8'.

There are a couple of tricks to installing Qt Creator on the Jetson TX2 from the Ubuntu repositories. Some folks have reported issues installing newest version like Qt 5.1 on the TX2, next steps will explain all the process for installation.

Installation:

1-Install Qt Creator from the repositories. Open a Terminal and execute:

- `sudo apt - get install qt5 - default qtcreator -y`

2-The compiler needs to be set up. Open Qt Creator, and go to: *Tools* -> *Options* -> *Build and Run* -> *Compilers* Click the 'Add' button and select 'GCC'. In the 'Compiler path:' text box, place the path to the gcc compiler. On a standard installation the path is: `/usr/bin/gcc`.

3-The last step is to add a kit which supports the GCC compiler. Click the 'Kit' tab. The 'Desktop' kit appears to have an issue with setting the compiler. This means that you can find the Desktop kit configuration file and manually modify it, or you can create a new Kit all together.

Qt creator is now ready for development, after installation JetsonTX2 Kit is not selected when creating a new project, so it's necessary to define JetsonTX2 Kit as the default compiler.

6.1.1 Boost library

In order to have all the libraries to establish a network communication 'Boost.asio' libraries provide developer some useful libraries which include different functions and structures for this type of compilers.

Just write the command **apt-get command** (requires sudo) In order to have the newest version of these Boost.asio library, its necessary to download 'Debian version', there are more of them, but this is the correct one for Ubuntu 16.04.

- `sudo apt-get install libboost-all-dev`

It's recommended to run a few examples to know if Boost.asio is working in compiler (Qt creator), to see that install 'aptitude library' and run one of the examples in 'Boost.asio for Linux' web page.

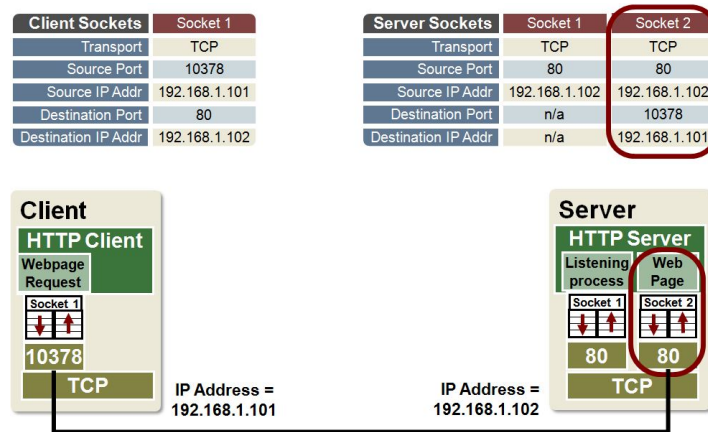


Bild 6.1. Socket Ip address.

- sudo apt-get install aptitude

6.2 Socket: End point of communication

To establish communication between the TX2 Jetson board and the two Arduino microcontroller with two different programming codes, it is necessary to create a socket program for two clients and one server. To understand better what is a socket and how does communication is done. Some research must be done before socket is implemented. What is a SOCKET? A socket is an end of communication between two systems on a network. To be a bit precise, a socket is a combination of IP address and port on one system. So, on each system a socket exists for a process interacting with the socket on other system over the network. this process can be related with ►Bild 6.1:

There are two types of network communication models:

1. OSI
1. TCP/IP

The communication over the network in TCP/IP model takes place in form of a client server architecture. i.e., the client begins the communication and server follows up and a connection is established. Sockets can be made in many languages like Java and C++ which are the most known for this type of communication, to understand the socket communication in its purest form, C++ programming language is implemented in this project. To observe its construction and functioning, observe next flow chart diagram starting with the libraries structure in ►Bild 6.2.

6.2.1 Flowchart diagram: Server

To start making C++ programming there are a few basic examples on internet related to Socket communication with TCP protocol, this program was based on a Socket with Chat

conversation between client/server, the main purpose of this program was to interact with the server sending messages through PC's keyboard. Server program's structure is divided in 7 parts in order to understand its functioning. 1. This C++ program contained all the Boost libraries available for Socket communication which can be observe in ►Bild 6.2, some of them are added to make strings conversions, synchronized timers and use of strings.

2. Most of the variables are related to socket programming, arrays with a specific size to send or receive strings, and floats to make some mathematical operations , observe this on ►Bild 6.3 .

3. Structure for incomming conections is nessesary for any type of socket, these condions allows the Server accept clients which are connected to its own ip address, a secuence of conditionals must be generated to get connected and accept next clients , observe this in ►Bild 6.4.

4. To exactly know if there is connection to the client, client must send a notification to the server to say is is connected and there will be information exchange in the future. As the previous image ►Bild 6.4 this is necessary to have client connection. Observe this on ►Bild 6.5.

5. The program execution after a Server's Acceptation, this main part of the code will be executed in all the cases server is connected to client, it contains a Synchronized timer with a boost::asio structure, it is practical to make a delay between each message send and another received, this message is written to the server through command *write()* and message read from client with *recv()*, observe this in ►Bild 6.6.

6. The calculus of the average is main part of the Synchronization algorithm, Adding the two moments of time in which a message is sent and received and divided by the number of processes within the system it is possible to get the average in a Client/server communication, observe this on ►Bild 6.7 .

7. Finally send the average to the client and expect an answer from the client receiving and printing the average on Monitor, a sequence is generated and the message exchange will be permanently changing each second according to ►Bild 6.8.

6.2.2 Flowchart diagram: Client

6.2.3 Synchronous timer

In order to obtain a better performance from the C code, a synchronous timer is implemented. The main advantage to these kinds of timers is that it does not require any knowledge about network programming compared to the other I/O objects provided by asio. The asio classes can be used by including the asio.hpp header file in the main file. To configure .pro file, it is necessary to add these libraries for synchronous timers:

```
–lboost_system –lboost_thread –lthread
```

Inside of the .cpp file is nessesary to follow next steps:

1. To use timers, the *Boost.Date_time* header file is included.
2. Declare an *io_service* object io before '*mainloop*' :

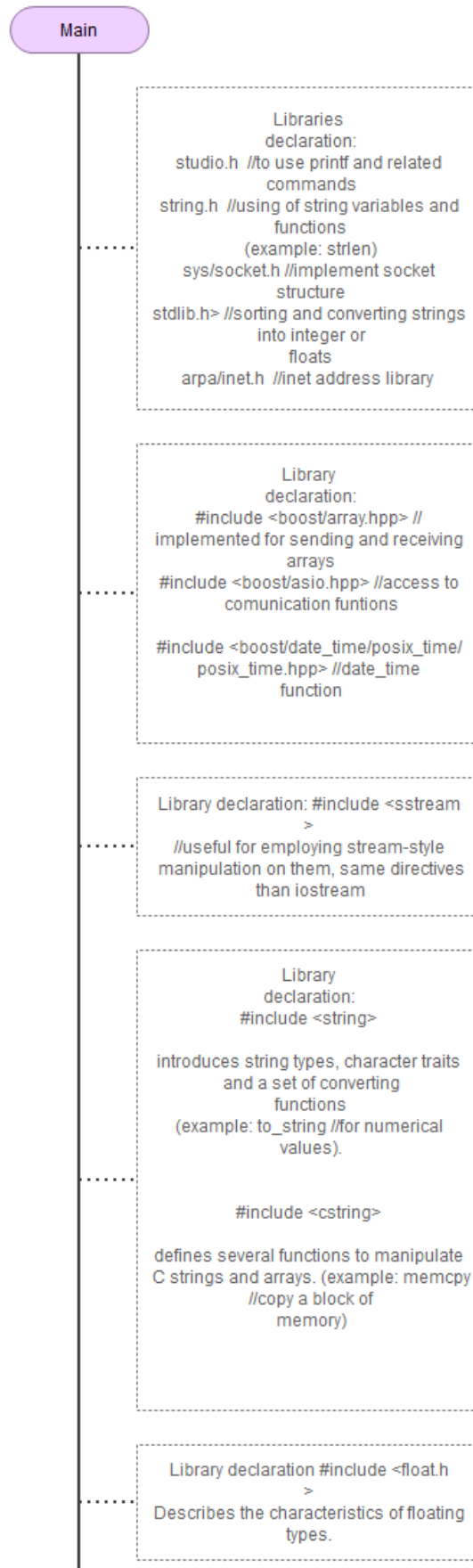


Bild 6.2. Libraries in Socket server.

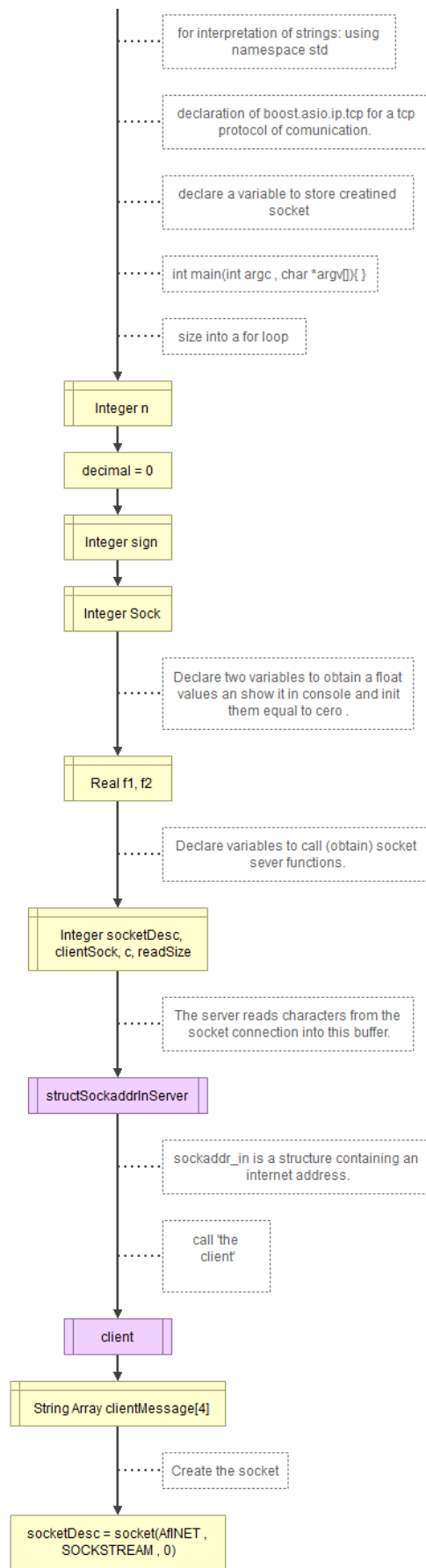


Bild 6.3. Variable declaration to call socket functions.

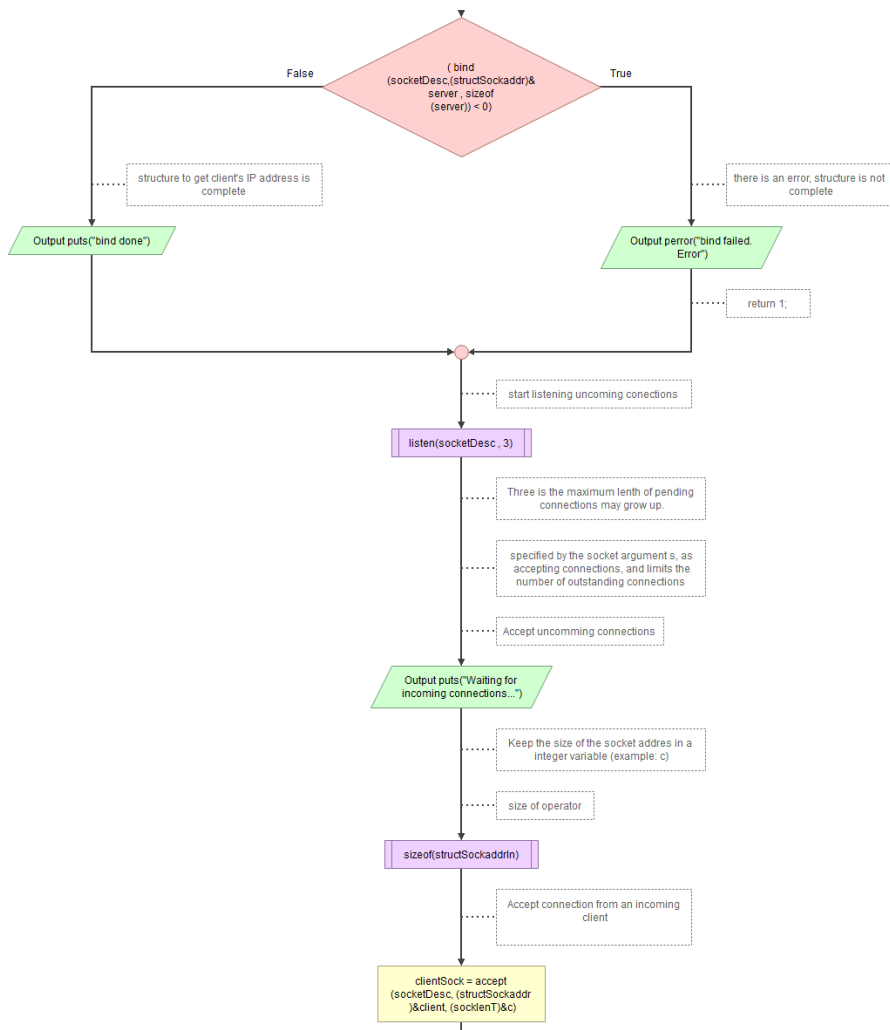


Bild 6.4. Structure to get Ip address and accept incoming connections.

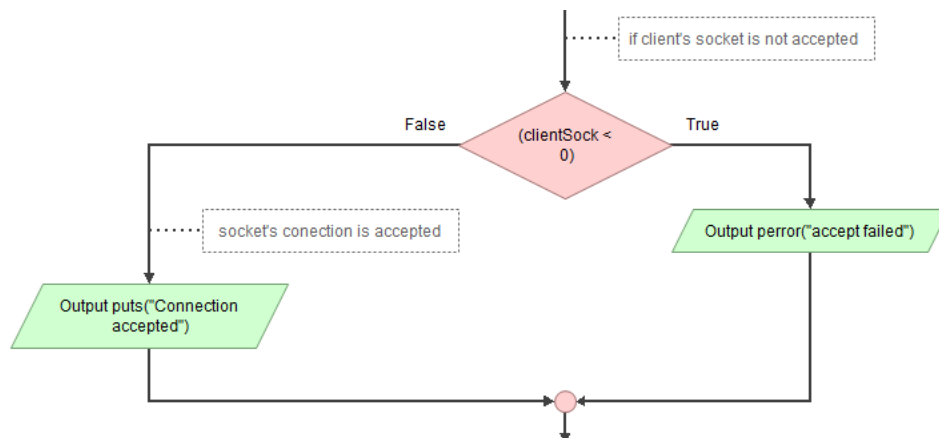


Bild 6.5. Notify server from incomming connections.

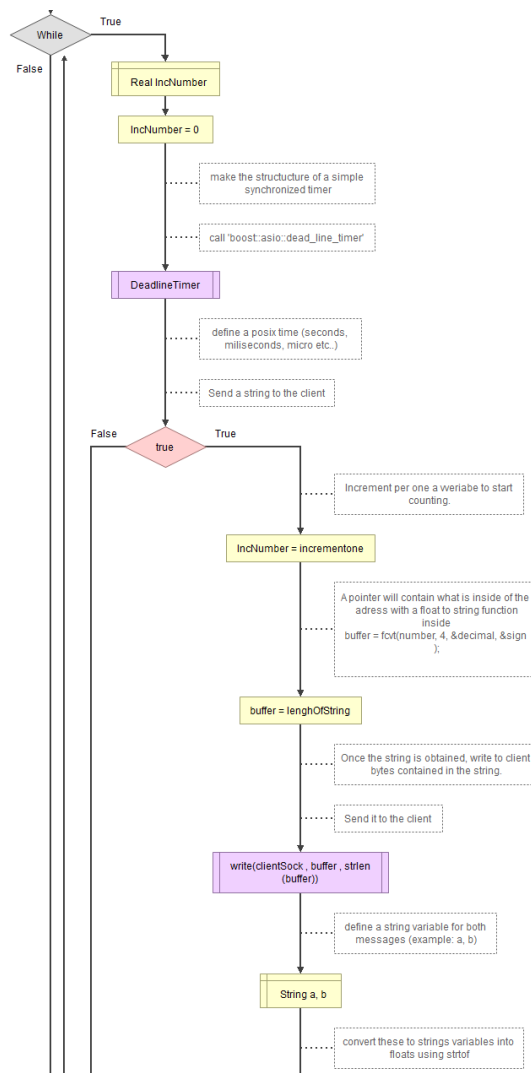


Bild 6.6. Infinite loop, synchronized timer, declared variables.

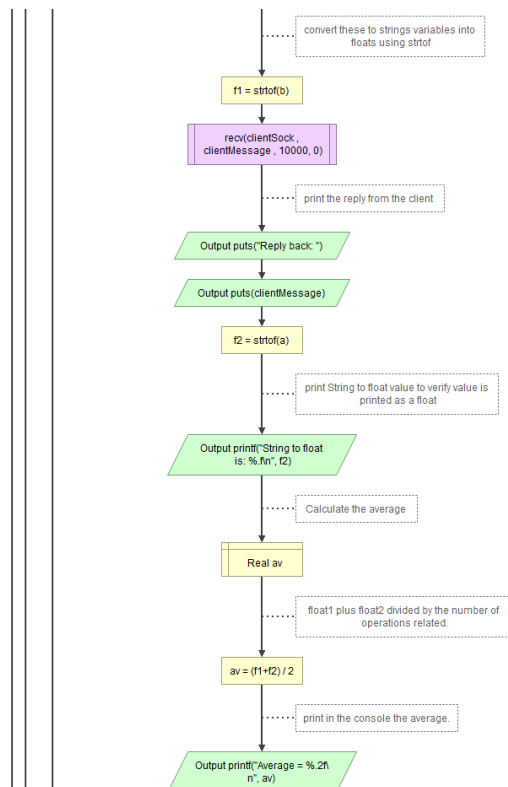


Bild 6.7. Calculus of the average.

