HAW HAMBURG

BACHELOR THESIS
Aynur Agnyanova Hasanova

Design and development of wireless communication systems for control and monitoring of an autonomous meteorological station

FACULTY OF ENGINEERING AND COMPUTER SCIENCE
Department of Information and Electrical Engineering

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Aynur Agnyanova Hasanova

Design and development of wireless communication systems for control and monitoring of an autonomous meteorological station

Bachelor Thesis based on the examination and study regulations for the
Bachelor of Engineering degree programme
Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner: Prof. Dr.-Ing. Lutz Leutelt
Second examiner: Prof. Dr. Marc Hensel

Day of delivery: 21 June 2021

Aynur Agnyanova Hasanova

Title of the paper

Design and development of wireless communication systems for control and monitoring of an autonomous meteorological station

Keywords

Weather Station, Microcontroller, Wi-Fi, Bluetooth, Wireless Communication, TM4C1294NCPDT, Wemos D1 mini, ESP8266, HC-05, UART, I^2C, AT commands

Abstract

The work detailed in the present document presents a part of an autonomous solar-powered weather station project of the Hamburg University of Applied Sciences. The project aims to demonstrate the skills acquired by students in the Faculty of Engineering and Computer Sciences at the end of their programs and to attract more students to the fields of Computer Sciences and Engineering.

This thesis project aims to design and implement Wi-Fi and Bluetooth communication channels for the autonomous weather station and the building of a framework for the development of the weather station. The Wi-Fi link is intended for weather data upload to a remote server. In contrast, the Bluetooth link is dedicated for demonstration of the weather station capabilities and maintenance using a Bluetooth connected handheld device. An investigation on weather stations previously developed by other university students and analysis on commercially available digital weather stations is done to determine the expected behaviour of the currently developed weather station and its subsystem specifics.

Aynur Agnyanova Hasanova

Thema der Bachelorthesis

Entwurf und Entwicklung von drahtlosen Kommunikationssystemen zur Steuerung und Überwachung einer autonomen Wetterstation

Stichworte

Wetterstation, Mikrocontroller, Wi-Fi, Bluetooth, drahtlose Kommunikation, TM4C1294NCPDT, Wemos D1 mini, ESP8266, HC-05, UART, I^2C, AT-Befehle

Kurzzusammenfassung

Bei der hier vorgestellten Arbeit handelt es sich um einen Teil eines autonomen, solarbetriebenen Wetterstationsprojekts der Hochschule für Angewandte Wissenschaften Hamburg. Das Projekt zielt darauf ab, die bis zum Studiumsende erworbenen Fähigkeiten der Studentinnen und Studenten der Fakultät für Ingenieurwissenschaften und Informatik zu demonstrieren und mehr Studierende für die Bereiche Informatik und Ingenieurwissenschaften zu begeistern.

Das Ziel dieser Arbeit ist es, Schnittstellen für Wi-Fi- und Bluetooth-Kommunikationskanäle für die autonome Wetterstation zu entwickeln und zu implementieren, sowie eine entsprechende Programmiergrundlage für die weitere Entwicklung der Wetterstation zu schaffen. Die Wi-Fi-Verbindung ist für das Hochladen von Wetterdaten auf einen ortsfernen Server vorgesehen, während die Bluetooth-Verbindung vor Ort für die Demonstration der Fähigkeiten und zur Wartung der Wetterstation genutzt wird. Dies geschieht mit Hilfe eines über Bluetooth verbundenen mobilen Handgeräts. Die vorliegende Arbeit analysiert darüber hinaus sowohl Wetterstationen, die von anderen Studierenden der Universität entwickelt wurden, als auch kommerziell erhältliche digitale meterologische Stationen in Bezug darauf welche Subsysteme integriert wurden. Anhand der gewonnenen Erkenntnisse konnte die Programmiergrundlage entsprechend derart entwickelt werden diese zu erwartenden Subsysteme ebenfalls zu berücksichtigen, also über die reine Unterstützung der Kommunikationsschnittstelle hinaus.

Contents

List of Tables

List of Figures

Abbreviations

AP	Access point
AT	Attention command
BC	Before Christ
CR	Carriage return
GNSS	Global Navigation Satellite System
GUI	Graphical User Interface
HAW Hamburg	Hamburg University of Applied Sciences
I²C	Inter-Integrated Circuit
IDE	Integrated development environment
ISR	Interrupt Service Routine
LF	Line feed, also known as
MCU	Microcontroller unit
OOP	Object-oriented programming
RTC	Real-time clock
SCL	Serial clock line (relates to I²C)
SDA	Serial data line (relates to I²C)
SoC	System-on-a-chip
SPI	Serial peripheral interface
TCP/IP	Transmission Control Protocol/ Internet protocol
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UTC	Coordinated Universal Time
Wi-Fi	Wireless Fidelity
WS	Weather Station
N/A	Not available

Terminology

Initiator/Leader	A device that can initiate a transmission and provides the clock signal in a serial communication system. These terms are used as an alternative to the “master” notation.
Responder/Follower	A device that waits for an instruction to participate in a serial communication system and uses a provided clock signal. These terms are used as an alternative to the “slave” notation.
Peripheral	A device directly connected to the CPU via the internal bus system with no easy option to separate it from the board.
Module	A device which is external to the board and needs to be connected to an Input/Output Peripheral of the board. An exception is the Hibernation module of EK-TM4C1294XL, which is part of the dedicated board.

1. Introduction

The history of weather sciences can be traced back to 3000 BC in India, where the first writings on weather, cloud formations and season cycles were done [1]. Later in 350 BC, Aristotle produced a written description on earth sciences, including weather and climate in “Meteorology”(written 350 BC). Even before those descriptions, sailors, farmers and shepherds, whose livelihood and safety relied on their understanding of weather, had developed rules for prediction and the concept of seasonal changes [2]. Around 200 BC, the first analogue hygrometer was invented in China (206 BC), followed by the invention of the compass (200 BC) and the windsocks (100 BC, presumably China or Japan as origin). Since then, multiple tools for quantifying weather phenomenons such as temperature, humidity, wind speed and direction, precipitation and others have been developed. In 1847 the first independent weather station using self-recording instruments designed on a kite was sent up in England, and in 1960 the first weather dedicated satellite was dispatched, which allowed weather monitoring on a global level. At the end of the 20th century, the first public weather broadcasts were made, making meteorology a concern not only for the domain of experts. Miniaturisation and advancement of electronic components allowed for the development of weather stations with more quantifying elements, compacter sizes and better precision and accuracy of the readings.

A modern meteorological station consists of several systems revolving around gathering and administering weather data (see *Figure 1*). Those systems are based on electrical components, such as sensors and power providing units, and software controlling these components at the application level. Making a meteorological station combines theoretical and practical knowledge of climatology, electrical and computer sciences. This property makes it an exemplary demonstration of skills acquired in universities of applied sciences, where students gain a multitude of experiences and knowledge in one or more of these disciplines.

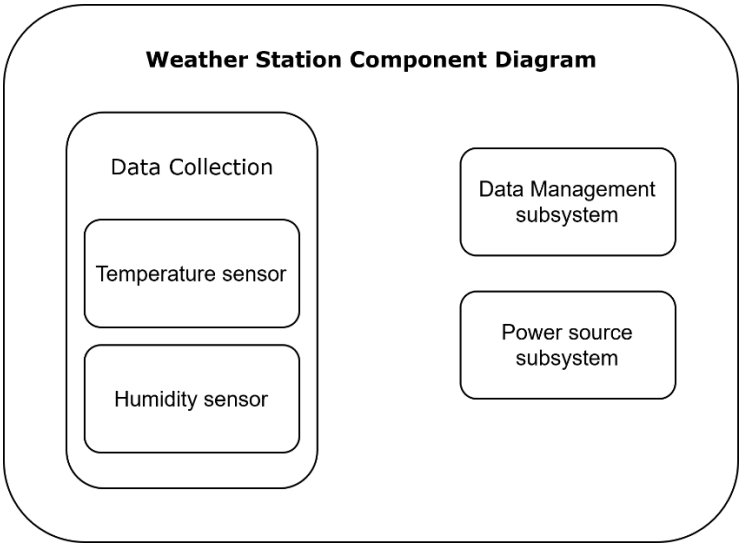


FIGURE 1: EXAMPLE OF COMPONENT DIAGRAM FOR A WEATHER STATION

In 2019, Prof. Dr.-Ing. Lutz Leutelt, a lecturer in the Hamburg University of Applied Sciences, together with several other involved parties, concluded that a meteorological station would be an exemplary display for the complexity of skills learned in Electrical Engineering and

Informatics programs. He concluded that a weather station project should be one of the showcases advertising the faculty to young students and other university visitors.

The final presentable solution for the showcase shall be an autonomous solar-powered weather station that provides several channels for accessing it wirelessly. The system is therefore separated into three major subsystems - data gathering, power management and communication management.

The work described in the present document concentrates on the design and implementation of wireless communication channels for the meteorological station. A Bluetooth channel for maintenance and control by a handheld device and a Wi-Fi channel for uploading weather data to a remote web server are to be developed (see *Figure 2*). As a starting point for the showcase product, an initial set of specifications for the weather station is derived based on industry standards and analysis of weather stations previously developed by students of the Hamburg University of Applied Sciences. This thesis project is the first subproject for the implementation of a fully functioning weather station. Consequently, a framework for the development of the whole project is built. It includes decisions about the microcontroller core of the weather station and the development tools to be used throughout the continuity of the weather station project.

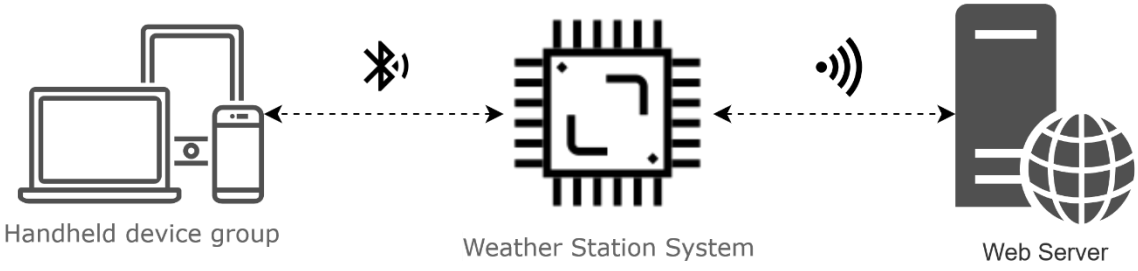


FIGURE 2: EXPECTED SYSTEM STRUCTURE

2. Fundamentals

2.1. Microcontroller

A microcontroller unit, also abbreviated as MCU or µC, is a complex computational unit that includes a Central Processing Unit (CPU), memory (RAM, ROM), input/output ports, peripherals, and interrupt processing all in one chip [3]. The block diagram in *Figure 3* displays the standard components in a microcontroller.

A specification that differentiates microcontrollers from microprocessors, even though both can be considered microcomputers, is that an MCU uses an internal bus system to interface its components, which means that the components included in the microcontroller cannot be separately used.

Another difference can be drawn between a microcontroller and a system on a chip (SoC). An SoC includes a microcontroller and integrates it on the same device or chip with advanced peripherals such as converters, oscillator, graphical processing unit (GPU), wireless modules (WLAN, Bluetooth, infrared) and more chipped solutions.

The main properties on which MCUs are assessed are:

- maximum system clock,
- processing data width,
- size of program memory (flash),
- operating voltage,
- power consumption,
- advanced architectural support for peripherals,
- development support (compiler, IDE, debugging)
- and most of all, cost of unit - both as a market price, and as an effort for integration [4].

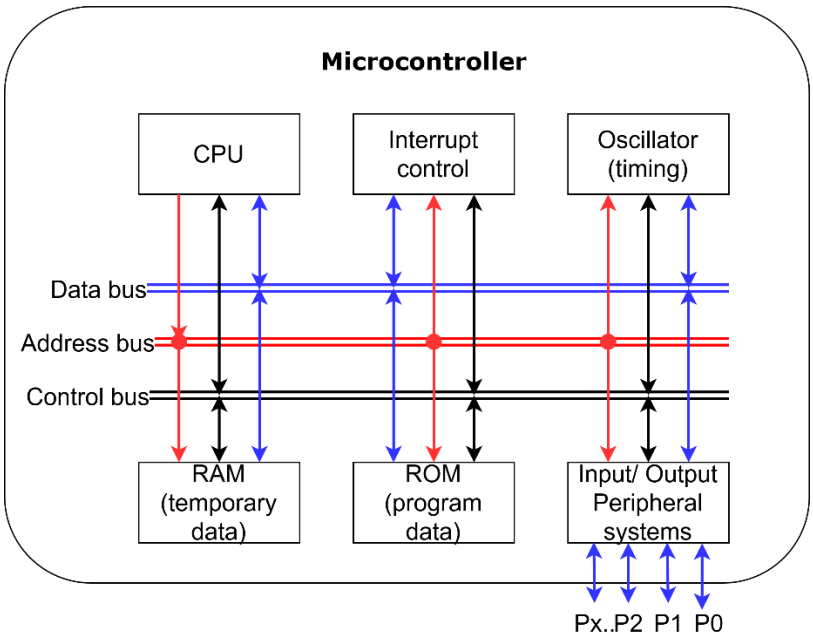


FIGURE 3: TYPICAL MICROCONTROLLER BLOCK DIAGRAM

2.2. Serial Communication Interfaces

One of the primary means of wired communication with a highly integrated device such as a microcontroller is via serial interfaces. Such an interface is represented by the sequential transmission of binary (bit) pulses on a wire [5, p. 39]. In a typical setting, a logical “1” is represented by sending of high voltage (usually 3.3V or 5V) and logical “0” is represented by low voltage (usually 0V). Because data is transmitted as a “binary string”, locating the start of the data is crucial. Different serial protocols provide different solutions for this issue.

2.2.1. Universal Asynchronous Receiver Transmitter

Universal Asynchronous Receiver Transmitter (UART) is a communication module that realises an **asynchronous serial protocol** that allows communication **between two devices** - typically a microcontroller with a peripheral device or another microcontroller [6]. It consists of two data wires – one for receiving (Rx) and one for transmitting (Tx) and their respective registers to hold the data (see Figure 4*).* As the communication is asynchronous, the transmitting and receiving devices should be preconfigured with the same clock. Otherwise, the transmission would be successful, but the receiver would not be able to correctly sample the data at the middle of each bit, which would lead to corrupted data.

Other characteristics of the protocol are the bitrate, which depends on the length of the cable connectors used and the packet format (portrayed in Figure 5) [6]:

There also exists USART - Universal Synchronous/Asynchronous Receiver Transmitter module definition. It allows for synchronous serial communication using the existing Tx and Rx lines and utilises a third line carrying a clock signal. Typically, USART is faster by a factor *s* than UART [7, p. 81]. A USART module combines logic for both synchronous and asynchronous communication.

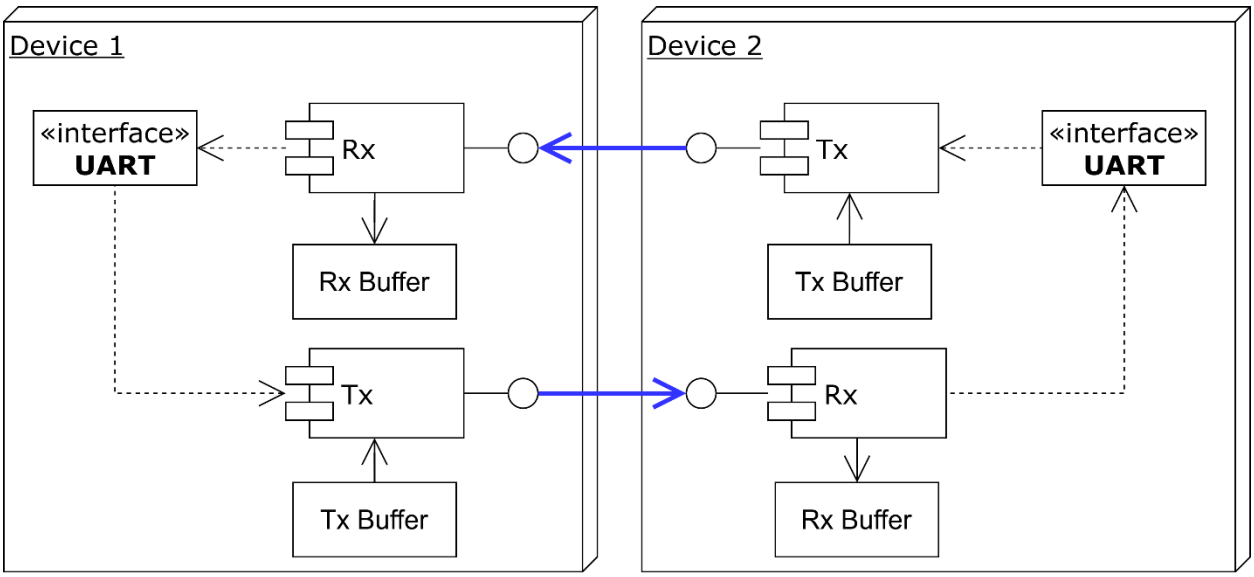


FIGURE 4: COMPONENT DIAGRAM OF COMMUNICATION BETWEEN TWO UART MODULES

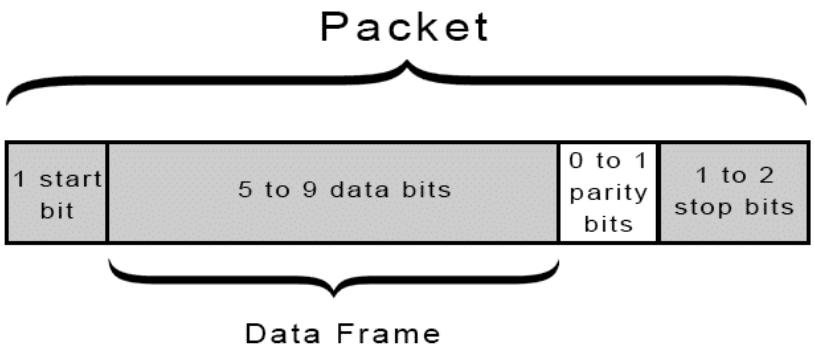


FIGURE 5: PACKET FORMAT OF UART COMMUNICATION PROTOCOL [8]

2.2.2. Inter-Integrated Circuit

The Inter-Integrated Circuit (I²C) bus is a **synchronous two-wire serial interface** developed by Philips Semiconductors in 1982 [7, p. 84] and currently owned by NXP. It is a **multi-initiator, multi-responder** protocol with a typical device address length of 7 bits. It **allows for up to 128 devices connected on the bus,** provided the maximum bus capacitance of 400 pF is not exceeded. It is widely used for short-distance interfacing between microcontrollers and peripheral devices and is the preferred module/protocol for multi-leader applications. The protocol allows several initiators, with the definition of a leader being a device that can initiate data transfers on the communication bus. A responder is defined by being addressed by an initiator [9, p. 45]. When an initiator sends data, it is visible to all connected responders, but only the responder whose address is part of the data frame can answer the query.

The I²C uses two bi-directional wires for communication: Serial Clock (SCL) and Serial Data (SDA), to which all participants, initiators and responders, are connected (see *Figure 6*) [7]. The interface has three speed modes: standard (100 kbit/s), fast (400 kbit/s) and high-speed (3.4 Mbit/s). The packet frame format for I²C is shown in *Figure 7*.

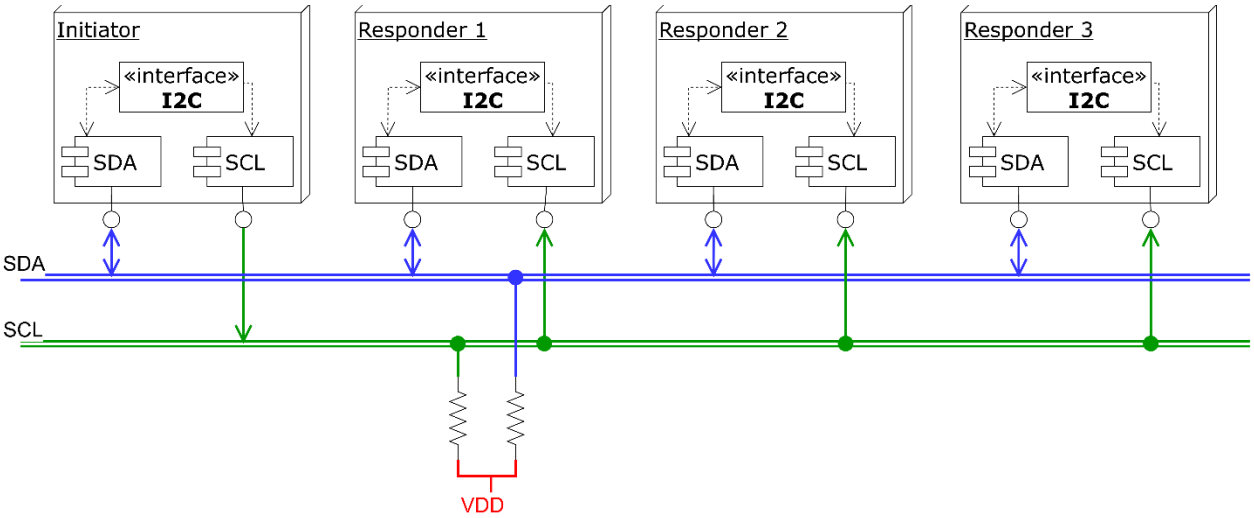


FIGURE 6: COMPONENT DIAGRAM OF A TYPICAL I²C NETWORK WITH THREE FOLLOWERS

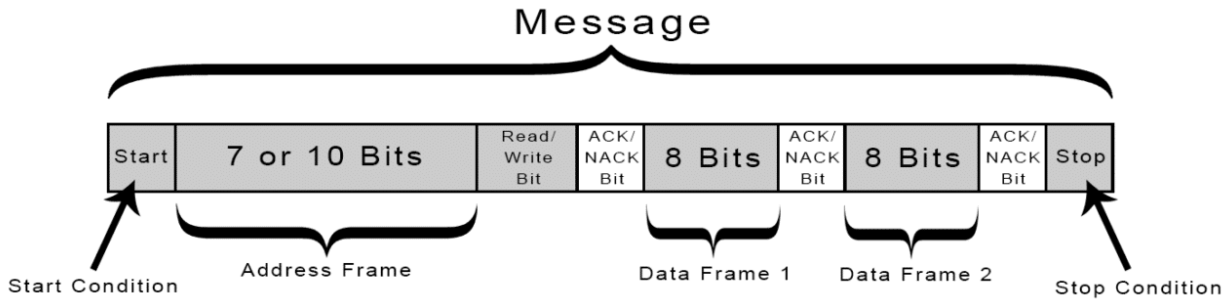


FIGURE 7: PACKET FORMAT OF I²C COMMUNICATION PROTOCOL [10]

2.3. Event Detection via Polling or Interrupt

Software and hardware systems alike are often designed to change their behaviour based on the change of status of an internal function or based on an external factor [7, p. 52]. This is typically done by the detection of these changes, called events, and the timely reaction to them via predefined routines. There exist two types of event discovery - polling and direct interrupt.

Polling is the repeated or perpetual checking for a flag of a resource (peripheral) if an event has occurred and the associated module needs servicing from the CPU ("software asks") [11]. Polling keeps the CPU continuously busy and does not allow it to go to power save mode. There are two types of polling - blocking ("busy waiting"), where the CPU continuously checks the status until the status changes, and non-blocking, where the CPU checks the status and continues with other tasks until the time has come to recheck the resource. This concept is easy to implement and debug as the program flow depends only on the main function and its sub-functions. However, some events can be missed if they occur when their flag is not checked or in the case of the busy waiting, the CPU cannot continue with other work until this flag has been set.

The other concept is via "**hardware tells**" by the associated peripherals, which send a signal to the CPU or a module dedicated to processing the interrupt before announcing it to the CPU [12]. The first requires the signal to be a vectored interrupt containing information about the priority of the interrupt and the expected interrupt service routine (interrupt handler) [9, p. 22]. The second implementation requires an additional module called Vectored Interrupt Controller, which receives the "hardware tells" when an event occurs and keeps track of the occurrence and priority of the events as well as the routine (interrupt handler or Interrupt Service Routine - ISR) to execute when a given interrupt occurs. This allows the CPU to possibly enter power save (sleep) mode when no interrupts are present. Additionally, it lowers the possibility of events being missed but requires more secure data integrity and is more challenging to debug as the main function can be interrupted at any time.

Figure 8 shows an approximation for the activity diagrams of a program for reading data using either the two polling concepts or the interrupt-driven concept.

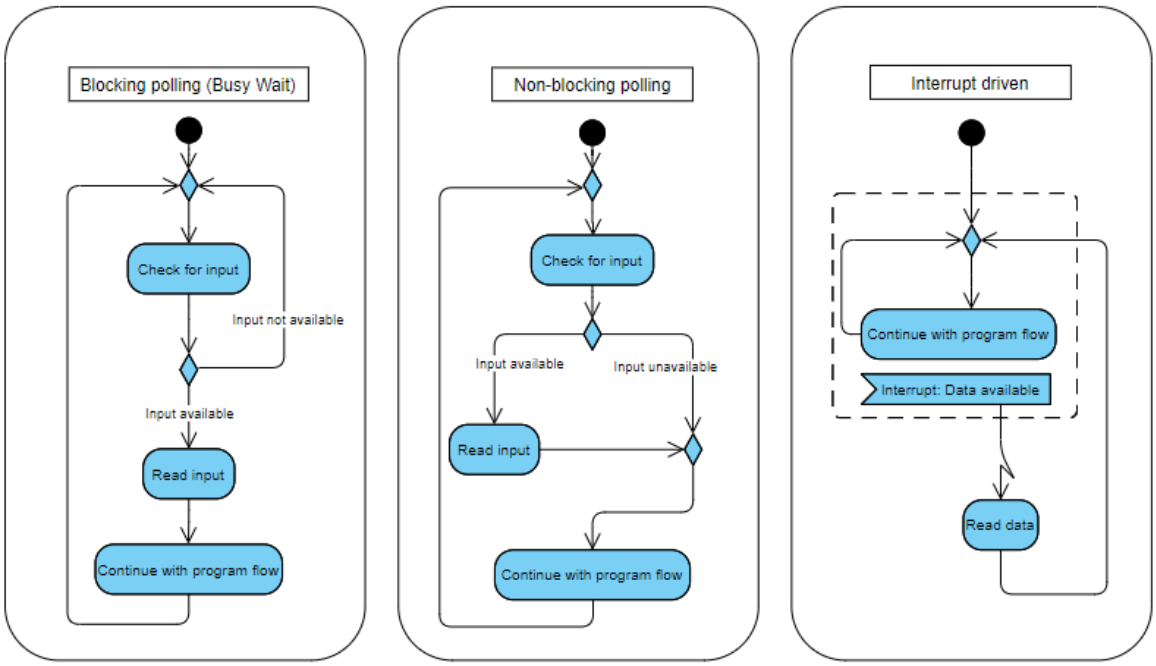


FIGURE 8: ACTIVITY DIAGRAMS OF BLOCKING POLLING, NON-BLOCKING POLLING AND INTERRUPT-DRIVEN ROUTINES

2.4. Attention Commands

The Hayes command set, also known as the “AT” command set, is a set of commands developed by Dennis Hayes, the founder of Hayes Microcomputer Products [13, p. 53]. It was designed to configure the settings of the “smart modem”, first introduced in 1977, without physically changing the hardware. Since then, the AT commands technology has evolved from purely modem control to a comprehensive middleware for mobile devices and networks.

An “attention” - “AT” command consists of three elements: the prefix -“AT”, the body of the command and the termination character <CR> [14, p. 6].

The standard defines two main types of commands - action and parameter commands. The action commands are either “execute” or “test” sub-type, where the first executes a particular function of the device. The second checks if the device supports specific functionality. The second primary type is separated into “set”, “read”, or “tested” command types. A “set” command stores values on the device, a “read” command reports the current value or values stored, and a “tested” command determines if the device supports a parameter and whether the value range matches.

There exist two syntax definitions for the commands - basic and extended, both of which follow the main attention command format **“AT<command><CR>”**.

The basic command syntax is the prefix followed by a body that combines a letter (A to Z) or “&” character and an optional numerical string, representing a decimal integer value. The letter corresponds to a predefined by the original standard application [15].

The extended command syntax is defined by names for the actions and parameters always precedented by a “+” character. A name (<name>) consists of one to sixteen characters, where the characters can be alphabetical (A-Z), numerical (0-9) or special (!,%,-,.,/,:,_). The first character after the “+” is a letter and implies the application of the command [14, p. 8]. If there are any parameters involved, they can be added after the name of the command by the addition of a “=” or other special character followed by the parameter value as a numerical or string constant (<value>). In the case that more than one parameter is involved, the values are placed in a “compound value” (<compound_value>) and are separated by a comma (“,”) character (see (1)). Optional parameters are placed in square brackets “[]”.

$$< compount\_value > = < value1 >, < value2 >, < value3 >, \ldots. \quad (1)$$

The command name should be defined using only upper-case characters.

Furthermore, the syntax for the extended command standard is as follows:

- Action execution command: *+<name>* or *+<name>[=<value>]* or *+<name>[=<compound_name>]*
- Action/Parameter test command: *+<name>=?*
- Parameter set command: *+<name>=[<value>]* or *+<name>=[<compound_value>]*
- Parameter read command: *+<name>?*

For return statements, a successful command returns an “OK” followed by a set of a <CR> and <LF> character.

2.5. Wireless Network Technologies and Standards

A wireless network is a wireless substitution of wired networks and is used for areas where the connected devices are moved often [16]. The technology uses antennas to transmit and receive data on a given radio frequency without a physical conductor. Within a LAN network, an address of a device is associated with its physical location. With WLAN, this rule does not always hold. An addressable unit (destination) for WLAN is a station (STA). Wireless standards work on the Physical and Data Link layers of the standard Open Systems Interconnect (OSI) model [17, p. 10]. There exist four types of wireless networks - Wireless Personal Area Network (WPAN), Wireless Local Area Network (WLAN), Wireless Metropolitan Area Network (WMAN) and Wireless Wide Area Network (WWAN), with the coverage varying from within a couple of meters to worldwide available [18].

2.5.1. Wi-Fi

Wi-Fi is a generalisation for the wireless network protocols under the IEEE 802.11 (WLAN) communication standards used for local area networking [17, p. 40]. A “Wi-Fi CERTIFIED” seal ensures that a product meets a set of criteria and can interoperate with other Wi-Fi certified devices on the same frequency band. It was developed to be a wireless substitution of the high-speed Ethernet internet protocol. It typically uses Star network topology with a router (playing the role similar to the hub in wired networks), providing an access point to which multiple devices can connect simultaneously (see *Figure 9*). Wi-Fi is commonly used for static

environments such as work or home, where a router can be placed, and devices within the building or campus can connect to. Typical applications for Wi-Fi include client-server based solutions where a certain degree of configuration for both sides is possible and high speed is required. A Wi-Fi network always has an access point and is usually password protected.

The most widely used Wi-Fi standards are “a”, “b”, “g”, “n”. The “a” standard stands for 54 Mbps data rate using a 5 GHz band. “B” stands for 11 Mbps on a 2.4 GHz band. “G” extends the data rate on a 2.4 GHz band to 54 Mbps and is compatible with “b” standard. “N” provides even higher data rates of 150,350 and up to 600 Mbps and maintains backward compatibility with the “a”, “b” and “g” standards [17, p. 140].

There also exists an “802.11u” standard which provides a generic, standardised approach for WLAN to work with non 802.11 networks such as Bluetooth and Zigbee (both WPAN networks) and WiMax (WMAN). However, this standard is rarely used in ready solutions. The most recent standards, “802.11ac” from 2013 and “802.11ax” from 2019, are both backwards compatible with “b”, “g” and “n” and provide higher security and performance capabilities [19]. A router with “802.11ac” would work with a device with “b”, “g”, or “n” standards. For using the benefits of the newest standards, both the router and the connected devices should be implementing the newest standard.

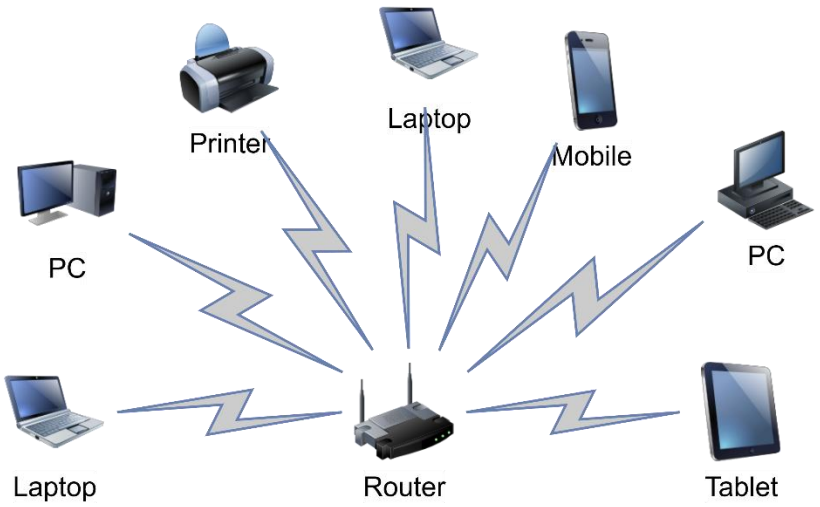


FIGURE 9: STAR TOPOLOGY IN WIRELESS NETWORKS

2.5.2. Bluetooth

Bluetooth is an IEEE 802.15.1 (WPAN) wireless standard with short-range capabilities which involves little to no infrastructure or direct (point-to-point) connectivity [20]. The communication is based on a leader-follower principle where a leader can connect to up to seven devices, but a follower module can typically only have one leader. A device can switch roles if the hardware allows it. At any time, data can be transmitted from the leader to one other device, where the leader chooses which device to address. If more than one follower is connected, the leader uses round-robin scheduling to address and transmit the needed data.

Bluetooth was designed to be a low-power alternative to other short-distance wireless protocols and requires a semi-visible optical path between the participants. The protocol is intended for portable equipment in any setting, both for personal and industrial use. Typical applications

include headsets and remote controls, where ease of connection with a minimal setup is required. All Bluetooth standards are backwards compatible.

2.5.3. TCP/IP Protocol Suite

The Transmission Control Protocol (TCP) and the Internet Protocol (IP) together form a protocol suite that became “the standard for interconnecting networks, devices and the Internet” [21, p. 2]. The suite is responsible for routing data packets within a TCP session using the IP address of the recipient attached to each data packet. TCP/IP provides an abstraction on top of physical networks and operating systems so that user applications do not need to know the physical architecture.

As with most networking software, TCP/IP is modelled in layers, otherwise referred to as the TCP/IP stack. The layers are defined as Application, Transport, Network, Data Link and Physical. Compared with the standard OSI model, the Application layer of the TCP/IP includes the Application, Presentation and Session layers of the OSI model (see *Figure 10*).

“It is important to note that when the layers of TCP/IP are on different systems, they are only connected at the physical layer. Direct peer-to-peer communication between all other layers is impossible. Therefore, all data from an application has to flow “down” through all five layers at the sender, and “up” all five layers at the receiver to reach the correct process on the other system.” [22].

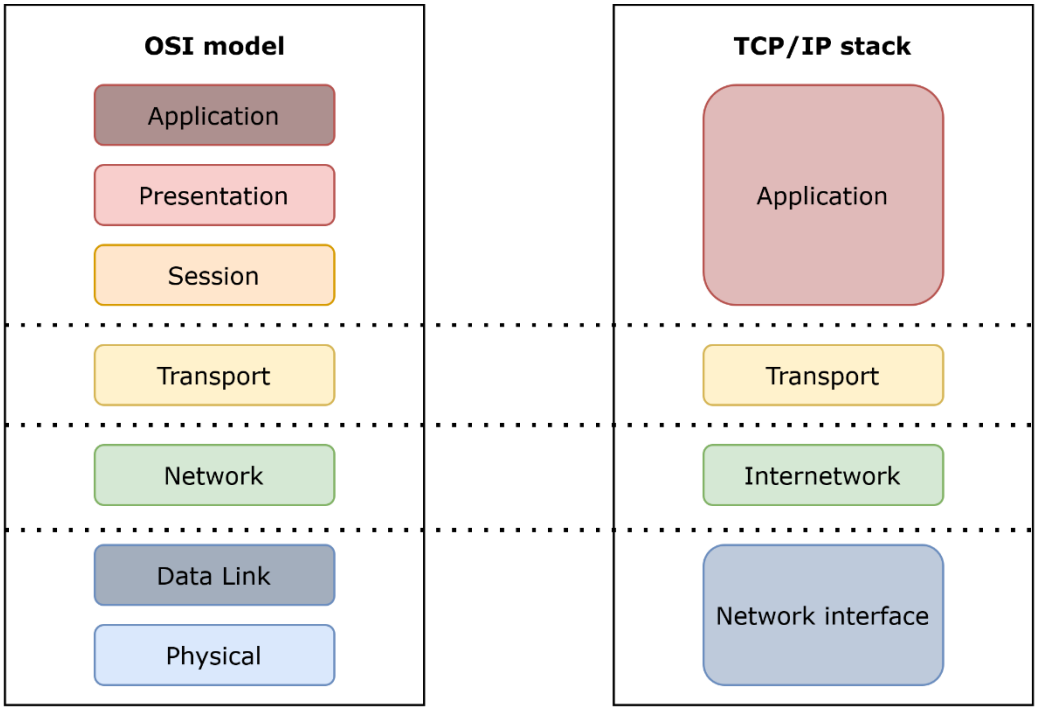


FIGURE 10: COMPARISON OF THE OSI MODEL AND TCP/IP STACK LAYERS

2.6. EK-TM4C1294XL LaunchPad

The EK-TM4C1294XL LaunchPad is an evaluation board based on the tm4c1294ncpdt microcontroller produced by Texas Instruments.

2.6.1. Microcontroller

The tm4c1294ncpdt is an ARM Cortex-M4F CPU microcontroller with a Nested vectored interrupt controller (NVIC), DSP and floating-point units, three sleep modes for optimised power consumption, 1024 kB of flash memory and operating at 120 MHz. It includes an internal ROM loaded with some of the functions available from TivaWare - an SDK provided from Texas Instruments for microcontroller software development - and a multitude of communication interfaces (UART, SPI, I^2C, USB, CAN). The microcontroller board pins operate at 3.3V. However, 3.3V and 5V output rails are available.

An additional Hibernation module peripheral allows the microcontroller to be powered off when put in hibernation mode, as long as the Hibernation peripheral has a separately provided battery attached to it.

2.6.2. Hardware

The EK-TM4C1294XL LaunchPad board provides access to 80 of the TM4C1294NCPDT microcontroller GPIOs via 4 sets of 20 pins (BoosterPacks) [23].

Figure 11 below shows the frontal pinout of the board with the possible pin configurations written in the tables on the side of the LaunchPad. The pins are grouped in 15 GPIO blocks (ports) - A, B, C, D, E, F, G, H, J, K, L, M, N, P, Q [24, p. 742]. Depending on a functionality set in the GPIO registers, most pins have alternate functions such as UART, I^2C, ADC, SPI. The BoosterPack pins provide access to six UART, four SPI, and three I^2C serial communication interfaces and 19 ADC pins where some of the pin functionalities overlap for several pins.

A side header for 98 connector pins is also available on the LaunchPad board, allowing for access also to UART0 and UART1 and other configurable pins (see *Figure 12*).

The LaunchPad board also provides:

- four user-operated LEDs (connected to PN1, PN0, PF4, PF0),
- two user-operated switches (connected to PJ0 and PJ1),
- one independent hibernate wake switch,
- one independent microcontroller reset switch,
- Ethernet connector,
- USB 2.0 micro A/B connector,
- integrated debug interface (ICDI) with available USB connector,
- jumper for selecting power source: ICDI USB, USB Device, BoosterPack.

If the LaunchPad is powered by USB, the expected power input is 5V. However, if the board is powered via a BoosterPack pin, the power supply needs to be 3.3V.

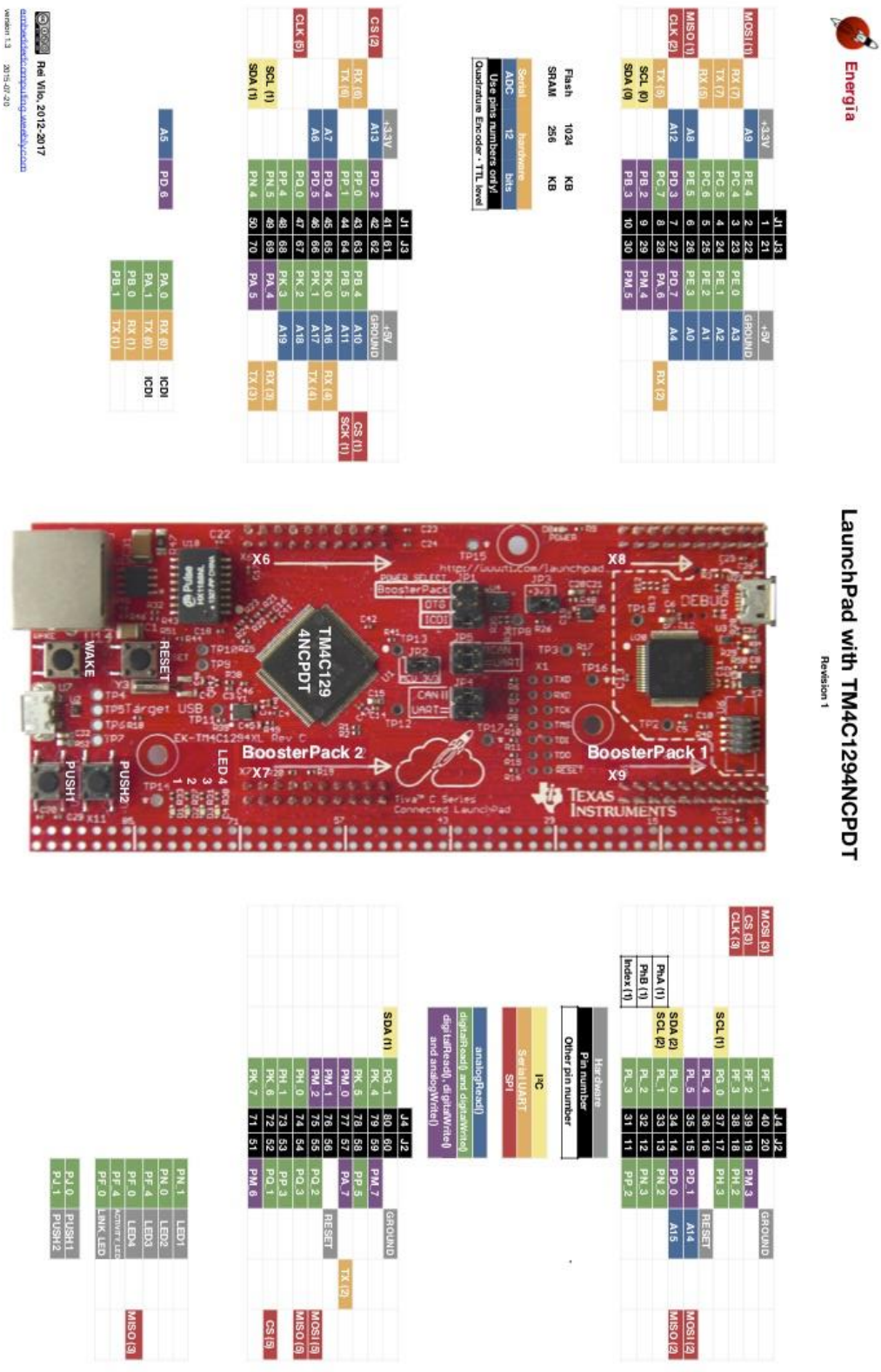


FIGURE 11: FRONT PINOUT OF THE EK-TM4C1294XL LAUNCHPAD BOARD WITH AVAILBLE BOOSTERPACKS BASED ON THE TM4C1294NCPDT MICROCONTROLLER [25]