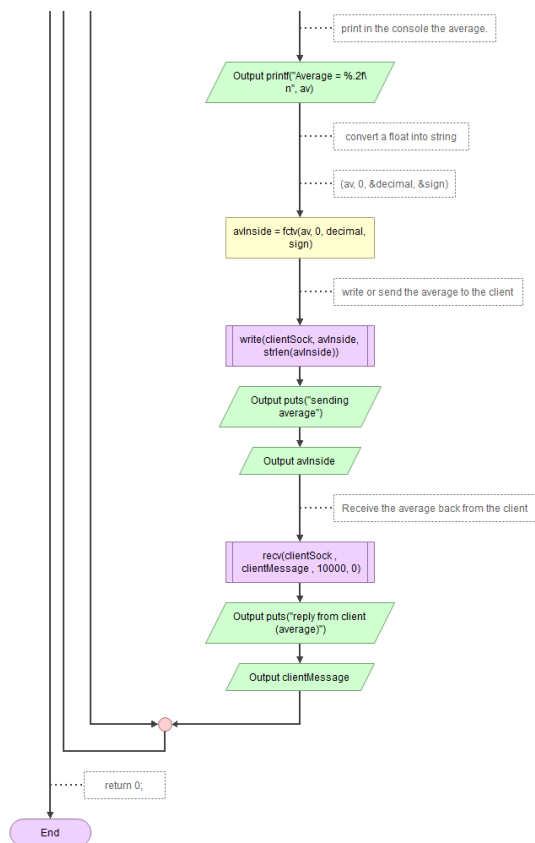


Bild 6.8. Sending and receiving the average.



Bild 6.9. Libraries declaration.

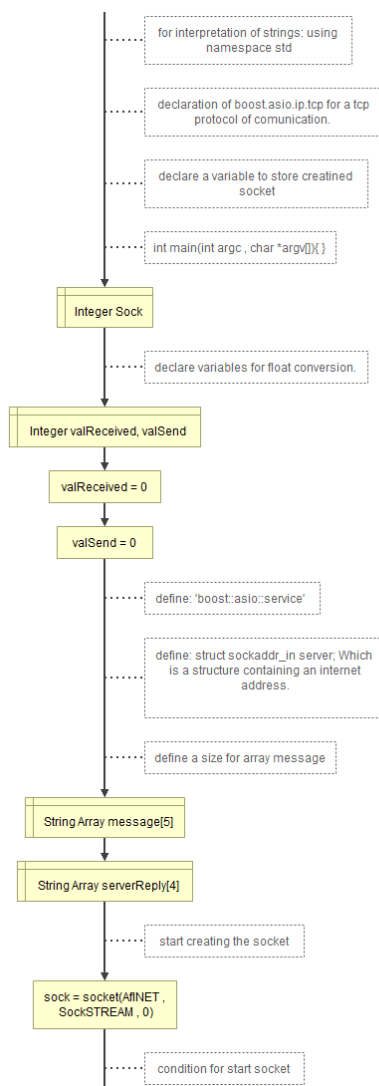


Bild 6.10. Variable declaration to create socket.

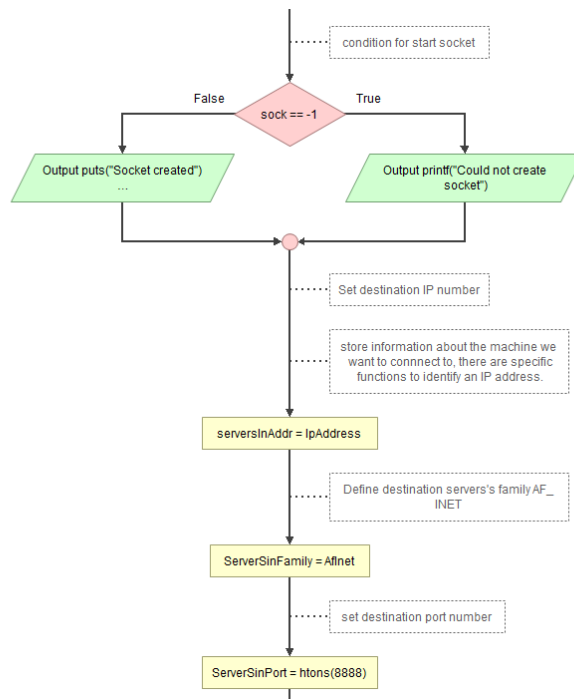


Bild 6.11. Client's Structure to get Ip address and accept incoming connections.

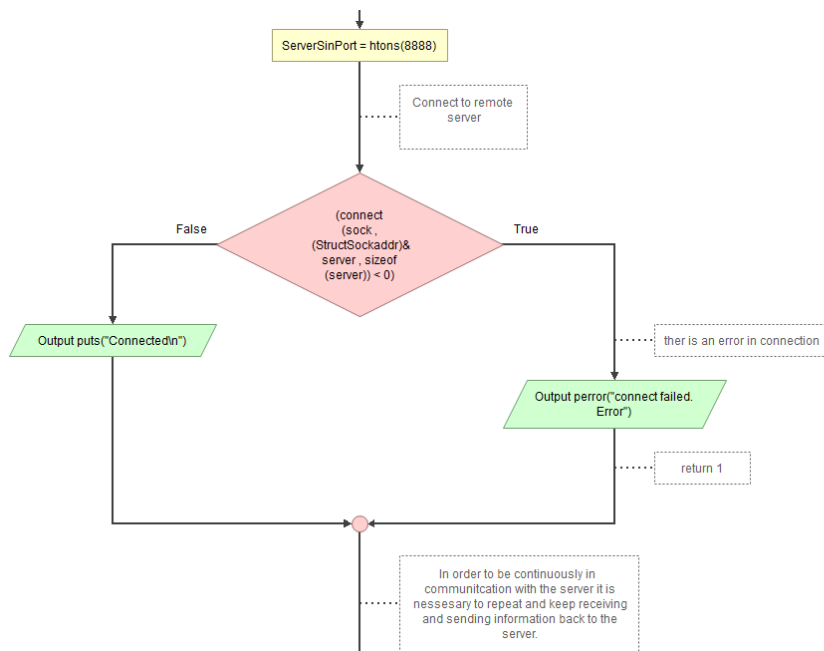


Bild 6.12. Notify client a connection from server and accept incoming connections.

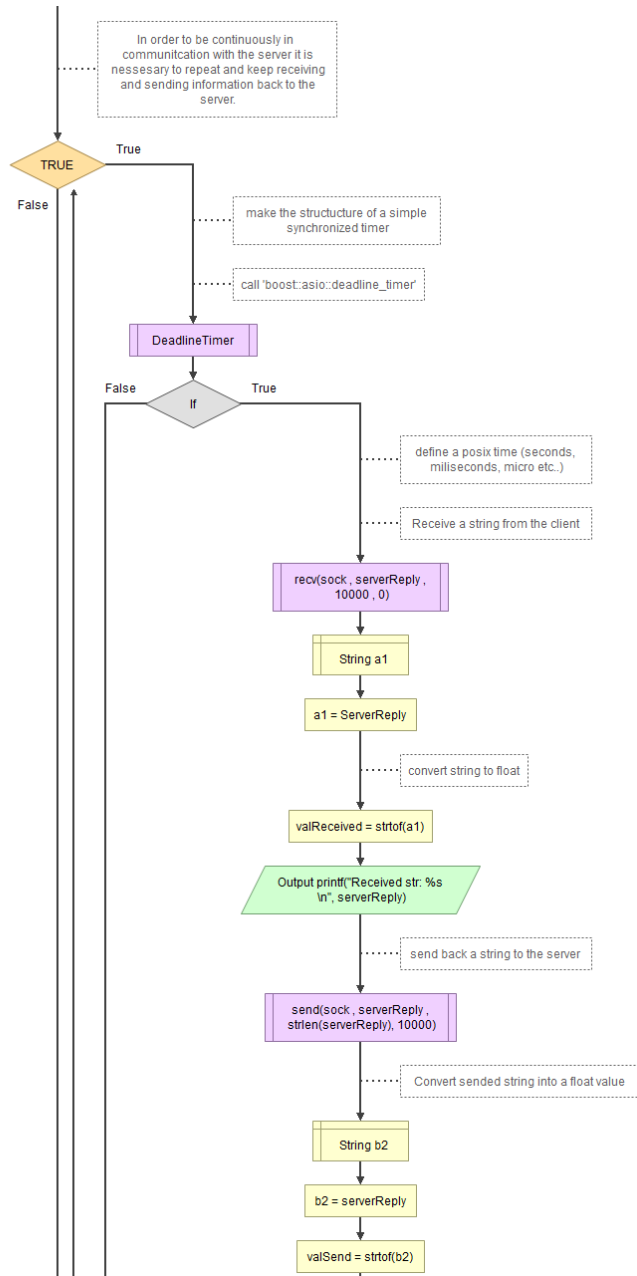


Bild 6.13. Receiving a real value and sending back to server.

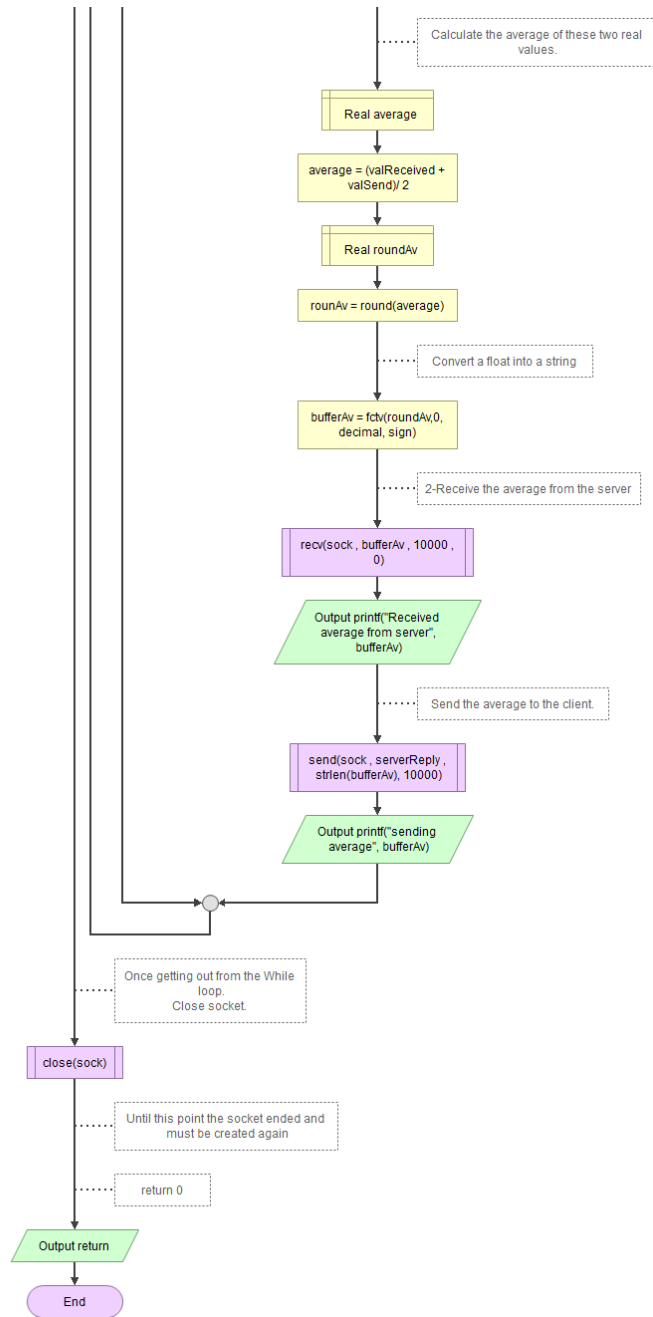


Bild 6.14. Receiving and sending the average (client).

```
boost :: asio :: io_serviceio;
```

3. Declare an object of type `boost :: asio :: deadline_timer`.

4. The asio classes that provide I/O (in this case timer) take a reference to an `io_service` as their first argument. The second argument sets the timer to expire in `i` seconds.

```
boost :: asio :: deadline_timer(io, boost :: posix_time :: seconds(i))
```

5. Declare a blocking wait on the timer. In other words, the call to `deadline_timer :: wait()` will not return until the timer has expired.

6.3 Communication Jetson TX2-Arduino over Serial port

Arduino devices are very simple microcontroller commonly encountered in embedded projects, and can communicate with a Jetson Tx2 via Serial UART or I2C, or can be programmed from a Jetson Tx2 through USB (ie: allowing you to develop Arduino code using a Jetson Tx2 instead of using a PC for development).

This kind of connection is needed to upload all the C++ programs into the Arduino ATMEGA 2560. Unfortunately the serial communication of the Jetson TX2 with this type of microcontrollers is not allowed because the NVIDIA developers give a Kernel (configuration of access to devices) customized to not be able to access serial communication with any other connected devices. This is done as a precaution to prevent any driver or controller installed by NVIDIA gets uninstalled by accident and the user feel safe with Ubuntu environment. It is possible to see the Arduino board connected to the Jetson TX2 writing on terminal:

```
ubuntu@tegra-ubuntu: lsusb Bus 002 Device 001 : ID 1d6b : 0003 Linux Foundation 3.0 root
hub Bus 001 Device 013 : ID 2341 : 0042 Arduino SA Mega 2560 R3 (CDC ACM) (Note: Even
if the device is recognized it doesn't mean it is possible to have serial communication.)
```

This can be noticed because when the arduino compiler is installed and a device is connected via usb the serial port appears disabled inside the compiler and the C program can not be loaded inside the microcontroller. For this you have to download a previous version of Kernel into the Jetson TX2 and in this way you can customize the downloaded Kernel with access to serial communication. Support for the FTDI converter device is not set in the kernel configuration by default on TX2 and it is possible to know it writing on terminal: `zcat /proc/config.gz | grep FTDI`

```
CONFIG_USB_SERIAL_FTDI_SIO is not set. CONFIG_USB_FTDI_ELAN is not set.
```

It is needed to compile the FTDI kernel module from source to use the device, to make this follow next instructions.

1. Download the kernel sources (`kernel_src.tar.bz2`) from <https://developer.nvidia.com/linux-tegra-rel-19> and copy the archive to the Jetson, into the ubuntu user home directory. This is a confusing process and has to be done with patience because the version of Jetpack 3.3 in this Jetson board does not allow to access to a Kernel of recent versions such as 4.11 as an example. Then extract the tarball to `/kernel`: (in the case of Jetson TX2 with JetPack 3.3 install the version 3.10.4).

```
tar xvjf kernel_src.tar.bz2
```

2. Copy over the Jetson's existing kernel configuration to the newly-extracted kernel source:

```
zcat/proc/config.gz > /kernel/.config
```

3. Build and launch the menuconfig tool to configure the kernel options. menuconfig requires ncurses to be installed, hence the apt-get command first, on terminal write:

```
sudo apt - get install ncurses-bin libncurses5-dev
```

5. On terminal access to the Kernel's folder: `cd/kernel`, and write

```
make menuconfig
```

Kernel will upload on Jetson TX2, this is will take a few minutes or one hour depending on the processor and Kernel version, terminal will display an screen with all the customized options.

6. Navigate to Device Drivers – > USB Support – > USB Serial Converter Support

7. Choose Module for USB FTDI Single Port Serial Driver and `FTDI_ELAN` port Serial Driver to allow them access to the main configuration.

Save changes and exit.

All this customization will be saved in a `.config` file inside your Kernel folder.

8. Verify that the FTDI component is now set to build as module writing `/kernelcat.config|grepFTDI`
`CONFIG_USB_SERIAL_FTDI_SIO=m CONFIG_USB_FTDI_ELANisnotset`

9. Build modules writing on terminal: `make prepare make modules_prepare`

10. Install the FTDI module writing on terminal:

```
sudo cp drivers/usb/serial/ftdi_sio.ko /lib/modules/(uname -r)/kernel sudo depmod -a
```

11. Verify installation, see the FTDI module loaded `and/dev` node assigned writing on terminal:
`dmesg | grep usb` Observe installation is completed.

```
usbserial: USB Serial support registered for FTDI USB Serial Device
```

6.4 Installing Arduino IDE compiler

Running the Arduino IDE on the Jetson is easy, as it's included in the Ubuntu 16.04 repository, and allows one to use the Jetson to develop upload sketches directly to the Arduino board. Write on terminal.

```
sudo apt-get install arduino arduino-core
```

(Note: This version of arduino is the oldest one available and some of the libraries are not installed by default, it is necessary to install (Dallas temperature, String library and Wire library) for the environment of this project.)

6.5 Communication Jetson TX2-Arduino over ethernet

Normally all the libraries related to ethernet connection are by default on Arduino IDE software: The libraries allow an Arduino board to connect to the internet. The board can serve as either a server accepting incoming connections or a client making outgoing ones. The libraries support up to four concurrent connection (incoming or outgoing or a combination). Ethernet library (Ethernet.h) manages the W5100 ethernet shield. Changing the library used allows to port the same code from Arduino Ethernet Shield to Arduino Ethernet 2 Shield and vice versa.

Arduino communicates with the shield using the SPI bus. This is on digital pins 50, 51, and 52 on the Mega. Pin 10 is used as SS. On the Mega, the hardware SS pin, 53, is not used to select the W5100, but it must be kept as an output or the SPI interface won't work.

The Arduino Ethernet Shield allows you to easily connect your Arduino to the internet. This shield enables your Arduino to send and receive data from anywhere in the world with an internet connection. There are some steps before ethernet communication is done.

1. **Setup:** Setting it up is as simple as plugging the header pins from the shield into your Arduino.