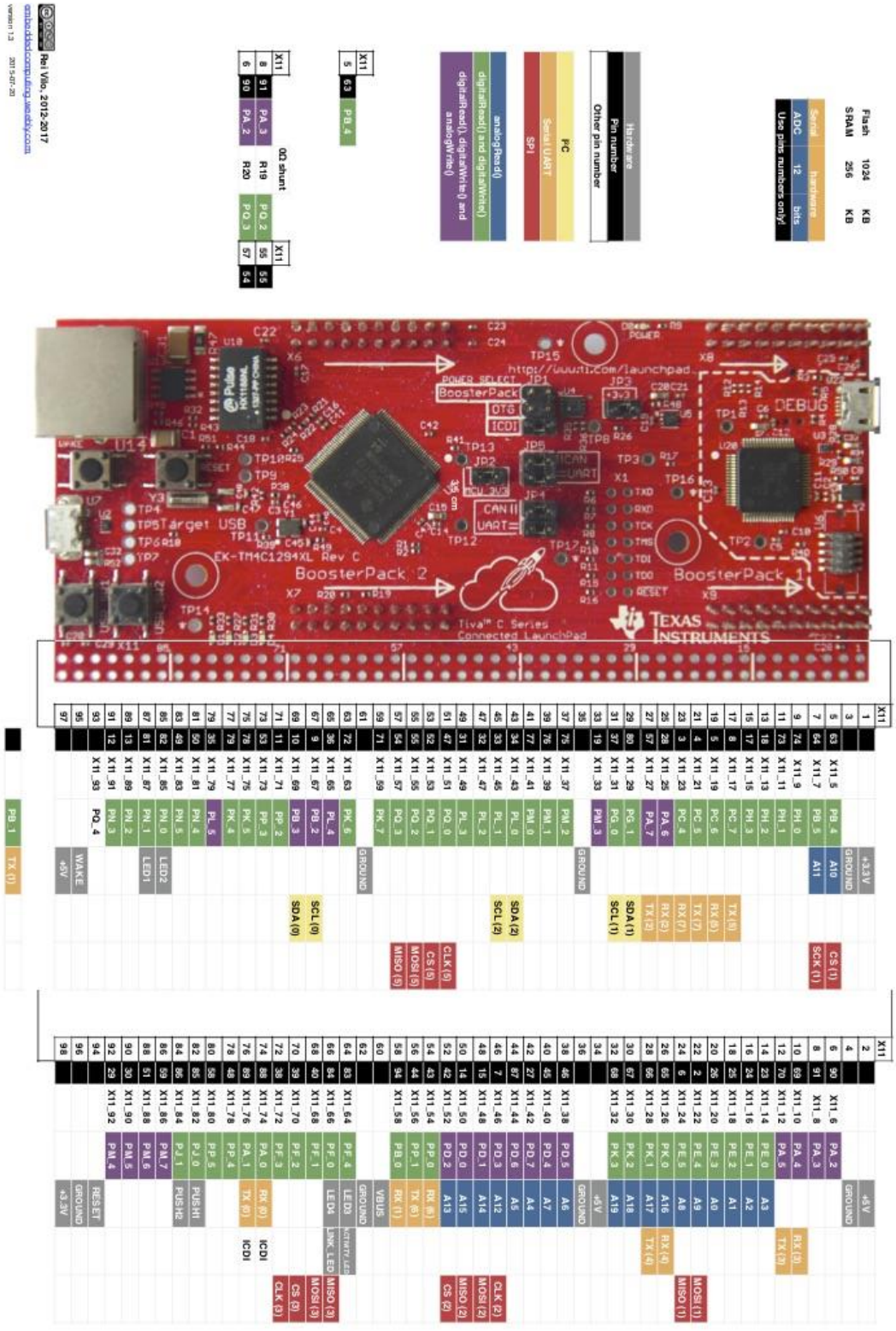


FIGURE 12: SIDE HEADER PINOUT OF THE EK-TM4C1294XL LAUNCHPAD BOARD BASED ON THE TM4C1294NCPDT MICROCONTROLLER [25]

2.6.3. TivaWare Software Development Kit

TivaWare is a software development kit written in C for the TM4C Cortex-M4F solutions family of Texas Instruments. It provides APIs for the initialisation and configuration of a microcontroller without direct access programming, thus lowering the overall time and production costs. The SDK is supported by Code Composer Studio (CCS), IAR Embedded Workbench and Keil Real View Microcontroller Development Kit, and several other development tool chains.

The SDK provides several libraries for peripheral access (driverlib), USB configuration (usblib), graphics (grlib), NFC (nfclib) or sensor (sensorlib) functionality, together with examples on how they could be used based on a board or peripheral.

Using these drivers could expedite a development process drastically. Consequently, to the design of the libraries, their implementation has a disclaimer that it might not be as efficient in terms of speed and code size as using direct access for programming the registers. The SDK also provides part-specific header files (tm4c1924ncpdt.h for the microcontroller base of the TM4C1294 LaunchPad, which accommodate register macros for uncomplicated direct access programming.

3. Requirements

This chapter gives insight into the requirements derivation process. It can be grouped into two main steps - elicitation and specification. The first one concentrates on defining the groups involved and researching the needs of the stakeholders and modality of weather stations solutions. In the specification phase, the gathered data is first filtered out for the requirements of the autonomous weather station and then narrowed down to provide the specification for the implementation of the wireless communication channels.

3.1. Elicitation

The requirements gathering process included identifying the involved parties (stakeholders) and their needs for the system, analysing those needs and narrowing them down to a set of requirements for the communication subsystem of the weather station. Only this subset of the requirements shall be implemented in this project. In the process, weather station solutions created by other university students and commercially available digital weather stations are evaluated.

3.1.1. Stakeholders

The stakeholders involved in the specification derivative were:

1. End-users:
    a. Professors, technical assistants,
    b. Student guests to the university,
2. Technical staff:
    a. University students involved in similar projects from the university in the past,
    b. University students, assistants and other workers involved in the developing, upgrading and maintaining of the meteorological station.

The first group, defined as “End-users”, would operate the final product to:

- see weather data provided by the station (a. and b.),
- demonstrate the functionality of the weather station (a.),
- illustrate various aspects and opportunities of electrical engineering and informatics (a.),
- arise interest in the study of information and electrical engineering (a. and b.).

The second group, defined as “Technical staff”:

- works on improvements of the current software and hardware of the weather station,
- develop further features for the “End-user” group,
- maintains the associated software and hardware,
- performs tests on separate features, and the whole system.

3.1.2. Requirement Derivation

The requirements derivation includes the breakdown of the specifications for a “fully-equipped weather station” and the communication links associated with it provided by the end-user stakeholder group. Secondly, an evaluation of existing solutions was done to derive further specifications about the expectations from the system. The analysis was performed on weather

stations created by other students in the Hamburg University of Applied Sciences (HAW Hamburg) and commercially available meteorological stations.

3.1.2.1. *Requirements for a Fully Equipped Weather Station*

A fully equipped weather station shall have dual use:

1. as a tool for demonstrating the possibilities and educational appeal of digital electronics and embedded systems,
2. as a tool for collecting weather data.

The station then shall support three modes:

- Demonstration mode, which showcases the weather station's functionality, such as manual adjustment of the solar panel towards the sun.
- Data collection mode, which focuses on the timely collection of data and its storage, and low power-consumption.
- Test/maintenance mode, in which a technician can perform maintenance tests to evaluate if any sensors or other hardware need adjustment or replacement. This mode and the demonstration mode would need a display.

Additionally, a concept for collecting, distributing, and visualising the data shall be designed. The solution shall:

1. collect data locally on an SD card.
2. transmit data wirelessly (e.g. GSM) to a central storage (e.g. cloud).
3. visualise data on an interactive local display or laptops/mobile devices (e.g. using a web server).
4. Enable setup and control of weather station features by local buttons, web interface or module app.

Therefore, a fully-equipped station shall include several subsystems – a data system, a power system, and a communication system.

The data system would include data gathering on several points: temperature, humidity, atmospheric pressure, precipitation, altitude, wind speed, wind direction and brightness.

The power system would include a solar panel whose positioning can be continually adjusted to that of the sun and a GPS and 3-axis magnetometer to measure the solar panel's direction and the location of the station.

The communication system shall include two types of wireless channels – a "Wireless Maintenance Link" and a "Data Upload Link".

The first link is intended for use by a service technician with a handheld device at the weather station site. This handheld device must:

- support Bluetooth communication
- provide touch-sensitive display
- allow implementation of control software with a graphical user interface (GUI)
- have an industrial appearance and feel to emphasise the embedded nature of electronics

The software on this handheld device shall:

- send commands to the weather station
- enable the display of sensor values as text in the GUI
- enable selection of the operating mode of the fully equipped weather station (demonstration, data collection or test mode)

- enable for selection of sensors, whose functionality can be showcased by lighting on/off an LED associated with a particular sensor
- enable forced solar panel adjustment towards the current sun position

The “Data Upload Link” is intended only for uploading sensor data via WLAN from the weather station to a remote web server. The web server has the task to store and display the data in an appropriate GUI format.

The entire setup shall have demonstration as its primary purpose:

- all electronics (sensors, microcontroller, solar panel, battery) shall be accessible and visible for inspection during a demonstration
- the sensor values shall be visualised in a manner that non-experts can understand
- the solution shall initially be developed for indoor use only.

The utilisation of the prototype can be based on existing, reused or refractured modules.

3.1.2.2. *Previously Developed Weather Stations in the HAW Hamburg*

3.1.2.2.1. Overview

Weather stations previously developed in HAW Hamburg play an essential part in identifying additional features about the system’s behaviour expected from the user group. Furthermore, they provide information about the implementation process and the possible concept development.

Each of the existing weather stations was developed by groups (teams) of students as part of a Sensor technology (Sensortechnik) project in the master programs in the Department of Information and Electrical Engineering in the HAW Hamburg. All teams were required to present a report summarising their approaches and chosen hardware and software implementations at the end of the semester. The teams varied in the number of people involved, and each team was required to create their definition of a “solar-powered weather station”.

As seen from *Table 1*, a total of 8 reports were available for analysis. The teams consisted of two to five people. A unique station ID was assigned to each report and then used instead of the reports full name.

Appendix A - Table with Detailed Overview of Features of Previously Developed Weather Stations presents an overview of the weather stations’ features developed and described in their respective reports by the HAW Hamburg master students.

The data from the table suggests that each of the teams came to the same base requirement set:

- the station should be autonomous, i.e. using a solar panel for power source
- should gather weather data (temperature, humidity, compass/direction, wind speed and direction)
- should provide at least one channel for data acquisition
- should provide time and date
- should save data on a local SD card.

Two of the weather stations, whose reports were provided, were still available physically as solutions in the university at the start of this project. As indicated in the end-user requirements, any parts from other stations fitting with the requirements can be reused in the current project.

The “Pro-Jan-17” solution worked until March 2020, when it encountered an issue with the solar panel movement and stopped sending data to its web server. The web server, consisting of a

router and a Raspberry Pi 3 server, database and web page displaying only the latest upload, was still available and working. It was considered for testing the WLAN communication channel implemented for the newest weather station. The technical staff on-site has indicated they would like to keep the rest of the weather station as is. Therefore, no other hardware could be acquired from it.

Name of report	Date	Number of people	Unique station ID (arbitrary)
Entwicklung und Aufbau eines Systems zur Erfassung von Messdaten und zur Steuerung eines Ausrichtungssystems für eine Sollarzelle	Jan-13	2	Entw-Jan-13
Entwicklung einer solarbetriebenen Wetterstation unter Verwendung eines Arduino Unos	Jan-14	2	Entw-Jan-14
Realisierung einer Wetterstation	Jan-14	2	Real-Jan-14
AWDAD – Autonomous Weather Data Acquiring Device	Jan-16	5	AWDA-Jan-16
Realisierung und technische Beschreibung einer Wetterstation basiert auf einem Arduino DUE Signalprozessor	Jan-16	5	Real-Jan-16
Wetterstation	Jan-18	5	Wett-Jan-18
Solarbetriebene, mobile Wetterstation	Jan-20	4	Solar-Jan-20
Projekt: Wetterstation	Jan-17	5	Pro-Jan-17

TABLE 1: GENERAL NAMING AND PROJECT DETAILS OF REPORTS ON PREVIOUS STATIONS

The other still physically available weather station in HAW Hamburg was “Solar-Jan-20”. It used an 8-bit STM microcontroller for the weather station base and included a Windows application for the data output of its Bluetooth communication channel. The solution provided weather readings for all the data points expected for a fully equipped weather station except for brightness. It was also autonomous in its data collection with the provided solar panel and battery. It would have been possible to use it as the weather station and expand it with the needed Wi-Fi module and brightness sensor. However, the microcontroller used in this solution was chosen to optimally provide the needed peripherals. It left only one SPI channel unused, which would not allow further development with the needed modules.

Consequently, this STM microcontroller could not provide the needed peripherals, and the “Solar-Jan-20” system could not be used as a base for the implementation of the wireless channels.

3.1.2.2.2. Graphical User Interface

Two of the solutions summarised in Appendix A - Table with Detailed Overview of Features of Previously Developed Weather Stations had a graphical interface described in their report. Both applications were developed to work as a base station and on a personal computer (PC).

Figure 13 shows the initial page of the Windows application provided by the team who developed “Real-Jan-16”. As seen from it, three main windows could be defined:

- a left window with most of the sensor data,
- a right window with a drawing of the weather station and identifier for the wind direction and the GPS data,
- and a bottom window with an interactive display for the user.

The blue fields of the sensors were made from interactive buttons and allowed a user to open a chart with the values over time for that sensor. The values were loaded from a local database. Underneath those two windows, several interactive buttons were placed:

- “View Data” for viewing of all data in a separate window in a table format
- “Show on Map” for showing a world map in a new window with indicated location of the weather station
- command buttons:
  - “Adjust Time” – for adjusting the time and date on the weather station
  - “Change Interval” – for setting the interval at which measurements should be sent from the weather station to the base station
  - “Change Counter”
  - “Calibrate” – for calibrating the solar panel position to the sun
  - “GPS ON”
  - “GPS OFF”
- buttons for control of the communication:
  - “Start COM”
  - “Show Available ComPorts”

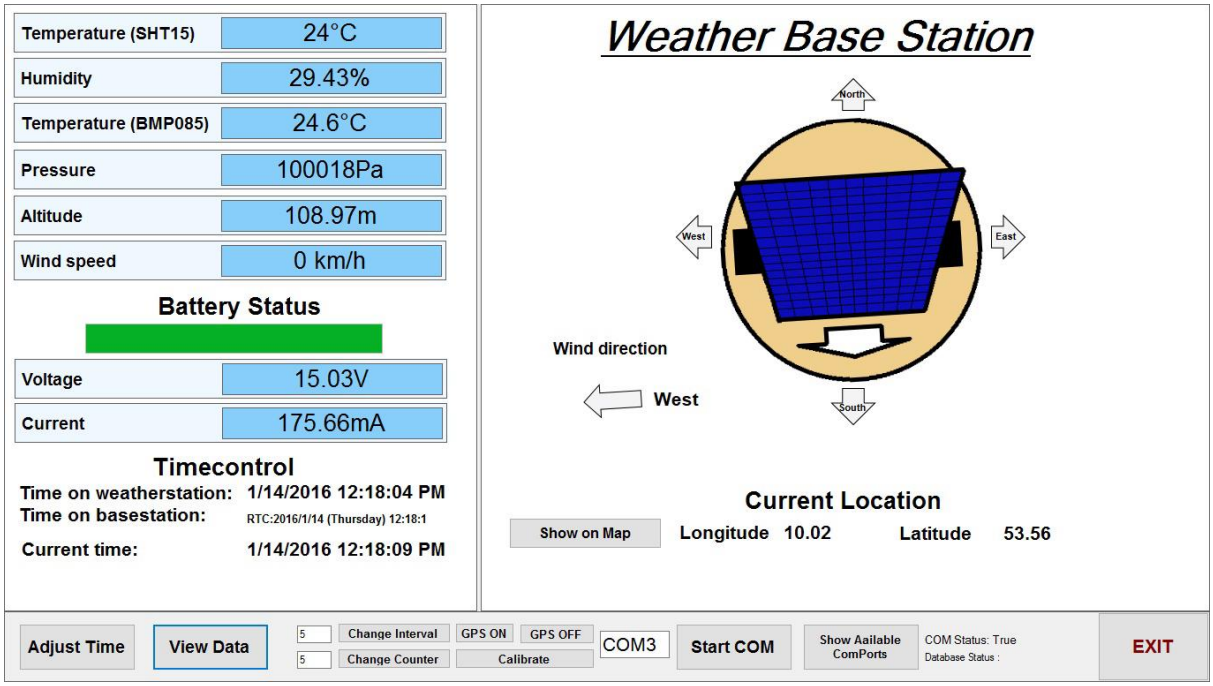


FIGURE 13: WEATHER BASE STATION APPLICATION GUI FOR “REAL-JAN-16”

The initial interface upon opening of the “Solar-Jan-20” PC application is shown in *Figure 14*. The application provided features for displaying randomly generated data in a chart format. Several parts could be distinguished:

- console for input and output at the bottom,

- data window with the chart graph - as seen in the picture, another type of data view was available - “Overview”
- data decision window – consisting of start and end time for the data to be displayed, two possible data fields with dropdown buttons and an “Update” button, which updates the data window
- and options bar at the top left corner consisting of options for Mode, View, Control and File

It was indicated in the report provided by the team that the Overview tab consists of sensor data in tabular form.

The options available for the “File” menu were: “Save” and “Open”, which were not implemented but were supposed to work with CSV files.

The options in the “Control” menu were:

- “Set Time”
  - UTC
  - Custom (not implemented)
- “Set position”
  - Hamburg
  - Custom (not implemented)
- “Adjust Orientation” (not implemented)
- “Set Update-Intervall”
  - 5 seconds, 1 minute, 15 minutes, 1 hour
  - Manual – with which the data must be requested and is not automatically sent to the Windows application
- “Set Measuring-Intervall”
  - 5 seconds, 15 seconds, 1 minute

Similarly, the “View” menu provided options for:

- Last-minute, last 15 minutes, Last Day, Last Week, Last Month
- Custom (not implemented) – the user is supposed to choose the start and end times from the main interface

The “Mode” menu consisted of 4 items:

- Enable Debug – enable the debug messages to be displayed in the console
- Disable Debug
- Enable COM – manually connect to the Weather station via Bluetooth, which was not needed to read measurement data automatically.
- Disable COM

FIGURE 14: INITIAL PAGE VIEW OF THE "SOLAR-JAN-20" WINDOWS APPLICATION

A note in the report indicated that the application worked in just one thread. Upon establishing communication with the weather station, it could not process any events triggered by the user, which proved problematic in terms of execution. Consequently, it was determined that this PC application would not make a good testing unit for the Bluetooth communication link. Instead, a commercial Bluetooth Serial Communicator with proven capability should be used to test the current wireless communications development. However, the thread issue should be passed on as a limitation to prevent any such occurrences with the developed application connected to the Bluetooth.

3.1.2.3. *Commercial Weather Provisioning Services*

From commercial standards, most of the requirements relate to the overall experience expected by the user, which is defined as the "user model" - "their mental understanding of what the program would do for them." [26]. The "user model" includes both the assumptions about what working with the solution might be like and the expectations propagating from previous contacts with other ones. Therefore, the result of any development must comply with both the expectations and the needs of the users. The end-user requirements give an overview of the intended use of the solution – the needs – but does not give specifics about the implementation of the features they automatically assume would be there without being discussed. Moreover, it was essential to look at solutions available on the market and identify features that, unless included, would lead to an unsatisfied user.

For this project, the end-user group would have contact with the weather station system via the weather application on the handheld device or the web page provided by the web server.

In this case, a look into portable digital weather stations with direct hardware access and web provisioning weather services was needed.

Most commercially available portable digital weather stations come either with a local display for showing current data or with a tablet/mobile or web application for collecting and analysing data over time and ultimately provide weather forecasts. Depending on the type of weather station, the provided data ranges from temperature, humidity and time measurement to including precipitation, wind speed, direction, and more.

If the user has direct access to the physical weather station, the following concerns were observed when choosing a weather station purchase [27]:

- How is the hardware powered - via changeable batteries, charging an embedded battery, or autonomously powered by a solar panel, and how long could the hardware go without needing a recharge/a change of the batteries?
- How easy would the weather station be install and set up?
- How easy would it be to operate the weather station if any additional settings are available?
- How many data values can be displayed on the local display of the weather station?
- What data is displayed and in what format?
- What are the connectivity options?
- How accurate are the readings?

Another commercially available weather service considered was forecasting websites or applications which use databases with already collected weather data and analyse them to produce a prognosis. The databases typically get data from a vast network of weather stations installed by individuals or institutions worldwide.

The similarities between the GUIs of popular web and mobile weather provisioning services such as “BBC Weather”, “AccuWeather”, “MyWorldWeather” suggest there are several features that are expected to be included in the GUI for a weather reporting service (see *Figure 15* and *Figure 16*):

- easy to read by the human eye data – font, font size, format (e.g. converted temperature data with up to 2 decimal points instead of raw data)
- hourly and daily forecast
- data on current temperature, humidity pressure, wind speed and direction, “real feel” temperature
- text summary of the weather with a form akin to “Partly sunny”, “Light clouds and a gentle breeze”
- selection of the period to be displayed - current weather, daily prognosis, weekly prognosis
- selection of location for the weather data to be displayed - options for user location and additional locations other than that of the user
- a scrollbar on the right side whenever more data is available on the page than is currently in the display window
- when on a mobile application: clickable menu icon either on the left or right upper corner. The menu icon should have a list of options, one of which is “Settings”. Other items in the menu list could be “Info”/” About us”, “Version”. The “Settings” menu should include unit selection for temperature
- when on a web page: menu bar with a selection of buttons for the period to display.

Nice to have features include:

- information about air quality

- minimum and maximum expected values for temperature
- sunrise and sunset times for a selected day
- weather map
- localised translation of the interface text depending on the location of the user accessing the service
- unit selection for data other than the temperature for mobile applications
- light/dark mode for mobile applications
- UV index, brightness

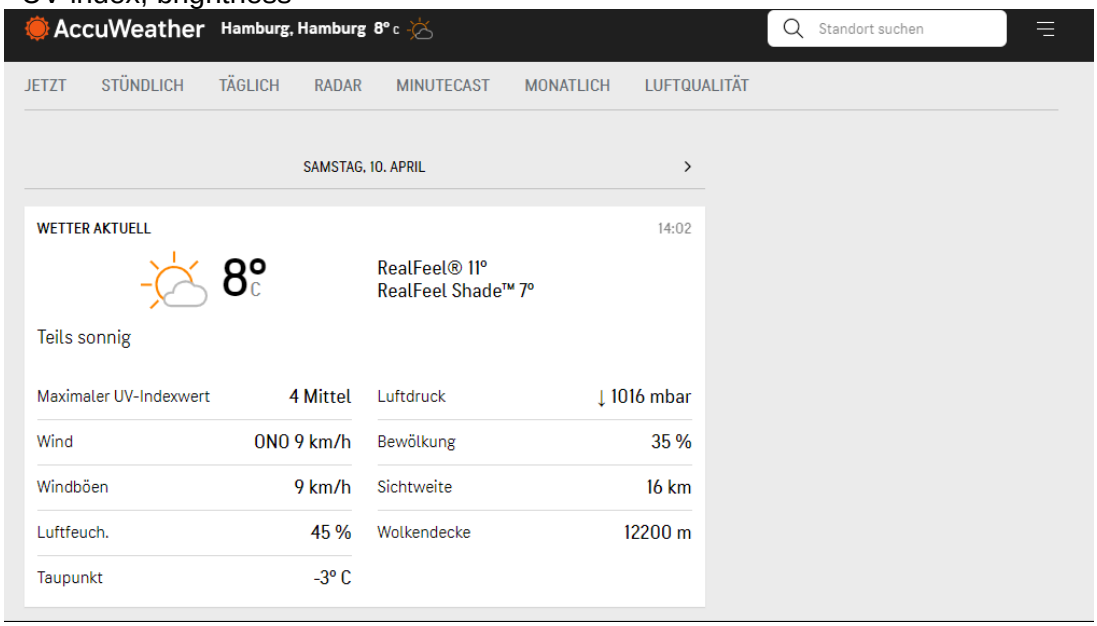


FIGURE 15: WEB PAGE OF ACCUWEATHER FOR CURRENT WEATHER WITH EXTENDED INFORMATION WINDOW

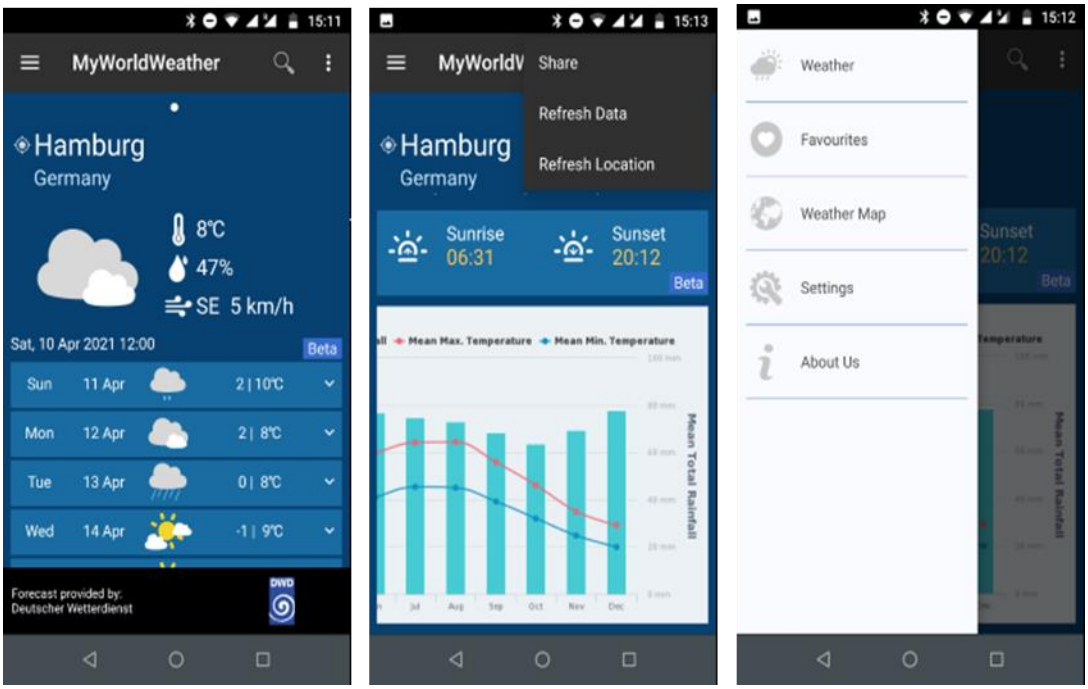


FIGURE 16: MYWORLDWEATHER ANDROID APPLICATION GUI AND MENUS

3.2. Specification

3.2.1. Fully Equipped Weather Station

A list of specifications applying for the fully equipped weather station project was derived and listed in *Table 2*. The specifications are based on the user requirements for a fully equipped weather station and the derived expectations set from previous stations created in the HAW Hamburg university, and some commercial standards about user interaction.

The requirements were grouped into **functional (F)** and **non-functional (NF)** and have a number assigned to them for later reference.

Additionally, they have been grouped into four main functional categories:

1. Communication: relating to the channels between the weather station, the handheld device, and the web server. It includes hardware and software needed for the transmission of data between the different modules.
2. Data collection: relating to data collection on the weather station from sensors and data collection on the handheld device from the weather station and the web server. It includes installing sensors and their interaction with the weather station and storing sensor data locally and remotely.
3. User interaction: relating to data display and options for customisation.
4. Other weather station functionality:
    a. Power system – relating to the inclusion of solar panel and battery and separate from power control subsystem to the weather station
    b. Other operational features

	Category	Requirement
F1	2	The weather station shall collect weather, location, and power data
NF1	2	Data collected shall include temperature, humidity, atmospheric pressure, solar panel orientation, wind speed, wind direction, altitude, longitude, latitude, time and date in UTC, current battery power
F2	2	Weather station shall save said data on local storage (e.g. SD card)
F3	1	The weather station shall send said data to the remote web server over an adjustable measurement interval
F4	1	The weather station shall connect to the remote web server via WLAN
F5	1	The weather station shall send any or all of the mentioned data to the handheld device
F6	1	The weather station shall send data to the handheld device over an adjustable update interval or on command
F7	2	The weather station shall collect data over an adjustable measurement interval
F8	1	The weather station shall connect to the handheld device via Bluetooth

F9	1	The weather station shall receive commands for data or control from the handheld device via Bluetooth
NF2	4	The weather station shall be an automated system
NF3	4	The weather station shall be solar powered
F10	4	The weather station shall have a low-power operating mode available
F11	4	The weather station shall have a demonstration, data collection and test operating modes available
F12	4	The weather station shall have local LEDs associated with components such as sensors, CPU, battery
F13	4	The weather station shall light LED on or off, indicating the location of said components
F14	4	Weather station shall have local buttons for change of the operating mode
F15	4	The weather station shall have all of its electronics visible and accessible for inspection
NF4	4	The weather station shall be initially developed for indoor use only - non-waterproof
F16	4	The weather station shall adjust solar panel orientation to that of the sun
NF5	4	The weather station shall be a real-time embedded system
F17	4	The weather station shall have a solar subsystem which includes a solar panel, a GPS module, a 3-axis magnetometer
NF6	3	The handheld device shall have a touch-sensitive display
F18	1	The handheld device shall have Bluetooth
F19	1	The handheld device shall have WLAN
NF7	3	The handheld device shall have an industrial appearance and feel
NF8	3	The handheld device shall allow for software with a graphical user interface
F20	1	The handheld app shall be able to use the Bluetooth functionality of the handheld device
F21	1	The handheld app shall be able to use the WLAN functionality of the handheld device
F22	2	The handheld app shall be able to receive a response to said commands via Bluetooth from the weather station
F23	1	The handheld app shall be able to request data from the weather station

F24	1	The handheld app shall be able to control certain functions of the Weather station: - set time, set measurement interval, set update data interval, set location, set operating mode, execute solar panel calibration, turn GPS on/off, set led on/off
F25	1	The handheld app should be able to request data from the web server
F26	2	The handheld app should be able to receive data from the web server
F27	3	The handheld app shall have a console display for input/output of a command to/from the weather station
F28	3	The handheld app shall have a window for the tabular display of data
F29	3	The handheld app shall have a window for the graphical display of data
F30	3	The handheld app shall provide options for data decision to the user: period selection, data to display selection
F31	3	The handheld app shall provide a Menu on the top left corner
F32	3	The handheld app menu shall include a list of commands that can be sent to the weather station
F33	3	The handheld app menu shall include an option for choosing parameters for the data display
F34	3	The handheld app menu shall include an option for unit selection for the data
F35	3	The handheld app shall provide text summarisation of the last measured data in the format “Hot and Raining.”
F36	3	The handheld app shall provide all measurement values converted to the appropriate unit
F37	3	The handheld app shall provide an option for weather station component selection where the LED associated with the component shall be lit on the weather station
F38	3	The handheld app shall allow user interaction and open communication channels at the same time
NF9	3	The handheld app shall provide all of the data in a readable format, including font and font size
F39	2	The web server shall store weather data
F40	3	The web server should display stored weather data

TABLE 2: LIST OF USER REQUIREMENTS

The web server, weather station, and handheld device could be considered systems with other subsystems. They interact with each other and with a fourth actor - a user, relating to the group of end-users. *Figure 17* displays the relations between these actors.

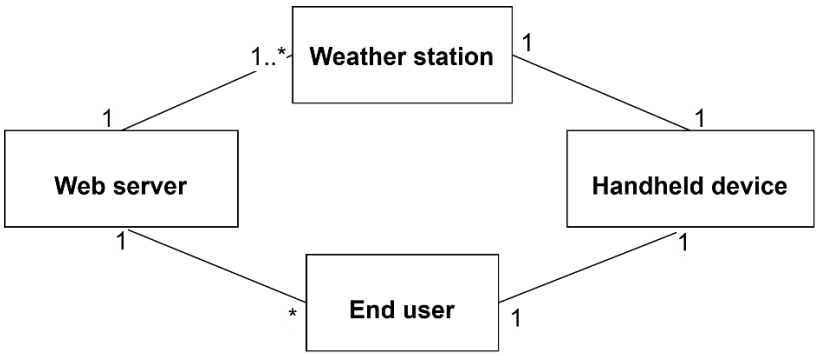


FIGURE 17: MINIMAL CLASS DIAGRAM WITH EXPECTED MULTIPLICITIES FOR THE FULL SYSTEM

The weather station system components are presented in *Figure 18*. The dotted channels indicate wireless communication. The filled lines indicate that physical access is intended.

The weather station itself was defined to have the following three subsystems:

- data collection,
- communication,
- and power subsystem.

The web server should have a database and a website displaying data from the database.

A mediator between the web server and the weather station should be a Wi-Fi-connected router, which might or might not, depending on the later implementation of the web server, connect to the internet.

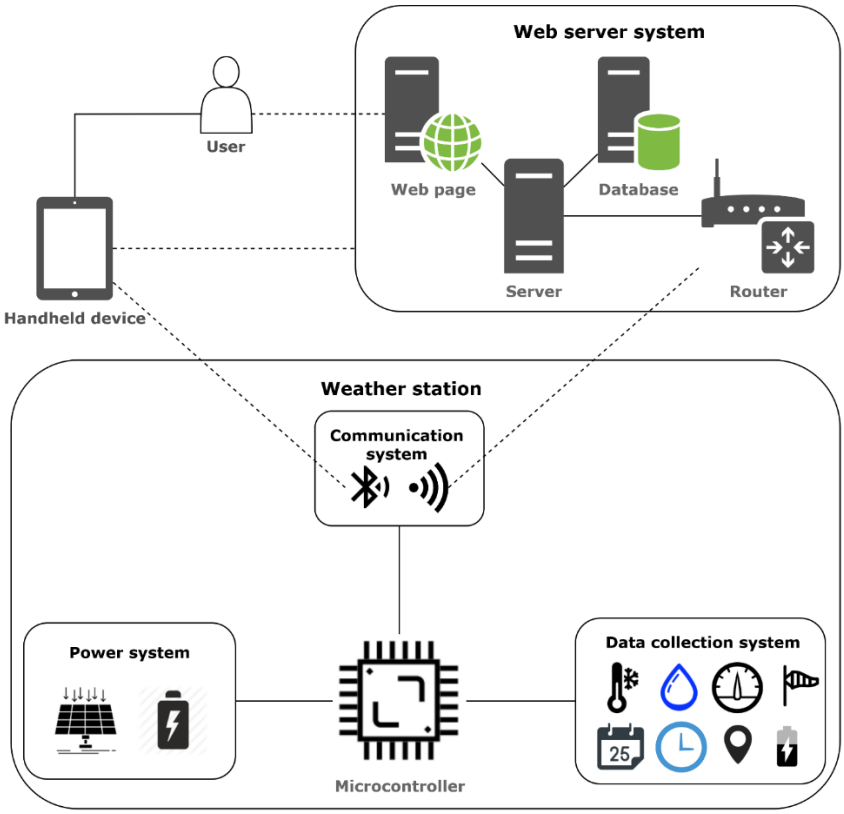


FIGURE 18: DATA RELATED COMPONENT OVERVIEW OF THE FULL SYSTEM INCLUDING WEATHER STATION, WEB SERVER AND HANDHELD DEVICE SUBSYSTEMS

The Handheld device system shall include Bluetooth and Wi-Fi hardware capability and a GUI application for the user (see *Figure 19*). The application shall use the Bluetooth capabilities for connecting to the weather station and WLAN to connect to the web server.

Figure 20 shows the data which is to be collected from the weather station. It includes several groups of data: weather data, real-time clock data, location data, solar panel data and power data. It was already indicated that a GPS module shall provide the location data, and the solar panel orientation shall be obtained from a 3-axis magnetometer.

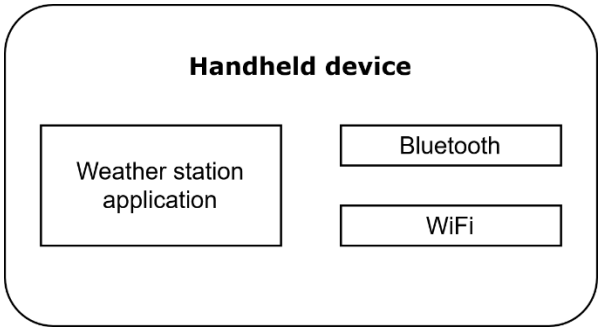


FIGURE 19: HANDHELD DEVICE SUBSYSTEM

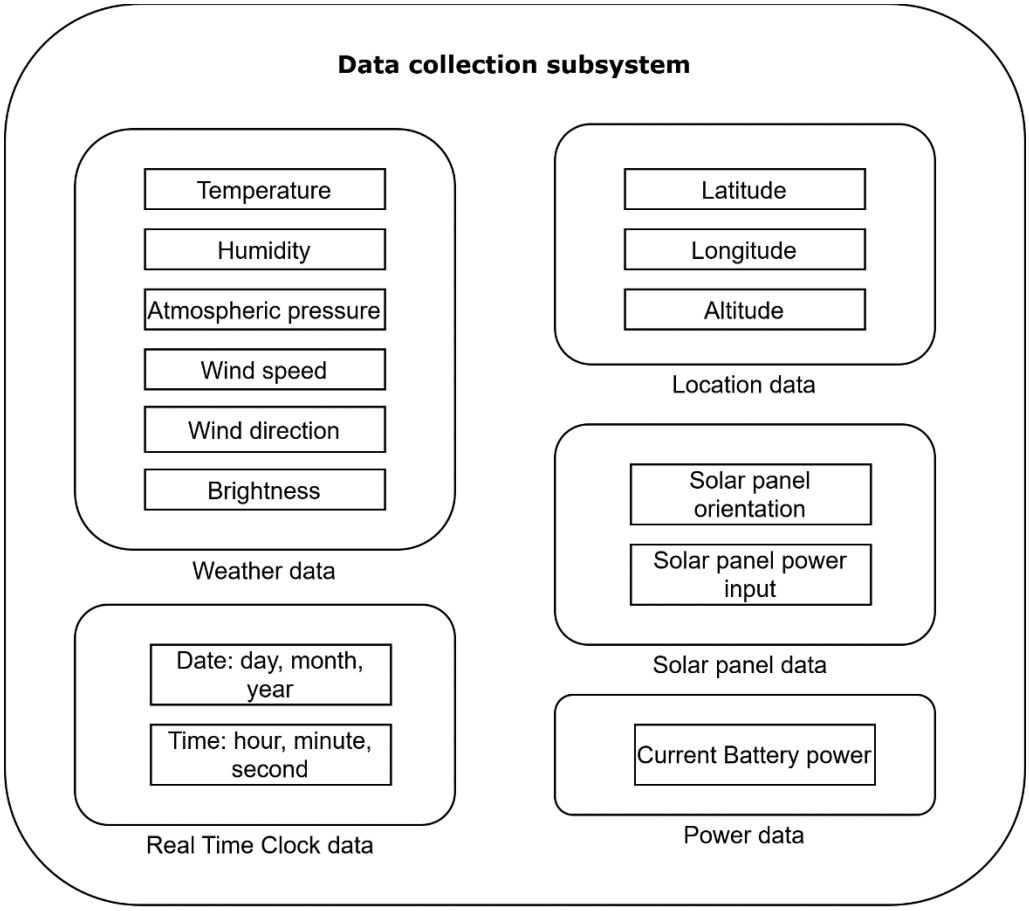


FIGURE 20: DATA COLLECTION SUBSYSTEM

Figure 22 and Figure 21 show the initial use case diagrams for the Handheld Application System and the Weather Station System. Further development of the specifications for each subsystem shall be created upon its implementation. This paper concentrates only on the Communication system.

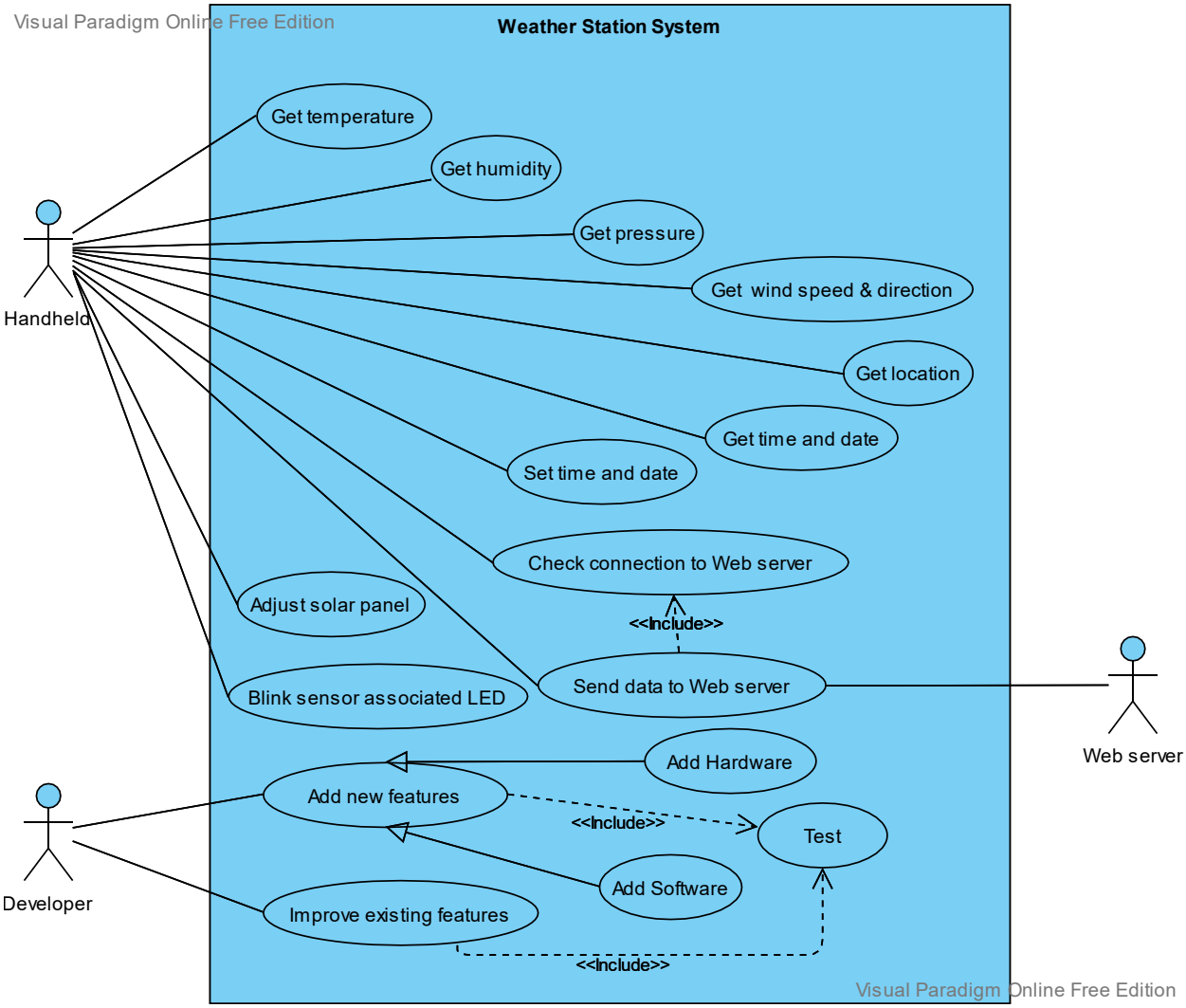


FIGURE 21: USE CASE DIAGRAM OF THE WEATHER STATION SYSTEM

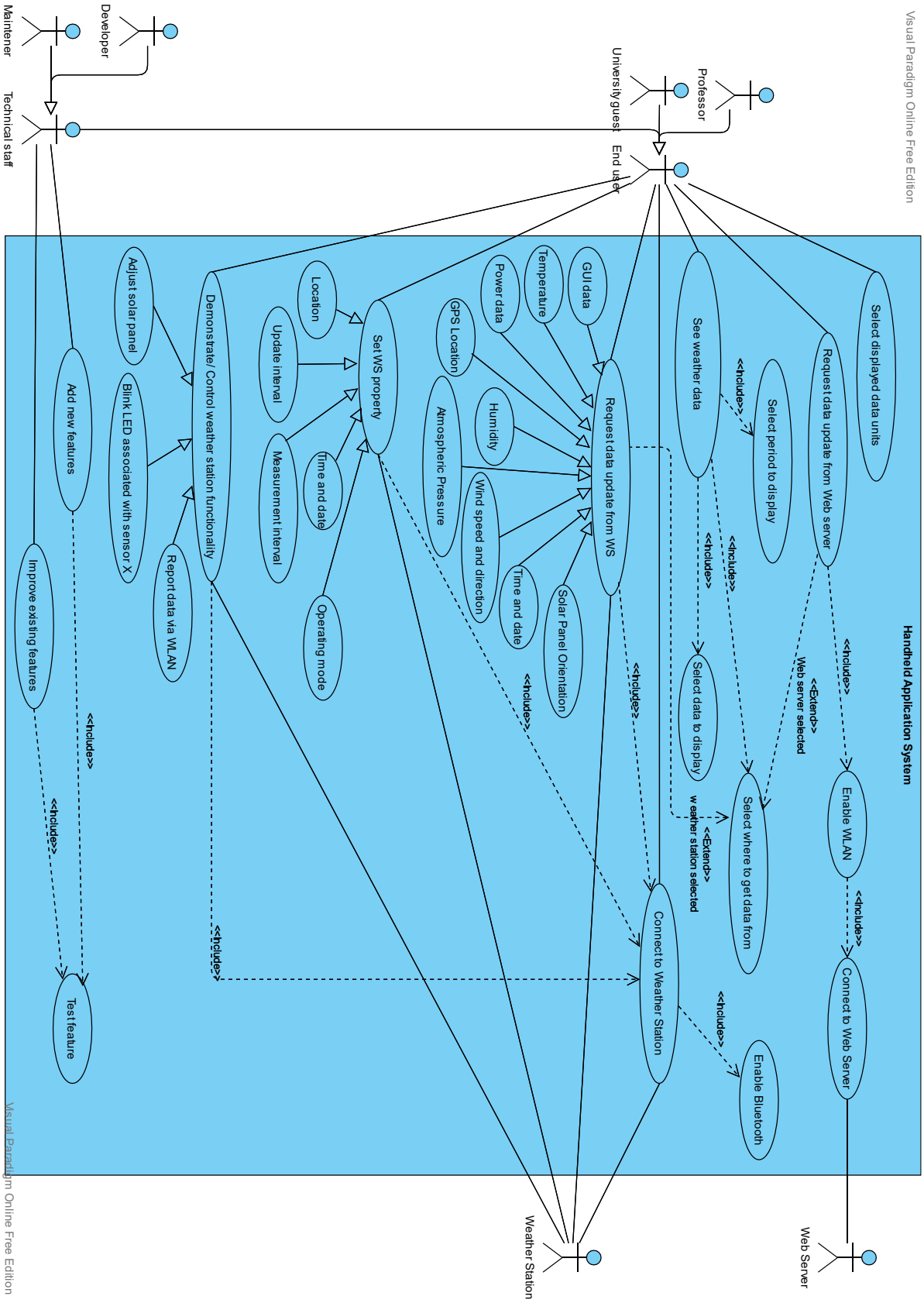


FIGURE 22: USE CASE DIAGRAM OF THE HANDHELD DEVICE SYSTEM

3.2.2. Communication System

The requirements for the communication system are listed in Table 3 and indicated by **CF for the functional** and **CNF for the non-functional specifications**, followed by a number.