2. The Ethernet Shield is based upon the W51000 chip: Shield Features:

- Internal 16K buffer.
- Connection speed: 10/100Mb.
- Micro SD slot

3. Get Started: Plug the Arduino into computer's USB port, and the Ethernet shield into Desktop switch (or direct internet connection). Arduino IDE 1.0 or later, does not require manually configuring an IP address. To figure out what IP address has been assigned to your board open *File* -- > *Examples* -- > *Ethernet* -- > *DhcpAddressPrinter*.

Or simply making up a unique mac address should work. If you are using multiple shields, make sure each one has a unique mac address. Once the mac address is properly configured, upload the sketch to your Arduino, and open the serial monitor. It should print out the IP address in use.

4. Connect Arduino Ethernet shield to a web server to load an HTML page or function as a chat server. It is possible also parse requests sent by a client, such as a web browser, make some example codes (*example* : *pushbutton*), and then refresh the browser page.

To make this work connect the positive lead an LED to pin 2, and the negative lead in series with a 220 ohm resistor to ground.

5. Replace the ip address in your web browser (*example* : 10 42 0 3) with your arduino ip address.

6. Connect arduino as a client. In other words, it is possible to read websites like a web browser. Reading information from websites typically involves parsing a lot of strings. To read other message, change the following bit of text:

- `client.println("GET /1/statuses/user_timeline.xml?screen_name=[NEW MESSAGE NAME HERE]count=1 HTTP/1.1");`

For this, the Jetson TX2 is connected to a Desktop switch and next two Arduino microcontrollers are connected in parallel with its own ethernet cables.

6.5.1 Servomotor control with Arduino ethernet shield

It is possible to connect small servo motors directly to an Arduino Atmega 2560 (in this system Adafruit servos work between 3 to 5v), enough to control the shaft position very precisely. Servo motors use feedback to determine the position of the shaft. As a result, servo motors are used to control the position of objects, rotate objects, move legs, arms or hands of robots, move sensors etc. with high precision. Servo motors implemented are small in size, but torque is totally enough for this kind of open/close environment.

Most servo motors have the following three connections:

- Black/Brown ground wire.
 - Red power wire (around 5V).
1. Brown or Yellow PWM wire.

Knowing this, connect the power and ground pins directly to the Arduino Atmega 2560 5V and GND pins (one 1k ohm resistor could be enough to protect Servomotor from incoming voltage), many tests will be implemented. The PWM input will be connected to one of the Arduino's digital output pins.

In order to make the first servo test, Arduino's library provide a list of basic examples, (example: the servo motor will rotate slowly from 0 degrees to 180 degrees). When the motor has rotated 180 degrees, it will begin to rotate in the other direction until it returns to the home position, this example code will be implemented for a Servomotor controller, in order to have a user interface to control the rotation of the axis to keep open/close the main door.

A simple circuit diagram has been implemented in this project with one servomotor: Many of some available example programs for Arduino run with some unknown libraries and some of them are not compatible with Arduino IDE 1.0.5 (which is the only version available for Jetson TX2), to solve this:

- 1. Implement a simple push button interface to see the behavior of the code and then try to add on/off buttons.
- 2. Understand the webpage customization on Arduino environment to edit future applications.

Example: `client.println(«H1>Project: Servomotor controller</H1>»);`

Println data is used to return a newline character to the web server which is connected to.

- 3. Start with a webserver connection Knowing which Ip address is going to be set, in a environment with many pc equipments.

It is possible to have multiple Ip address running in the same area and even if your program is right, the ip address will be occupied in other PC, so the connection will not be possible.

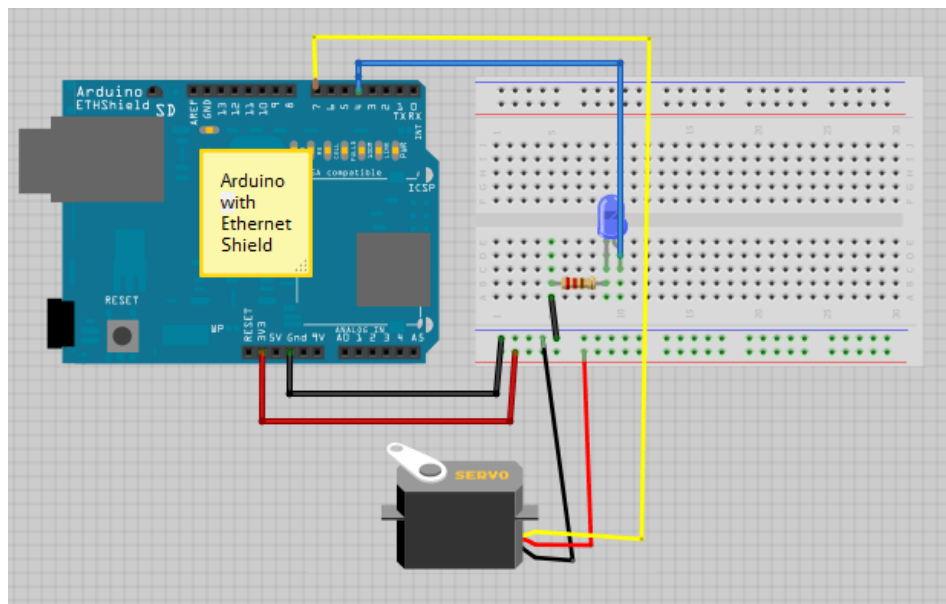


Bild 6.15. Ethernet shield with servomotor and led connected.

Position: 28



Bild 6.16. Servomotor slider webserver.

- 4. To check that send a ping directly to the Ip address desired (example: ping 10.44.0.83) if is not used then it can be set as Ip address in a webserver.
 - 5. Make a simple diagram on 'Fritzing' software, observe this in ►Bild 6.15, there are multiple Electronic design makers available on internet for Arduino.
 - 6. Start making a simple servomotor control with Arduino just to see the reaction of the axis (the reaction with TX2 serial communication takes some time compared with a conventional PC).
 - 7. To see the reaction of the servomotor to rotation, it is possible to create a webpage that contains a slider. Sliders are very useful interfaces to control not only rotation, velocity and position in servomotors, this interface must be related to the position values of axis and they must appear on webserver, observe this in ►Bild 6.16.
8. Make a push button interface to rotate *left/right*, this demo application is even more complicated, your servo variables must be related to the position of the axis (example: 0 to 180 grades).

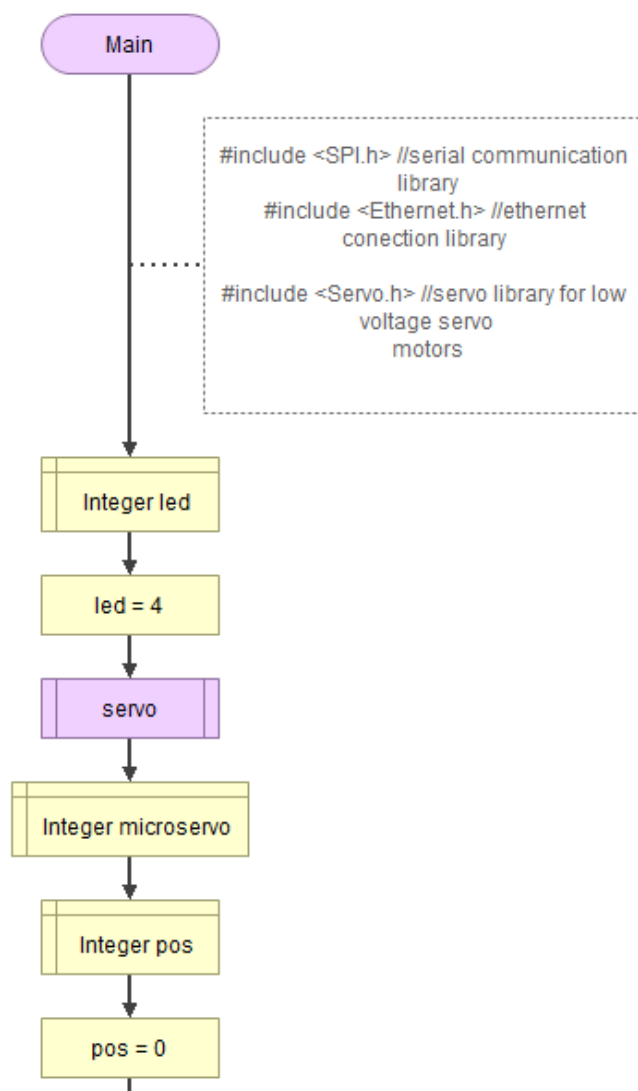


Bild 6.17. Libraries to implement servomotor and Ethernet connection.

It is possible to have multiple servomotors running with the same webserver the response will be in parallel, but they will not have the same time's response.

- In order to have connetion with the Webserver created, it is nessesary set an ip address in order to get connection to the webserver, the subnet mask is the class of network the server is connected to, ►Bild 6.18.
- Declare a port in order to obtain ethernet's conection, in many specific examples they use port 80 to stablish conection with webserver, ►Bild 6.19.
- To be connected to the local ip addresss it's nessesary to setup Ethernet client as the receiver of the webserver, observe this basic configuration on ►Bild 6.20.

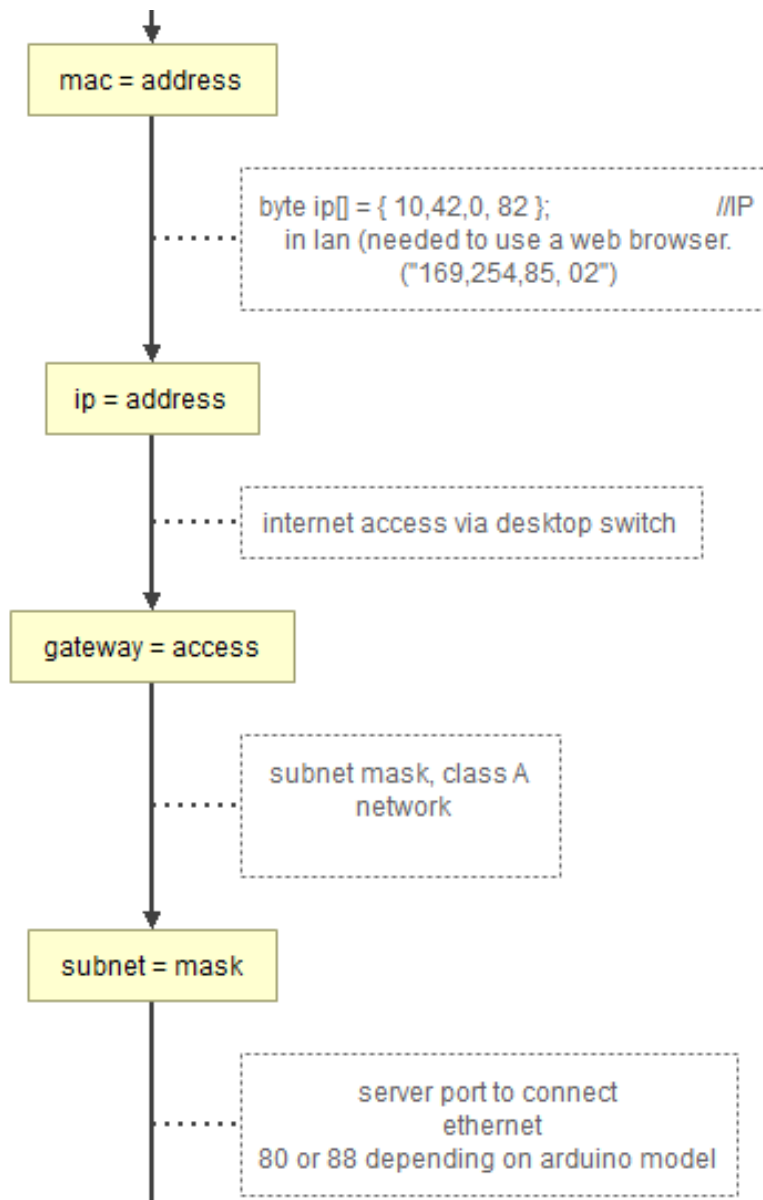


Bild 6.18. Ip address and Subnet mask.

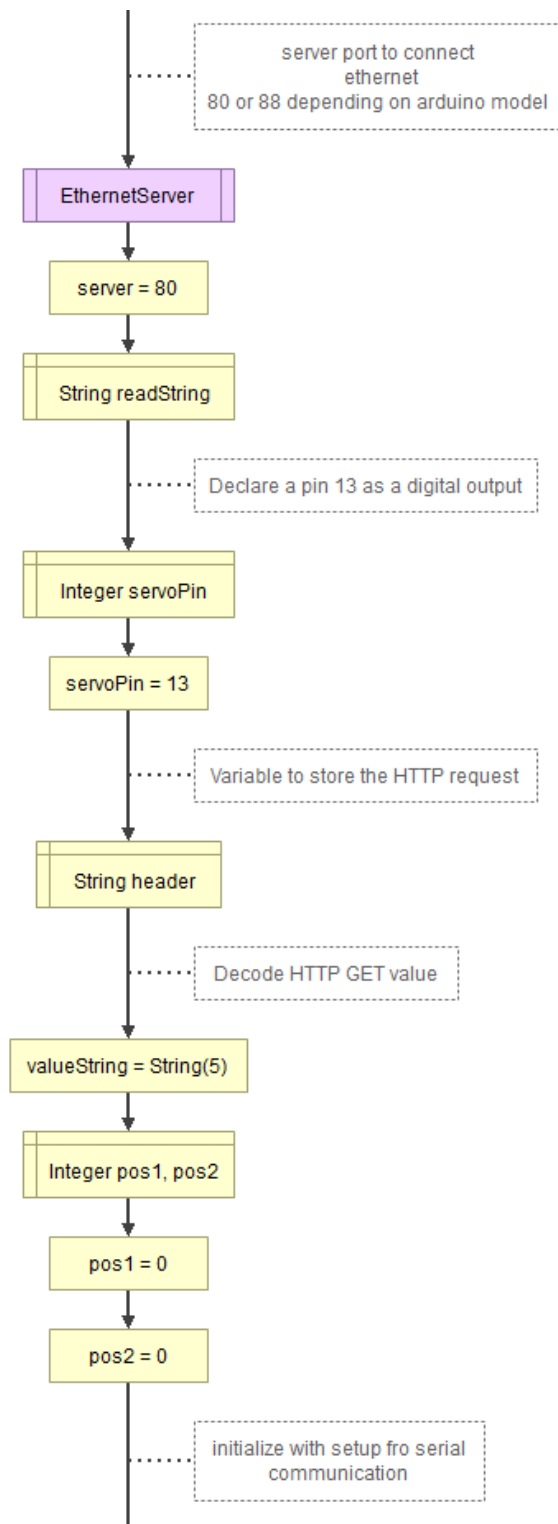


Bild 6.19. Ethernet port and variables declaration.

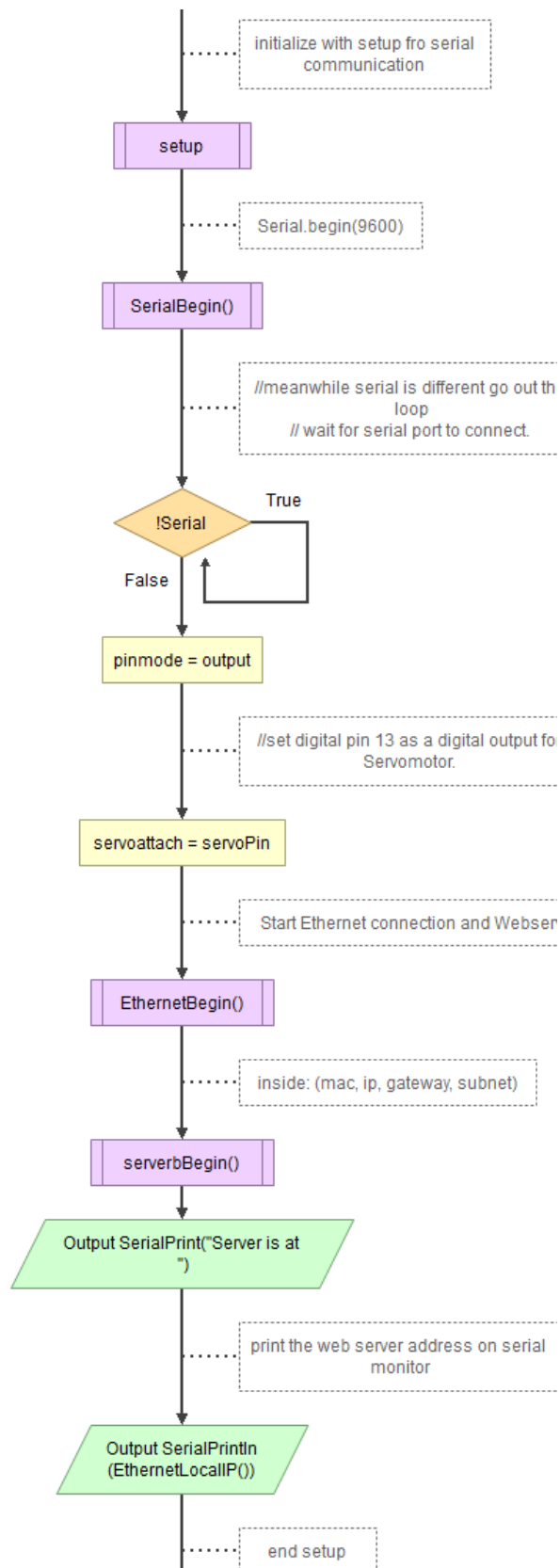


Bild 6.20. Webserver setup.