	Requirement
CF1	A proof of concept shall be the end product of this thesis project.
CF2 (extending F4)	The concept would include that the initial weather station can use WLAN.
CF3 (extending F8)	The concept would include that the initial weather station can use Bluetooth.
CNF1	It should be further extendable with data collection and solar power systems.
CF4	A single data module (e.g. sensor, RTC) shall be integrated.
CF5	The changing data of the data module shall be sent over the communication links together with hardcoded values for the other expected data fields.
CF6	The solutions for both communication links shall allow for easy extension or change of the current functionality.
CF7	The microcontroller and the accompanying modules in this proof of concept shall be able to be powered separately, e.g. from a battery
CF8 (extending F15)	The modules used in the proof of concept shall be development boards, a.k.a. shall not just be a microchip, but have pinouts that allow for breadboard testing and possibly a USB connection for separately debugging each module.
CF9 (extending F9)	The Bluetooth connection shall use commands for instructing the weather station about a needed action. Any compatible Bluetooth device or software can be used for testing this property.
CF10	A handheld device shall be chosen, and the concept for its integration with the Bluetooth communication link shall be done. Any further development on the basis of this concept would be out of the scope of this project.
CF11	The proof of concept shall use the existing **"**Pro-Jan-17**"** web server infrastructure (see *Figure 23* and *Figure 24*) for testing the implemented WLAN communication link.
CF12	Data sent from the weather station to the web server shall be successfully uploaded and seen as a new value in the database.

TABLE 3: LIST OF COMMUNICATION SYSTEM REQUIREMENTS

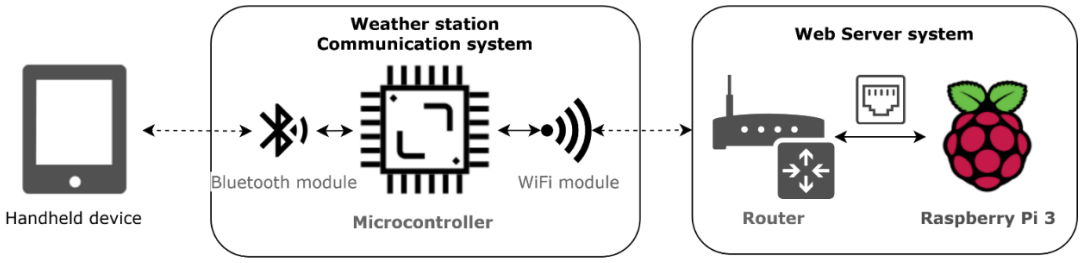


FIGURE 23: HARDWARE COMPONENT DIAGRAM OF WEATHER STATION AND ASSOCIATED WIRELESS DEVICES

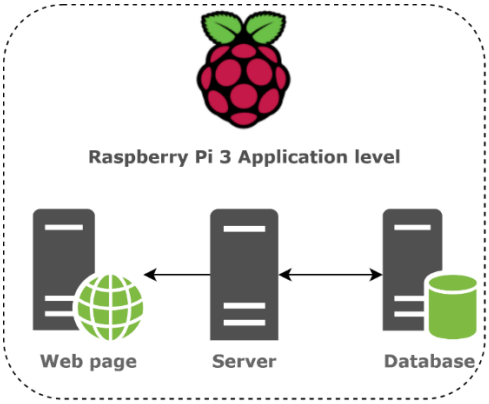


FIGURE 24: RASPBERRY PI SERVER STRUCTURE OVERVIEW

The core functionality of the weather station communication system can be seen in the use cases shown in *Figure 25*.

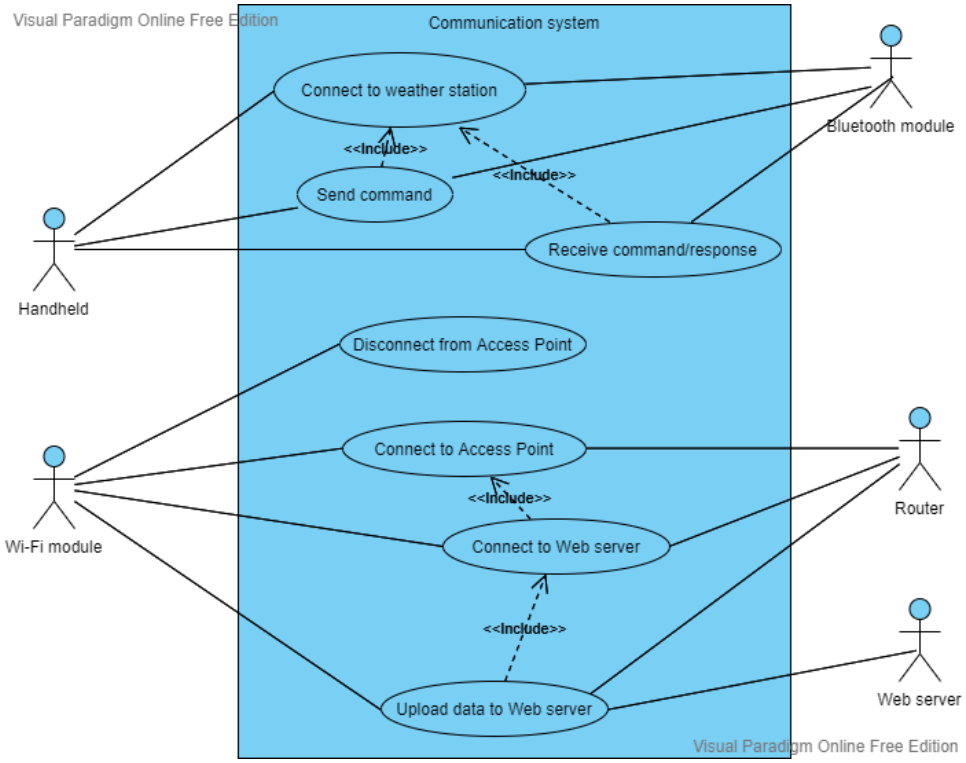


FIGURE 25: USE CASE DIAGRAM OF THE COMMUNICATION SUBSYSTEM

4. Concept

This chapter describes the decisions made about which hardware modules to use to fulfil the requirements defined in the previous chapter and the software design concept, which shall work as a base for the firmware architecture.

4.1. Hardware

The first step of the concept process was the decision on hardware components:

- a microcontroller for the base of the weather station,
- a WLAN and a Bluetooth module to enable wireless communication links,
- a data providing module for testing the data integrity of the communication channels,
- and a possible handheld device solution.

4.1.1. Microcontroller

The microcontroller plays an essential part as the heart of the weather station system – the data line for each module shall end at the microcontroller either for data gathering or control of the functionality. Therefore, it was essential to define several criteria for choosing a microcontroller, which can continue working as the weather station base past the scope of this thesis project, which involves only adding the communication links to the station.

One criterion should be that the program (flash) memory is sufficient for all modules intended for the final fully functioning weather station. In the reports of the previously developed weather stations in 2013 and 2014, using Arduino UNO provided some difficulties when integrating all modules in the firmware due to insufficient flash memory.

Another criterion already defined in the requirements was that the microcontroller could operate in low-power mode. This requirement was looked at from two perspectives – one is the microcontroller to have a sleep mode providing low-powered mode and how much current it would need with all peripherals integrated. However, the second criterion was defined mainly by the software development process and not by the hardware and was disregarded.

The third criterion was the availability of peripheral interfaces. As already discussed in the requirements elicitation, “Solar-Jan-20” would have been a great candidate for being extended with the needed functionality for a fully equipped weather station. However, the base microcontroller of the solution did not provide enough peripherals to integrate the needed additional modules.

Due to the scope of the final fully equipped weather station, it could be expected that even more modules might need integration besides the ones defined in the specification for the fully equipped station. Consequently, the following list of criteria for the microcontroller interfaces was decided:

- The microcontroller shall have at least one I^2C interface peripheral. According to the I^2C protocol specification, a single I^2C bus would allow 128 devices to be connected due to the 7-bit addressing mechanism of the I^2C protocol. On the other hand, not all sensor circuit boards provide I^2C interfacing. Therefore, it was necessary to give other interface options, thus enabling teams continuing the work on the weather station to have a certain amount of freedom when choosing their modules.

- An SD card development board typically provides only an SPI interface. An SD card would be needed to ensure that data is stored locally when it is impossible to push it to the web server. Due to the nature of the SPI interface, where one leader can have a maximum of two followers (if the negative of the chip select signal is used), the microcontroller board base should have a minimum of two SPI interfaces. The first one would be dedicated to the SD card module. The second one would be in case more boards provide only SPI interface (allowing for F2 to be implemented).
- When it comes to UART interfaces, one UART interface should be reserved for debugging purposes to allow the developers to pinpoint issues with the software, which is already troublesome in an embedded solution due to the frequent lack of debugging tools. Therefore, a minimum of two UART peripherals shall be available on the microcontroller chip.
- The program would need at least two Timers: one for timing the autonomous data collection (F7) and a second one for counting the Bluetooth data upload interval (F6). If the data upload to the server is different than those two intervals, a third one should also be available. Therefore, the microcontroller base should have a minimum of three general-purpose timers.

Another criterion for the decision matrix of the microcontroller was the provision of tools alongside the microcontroller development board, such as available IDEs, debugger, libraries. Those stand for the ease of integration and use of the chosen board.

The microcontroller should be available on a development or another board, providing access to the microcontroller functionality via standardly spaced at 2.54 mm pins. This would ensure fast development of the proof of concept for the current project and allow modules to be directly connected to the microcontroller board via standardly spaced connectors.

Additionally, the price point of the available solutions was to be observed.

Table 4 below shows an overview of the so far used microcontroller development boards for the weather base station from previous solutions created in HAW Hamburg.

Compared to the other microcontrollers, the Arduino UNO has a relatively small Flash memory and would not provide enough peripherals. The Stellaris LaunchPad Evaluation Kit LM4F120 (EK-LM4F120XL) has been discontinued, and an EK-TM4C123GXL evaluation board has been suggested as a substitute from its producer (Texas Instruments).

Arduino Due provides a sufficient number of peripherals and a lot of Flash memory. On the other hand, compared to the other boards, it has higher current consumption in its lowest power mode. It provides an internally integrated RTC, which cannot be powered separately. Furthermore, the price point is two to three times higher than other available boards. Most importantly, the board does not provide an integrated debugger, which would make the code implementation phase take longer.

The last two development boards were the NUCLEO-8L152R8, used in “Solar-Jan-20”, and the MSP-EXP432P401R, used in the “Pro-Jan-17” from which the web server would be used for testing the current weather station Wi-Fi implementation. As deducted from the requirements analysis phase, the station created with the NUCLEO-8L152R8 consisted of almost all features to be a fully equipped weather station as previously defined by this work. However, it would not have sufficient peripherals for adding a WLAN module and other sensors. This left the low-power MSP432 development board as the best board option used as a base for the previous weather stations. Given that the EK-LM4F120XL would have provided similar features, and Texas Instruments produced both boards, an investigation on the currently available development boards provided by TI was performed.

	Arduino UNO	Arduino DUE	EK-LM4F120XL	MSP-EXP432P401R	NUCLEO-8L152R8
MCU	ATmega328P (8-bit AVR)	SAM3X8E (32-bit ARM Cortex-M3)	LM4F120H5QR (32-bit ARM Cortex M4F)	MSP432P401R (32-bit ARM Cortex M4F)	STM8L152R8 (8-bit STM8)
Flash Memory	32 KB	512 KB	256 KB	256 KB	64 KB
CPU	16 MHz	84 MHz	84 MHz	48 MHz	16 MHz
Sleep modes	N/A	Sleep, Wait and Backup modes, down to 2.5 µA in Backup mode with RTC, RTT, and GPBR	Deep-sleep mode (sys clock at 30 KHz) with 1.05 mA, Hibernate mode with 1.7 uA for RTC enabled	Low-power mode with RTC with 660 nA	Halt mode with 400 nA
GPIOs	14 (6 PWM)	54 (12 PWM)	40 (24 PWM)	40 (4 PWM)	54 (3 PWM)
Peripherals	1x UART 1x I^2C/TWI 1x SPI 1x USB 6x ADC 3x Timer	4x UART 2x I^2C/TWI 1x SPI 2x USB 2x DAC 12x ADC 9x Timer RTC	4x UART 4x I^2C 4x SPI 2x 12-bit ADC 6x Timer RTC	4x UART 4x I^2C 8x SPI 2x 12-bit ADC 6x Timer RTC	3x UART 1x I^2C 2x SPI 2x DAC 1x ADC 5x Timer RTC LCD
IDE	Arduino IDE/ Atmel Studio	Arduino IDE/ Atmel Studio	Code Composer Studio	Code Composer Studio	STVD
Debugger	No/ yes(external)	No/ yes(external)	In-Circuit Debug Interface	Build-in	yes
MCU Library	Arduino/ ASF	Arduino/ ASF	StellarisWare	SIMPLELINK-MSP432-SDK	STM8-SPL
Price point (mouser.de)	17,70 €	31,68 €	Obsolete (discontinued)	22,52 €	9,10 €

TABLE 4: COMPARISON TABLE OF MICROCONTROLLER DEVELOPMENT BOARDS USED IN PREVIOUSLY DEVELOPED WEATHER STATION SOLUTIONS

The portfolio of microcontrollers of Texas Instruments was divided into three major families: ARM-based, C2000 real-time, and MSP430. All the solutions come with a microcontroller peripheral library available when using Code Composer Studio.

The ARM-based products were grouped into several families: Arm-M4F (TM4C) and Arm Cortex R4&R5.

C200 MCUs provide control for power electronics requiring advanced digital processing.

The last group consists of 16-bit microcontrollers with a concentration on system costs.

The microcontroller of the EK-TM4C123GXL recommended by Texas Instruments as a replacement for the EK-LM4F120XL Stellaris LaunchPad was part of the TM4C family which is defined by a 32-bit ARM Cortex-M4F and multiple communication peripherals such as CAN, USB and Ethernet.

Table 5 below shows an overview of the microcontroller series part of the TM4C family [28]. The family consists of two production series, namely the TM4C123x and the TM4C129x. Both groups have more than sufficient peripherals, flash memory, and provide several tools for fast development: IDE, TivaWare SDK, and have a development board based on them.

The first group was characterised by up to 80 MHz CPU, Flash memory of up to 256 KB, and power consumption in deep-sleep mode down to 1.6 µA in hibernation mode.

Compared to the TM4C123x group, the TM4C129x group provides even more peripherals and memory and lower power consumption. Consequently, a microcontroller base from the TM4C129x group was determined to have better properties.

	CPU	Memory			Lowest power consumption	Peripherals	Tools
		Flash	SRAM	EEPROM			
TM4C123x	Up to 80MHz	Up to 256KB	Up to 32KB	2KB	1.6 µA in Hibernate mode	64-168 GPIO (Up to 40x PWM) 8x UART 6x I^2C 4x SPI/SSI 2x CAN 1x USB 2.0 2x 12-bit ADC RTC	Code Composer Studio IDE TivaWare SDK
TM4C129x	Up to 120MHz	Up to 1MB	Up to 256KB	6KB	1.30 µA in Hibernate mode with RTC enabled	128-212 GPIO (Up to 8x PWM) 8x UART 10x I^2C 4x QSPI/SSI 1x 1-wire leader interface 2x CAN	Code Composer Studio IDE TivaWare SDK

						1x USB 2.0 1x Ethernet 1x LCD controller 2x12-bit ADC 8x 32-bit Timer RTC	

TABLE 5: OVERVIEW OF TM4C MICROCONTROLLER FAMILY SERIES

An evaluation of the development boards based on TM4C129x microcontrollers is presented in Table 6. There are two solutions available: the TM4C1294XL evaluation kit and the TM4C129x development kit. Both solutions provide an in-circuit debug interface (ICDI). However, the price point of the Development Kit based on the TM4C129XNCZAD microcontroller is more than several times higher than that of the EK-TM4C129XL. The EK-TM4C1294XL was still in a comparable price range with other solutions mentioned before, whilst offering more versatile properties than many of the above-discussed solutions. It was consequently chosen as the base board of the weather station. The suggested tools for software development by the board producer were CCS IDE and TivaWare. They were chosen to be used for the development of the proof of concept.

	MCU	Interfaces	Price Point (mouser.de)
Evaluation kit (EK) TM4C1294XL	TM4C1294NCPDT	2x stackable BoosterPack XL connection interfaces (access to 80x GPIO in total), Ethernet port, Reset and Wake push buttons, 1x micro A/B USB connector, 2x user switches, 4x user LEDs, 1x USB Debug port, External debug connection, 98x Breadboard connection Headers [23, p. 4]	22,52 €
Development kit (DK) TM4C129X	TM4C129XNCZAD	1x BoosterPack and 1x BoosterPack XL interfaces(access to 104 GPIO in total), Integrated LCD, speaker, microSD, user tricolour LED, power and Ethernet LEDs, Up and Down buttons, Select and Reset buttons, Ethernet port, 1x USB AB OTG, 1x USB micro B Debug/power [29, p. 4]	224,18 €

TABLE 6: OVERVIEW OF TM4C129X BASED DEVELOPMENT BOARDS

4.1.2. WLAN Module

The requirements for the WLAN module were to utilise the Wi-Fi (IEEE 801.11) protocol and to be breadboard compatible for the proof of concept. The router available as part of the web server solution for testing the implementation of the WLAN module was a TPlink TL-WR841N. The device works with the “n” standard of Wi-Fi, making it compatible with “b” and “g” Wi-Fi connected devices, which comprise the multitude of commercially available devices for wireless connectivity. Moreover, the web server available worked with TCP/IP, which required the Wi-Fi module to utilize TCP/IP.

Table 7 below shows several Wi-Fi modules for achieving the WLAN link.

The CC3100BOOST [30] module would have been the recommended module to use as it is provided by Texas Instruments and integrates well with the chosen TM4C microcontroller. However, its price point is higher than that of the base microcontroller, making it a bad match for this project. Additionally, it would need to be placed directly on top of the microcontroller, which would block the direct view of the microcontroller board.

The ESP8266EX is a system on a chip (SoC) often used in non-professional and student IoT projects. It provides several connection interfaces, including UART, SPI and I^2C. The microcontroller of the ESP8266 comes with an SDK available from its producer, Espressif, and implements the standard TCP/IP stack. The SDK has an integrated set of Hayes-style commands for easy TCP/IP connections [31] and three sleep modes, where the power consumption could go as low as 20 µA [32]. Multiple development boards and modules, such as the NodeMCU and the Wemos D1 mini [33], integrate this chip.

The NodeMCU itself is a development kit for using the NodeMCU open-source firmware, which is layered on top of the existing Espressif SDK. Both modules provide similar features matching the requirements for the Wi-Fi module for this communication solution. The NodeMCU, however, has a slightly larger size and is slightly more expensive than the Wemos D1 mini board. Therefore, the Wemos D1 mini board was chosen to implement the Wi-Fi link on the weather station.

	Features	Price point
NodeMCU (ESP8266 based)	Flash: 4MB 10 GPIOs I^2C, 1-wire, 1 ADC, micro USB Weight: 50g Size: 4.8cm x 2.38cm	8.99 € (conrad.de)
Wemos D1 mini (ESP8266 based)	Flash: 4MB 11 GPIOs UART, SPI, I^2C,1 ADC Status LED, micro USB Weight: 3g Size: 3.42cm x 2.56cm	7.51 € (conrad.de)
CC3100BOOST	2x20 GPIOs 4x status LEDs 2x push buttons UART, SPI, USB type B	55,20 € (mouser.de)

TABLE 7: COMPARISON TABLE OF WI-FI TCP/IP PROVIDING MODULES

4.1.3. Bluetooth Module

For the Bluetooth module, the requirements related only to the hardware, where it shall allow for Bluetooth communication and be compatible with a breadboard.

The Bluetooth module from “Solar-Jan-20” was still available and could be used as it meets the criteria.

It is an HC-05 board based on the EGBT-045MS microchip [34]. The board itself comes with a UART interface with Serial-to-Bluetooth conversion and two additional pins for Connection Status and mode selection.

The microchip comes with a preinstalled AT command set. However, due to the need for custom commands, the development team decided to permanently put the module in “Data Mode”, where all data is directly transferred over the UART via connecting the Mode Selection pin to a 0V internal pin. This enabled the team to create custom commands for operating their weather station functionality via Bluetooth. The “Data Mode” puts the Bluetooth module in follower mode, where it can be only connected to but cannot initiate a connection itself. Because of these changes, the UART configuration of the module is permanently set to 9600bps, 8 data bits, 1 stop bit, no parity, and no handshake.

Additionally, the connection parameters for the module were set to:

Passkey:	1234
Device Name:	Wetterstation_1a

TABLE 8: BLUETOOTH CONNECTION PARAMETERS

The HC-05 module also provides a red LED indicating the status of the connection. If the module is in broadcast mode and not connected, it would continuously blink. When the module has been paired and connected to, it will blink twice with an interval of approximately 5 seconds.

4.1.4. Data Provisioning Module

As described in the requirements for this package, a module for providing variable data was needed to test that the wireless connections transfer data correctly. The TM4C1294XL LaunchPad, chosen as the base board for the weather station, provided an onboard Hibernation module incorporating a Real-Time Clock (RTC). This RTC could be used for providing new data over time to test if data is stored and extracted correctly in the main routine of the microcontroller and was therefore chosen as the data provisioning module.

4.1.5. Handheld Device for Monitoring and Maintenance

For the handheld device on the other end of the Bluetooth connection, several options were available: to create the device from scratch or to use an already available module on the market.

An in-house made device would require a touch-sensitive display, base microcontroller with graphic capabilities, Wi-Fi and Bluetooth communication, battery. The integration itself would have been a whole other project, so it was not feasible in the scope of this thesis work.

Consequently, the handheld device was decided to be a store-bought solution.

Two sets of tablets were derived for comparison: computer modules with display and rugged industrial tablets, both presented in Appendix B - Comparison Tables for Handheld Devices . All of the included devices covered the requirements for the handheld device to have an industrial feel and have Wi-Fi and Bluetooth communication channels.

The first set consisted mainly of separate from each other display and processing unit, whereas the second set was with ready-to-use handheld solutions with mobile capabilities (see Figure 26).

The devices listed in Table 17 required prices quoting from the distributing or production company. The tablets catalogued in Table 18 were available with price tags for purchasing either via the manufacturer or via online electronics distributor platforms based in Germany. This proved to be an important property at the time of development of the currently discussed project due to limitations of import products of countries outside the European Union.

The two selection sets were presented to two of the involved stakeholders from the end-user group – Prof. Dr.-Ing. Lutz Leutelt and technical assistant Detmar Rüdiger. A decision was made that the device would be purchased by invoice, and the price shall not exceed 500 €. Therefore, the list of computer modules with a display was removed entirely from the selection due to the needed request for a price quote. From the selection of industrial tablets, only the Apglos Armour tablet was available by a European distributor, wherefore purchasable by invoice in the decided price range. The Apglos Armor was chosen as the handheld device to implement the Bluetooth link.

FIGURE 26: FRONT AND BACK IMAGES OF THE SELECTION OF INDUSTRIAL TABLET

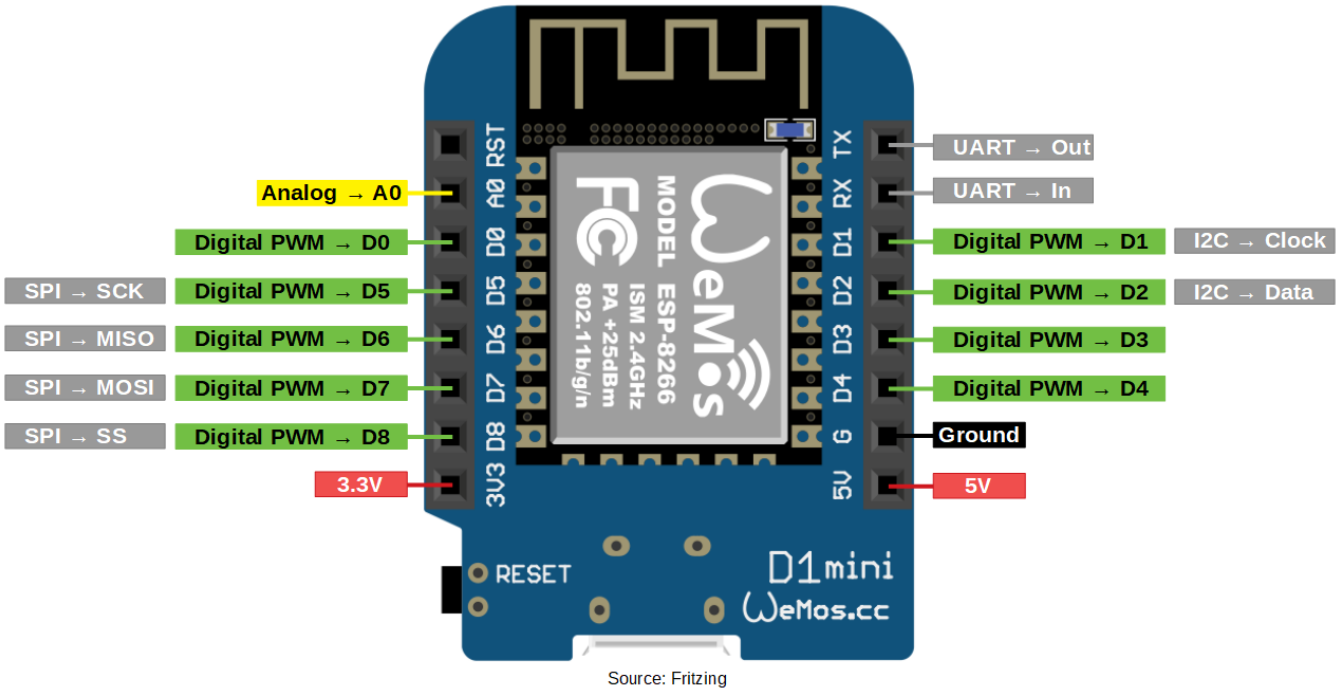
4.2. Hardware Structure

This subchapter analyses the available hardware interfaces of the modules and defines the type of interfaces needed for connecting the wireless modules to the chosen microcontroller development board.

4.2.1. Wemos D1 Mini

The Wemos D1 mini chosen for the Wi-Fi communication provides single UART, SPI and I²C interfaces for connecting to other devices (see *Figure 27*). Using the SPI and I²C interfaces require software programming and setup via the existing Serial to USB interface of the module to configure the mode of operation (leader or follower) [35, p. 3]. Therefore, to directly use the AT command driver and the TCP/IP stack available with the ESP8266 core firmware of the module without any further configuration of the microchip, the UART interface was decided to be used for connecting to the weather station system.

The Wemos D1 board provides access to two UART interfaces - UART0 and UART1. UART0 had dedicated hardware pins working with default transmission speed 115200. UART1 was connected to the flash, and only the transmit signal pin (D4) could be used, typically for debugging purposes. Ergo, the fully-functioning UART0 of the Wemos D1 mini module was chosen for interaction with the LaunchPad.



Source: Fritzing

FIGURE 27: PINOUT OF THE WEMOS D1 MINI BOARD [36]

4.2.2. HC-05

Figure 28 shows the physical appearance of the HC-05 Bluetooth module and its pinout. The module provides only a single UART interface, which was chosen for the connection to the microcontroller. It is important to note that the voltage required to power the module is 5V. However, the other board pins operate at 3.3V. The State pin indicates whether a device has connected to the Bluetooth module when high level (1), and the Enable pin (EN) can be set low (0) from the microcontroller when the module should be disabled.

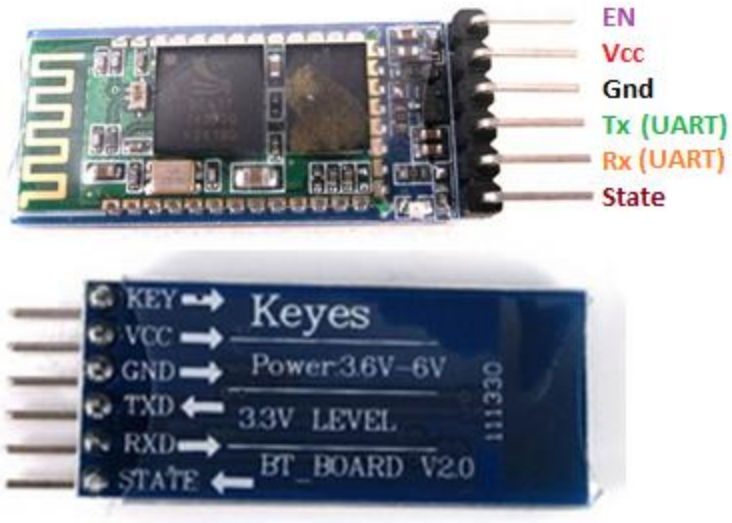


FIGURE 28: PINOUT OF THE HC-05 BOARD [37]

4.2.3. EK-TM4C1294XL LaunchPad Connections to the Wemos D1 Mini and HC-05 Boards

The Wi-Fi and the Bluetooth modules each used a UART interface and were defined to occupy two of the existing UART peripherals of the LaunchPad. The Bluetooth State pin was to be connected to one of the general-purpose pins as input and used to determine the connection status of the Bluetooth module.

Figure 29 shows the expected hardware component diagram, where the LaunchPad is one component with several subcomponents derived from the microcontroller block diagram ([24, p. 54]). GPIOs represent the UART peripherals in the real world, where the GPIO can have an alternate functionality as UART. The HC-05 and Wemos D1 mini modules are external components that use the provided interfaces (pins) from the LaunchPad board.

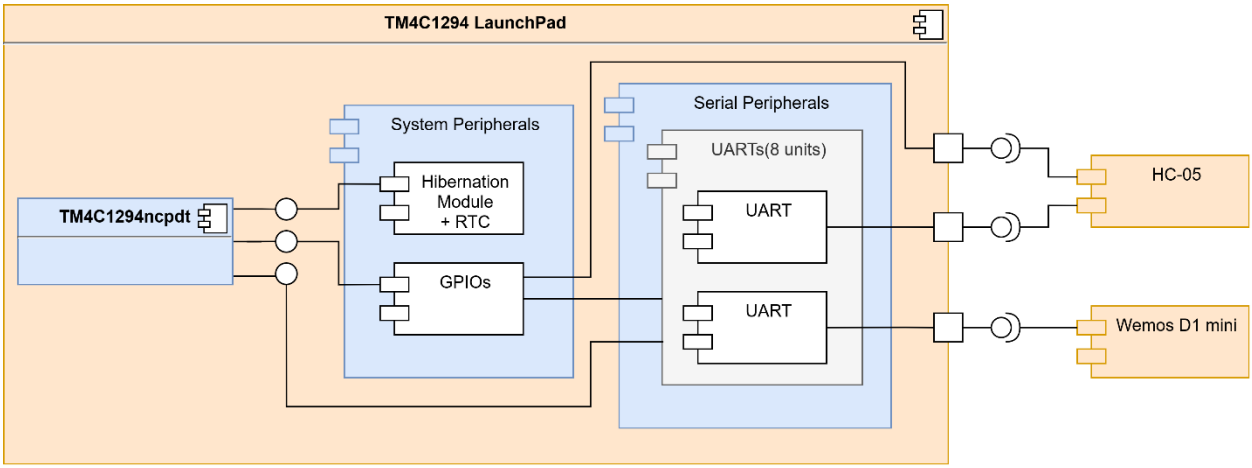


FIGURE 29: COMMUNICATION SUBSYSTEM HARDWARE COMPONENT DIAGRAM

4.3. Software

After the decision of the hardware components and their connection types, an outline of the software design was derived. It included analysis of the expected behaviour and construction of the needed software model.

4.3.1. Structures

4.3.1.1. *WLAN Structures*

Based on the source code provided with the respective report for the weather station and accompanying web-server developed in 2017, the web-server expects a string wrapped with an "A:" at the beginning and a "\n!" at the end with semicolon-separated values (“;”). The total number of expected value fields is 21. Whenever no new value is present compared to the last measurement, the data slot should be left empty with no spaces between, just the semi-colon separating it from the following data field.

Figure 30 shows the expected string with the parameters. The parameters should be replaced by numerical values representing raw data.

This structure was expected to be sent from the new weather station to the current web server to test the Wi-Fi communication channel.

Following this data upload format, the proof of concept can assume that the data was already available and in the correct format. As indicated in the requirements, all data values, except for the ones provided by a single data provisioning module - in this case, the already defined RTC - shall be hardcoded (CF5).

```
A:year;day;month;hour;minute;second;currentSupply;currentSolarPanel;voltage;windDirection;
windSpeed;pressure;temperature1;latitude;longitude;compass;humidity;temperature2;freeSDspace;
solarPanelAngle;\n!
```

FIGURE 30: WEB SERVER EXPECTED DATA FORMAT

Furthermore, based on the code source provided, the parameters for connecting to the server were extracted and are displayed in Table 9.

	Value
Access point name (<ssid>)	Wetterstation
Password (<pwd>)	wetterdeluxe
Medium (<type>)	TCP
IP (<ip_addr>)	192.168.178.82
IP Port (<port>)	100

TABLE 9: WEB SERVER CONNECITON PARAMETERS

4.3.1.2. *Wi-Fi Commands*

The ESP8266 microchip of the Wemos D1 module provides a driver with AT commands, including basic AT commands for the operation of the module hardware, Wi-Fi commands, and TCP/IP-related AT commands [31].

Considering the data provided from Espressif for connecting to a TCP/IP server and uploading of single data strings [38], the following subset of AT commands were decided to be implemented on the weather station to enable data upload to the server via using the ESP8266 driver (see Table 10).

	Description	Expected Response
AT	Test AT module	OK
AT+RST	Restart the module	OK
AT+CWJAP_CUR=<ssid>,<pwd>	Connect to an Access Point (AP)	OK
AT+CWQAP	Disconnect from the AP	OK
AT+CIPSTATUS	Get the connection status	+CIPSTATUS:<link ID>,<type>,<remote IP>,<local port>,<tetype>
AT+CIPSTART=<type>,<ip_addr>,<port>	Establish a TCP, UDP or SSL connection	OK
AT+CIPSEND	Send data	SEND OK
AT+CIPCLOSE	Closes the TCP, UDP or SSL connection	OK

TABLE 10: WI-FI AT COMMAND SET TO BE IMPLEMENTED

4.3.1.3. *Bluetooth Commands*

The HC-05 Bluetooth module taken from “Solar-Jan-20” could not use the AT command set described in its datasheet due to working in “pass-through” mode. The weather station and the handheld device were defined to share a set of commands and implement them on both sides of the Bluetooth link to ensure compatibility for the communication.

The command format could be as simple as singular numerals or letters. However, to provide a universal feel of the solution and allow for further development of the list of commands, a command set following the extended “AT command syntax” rules were defined to be used.

“Solar-Jan-20” also implemented an extended AT command style set for the Bluetooth communication (see Table 11). These commands were expected to be part of the new Bluetooth solution.

	Description	Expected Response
AT	Test connection	OK
ATE<n>	Activate/Deactivate remote echo	OK
AT+CTEMP?	Get temperature measurement	+CTEMP: <bme>,<cpu>,<qmc>,<mpu> OK
AT+CPRES?	Get atmospheric pressure measurement	+CPRES: <bme> OK
AT+HUM?	Get humidity measurement	+CHUM: <bme> OK
AT+CWIND?	Get wind direction and speed	+CWIND: <w_dir>,<w_spd> OK
AT+CTIME?	Get weather station time and date	+CTIME: <YY>,<MM>,<DD>,<hh>,<mm>,<ss> OK
AT+CTIME=<YY>,<MM>,<DD>,<hh>,<mm>,<ss>	Set time and date of the weather station	OK
AT+CALIGN?	Align the weather station solar panel position to that of the sun	+CALIGN:<azm>,<zen>
AT+CGNSPOS?	Get GPS position	+CGNSPOS: <lat>,<lon>,<alt> OK
AT+CGNSPOS=<lat>,<lon>[,<alt>]	Set GPS position	OK
AT+CPWR?	Get power statuses	+CPWR: <v_bat>,<i_bat>,<v_solar>,<i_solar>,<v_sys> OK
AT+CINTV=<intvl>	Set the measurement interval	OK
AT+CGUI?	Get weather station data for graphical representation	+CGUI:<YY>,<MM>,<DD>,<hh>,<mm>,<ss>,<t_bme>,<t_cpu>,<t_qmc>,<t_mpu>,<w_dir>,<w_spd>,<pres>,<hum>,<zen>,<azm>,<lat>,<lon>,<alt>,<v_bat>,<i_bat>,<v_solar>,<i_solar>,<v_sys> OK

TABLE 11: LIST OF BLUETOOTH AT COMMANDS IMPLEMENTED IN "SOLAR-JAN-20"

However, this set of commands did not cover all the functionality required for the showcase weather station. For that reason, it was extended with commands allowing the missing functionality:

- set the Bluetooth data update interval (F24)
- set weather station operating mode (F24)
- set GPS on/off (F24)
- set LED on/off (F24)

A framework for the Bluetooth AT commands was developed as follows:

- Commands contain “AT” at the beginning of the string and are followed by a plus (“+”) sign and the command name. The command name is an abbreviation indicating the purpose of the command and is written with uppercase letters. The first letter of the name string is “C”, indicating it as a command.
$$AT+<command\_name>$$
- Any parameters to the set commands are added by an equal (“=”) sign after the command name and the parameter's value. If more than one parameter is applicable, the parameters are separated by the comma (“,”) character with no spaces between them.
$$AT+<command\_name>=<value\_1>,<value\_2>$$
- Similarly to the developed concept for “Solar-Jan-20” parameter “read” commands contain a question mark (“?”) character after the command name.
$$AT+<command\_name>?$$
- A command which is an “execute” type and does not require any return parameters shall have the exclamation mark (“!”) character. By definition of the AT command syntax, this is permitted.
$$AT+<command\_name>!$$
- A “read” command should return a string with parameter values together with the command name. The “+” sign is placed before the command name, and a “:” sign after it is added. The parameter values are separated with a comma (“,”) if more than one is applicable.
$$+<command\_name>:<value\_1>,<value\_2>$$
- If a command has been executed successfully, the weather station shall return an “OK” line on the Bluetooth link.

The complete list of the commands to be implemented can be checked in Appendix C - Bluetooth Commands to be Implemented and Command Parameter Definitions. A list of the parameters used in these AT commands for clarity of the definitions is also included.

4.3.1.4. *RTC Holder*

As part of the Hibernation module, the Real-Time Clock had already defined software functionality by the TivaWare drivers. However, a wrapper for assigning and reading the time of the module needed to be created. Furthermore, based on the data structure required for the Wi-Fi Upload link, the following structures were defined to be part of the RTC wrapper:

- date: year, month, day
- time: hour, minute, second

4.3.2. Behaviour

4.3.2.1. *Weather Station System*

The primary purpose of the weather station's communication system was to enable control and data monitoring from outside sources, namely a handheld device and a webserver. *Figure 31* illustrates the operating modes described in the requirements phase as state behaviour of the fully functional weather station.

Based on the specification in the requirements phase, the weather station was defined to have three operating modes (F11):

- data collection (regular operation),
- demonstration,
- and test(maintenance) mode.

The first was defined by the autonomous operation of the weather station such as:

- the collection of data over a set interval (F7),
- data upload to a remote web server or save on a local SD card if there is no access to the webserver (F3 and F2),
- adjustment of the solar panel orientation for optimal sun power input (F16),
- and entering in low-power mode when those actions are not performed (F10).

The second operating mode was defined by the shift of the control to the device connected by the Bluetooth link. As long as the device is connected, the user on the device end can control the weather station and its connected devices.

The third operating mode should incorporate all the functionality of the second mode. However, it should also include the performance of selected self-maintenance tests on the weather station and their report over the Bluetooth link.

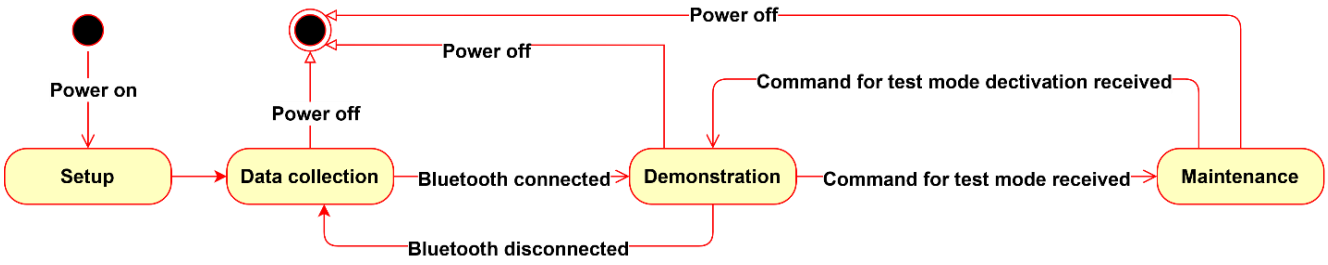


FIGURE 31: WEATHER STATION MODES - STATE MACHINE DIAGRAM

Consequently, the general flow of the program was decided to be as follows (see *Figure 32*):

1. power on the microcontroller,
2. peripherals setup,
3. infinite loop.