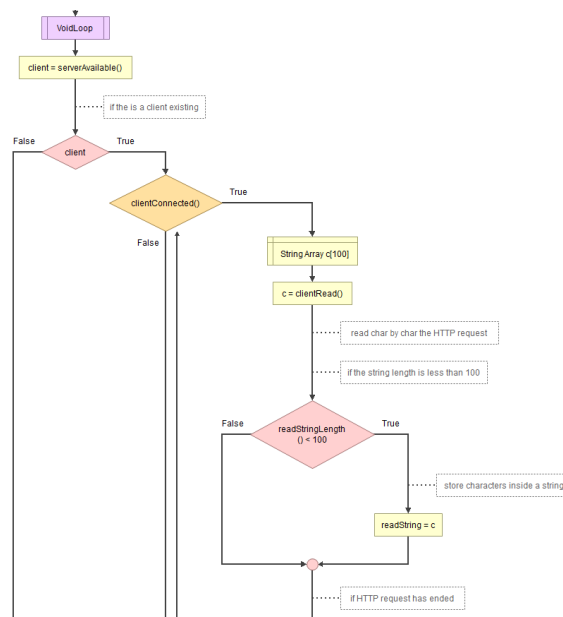


Bild 6.21. Connection with client and confirmation.

- Next step is to have a confirmation message from the client printing its ip address on serial monitor, ►Bild 6.21.
- There is a specific form to start with a HTML format for a webserver, it is necessary to create all the titles and spaces in order to have all the displayed information in order, within webpage, ►Bild 6.22.
- The end of the webpage must appear before the functions for the servomotor control started, ►Bild 6.23.

Creation of the slider: Normally it is possible to call different functions from the main loop in order to not have to much code within the main loop of the program, observe this in ►Bild 6.24.

- To have a reaction when a button is pressed a simple example is to make on/off buttons, in order to use digital outputs of the microcontroller, observe this in ►Bild 6.25.
- Knowing previous point, buttons left/right can be implemented in the program, using the servo object as a receiver to obtain the digital output values, observe this in ►Bild 6.26.

6.6 Temperature controller with Arduino ethernet shield

For this project a DS18B20 temperature sensor is been implemented as a detector for the optimal conditions in a refrigerated environment. DS18B20 is Wire digital temperature sensor from Maxim IC. Reports degrees in Celsius with 9 to 12 – bit precision, from –55 to 125

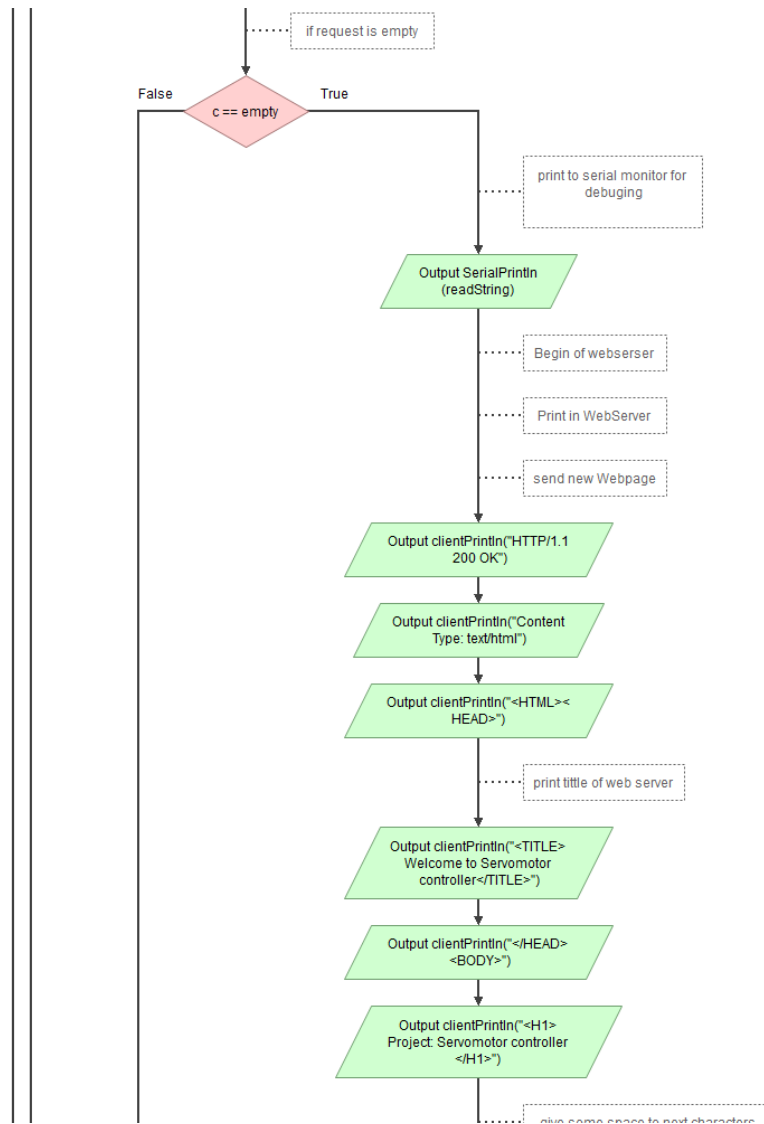


Bild 6.22. Create Webpage body.

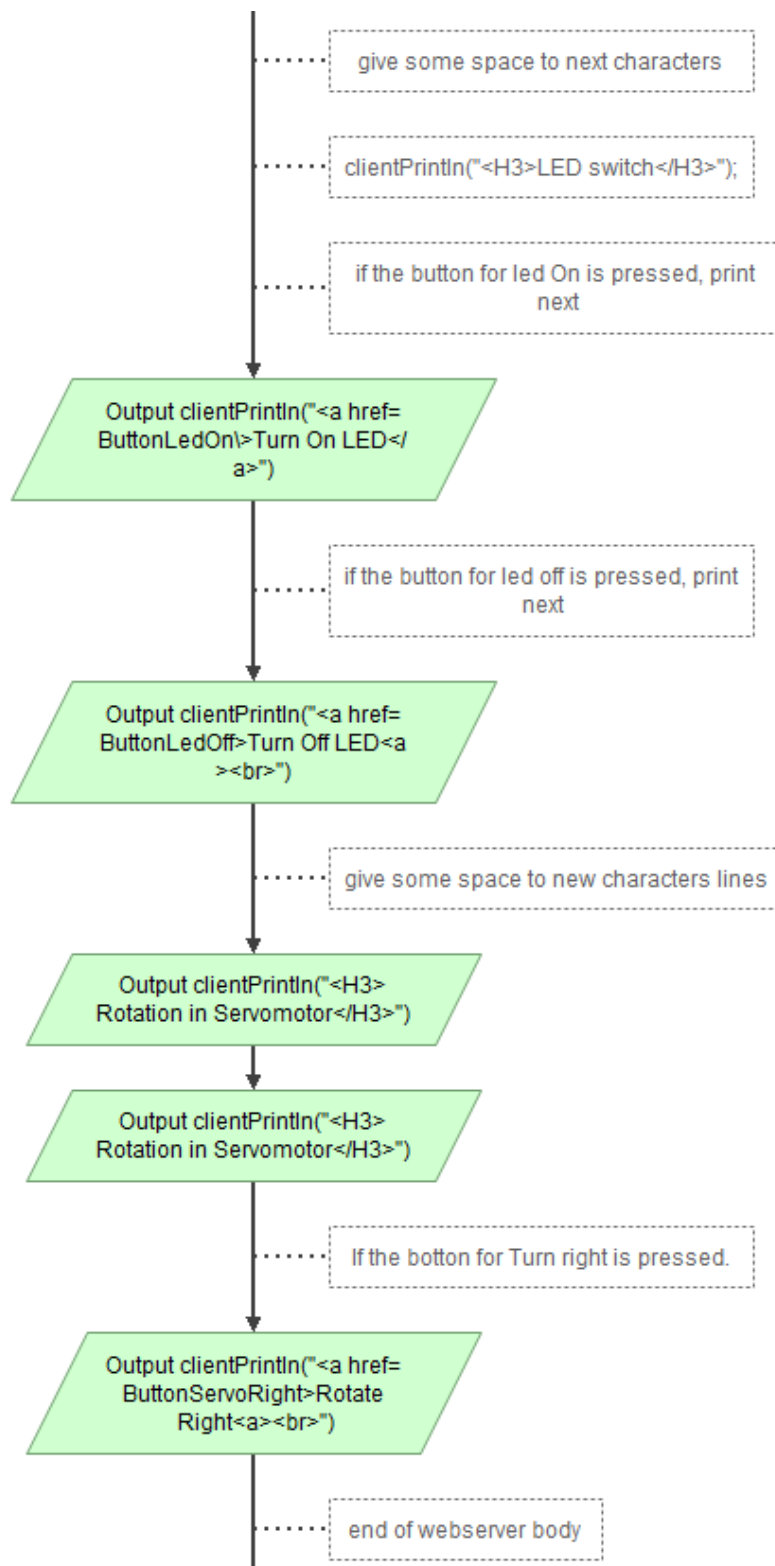


Bild 6.23. End of Webpage body.

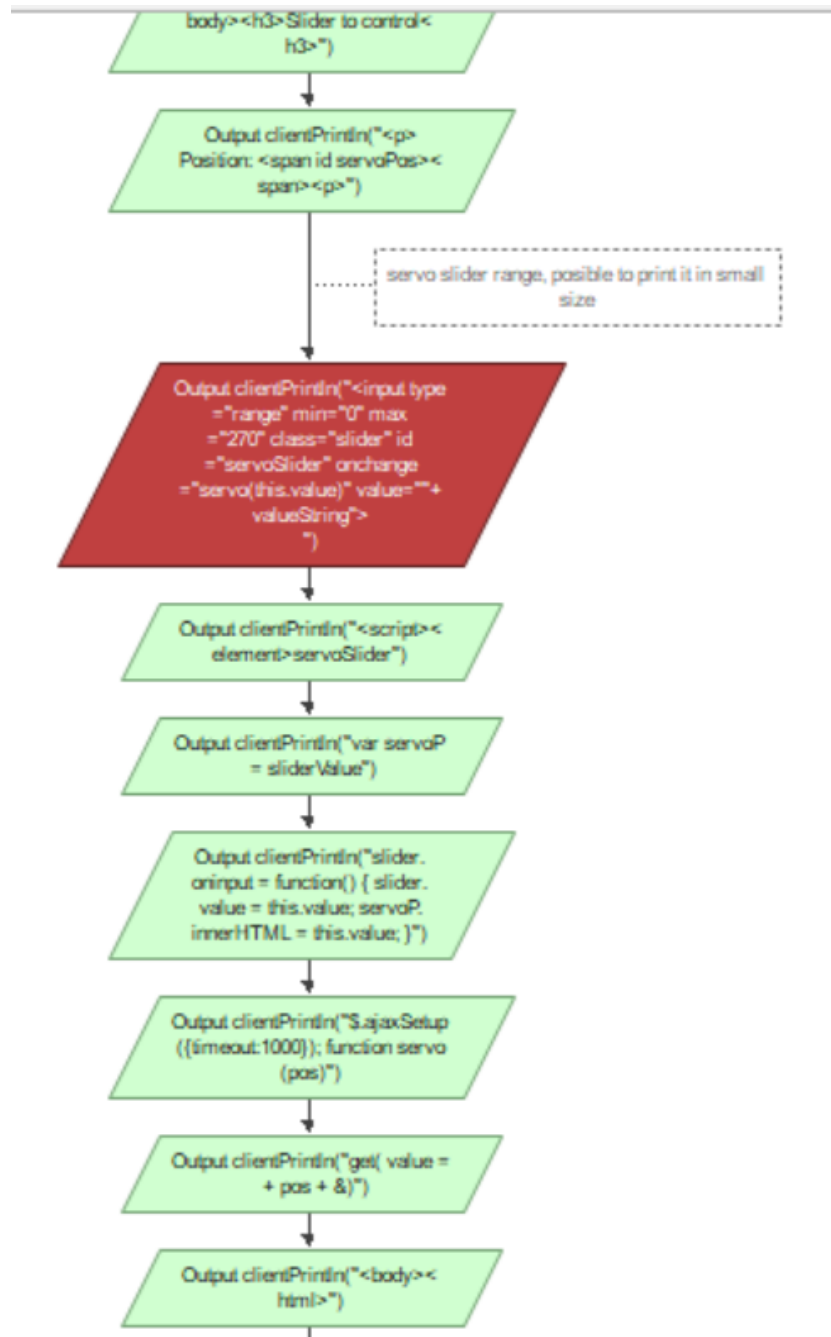


Bild 6.24. Slider body on webpage.

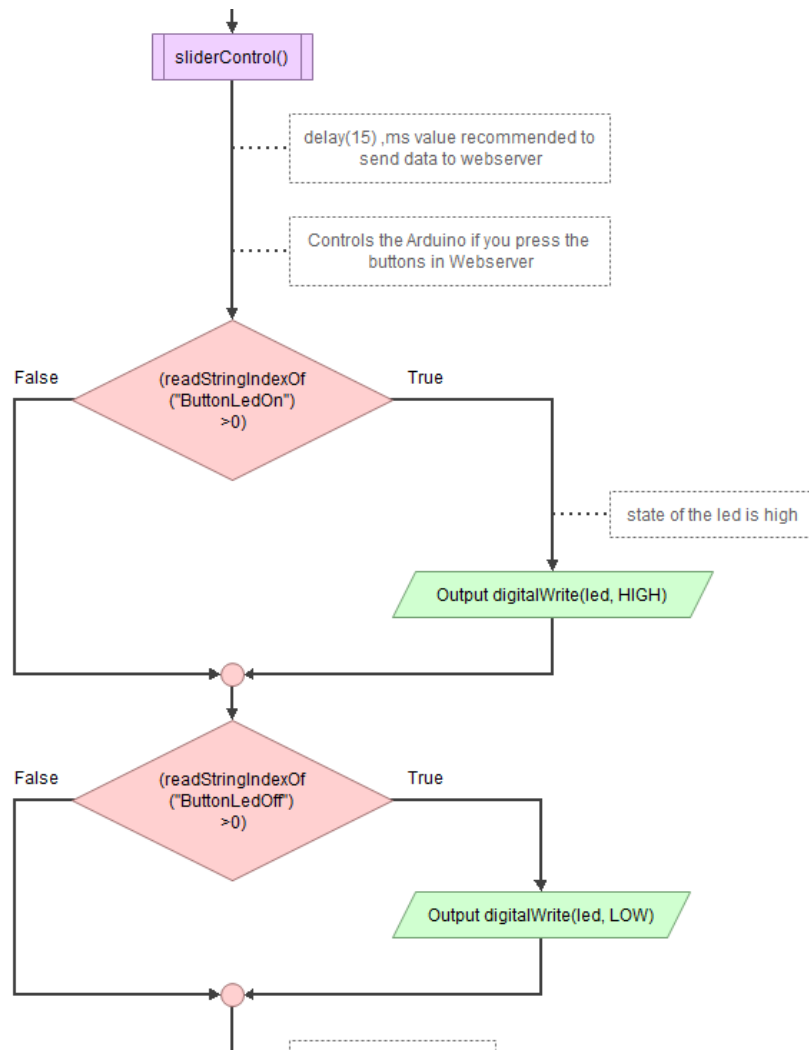


Bild 6.25. Slider body on webpage.

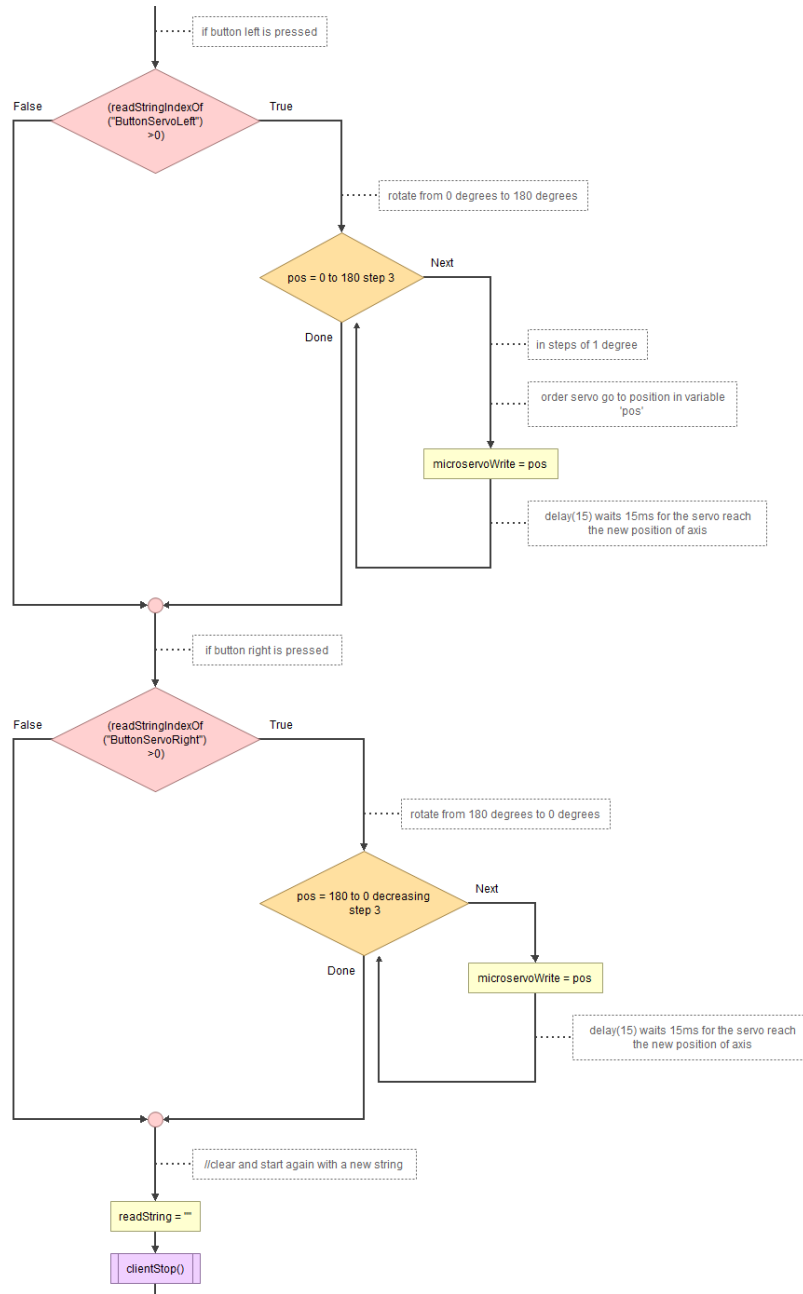


Bild 6.26. Buttons to rotate axis left/right.