The loop would ensure that the weather station software would continue to run as long as the hardware is powered on.

The peripherals setup refers to the powering and enabling of the microcontroller and its peripheral modules (general clock, GPIOs, UARTs, interrupts) and is typically done once at the beginning of the program and is not changed through the run of the program.

The loop is a general infinite loop ensuring that the program continues running as long as the device is powered on and would continue to execute actions.

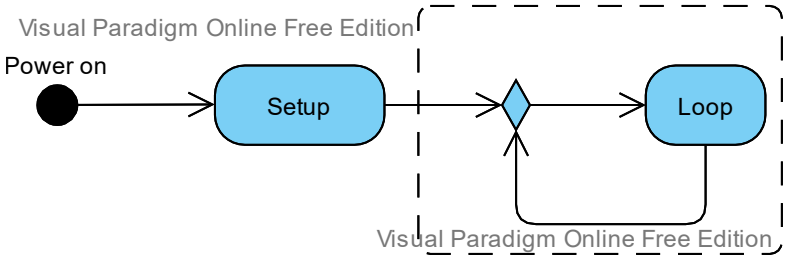


FIGURE 32: GENERALIZATION OF EMBEDDED PROGRAM FLOW

There were two methods to implement this program flow in an embedded solution – sequential and interrupt-based. One of these flows needed to be decided to define how the weather station would operate.

The sequential program flow is defined by all actions being performed sequentially in the loop. This would mean that the microcontroller would need to be always active (powered on) and wait to perform a given action specified in the routine only when its turn in the sequence has come up, consequently possibly missing on events if they have a short lifetime.

The second one, interrupt-based routine, allows for the timely reaction to change of status. The weather station needs to be a real-time reactive system. Therefore the interrupt approach was preferred and chosen for the implementation to ensure any event occurring, such as a restart or a connection initiated via the Bluetooth module from the handheld device, was recognized and reacted to accordingly.

4.3.2.2. *Communication Modules*

4.3.2.2.1. Bluetooth Module

According to the requirements, the Bluetooth connection should be a primary trigger for the change of the status of the weather station.

This would mean that once the handheld device has been connected to the Bluetooth module, the weather station should prioritise actions requested by that link. Depending on the type of command received, the microcontroller could enter its test mode or need to execute an action such as read and send data over the Bluetooth link, power GPS module on or off, or align solar panel.

While the command is processed and executed, other software-defined interrupts should not be able to disturb the routine so that the action required by the command can be completed, which makes the command processing a “blocking” function. Figure 33 shows the expected states of the Bluetooth module derived from these definitions.

The status of the Bluetooth module should change from “Disconnected” to “Connected” when the handheld has discovered the weather station Bluetooth module on the Bluetooth network via its name and has successfully “paired” to the weather station BT via entering the authentication passkey. The authentication is done automatically by the BT module firmware even in the “pass-through” mode, and the connectivity is realized by a continuous “high” level signal sent from the device to the microcontroller via the Bluetooth status pin.

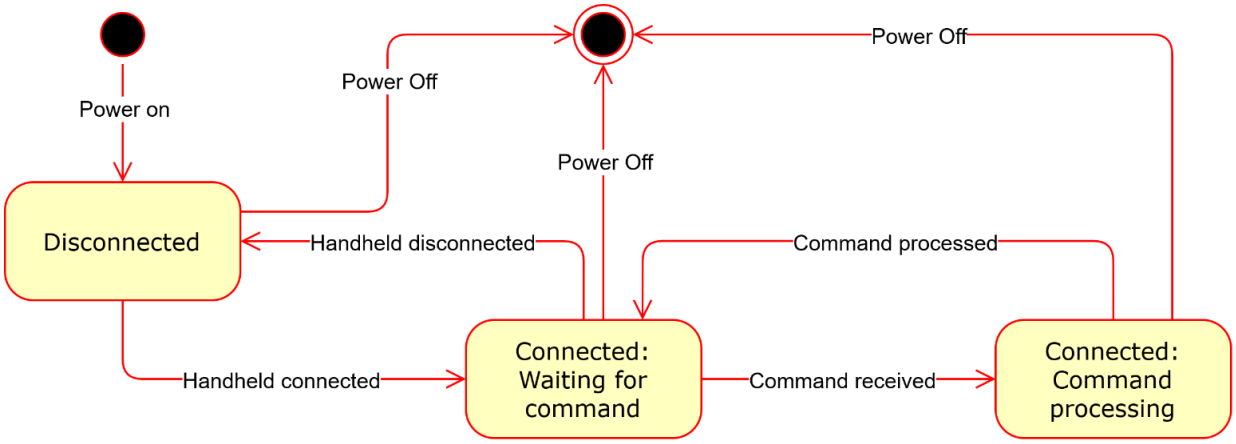


FIGURE 33: BLUETOOTH MODULE STATE MACHINE

4.3.2.2.2. Data Upload

The Wi-Fi connection was defined to be started either via a command received on the Bluetooth link and processed for requesting an in-time data upload or via a timed interrupt. The timed interrupt would trigger the data collecting routine and the Data upload sequentially, or there could be an interrupt with a different interval only triggering the Wi-Fi Data upload.

As the proof of concept required only proof that the communication channel works and the upload is successful, only the Bluetooth triggered Wi-Fi connection was decided to be part of this weather station implementation. Therefore, the two concerns defining the Wi-Fi link were the connectivity to the web server and the correct formatting of the uploaded data.

As defined in the AT command example for the ESP8266 [38], the upload was characterized by three main steps in sequential order:

1. connecting to the router Access Point (AP),
2. connecting to the TCP/IP defined server,
3. sending the data to the web server using the TCP/IP session.

Figure 34 shows the expected sequence diagram for a Data upload event.

The microcontroller receives a signal for upload, then initializes (powers on) the Wi-Fi module and checks if the AP associated with the web server is available. If it has been available, it connects to it via using the predefined AP connecting parameters. If the connection is successful, the microcontroller initiates a TCP/IP session to the web server and uploads the data.

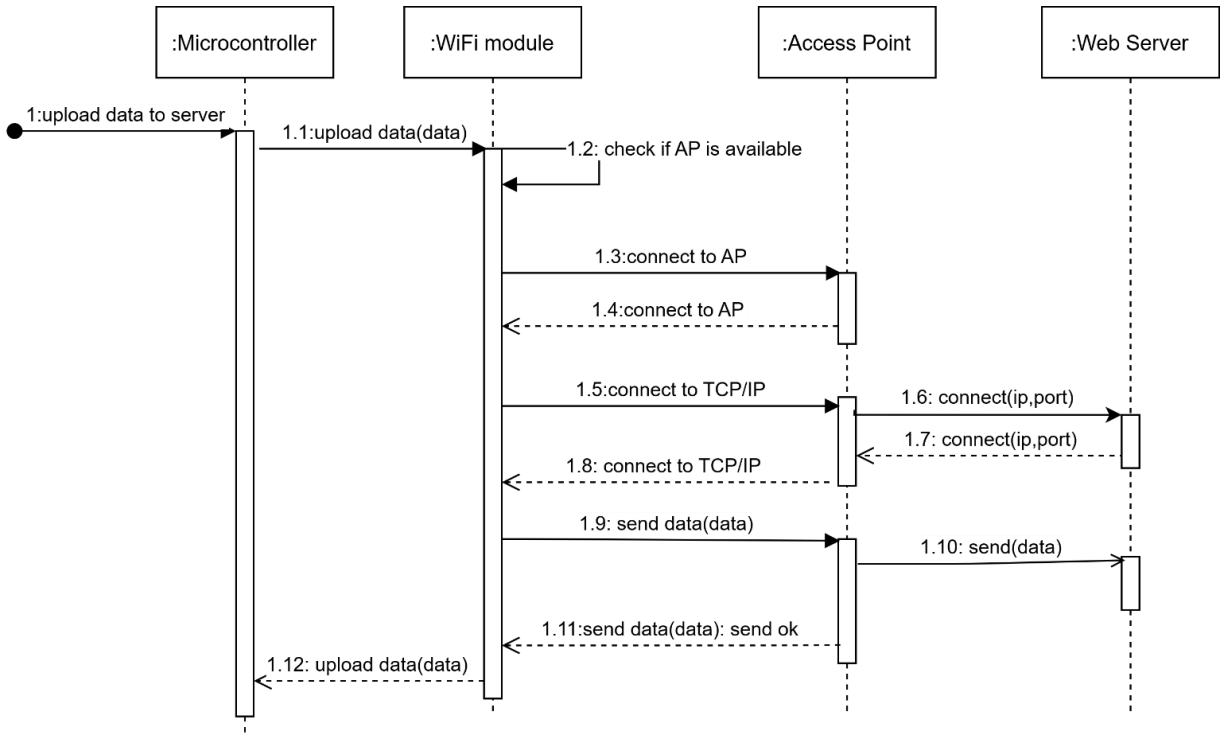


FIGURE 34: SEQUENCE DIAGRAM FOR UPLOADING DATA TO THE WEB SERVER

Taking into account the connectivity statuses, the Wi-Fi module could be defined to have four primary states (see Figure 35):

- Idle: when the module has been powered on, but no AP connection was made.
- AP connected: when an upload is requested, and the Wi-Fi module successfully connects to the access point of the web server associated router.
- TCP connected: when the module has established an AP connection and has successfully connected to the TCP socket. When the Wi-Fi module is in this state, the microcontroller can send the data to the web server.
- Waiting for response: for any type of AT command action required, the software should wait for a response from the ESP8266 driver received on the UART receiving line (busy waiting). If the AT command is correctly typed, recognized by the driver and the action is performed correctly from the Wemos D1 mini module, the response will be “OK”.

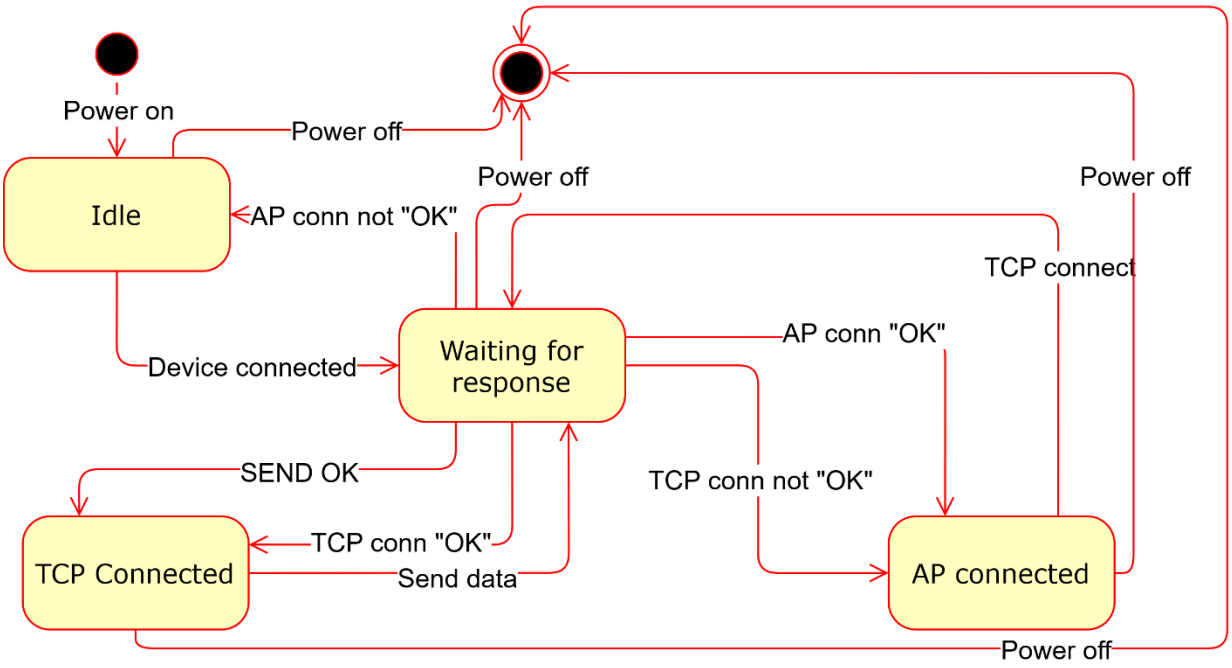


FIGURE 35: WI-FI MODULE STATE MACHINE

4.3.2.3. *Debug Console*

The debug console using the UART should output information about all the actions performed on the “modules”. For example, if the Wi-Fi module attempts to connect to the access point, the console should output a message indicating that.

The writing of the information on the serial output line should happen immediately to ensure that a developer can check the flow of the program in time. Therefore, the debugging console was decided to also implement an interrupt routine for printing on the serial line.

The interrupt routine would be triggered for character input or output on the console. When a character was typed in, the console should do nothing with it unless the prompt was given to the user. If the input was requested from the program, the typed character should be printed out on the console to provide feedback to the user that he has typed that character. At the same time, it should be saved in a buffer provided for the Debug Console input and then assigned to the function which requested console input.

4.4. Test Concept

This chapter describes the testing processes which would ensure that the core functionality listed in the requirements chapter has been sustained.

There were several scenarios that needed to be tested for the verification of the solution. As the goal of this project was proof of concept (CF1), only the standard use case scenarios were assumed. This would mean that the system was tested for working under the designed conditions and does not demonstrate that it is free of defects [39, p. 206]. The designed conditions were that the web server and access point from “Pro-Jan-17” are available and

discoverable from the weather station, and a serial Bluetooth terminal application with proven capabilities is used for connecting to the HC-05 module.

The test scenarios were given an identification string by “TS” (standing for “test scenario”) and a number following it (example “TS1”). They were grouped by “Hardware” and “Software” tests.

The “Software” tests were defined to test the base functionality of the Wi-Fi, Bluetooth and RTC modules such as sending or receiving commands. A complete list with the derived tests scenarios can be seen in Appendix G - Test Scenarios.

An automated testing environment can be a project of its own. Ergo the testing concept assumed only manual testing.

5. Implementation

This chapter concentrates on the implementation of the communication channels and the specifics of the microcontroller and modules used. This includes the hardware setup of the devices and software created to fulfil the requirement specifications.

5.1. Hardware

As mentioned in 4.2, the TM4C1294NCPDT microcontroller base of the LaunchPad evaluation board provides access to 80 general-purpose input/output pins via BoosterPack headers, which can be configured to work with alternative functions. The alternative functions give access to the microcontroller UART interfaces 2 to 7, SPI interfaces 0,1,2,3 and 5, and I^2C interfaces 0 to 2.

5.1.1. Pin Assignment

The HC-05 and the Wemos D1 modules both needed UART interfaces for their connection to the LaunchPad.

The Wemos D1 had two UART hardware interfaces - UART0 and UART1, where UART0 could be used for communication and UART1, namely the Tx, could be used for debugging purposes. The Wemos D1 was connected to the microcontroller for control of the AT command software via the UART0.

The decision which UART interfaces of the LaunchPad to use was made based on the proximity of the pins on the board in such a way that only one of the BoosterPack pin sets was used. Therefore, the Wi-Fi module was placed to use UART5, and the Bluetooth module was connected to the UART7 interface.

Additionally, the status pin of the Bluetooth required a connection to a GPIO pin, for which the same strategy applied. Hence, pin 5 of port M, also on the same set of pins with the two UARTs, was used for the Bluetooth Status pin.

Figure 36 shows the hardware connection between the two communication modules and the LaunchPad evaluation board.

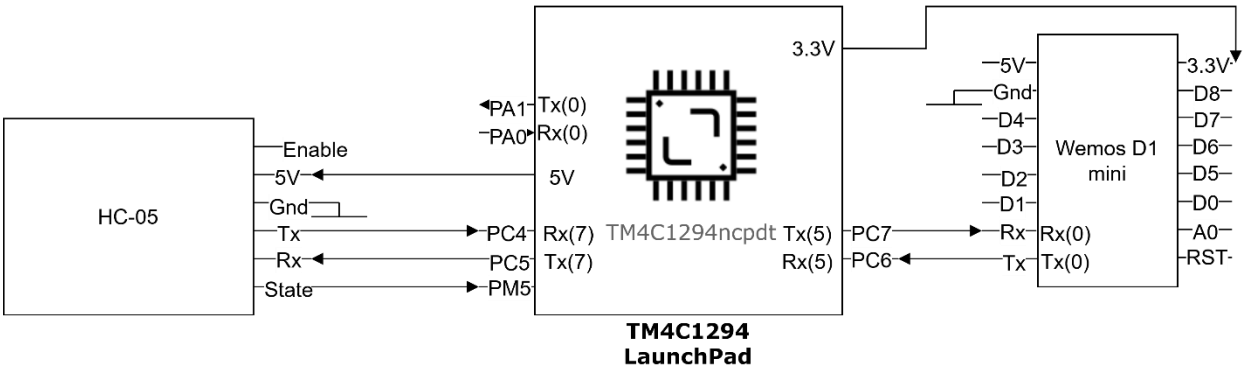


FIGURE 36: TIVA C LAUNCHPAD PIN CONNECTIONS TO HC-05 AND WEMOS D1 MINI MODULES

In addition to these modules, a Debug Console interface for printing out messages from the microcontroller using a UART interface was defined in the concept phase. The LaunchPad provides an internal conversion between USB and the UART0 interface, allowing a USB cable to transport the serial data in and out of a connected personal computer. Ergo, the UART0 interface was chosen for the Debug Console.

Table 12 shows the intended functionality of the modules and the pins with their alternative functions and settings for implementing the communication system of the weather station.

	Module pin	LaunchPad Peripheral	LaunchPad pin	LaunchPad Alternative function	Settings
Debug Console		UART0	Port A pin 1 – output	U0Tx	Baud rate: 112500 Frame format: 8N1
			Port A pin 0 – input	U0Rx	
Wi-Fi	Wemos D1 mini Rx	UART5	Port C pin 7 – output	U5Tx	Baud rate: 115200 Frame format: 8N1
	Wemos D1 mini Tx		Port C pin 6 – input	U5Rx	
Bluetooth	HC-05 D1 Rx	UART7	Port C pin 5 – output	U7Tx	Baud rate: 9600 Frame format: 8N1
	HC-05 D1 Tx		Port C pin 4 – input	U7Rx	
Bluetooth State	HC-05 State pin	GPIO	Port M pin 5 - input	none	Interrupt: On input high Priority: highest

TABLE 12: DEFINED HARDWARE FUNCTIONALITY OF THE LAUNCHPAD FOR THE BLUETOOTH AND WI-FI MODULES AND A DEBUG CONSOLE

5.1.2. Power Supply

5.1.2.1. *Wemos D1 Mini Breakout Board*

The Wemos D1 mini board needed a 3.3V supply. The LaunchPad provided a 3.3V power rail, so the Wi-Fi module was powered on by that connection for this iteration of the project.

However, the module could also be powered on by plugging in a 5V micro USB supply source. Suppose the USB output of the Wemos board is connected to a computer. In that case, the module is powered on by the computer, and the computer can serve as an output for the AT driver software via a serial console such as “PuTTY” or “Arduino IDE Serial Monitor”. The computer console would then need to be configured with the same settings for the Wi-Fi module setup on the LaunchPad. Thus, the computer console could output all the commands sent or received on the Wemos D1 mini board via the UART0 interface.

Provided that a 3.3V supply powers the Wemos D1 mini board and it is at the same time it is connected via the USB interface, this does not impact the performance. This feature was used at a later point in the testing phase for the module.

5.1.2.2. *HC-05 Breakout Board*

The HC-05 Bluetooth module needs a 5V supply. As the LaunchPad provides a 5V power rail, the module was powered on by that connection.

5.1.2.3. *LaunchPad Board*

The LaunchPad has several options for powering. A “power select” jumper (JP1) decides the power source - BoosterPack, target USB cable, or the on-board debug interface (ICDI) USB cable. While connected via its USB debug interface, the evaluation board can be at the same time powered by it.

Both USB connections allow for 5V to be connected to the board. However, if the BoosterPack is selected as a power source, the voltage should be 3.3V.

For the CF7 requirement, the weather station should be set up to use a separate battery as a power source. Considering that the Wemos D1 mini and the HC-05 modules can both use the LaunchPas for their power needs and the LaunchPad itself is powered by a battery via its target USB, then all of the modules could be powered altogether.

In the proof of concept, the Wemos D1 was powered on via the 3.3V power rail on the LaunchPad. The HC-05 was powered on via the 5V LaunchPad power rail, and the LaunchPad itself was powered on via the default debug 5V USB connector.

5.2. Software

This subchapter describes the software implementation for the communication channels and the weather station functionality determined in the Concept chapter. It includes three main points of the development:

1. initialization and configuration of the microcontroller and its peripherals (GPIOs, UARTs, Hibernation module)
2. command processing software developed for the separate Wi-Fi and Bluetooth communication channels.
3. command sets available for use for the two wireless channels of the weather station.

As this was the beginning of the new weather station, a new project was created in Code Composer Studio (CCS), the recommended by Texas Instruments IDE, and the C programming language was used, as this is considered the standard language for microcontroller programming with CCS. Additionally, the TivaWare SDK was used for the rapid development of the software.

5.2.1. MCU Initialization and Configuration

As mentioned in 4.3.2, the microcontroller program could be separated into two major parts: setup and loop. The setup consists of the initialization and configuration of the microcontroller and its peripherals. The loop typically concentrates on the functionality intended for the solution.

For the setup phase, the existing TivaWare drivers were used to initialize the system and its components.

5.2.1.1. *System Clock*

First and foremost, the clock system for the microcontroller had to be set up. The tm4c1294ncpdt microcontroller allows a system clock of up to 120 MHz. However, as there was no requirement about the expected clock of the system, the system clock was directly selected from the 16 MHz main external oscillator of the microcontroller.

5.2.1.2. *Peripherals setup*

After that the setup of the system clock was done, there was no particular sequence required for the setup of the peripherals, so it was arbitrarily chosen as:

1. Setup of UART0 and Debug Console instance
2. Setup of UART7 and Bluetooth instance
3. Setup of UART5 and Wi-Fi instance
4. Setup of Hibernation module and RTC
5. Setup of GPIO PM5 as input for Bluetooth Connection Status pin and setup an interrupt for it

5.2.1.3. *UART*

The two UART instances for the communication modules were included in the definition of the appropriate module (UART5 in a file associated with the Wi-Fi and UART7 associated with the Bluetooth) to provide a certain level of abstraction for the command software. Similarly, the GPIO configuration for the Bluetooth status pin was also part of the Bluetooth definition.