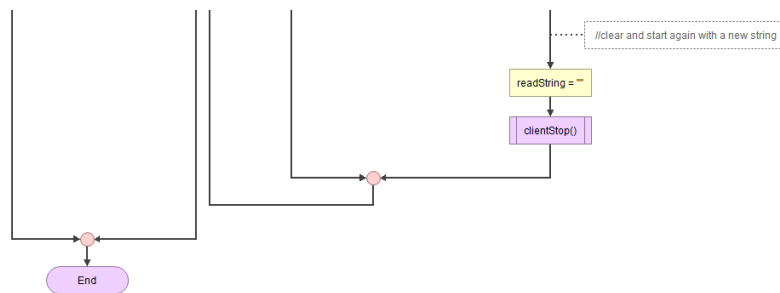


Bild 6.27. Client stop and end of the loop.

(± 0.5). Each sensor has a unique 64 – *Bit* Serial number attached into it - allows for a huge number of sensors to be used on one data bus. Some important features are the next ones:

- Port pin for communication.
- Power supply range: 3.0V to 5.5V.
- Range of measurements: $\sim 55C$ to $+125C$ ($\sim 67F$ to $+257F$) 0.5C accuracy from $\sim 10C$ to $+85C$.
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system.

Libraries for implementation: 1. Wire bus 2. Dallas Temperature

Dallas temperature library will take care of communicating with the temperature sensor. It's needed to tell the Arduino what pin the sensor is connected to. In this project digital pin 2 is connected to the sensor. Also it is necessary to create a DS18B20 object to have access to the sensor data.

A basic webserver allows to connect Arduino Ethernet shield to Jetson TX2, it's a simple application but practical to display sensor's values. If the connection is successful it will display the current temperature as read from the DS28B20 sensor. The browser will refresh every 3 seconds the web page to update the sensor's values. For ethernet communication:

1. Connect to the Arduino using the ip address its assigned in the sketch. This would be the private network in your location.
2. Add Arduino Ethernet library to communicate with the W5100 Ethernet controller. It communicates with the ethernet controller using the SPI bus ►Bild 6.28.
3. To allow Arduino to communicate with the ethernet controller include 2 libraries: 1.SPI.h is to enable the SPI interface to talk to the ethernet chip. 2.Ethernet.h which acts as an interface to the ethernet side of things within the controller chip.
4. Setup function is to initialize the ethernet controller and then start listening for clients to connect with them ►Bild 6.29.
5. In the main loop, if a client connection is detected, process the request looking for a blank line with a newline character at the end. If this is true then send a response.

6. Use a while loop to process each character received from the client. It checks the character value which is stored in a char variable called *c* to determine if it's a newline character.
7. During the process of sending the response of the temperature sensor using *ds.getTemperature_C()* function, it is possible also to use *ds.getTemperature_F()* for Fahrenheit ►Bild 6.32.
8. Tell the browser that the response is *text/html* using the Content Type: *text/html* header. Refresh: 3 to tell the browser to reconnect to the Arduino every 3 seconds to read the temperature again.
 - The structure of this main loop the same as the servomotor's webserver, it is necessary to the client to stay connected with the webserver at any time to receive messages from ►Bild 6.30.
 - The webpage is implemented for display temperature values on webserver, the structure is even more simple than the servomotor controller. This webserver displays new values of temperature sensor each 5 seconds or if the user desire have new values, it is possible to refresh the webpage to have new values but starting again with the webserver ►Bild 6.31.
 - In order to print the values of temperature on this webserver it is necessary to have some functions that the Dallas library provide for this class of detection, some functions obtain the value of temperature in celcius and fahrenheit , there is an specific format to print this values, observe this in ►Bild 6.32 and ►Bild 6.33.
 - To alert the user the temperature is increasing or decreasing, it is possible to define some conditional in order to have a range for the desired temperature values ►Bild 6.34.
 - After all the process, the body of the webpage is completed and the loop is closed for next incoming connections, they could start with a new space or with a new character on space ►Bild 6.35.
 - If there are not incoming connections from the webserver the client will stop in a short period of time and it's necessary to start again the webserver.

6.7 Clock Synchronization algorithm in Webserver

There is a special order to send and receive messages from webserver and then to make a response from client, the purpose getting the average in the same compare with socket in Linux, but the structure of the program change because there is the client and server response in the same infinite loop, observe this in ►Bild 6.37.

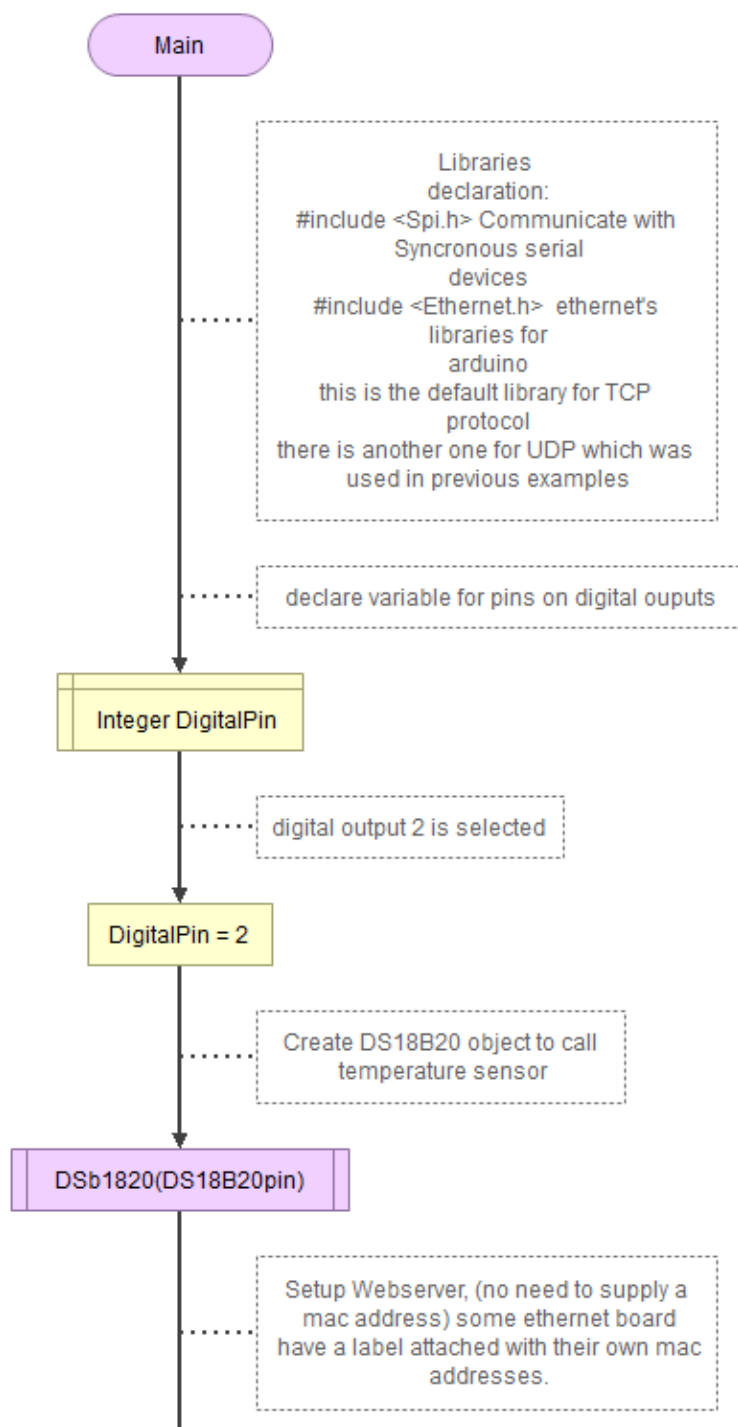


Bild 6.28. Libraries declaration and port designation.

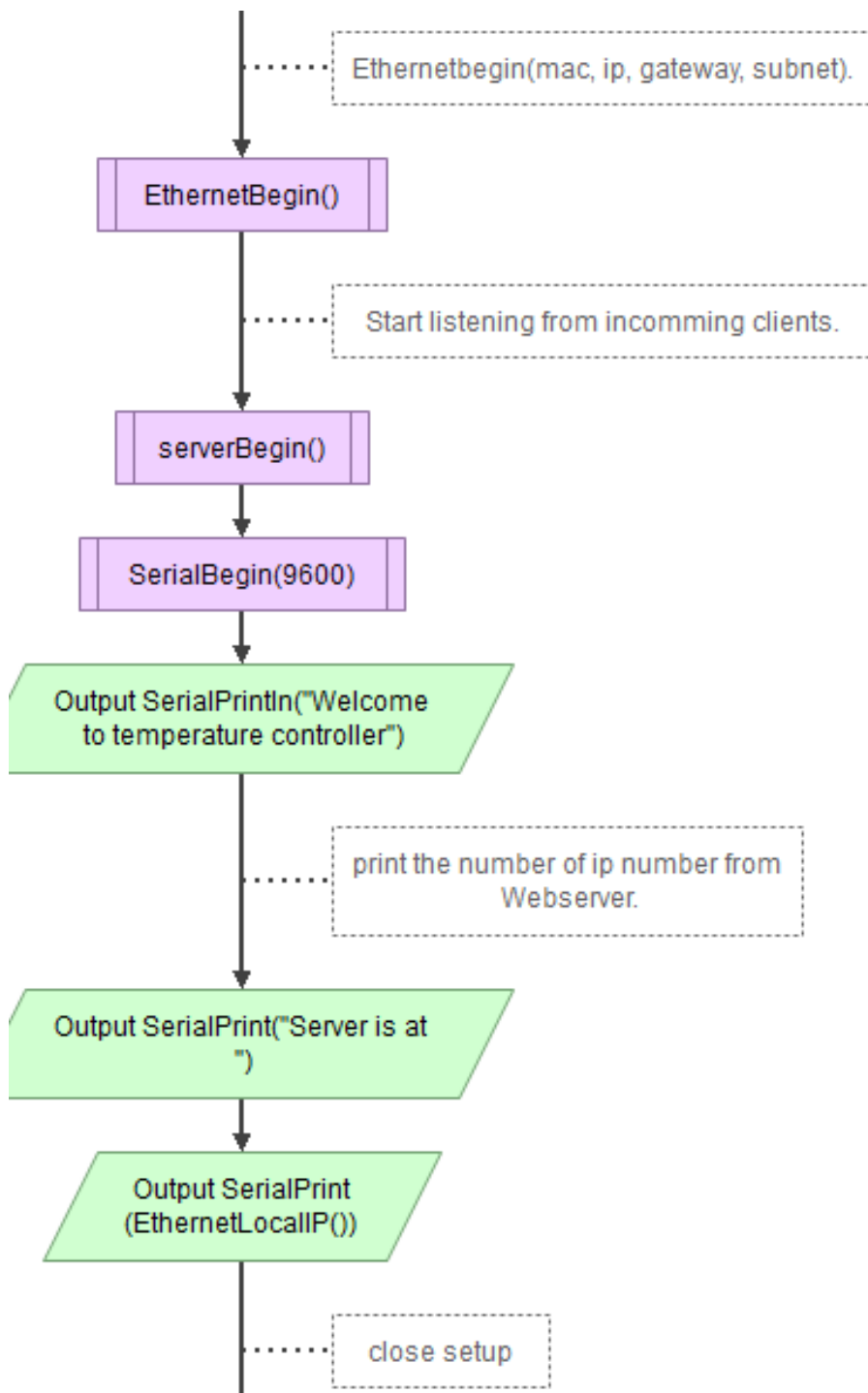


Bild 6.29. Setup and start listening for incoming connections.

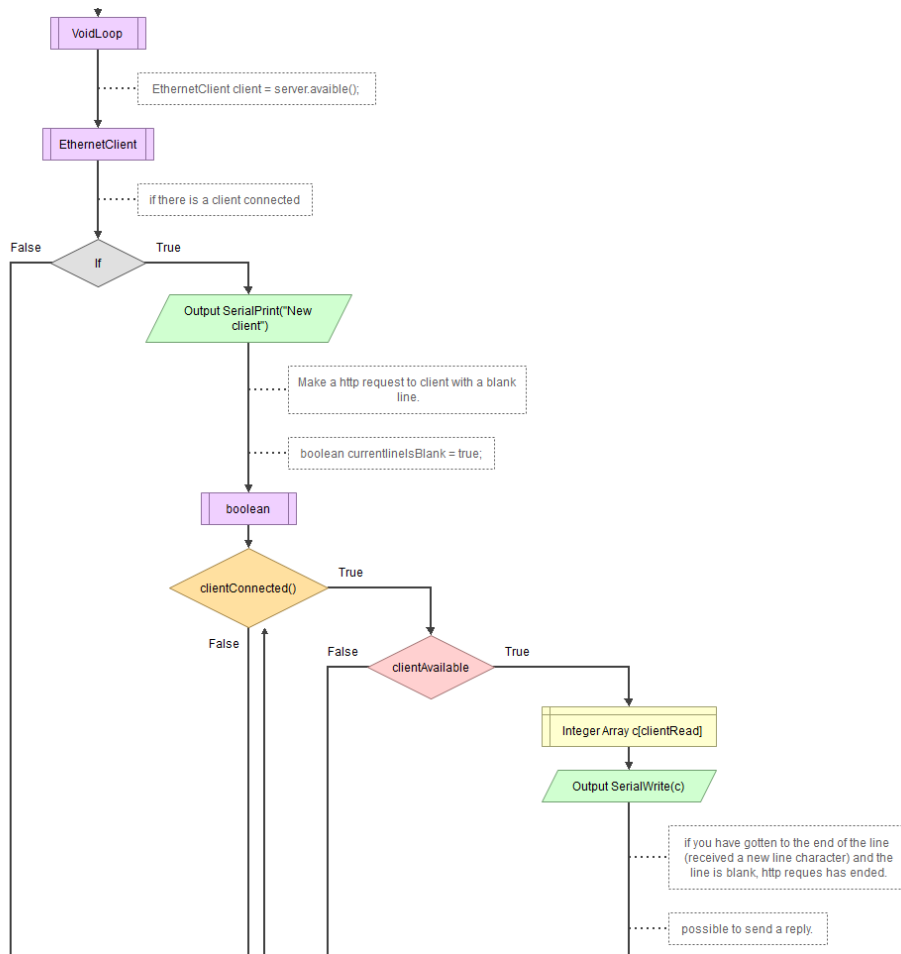


Bild 6.30. Main loop structure and Client available.

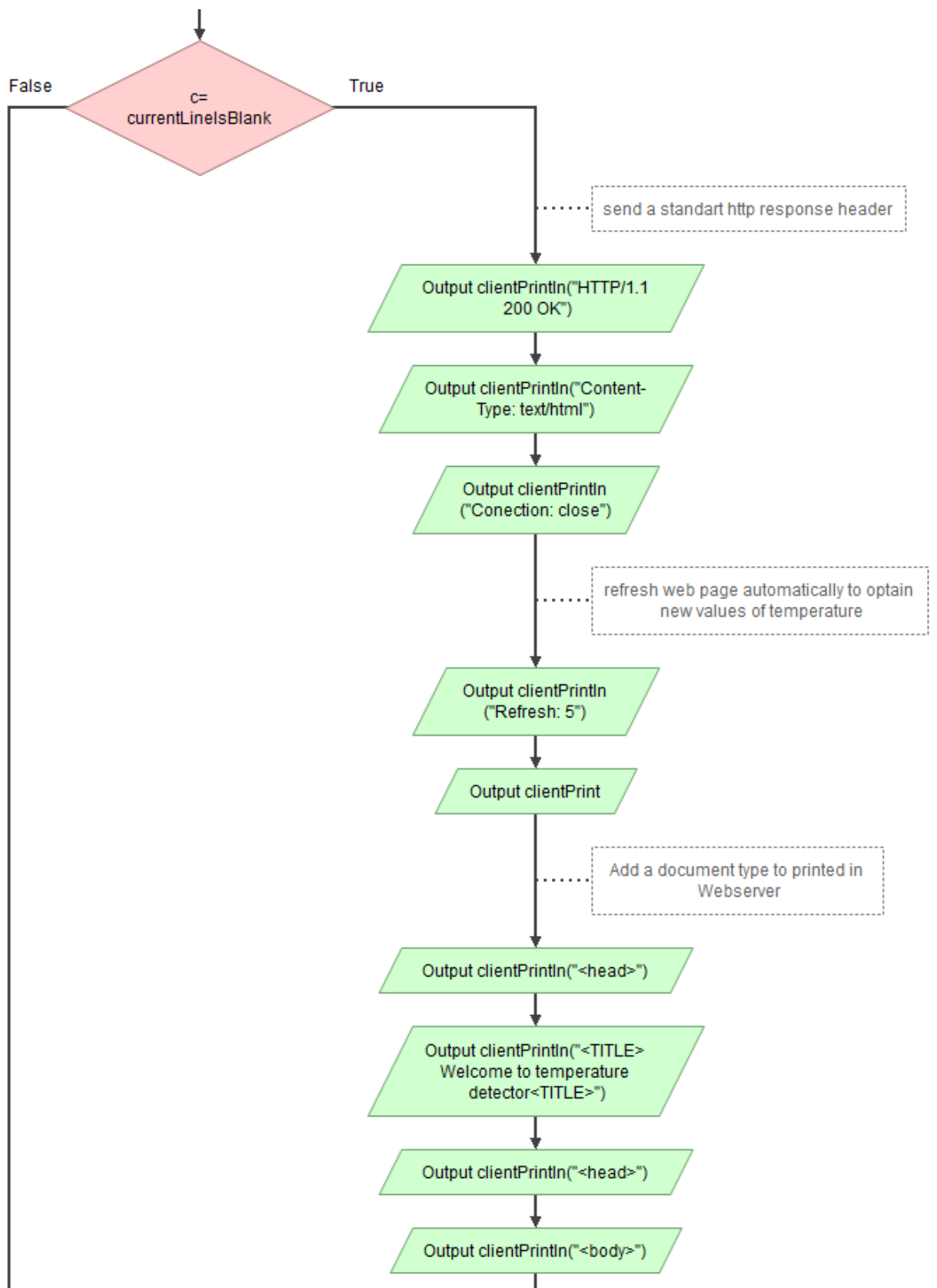


Bild 6.31. Beginning of the webpage.

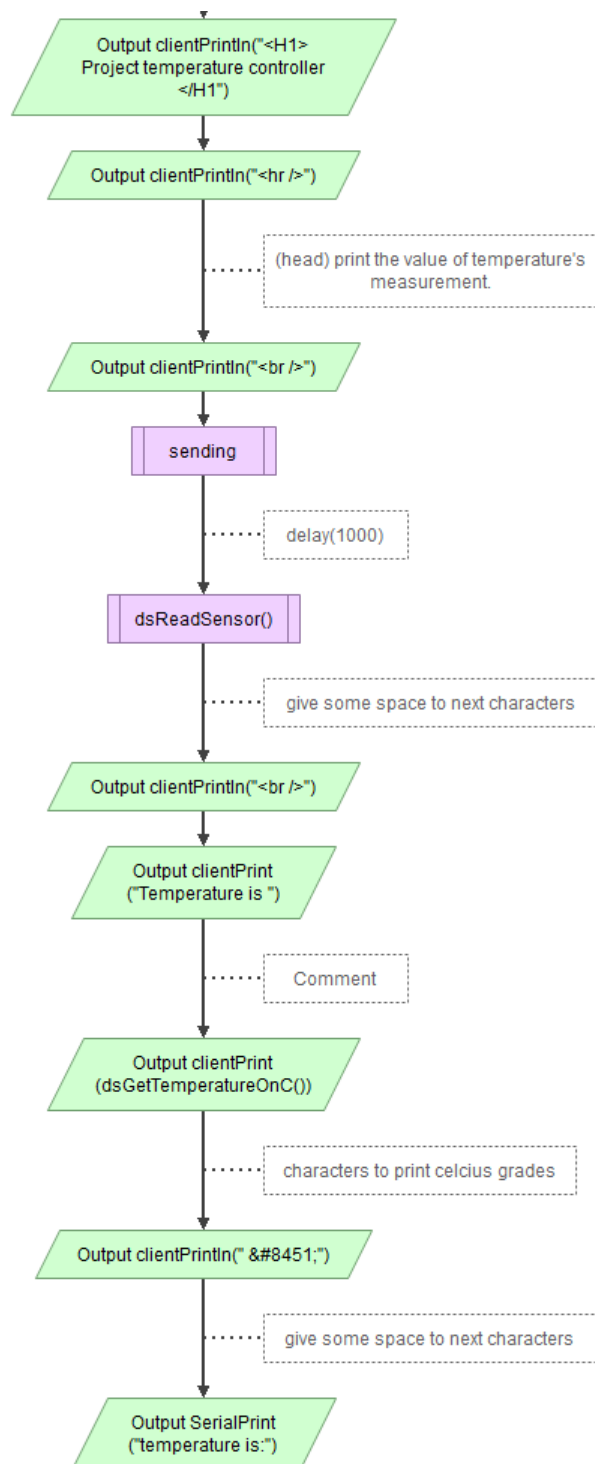


Bild 6.32. Printing the values of the temperature controller on webpage.

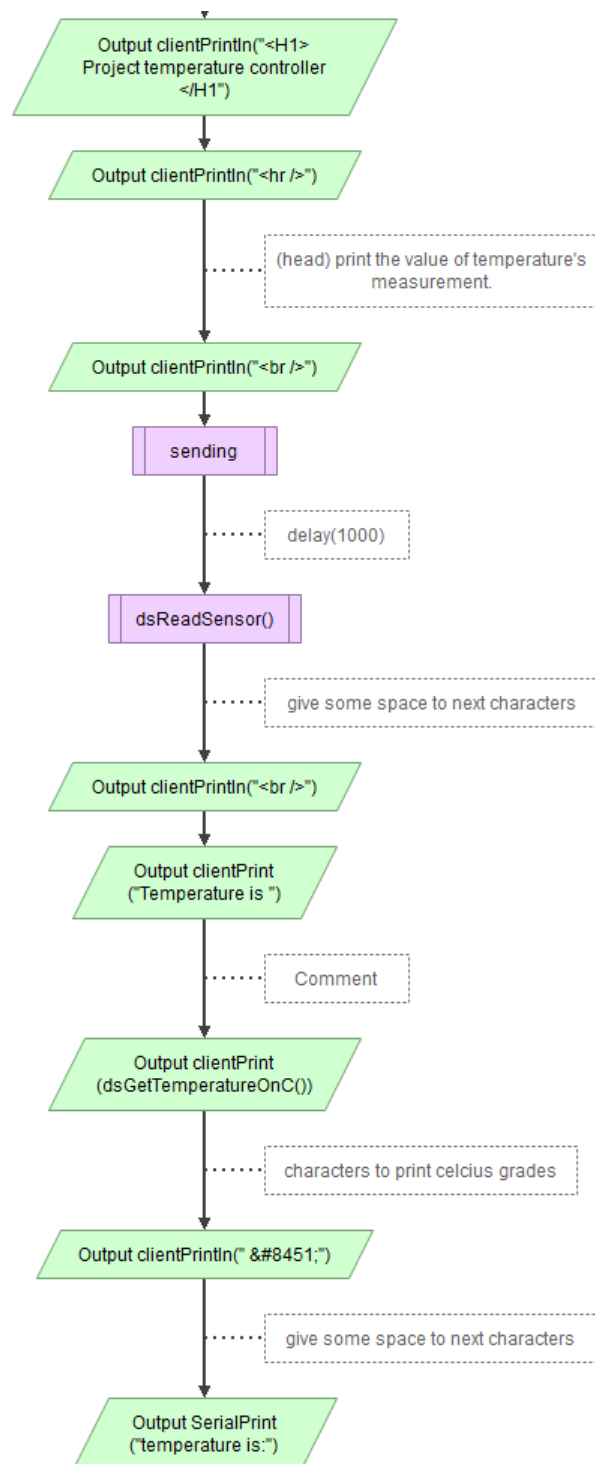


Bild 6.33. Printing the values of the temperature controller (Fahrenheit).

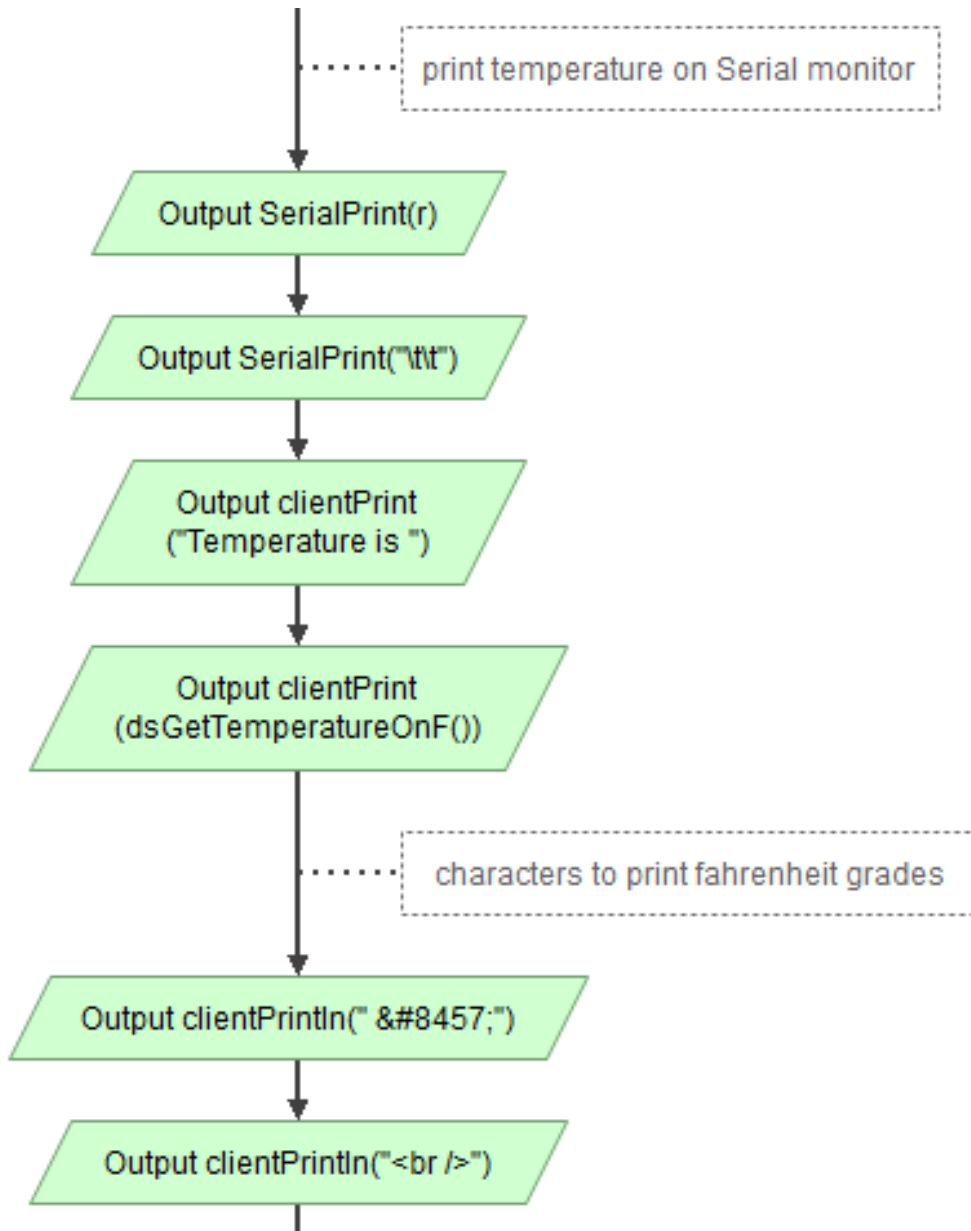


Bild 6.34. Printing Alert messages for temperature increasing/decreasing..

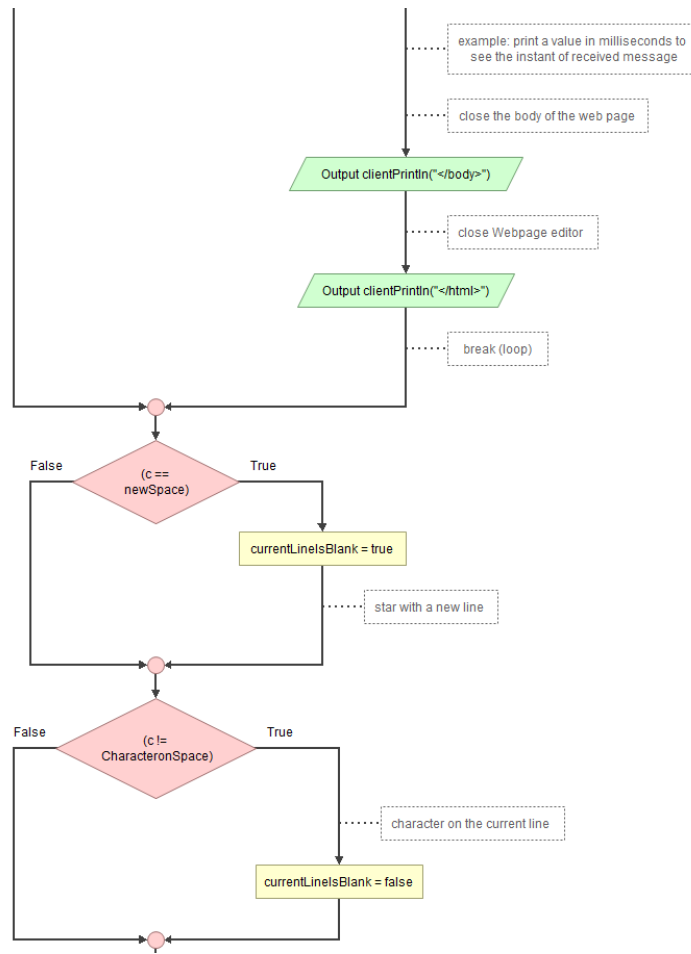


Bild 6.35. End of the webpage and state of the current space.

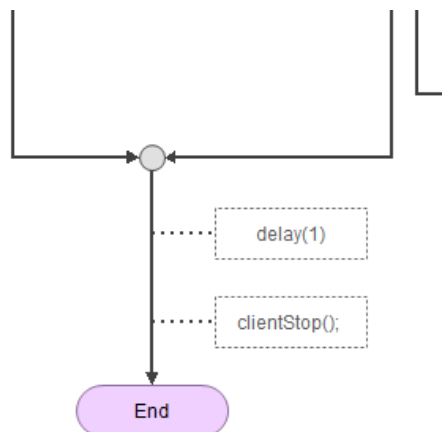


Bild 6.36. Client stop and end of the program.

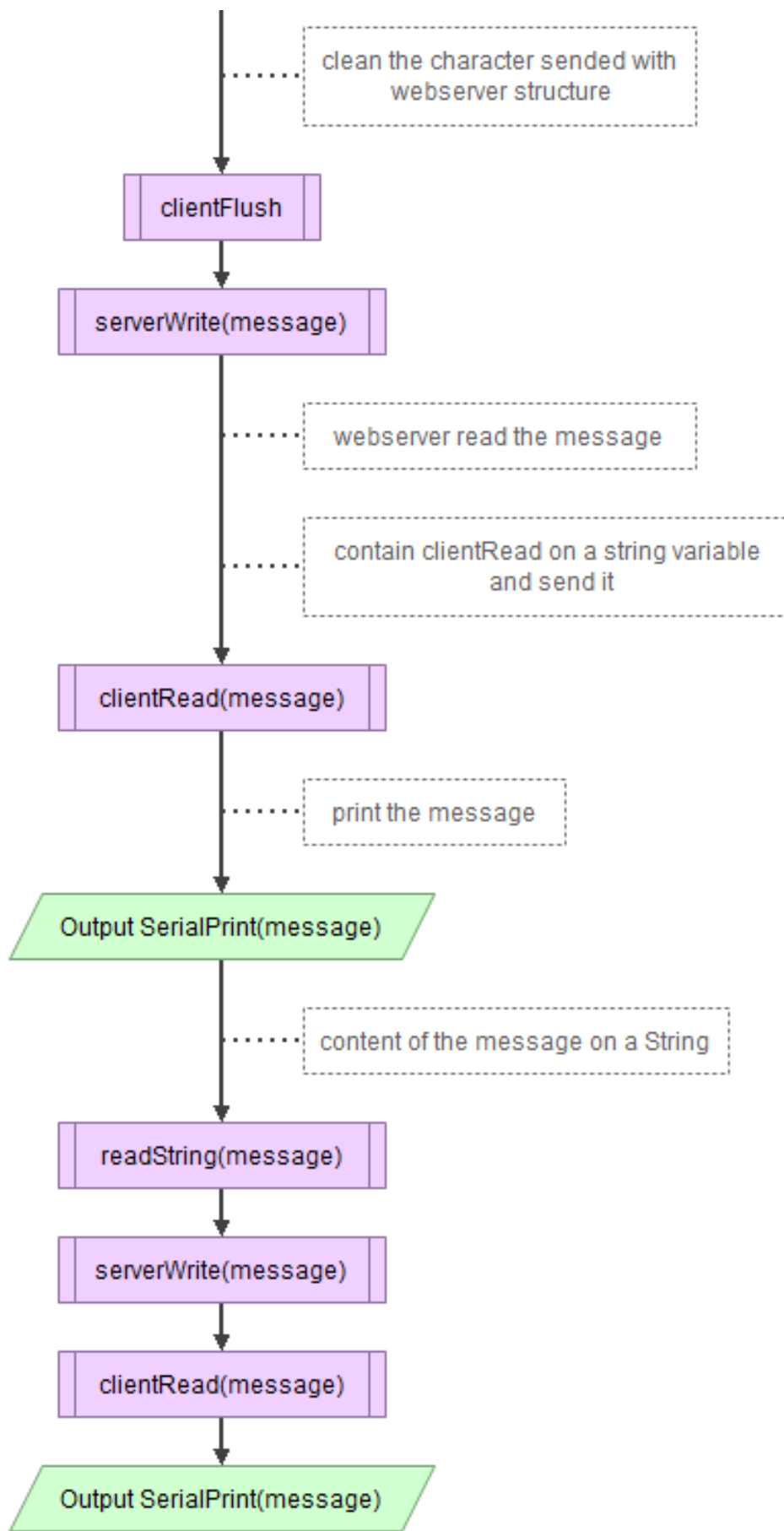


Bild 6.37. Sending and receiving messages from webserver to client (serial monitor).

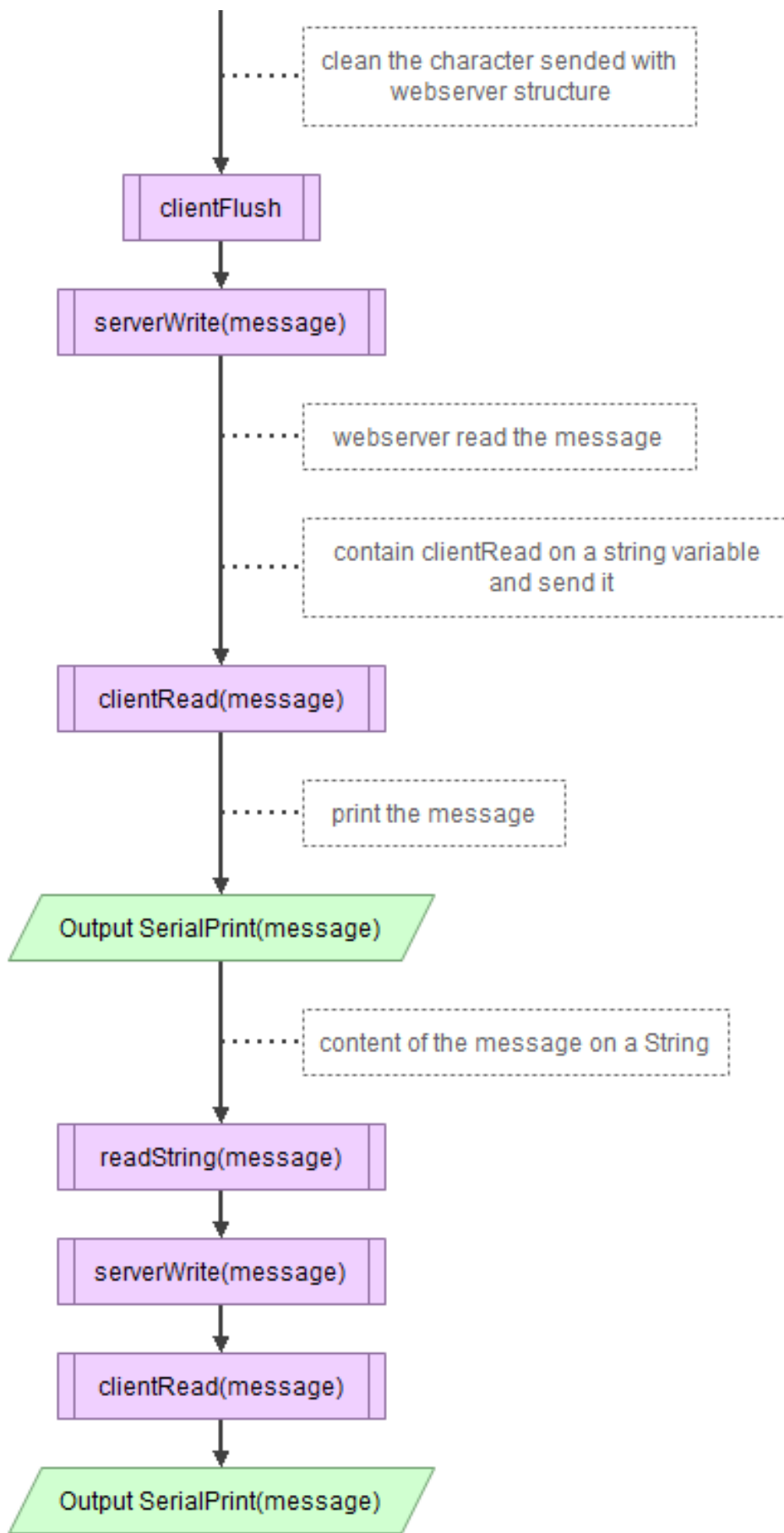


Bild 6.38. Calculus of the average in Arduino and Webserver.

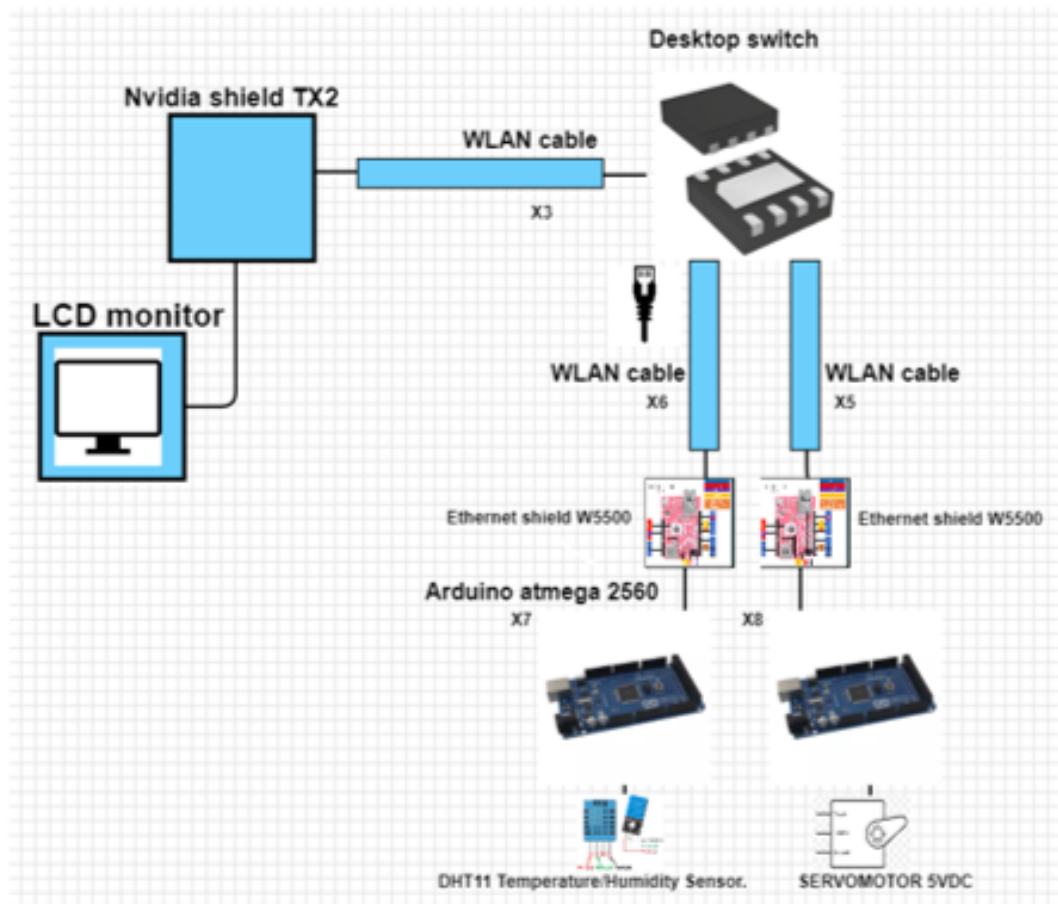


Bild 6.39. Functioning diagram.

6.8 Clock Synchronized embedded system's functioning diagram

In order to clarify which environment is going to be implemented for a Clock Synchronized embedded system a Functioning diagram is required, observe this in ►Bild 6.39. Many web pages provide to user many useful interfaces to design your own Functioning diagrams with the purpose to buy the list of materials or devices needed, for this diagram 'digikey.com' was implemented as electronic web designer.

The system is composed of 2 Arduino Atmega 2560 which are programmed with different functions, one is a temperature controller and the other is a servo motor controller. The purpose of this microcontrollers is to be synchronized with the server (Jetson TX2) at the same time they are running.

Implementing the same synchronization algorithm that was used for the Socket from Jetson TX2 to Ubuntu PC, the speed of information processing varies in comparison with the communication to a PC because the libraries that are used within Boost.asio are designed for a communication based on the TCP communication protocol, in addition to the speed of the processor and the large amount of information that can be sent.

7 Experiments and results

7.1 Socket chat server

To obtain the results with the Synchronization Algorithm created in Ubuntu, it was necessary to make multiple communication tests with a socket created from a client / server chat, this chat provides server a response from the client, but not a response of the client from the server. The initial program was designed for the server could write a message through a keyboard, the client read this message and send a response. Those were the bases that we implement to obtain the automatic message communication between server and client, observe this in ►Bild 7.1.

The previous Chat server was structured as an automatic client server communication, this was possible since messages can be written to the client from the server, the client is able to read the messages (read the bytes inside) and store the messages within a variable, this variable can be transferred to a function which is Write a message to the server. It does not have as much complexity since the number of characters stored within the message sent must be the same as the one received, that is why all messages have the same size.

It's necessary to be sure that the content of the characters chains is exactly the same if you send or receive, because an error within the program has the consequence that the message is not printed correctly in one of the two terminals, even knowing that the server program is correct it is possible that the client is receiving bytes with unknown content and for that reason is not able to recognize the message sent.

1. One of first tests was performed in the socket structure is send a counting value from server to client this is possible with the help of a 'Boost deadline timer' which was the responsible of make a delay between each iteration this was the beginning to start with an automatic communication between client and server, observe this in ►Bild 7.2.
2. With the Boost libraries it is possible also to send the date and time from server to client, that's a common application on Boost socket not usefull to synchronized client/server, but usefull to understand how to send entire strings from server to client ►Bild 7.3.
3. This message is contained in a string which provide the lenght of the string, all the characters of the message and all the numbers included in the entire date/time message ►Bild 7.4.

Making this previous experiments about socket's structure ang how is the communication between client and server, it is possible to start with the performing of the Clock Synchronization Algorithm in the Socket.

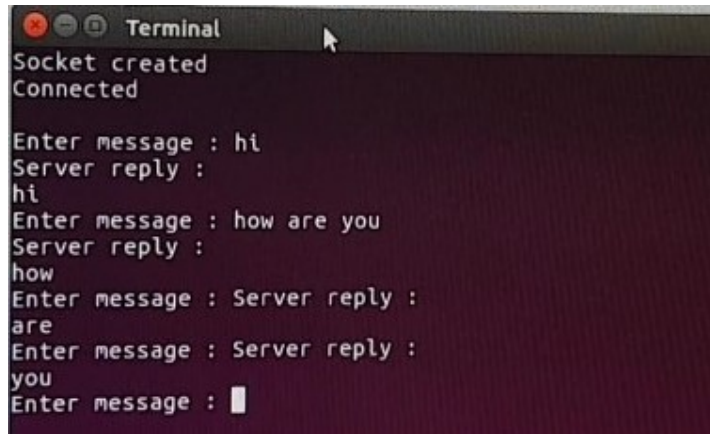


Bild 7.1. Socket chat server.

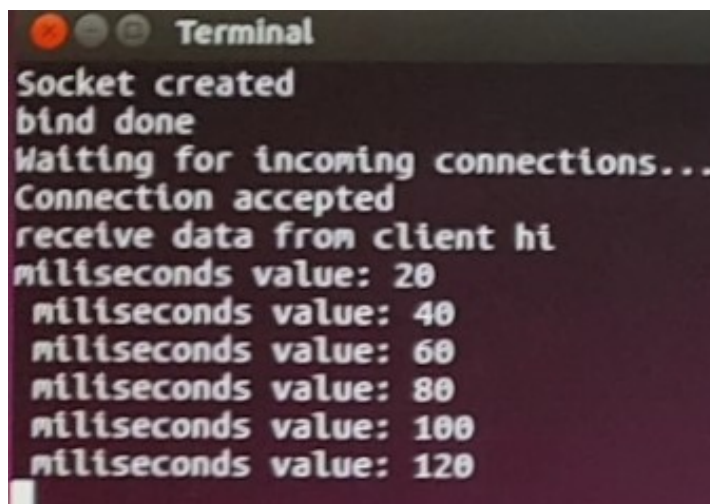
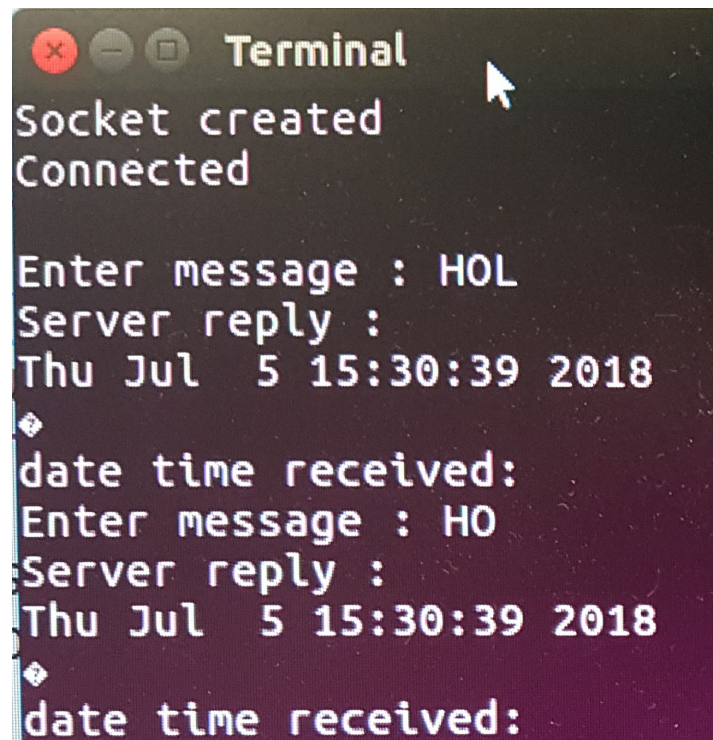


Bild 7.2. Sending counting values from server to client.



```

Terminal
Socket created
Connected

Enter message : HOL
Server reply :
Thu Jul  5 15:30:39 2018
◆
date time received:
Enter message : HO
Server reply :
Thu Jul  5 15:30:39 2018
◆
date time received:

```

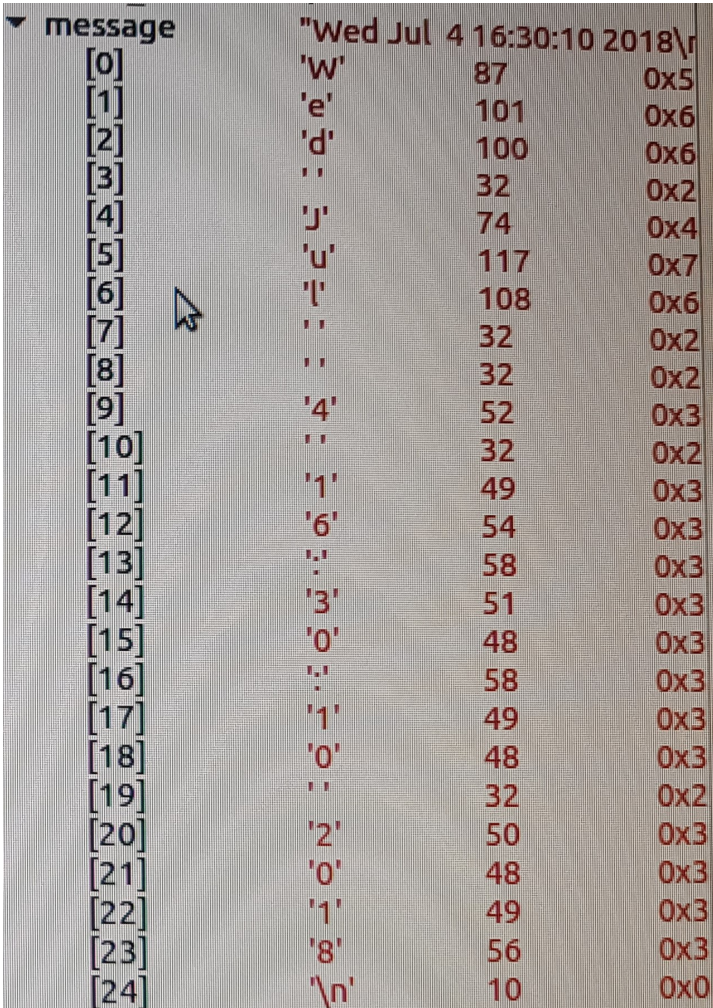
Bild 7.3. Sending date from server to client.

7.2 Socket with a Clock synchronization algorithm on Linux environment

These results show the sending and receiving of the average from the Clock synchronization algorithm performed, these values are generated from the average obtained from the actual value sent plus the actual value received divided by the number of processes inside the system. It is a mathematical operation that has no complexity. The size of messages is longer because the number of decimals inside of arrays, they cannot be reduced to its minimum expression because if it is reduced the remaining spaces will appear with an unknown message or unknown addresses, that is not very esthetic and the value is not perceived in the same way, so it is possible to reduce the number of characters in this way and have a real value with a more adequate size. In the following result see there is only the result of the sending and the receiving of the average which are the two most important messages to know if the user obtained a correct result with the synchronization algorithm observe this in ►Bild 7.5 and ►Bild 7.6 .

7.3 Webserver with Servomotor controller

The result of this webserver is produced by a socket structure in Arduino environment, the process to accede to the webserver is simple, and the connection client/server is even faster than in Linux environment because is a direct ethernet communication and the webserver can be easiest recognized by client (Arduino environment). The harder part to understand is the



A screenshot of a debugger's memory dump for a variable named 'message'. The dump shows a sequence of 25 elements, indexed from [0] to [24]. Each element contains a character, its decimal value, and its hexadecimal value. The characters form the string 'Wed Jul 4 16:30:10 2018\r\n'. A mouse cursor is visible over the character 'l' at index [6].

Index	Character	Decimal Value	Hex Value
[0]	'W'	87	0x5
[1]	'e'	101	0x6
[2]	'd'	100	0x6
[3]	' '	32	0x2
[4]	'J'	74	0x4
[5]	'u'	117	0x7
[6]	'l'	108	0x6
[7]	' '	32	0x2
[8]	' '	32	0x2
[9]	'4'	52	0x3
[10]	' '	32	0x2
[11]	'1'	49	0x3
[12]	'6'	54	0x3
[13]	'.'	58	0x3
[14]	'3'	51	0x3
[15]	'0'	48	0x3
[16]	'.'	58	0x3
[17]	'1'	49	0x3
[18]	'0'	48	0x3
[19]	' '	32	0x2
[20]	'2'	50	0x3
[21]	'0'	48	0x3
[22]	'1'	49	0x3
[23]	'8'	56	0x3
[24]	'\n'	10	0x0

Bild 7.4. String with date and time.

```
l
Socket created
bind done
Waiting for incoming connections...
Connection accepted

  Sending average:
1000

  Reply from client(average):
1000

  Sending average:
2000

  Reply from client(average):
2000

  Sending average:
3000

  Reply from client(average):
3000

  Sending average:
4000

  Reply from client(average):
4000

  Sending average:
5000

  Reply from client(average):
5000
```

Bild 7.5. Results of server's terminal with a clock synchronized response.

```

Socket created
Connected

Received average from server: 1000 sending average: 1000
Received average from server: 2000 sending average: 2000
Received average from server: 3000 sending average: 3000
Received average from server: 4000 sending average: 4000
Received average from server: 5000 sending average: 5000

```

Bild 7.6. Results of client's terminal with a clock synchronized response.

serial communication with Arduino and how the webserver can give a response to the client. All the webpage was designed to control the servomotor rotation and know the exact position of the axis in any instant of time. To understand how the response from webserver to client, is the first test was performing a simple push button to turn on/off a led connected to a digital output, with that structure of the program it was easiest to start with the servomotor controller in order to rotate left/right 180 grades in case is needed, for this function two buttons were designed on webserver. All the webserver must have their own ip address and it's necessary to specify it in the program. This webpage was designed with HTML structure to simplify its functioning, observe this in ►Bild 7.7.

7.4 Webserver with Temperature controller

The Webserver displays the temperature values obtained from the temperature sensor, those real values can be transferred to the webserver with some different functions from the sensor library repository, this functions provide just the analog lecture from the sensor, there is an specific format to transmit these values to a webpage and it is necessary to identify these formats to print on webserver the temperature value in Celsius and Fahrenheit. Also, there is an application to identify the specific instant of time in milliseconds the lecture was realized to identify how much time the sensor takes to read the actual temperature value displayed. Finally, a counter was performed to know which temperature value has the highest or lower temperature in the environment, observe this in ►Bild 7.8 .

7.5 Clock synchronization algorithm on Webserver

Finally the result of the Clock synchronization algorithm on Webserver with Servomotor controller was obtained with the structure of a Websocket in Arduino environment, one of the main advantages of this socket is that the values of this sending and receiving messages can be perfectly observed in the same screen opening the webserver and the serial monitor in Arduino



Bild 7.7. Results of webserver with Servomotor controller.

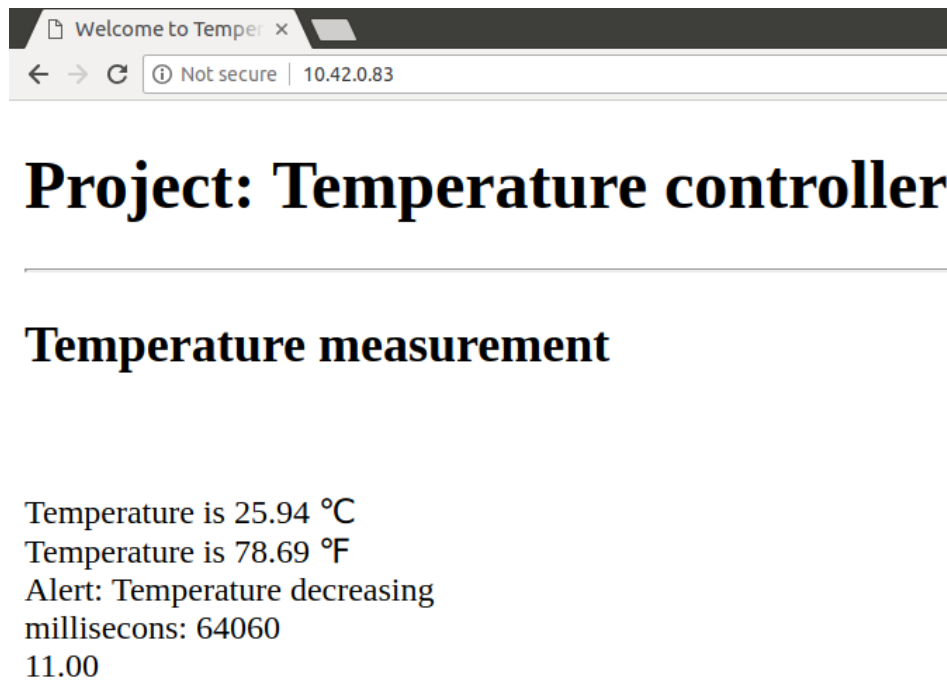


Bild 7.8. Results of webserver with Temperarure controller.

software, both are related, so if the webserver displays a message to the client, We must open the serial monitor to send a response to the webserver and have communication. This Clock synchronization algorithm display 4 values in serial monitor and 4 values in webserver, 2 values represent the sending and receiving actual values and the other 2 values represent the sending and receiving of the average. This application takes more time to be performed because serial communication on Jetson TX2 is slower than ethernet communication, but the functioning is the same compared with the Socket in Linux environment. Both webserver have the same displayed messages and the same values for the average obtention, with these values displayed, it is possible to guarantee that client and server are properly Clock synchronized, observe this in ►Bild 7.9.

The main difference between these two webserver is only that in servomotor controller there are buttons and sliders as effectors. The purpose of the temperature controller is to display temperature values, time of actual values, and number of event, also all the structure of the webpage in by default is printed ►Bild 7.10.



Bild 7.9. Results of Clock synchronization on Webserver and client in Serial monitor.

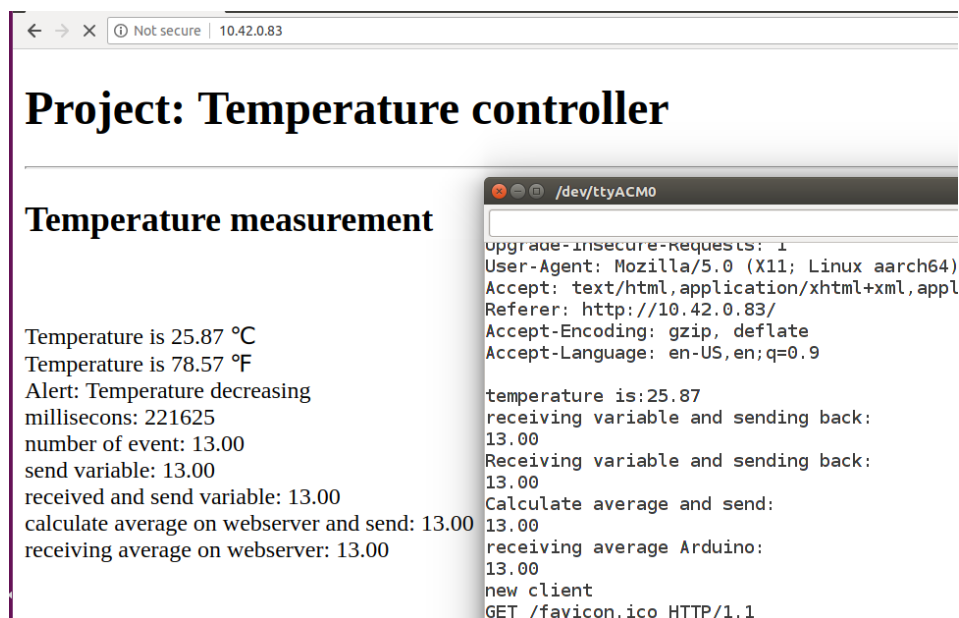


Bild 7.10. Results of Clock synchronization between Webserver with Temperature controller and client in Serial monitor.

8 Conclusions

Over time has been the questioning of the different actions that can be programmed in a microcontroller, this represents an expenditure of time necessary, within a mechatronic system you can work with one or more microcontrollers for the design of a control plant for a factory, company or a social environment. The use of one or more synchronized microcontrollers is a frequent necessity that sometimes it is required to perform different programmed actions in different devices which have a common purpose in a close loop of time, the reaction of these microcontrollers must be often immediate, and the time is vital for the conservation of different devices and / or materials. Flexible clock synchronization for a knowledge-based reasoning system would be the solution to this type of needs and many others in which time is a key factor for the actions of some microcontrollers, working together in the same mechatronic system.

- [1] Artificial intelligence,
<https://www.techopedia.com/definition/190/artificial-intelligence-ai> [visited at 13. 04. 2018]
- [2] Jean-Claude Latombe, The Role of Reasoning in Knowledge-Based Systems,
https://link.springer.com/chapter/10.1007/978-3-642-88719-2_14[visited at 13.04.2018]
- [3] So-Young Hwang, Dong-Hui Yu, Ki-Joune Li, Clock Synchronization in embedded systems,
https://web.stanford.edu/class/ee183/handouts_spr2003/synchronization_pres.pdf
<https://www.avrfreaks.net/forum/time-synchronization-two-avrs>[visited at 19.04.2018]

Tabellenverzeichnis

Abbildungsverzeichnis

1.1	Signature.	3
5.1	Intelligent agent's structure.	15
5.2	Intelligent Agents.	16
5.3	AI parts.	17
5.4	Cognitive system.	18
5.5	AI Quiz.	18
5.6	Topics KB.	18
5.7	Knowledge base system model/Architecture	19
5.8	Refinement for knowledge-based agent.	20
5.9	Axes of our representation system.	22
5.10	Nodes in field bus.	23
5.11	Clock synchronization phases.	25
5.12	Suitability of some design decision.	25
5.13	Table symetric vs asymeric.	26
5.14	General computer clock model.	28
5.15	Communicate using network.	29
5.16	Number of replicas.	30
5.17	Commands of paxos.	30
5.18	Clients and leader.	31
5.19	Executing a command.	32
5.20	Message exchange.	32
5.21	Synchronization order.	32
5.22	Faster reads.	33
5.23	Partition.	33
5.24	Lack of global time.	34
5.25	Interrupt handling time	35
5.26	Relation for clock adjusting	37
5.27	Calculus of optimum time	38
5.28	Difference between discrete clock and continuos clock	38
5.29	Components of synchronization algorithm	39
5.30	Algorithms classification	39
5.31	Advantages and disadvantages of Network switch.	41
6.1	Socket Ip address.	44
6.2	Libraries in Socket server.	46
6.3	Variable declaration to call socket functions.	47
6.4	Structure to get Ip address and accept incoming connections.	48
6.5	Notify server from incomming connections.	48
6.6	Infinite loop, synchronized timer, declared variables.	49
6.7	Calculus of the average.	50

6.8	Sending and receiving the average.	51
6.9	Libraries declaration.	52
6.10	Variable declaration to create socket.	53
6.11	Client's Structure to get Ip address and accept incoming connections.	54
6.12	Notify client a conection from server and accept incoming connections.	54
6.13	Receiving a real value and sending back to server.	55
6.14	Receiving and sending the average (client).	56
6.15	Ethernet shield with servomotor and led connected.	61
6.16	Servomotor slider webserver.	61
6.17	Libraries to implement servomotor and Ethernet connection.	62
6.18	Ip address and Subnet mask.	63
6.19	Ethernet port and variables declaration.	64
6.20	Webserver setup.	65
6.21	Connection with client and confirmation.	66
6.22	Create Webpage body.	67
6.23	End of Webpage body.	68
6.24	Slider body on webpage.	69
6.25	Call slider function and Buttons on/off condidionals	70
6.26	Buttons to rotate axis left/right.	71
6.27	Client stop and end of the loop.	72
6.28	Libraries declaration and port designation.	74
6.29	Setup and start listening for incomming connections.	75
6.30	Main loop structure and Client available.	76
6.31	Beginning of the webpage.	77
6.32	Printing the values of the temperature controller on webpage.	78
6.33	Printing the values of the temperature controller (Fahrenheit).	79
6.34	Printing Alert messages for temperature increasing/decreasing.	80
6.35	End of the webpage and state of the current space.	81
6.36	Client stop and end of the program.	81
6.37	Sending and receiving messages from webserver to client (serial monitor).	82
6.38	Calculus of the average in Arduino and Webserver.	83
6.39	Functioning diagram.	84
7.1	Socket chat server.	86
7.2	Sending counting values from server to client.	86
7.3	Sending date from server to client.	87
7.4	String with date and time.	88
7.5	Results of server's terminal with a clock sincronized response.	89
7.6	Results of server's terminal with a clock sincronized response.	90
7.7	Results of webserver with Servomotor controller.	91
7.8	Results of webserver with Temperature controller.	92
7.9	Results of Clock synchronization between Webserver with Servomotor controller and client in Serial monitor.	93
7.10	Results of Clock synchronization between Webserver with Temperature controller and client in Serial monitor.	93
.1	Formular: Wichtige Namen, Anschriften und Telefonnummern	104
.2	Working plan for activities.	105

.3 Master thesis topic 106

Stichwortverzeichnis

L^AT_EX, 1

LibreOffice, 1

MS-Office, 1

MS-Word, 1

Richtlinie, 1

Sinn, 1

SoftmakerOffice, 1

USB-Stick, 1

Word, 1

Writer, 1

.1 Bildanhang

Wichtige Kontaktdaten

Thema:	<input type="checkbox"/> Bachelorarbeit	<input type="checkbox"/> Masterarbeit
Vorname und NAME der Kandidatin bzw. des Kandidaten	Vornamen NAME	
Heimatanschrift:	Straße PLZ Ort	
Semester-/Wochentagsanschrift:	Straße PLZ Ort	
Telefon	+	
E-Mail (HsKA)	@hs-karlsruhe.de	
Name der Organisation		
Anschrift		
Telefon (Zentrale)	+	
Telefon -Druchwahl		
E-Mail (Organisation)		
Name der Betreuerin/des Betreuers		
Titel (z. B. Dipl.-Ing.)		
Position (z. B. Abteilungsleiter)		
Abteilung		
Telefon	+	
E-Mail (Organisation)	@	

File: NAME KontaktDaten yymmdd
Date: 2012-05-04

Bild .1. Wichtige Namen, Anschriften und Telefonnummern

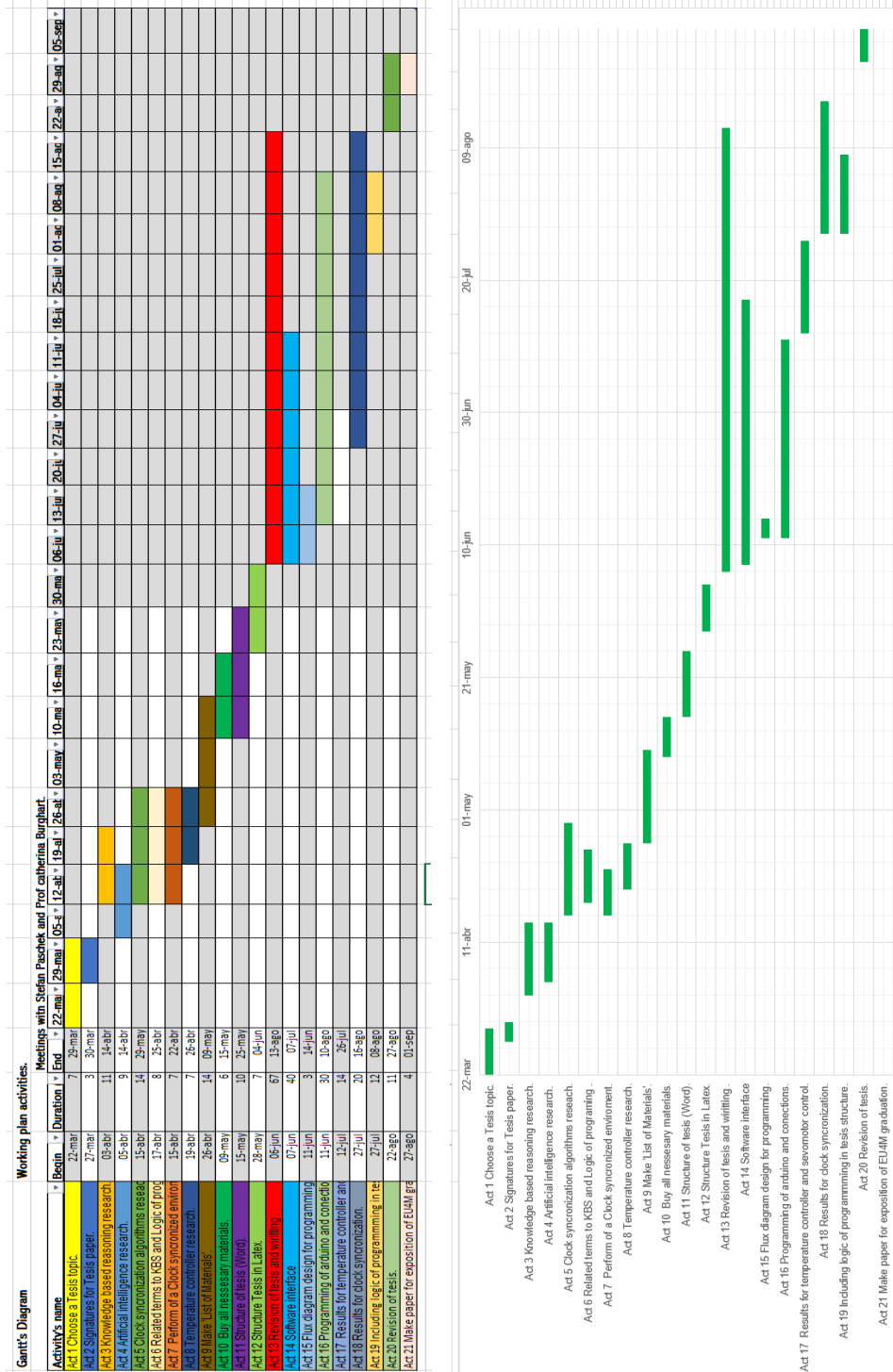


Bild .2. Working plan for activities (Gantt's diagram).