Due to using a utility provided by the TivaWare software for UART0 as a Debug Console, there was no need for further setup of reading/writing functions, software buffers, or interrupts as they came with the utility.

A general UART wrapper was created to send and receive on the serial Rx and Tx lines. It received references from the Wi-Fi and Bluetooth modules for the UART peripheral it should print to/read from. The following functionality was supported in this wrapper:

- Waiting for a line: The <CR><LF> characters sequentially received define the end of a line.
- Waiting for a specific string line: A comparison is made between the expected string and the current line.
- Waiting until all data sent is received: The program receives lines and stores them in the indicated receive buffer until the sequence “OK”<CR><LF> is received.
- Sending a command: The command is sent together with the <CR><LF> character sequence to indicate the end of transmission.
- Sending data: The command sent is intended for the data upload to the server. The prefix “A:” and the postfix “<LF>!” are added to the data. The composed data string is thus transmitted on the serial line.
- buffers for used for storing received lines or lines to transmit
- buffer cleaning

The <CR><LF> character sequence at the end of commands is part of the typical command format for data sent over to servers or console outputs, which is why this termination sequence is commonly used in AT command software.

5.2.1.4. *Hibernation Module*

A wrapper was created for the Hibernation module providing two formatting options for the RTC data:

- “YY.MM.DD hh:mm:ss” - for data requests from the Bluetooth or data upload via the Wi-Fi link
- "MMM DD, YYYY hh : hh : ss AM/PM" - for displayed data on the Debug Console

The RTC requires the time and date first to be set and would otherwise return “INVALID DATE AND/OR TIME” upon a time query. The wrapper allows the time and date of the RTC to be set via the Debug Console or via a “set” command from the Bluetooth module.

Setting from the Debus console is made after the prompt requests input via the “>” sign followed by the following lines, each of which requires entry:

- “Enter hour. Range: {0, ..., 23}:”
- "Enter minutes. Range: {0, ..., 59}:"
- "Enter date in format YY.MM.DD:"

The Hibernation module wrapper also provides functions for entering “hibernation mode”, which is one of the microcontroller’s low-power modes.

As part of the investigation on the Hibernation module, it was found that the pin for connecting separate battery source just for this module on the tm4c1294ncpdt microcontroller (v_bat) was not taken out on the LaunchPad breakout board. This consequently meant that the board needed to be powered on at all times to keep the saved data when entering hibernation mode. Suppose a method for accessing the v_bat pin on the microcontroller is found. In that case, the microcontroller could also execute hibernation mode where all power supplies are turned off to the rest of the microcontroller and only the Hibernation module’s circuit is active [44, p. 245].

An external event on a switch or a predefined GPIO or an RTC match can wake the microcontroller back and get all of the other microcontroller circuits powered on. The processor then starts the typical sequence of running code.

An interrupt for waking up the peripheral 10 seconds after entering hibernation mode was configured and enabled. *Figure 37* shows a structogram with the setup for the Hibernation module interrupt. *Figure 38* shows the program flow of the associated interrupt handler.

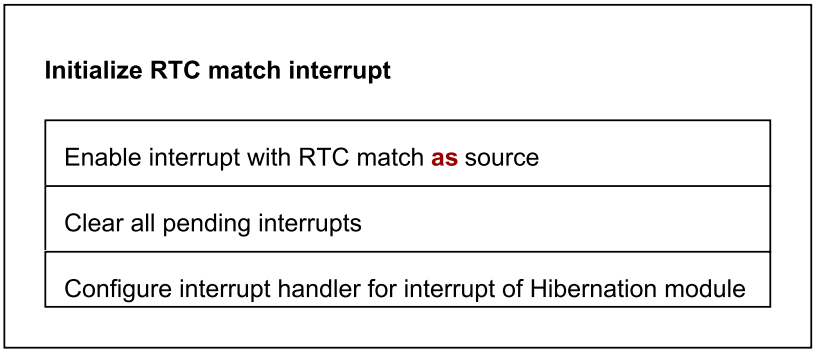


FIGURE 37: STRUCTOGRAM OF HIBERNATION MODULE INTERRUPT SETUP

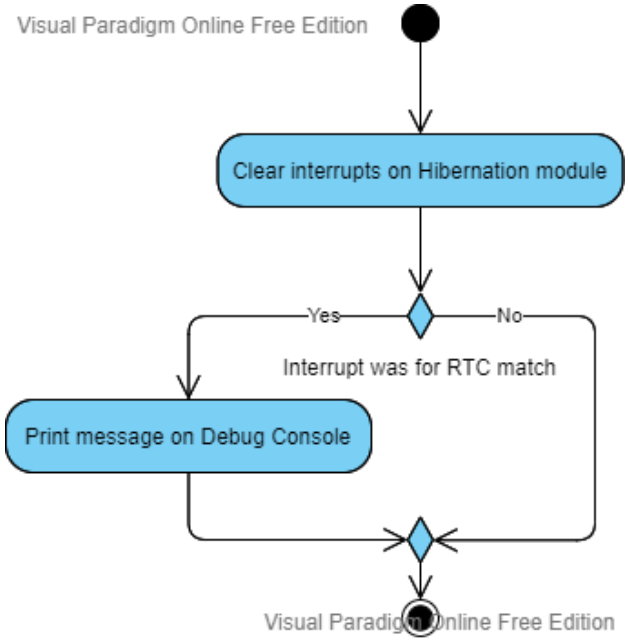


FIGURE 38: ACTIVITY DIAGRAM OF HIBERNATION MODULE INTERRUPT HANDLER

The wrapper was created based on an example for the Hibernation module peripheral provided by TivaWare SDK.

5.2.1.5. *UART Debug Console*

For the user Debug Console setup, a utility UART driver wrapper was already provided by TI. The utility is part of the TivaWare SDK and allows for buffered or unbuffered console output/input. As part of the utility, an interrupt routine for the UART0 was included reacting to input from the console or output from the application. The printing function was modified to ensure that data is pushed from the software buffer to the UART0 Tx buffer.

5.2.1.6. *GPIO*

As part of the proof of concept, an interrupt for changing the station into demonstration mode was set up based on the status of the Bluetooth Status pin. The GPIO pin 5 of Port 5 on the

LaunchPad was configured as an input pin connected to the Bluetooth Status pin with an interrupt for an achieved “high” level on the line.

The interrupt and its associated handler were added to the application’s startup code (tm4c1294ncpdt_startup_css.c) as instructed by the TivaWare Peripheral Driver Library User’s Guide [40, p. 353] to ensure the fastest interrupt response time. *Figure 39* shows the program flow for the Port M handler.

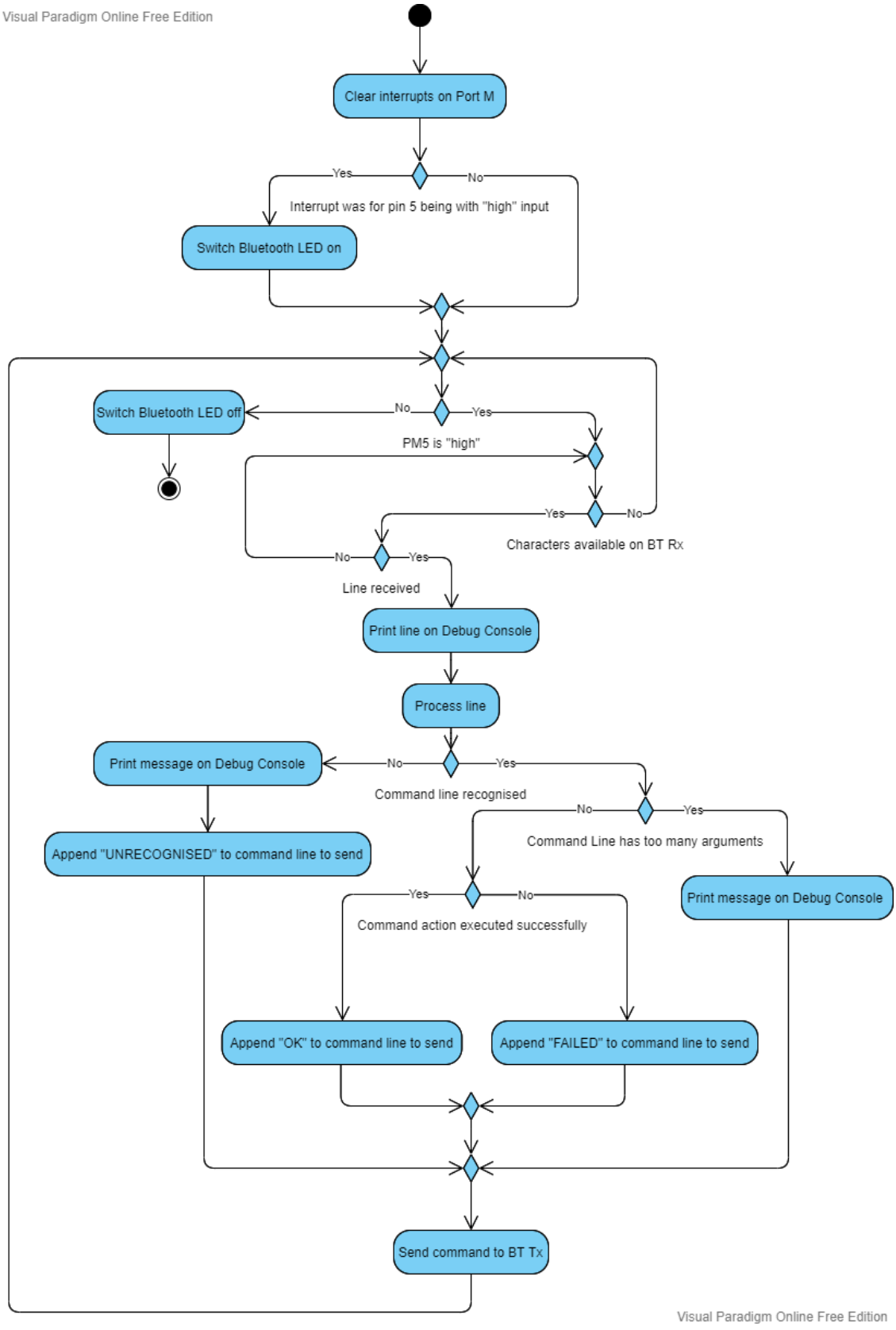


FIGURE 39: ACTIVITY DIAGRAM FOR PORT M INTERRUPT HANDLER

5.2.2. Command Software

The command software used for communication between the weather station and the web server or handheld controlling device is described in this section. The interfaces for the Bluetooth and the Wi-Fi modules differ in the command model. This is because the communication with the Wi-Fi is initiated by the weather station, but an external Bluetooth handheld device initiates the communication with the Bluetooth.

There were two versions of the command software implementation. The first version consisted of hardcoded command functions where a function needed to be implicitly called to execute a command. The second was developed to shorten the amount of needed programming space and relied on a single processing function and a grouping of the commands by type. This model was inspired by the examples presented from TivaWare for the Hibernation module.

5.2.2.1. *Processing*

5.2.2.1.1. Bluetooth

The processing of the Bluetooth commands was done according to their AT type. There were four groups of commands defined: “test”, “execution”, “set”, and “get”, and for each type, a two-dimensional array with command definitions was created.

Upon a line received, the line is separated into arguments based on the number of special characters, namely “:” or “+”. Up to three arguments are accepted for the command. The number of arguments partially determine the type of the command.

A “test” command has only one argument, meaning there are no additional special characters in the command. It starts with the “AT” prefix and is accompanied by a letter or a combination of a letter and a numeral, similarly to the original Hayes command set.

A “get” command consists of two arguments, where the first is the “AT” prefix and the second is the command name. A special character “?” at the end of the command name distinguishes the command as “get” type.

In a similar manner, an “execution” command consists of two arguments but does not have a “?” at the end of the command. In the current list of “execution” commands, the commands have a “!” attached to the command name. The algorithm, however, will also accept other special characters or no special characters after the command name, as is generally defined in the extended Hayes command style. This change was made so that if a different HC-05 module or another AT-command-supporting Bluetooth device is connected, the developers can change the list of supported commands without a need to retouch the processing of the commands.

A “set” command consists of three arguments. The first one is the “AT” prefix, followed by the connective “+” sign. The second is the name of the command followed by a “=” sign. The third contains the command parameter arguments. The “=” is by standard the character used for separation between the command and the parameters in the extended Hayes command style.

If a command has not been found among the tables provided, the algorithm would make it so that “UNRECOGNISED” is sent back on the Bluetooth line. If the execution of the associated command has been successful, an “OK” line would be sent back, and if it were not successful, a “FAILED” line would be sent back.

The activity diagram in Appendix D - Activity Diagram of Bluetooth Command Line Processing provides further information on the command processing algorithm.

5.2.2.1.2. Wi-Fi

As already mentioned, the Wi-Fi command processing model was different from that of the Bluetooth module due to the initiation of the communication coming from the weather station.

The Wi-Fi processing function is called when a command needs to be sent to the Wi-Fi module. A single parameter is passed to the function, indicating the command that would be sent. A check upon an entry of the existing single command table is done for matching the command. If the entry matches the command, a new string for sending with the command prefix “AT” and accompanying name connected with a “+” sign is created. After that, depending on the command and the command type, the following checks are made:

- If the command would be for sending data, an attempt for connecting to the TCP/IP server is made. If the connection is successful, the data to be sent would be appended to the started command build and send over to the server. After the send, the weather station is to wait for a “SEND OK” string for successful execution status, and upon receiving it close the TCP/IP connection.
- If the command is to restart the module, the command is sent, and the weather station should wait for a “ready” string for successful execution status.
- For all other commands, the type of the command matters:
    - Execute: The command is sent, and the weather station waits for an “OK” string to be received for successful execution status.
    - Set: The needed parameters are appended to the command via an additional function call. After that, the command is sent. The weather station waits for an “OK” string to be received for successful execution status.
    - Get: The command is sent. The weather station reads received lines and parses them to internal parameters until an “OK” string line is received. This indicates a successful execution status.
    - Set_get: The needed parameters are appended to the command via an additional function call. After that, the command is sent. The weather station reads received lines and parses them to internal parameters until an “OK” string line is received for successful execution status.

If the command is not that of the current command entry, the algorithm continues until it reaches the end of the command set table.

The command is always found within the table as the commands available for use are all in the table.

If the execution was successful, the flow terminates by returning “OK” status. If the execution was not successful, this is also reported back by a different return status, depending on the failure reason.

5.2.2.2. *Structures*

5.2.2.2.1. Bluetooth

Four separate 2D array tables for the four Bluetooth command types were created to assist in the command processing.

Each of the command entry types has:

- a pointer to a string holding the command name (e.g. “CWTIME?”),
- a function pointer for the action needed to be executed,
- and a pointer to a string with a brief description for the command.

The “get” command type has an additional pointer holding the expected return command name.

5.2.2.2.2. Wi-Fi

More diverse structures were associated with the Wi-Fi module compared to the Bluetooth module. This was due to the vast number of parameters that can be queried or set on the Wi-Fi module because of the installed AT firmware.

A single table was defined for holding all of the commands. An entry in the command table consists of seven parameters:

- a pointer to a string holding the name of the command together with the “AT” prefix (e.g. “AT+CIPSTART=”),
- an enum stating the command (e.g. *connectTCP_IP*),
- an enum with type generalisation of the command type (e.g. *set*),
- a function pointer to a function which appends the needed by the command parameter values and is used for “set” and “set_get” command types,
- a number indicating the expected number of lines before an “OK” is received. The value is set to “0” for “set” and “execute” command types,
- and a pointer to a function that sets weather station parameters from received input from the module defined only for the “get” and “set_get” command types.

An enum structure was created as part of the Wi-Fi module definition for switching the states of the module as needed (see *Figure 40*). This parameter was based on the state returned from the AT software for the Wi-Fi module. It also fulfilled partially the definition of Wi-Fi states mentioned in 4.3.2. The status parameter of the module is assigned when a query to the module for checking module status is made. Additionally, if a Bluetooth command requesting the module status is made, the value of this parameter is returned a parsed to a string value to be sent back to the Bluetooth connected device.

```
typedef enum WlanStatus{
    inactive,
    idle,
    IPConnected,
    transmitting,
    IPDisconnected,
    APDisconnected
}tWlanStatus;
```

FIGURE 40: ENUM HOLDING THE POSSIBLE WLAN STATES FOR THE WI-FI MODULE

Other structures which were defined were for keeping the current operating mode of the Wi-Fi (as “Station”, “Access Point”, or both) and for keeping the enum types needed for the table.

5.2.2.3. *AT Command Sets*

5.2.2.3.1. Bluetooth

The Bluetooth commands defined in 4.3.1.3 were implemented with the addition to several other commands:

- uploading data via the Wi-Fi link,
- checking the status of the Wi-Fi connectivity,
- checking or setting the tracking of the solar panel position,
- reading and writing data from/to the weather station SD card,
- putting the weather station into hibernation mode.

A list with the implemented commands can be checked in Appendix E - Bluetooth AT Commands Implemented.

All “set” commands, except for date and time set, were configured to return that the action was not executed. This is due to the fact that the current implementation does not include the needed modules whose data is to be set.

In a similar manner, “get” commands return hardcoded data, except for data acquired from the RTC module. Although a hibernation function was defined, due to the complications described in 5.2.1.4, the execution of the command does not happen.

Similarly, the “Bluetooth Echo” was partially configured - a handler was written, but no actual interrupt was configured for UART7. Therefore, the function currently returns that enable/disable occurs, but as no interrupt has been configured, the echo parameter is never evaluated. This implementation was done due to concerns for constantly entering the interrupt for the UART7 and printing data back when the “BT Echo on” was selected. Additionally, this could have a user on the BT line confused as data sent to the weather station would be immediately returned. Consequently, if the functionality is to be used, an additional interrupt for UART7 on Receiving should be configured in the module setup phase.

For the “CGUI” command, semicolon-separated hardcoded values are placed and returned. The data used is the same as the one for the Wi-Fi data upload link, where 21 values are sent, and six of them are the changing RTC values.

The parameter names used in the command definitions are listed in *Table 20: List of Bluetooth Command Parameters and Their* Meanings as part of Appendix C - Bluetooth Commands to be Implemented and Command Parameter Definitions.

A set of tests proving that the commands defined work as intended was added to the test scenarios in Appendix G - Test Scenarios.

5.2.2.3.2. Wi-Fi

The Wi-Fi commands supported by the weather station are listed in Appendix F - Wi-Fi AT Commands Implemented. The list includes the connection commands defined in 4.3.1.2 and

extends with several others from the ESP8266 AT command software [31], which could be helpful in the future implementations of the weather station.

For a command to be used, a call to the Wi-Fi processing function needs to be made. The processing function requires the command defining enum parameter to process the command.

In the current implementation of the Wi-Fi commands, all of the parameters used are predefined and hardcoded, except the length of the upload data. The length of the upload data is determined in time based on the current data string, whose length could vary due to the RTC values included. For a complete list of the parameters used, see Table 26 in Appendix F - Wi-Fi AT Commands Implemented.

Most “get” functions were left blank, meaning they do not assign any Wi-Fi module data due to a lack of module parameters for setting. Due to the supported commands, a rework of the Wi-Fi module is encouraged to incorporate more of the ESP8266 parameters.

5.3. Additions

During the implementation process, two features outside the derived concept were developed. The first was a user-defined LED on the LaunchPad as a visual help for determining when the Bluetooth has been connected. The second was an implementation of the I^2C0 interface on the LaunchPad with the microcontroller as the initiator of the communication. In addition, a BME280 module retrieved from “Solar-Jan-20” was connected to the I^2C0 interface to test the connection and possibly acquire additional data for uploading.

5.3.1. Bluetooth LED

An LED associated with Port N pin 1 of the LaunchPad was defined to output the status received on PM5, thus visually informing if the Bluetooth has been connected to. The setup of the MCU for this addition is listed in Table 13.

	MCU Peripheral	MCU pin	MCU Alternative function	Settings
Bluetooth State LED	GPIO	Port N pin 1- output	none	Output the same signal as Port M pin 5 input

TABLE 13: HARDWARE DEFINITION OF THE BLUETOOTH STATUS LED

5.3.2. I^2C

An I^2C interface was configured to accommodate sensors that would be included in later versions of the weather station. A BME280 sensor retrieved from “Solar-Jan-20” was used to test the interface. The BME280 is a sensor from Bosch Sensortec that provides temperature, humidity and barometric pressure readings [41]. Typically the sensor provides SPI and I2C interfaces. However, the development board module which was used only extended the I2C interface (see Figure 41).

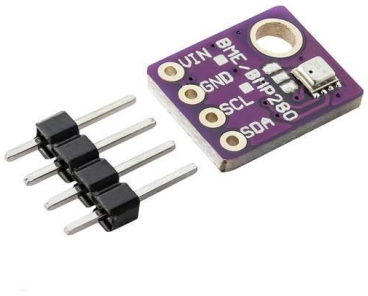


FIGURE 41: PINOUT OF THE BME280 MODULE BOARD [42]

5.3.2.1. *Hardware*

The connection from the BME280 sensor to the microcontroller was made so that the same LaunchPad BoosterPack previously used for the WI-Fi and Bluetooth module connections was utilized. The I^2C number 0 with GPIO pins PB3 for the Serial Data line (SDA) and PB2 for the Serial Clock line (SCL) was enabled for communication. The connections to the microcontroller for the inclusion of this module are listed in Table 14.

The BME280 sensor board needed a 3.3V supply and was powered up by the 3.3V power line of the LaunchPad.

	Module pin	MCU Peripheral	MCU pin	MCU Alternative function	Settings
BME280	SCL	I^2C0	Port B pin 2-output	SCL	Microcontroller as leader of the connection
	SDA		Port B pin 3-Output/input	SDA	

TABLE 14: HARDWARE DEFINITION FOR USING I^2C0 FOR CONNECTING A BME280 SENSOR

5.3.2.2. *Software*

5.3.2.2.1. I^2C wrapper

A wrapper for using the I^2C0 serial interface with the microcontroller as the leader of the communication channel was defined. The microcontroller feeds the clock to the SCL line, thus controlling the devices connected on the I^2C0 bus.

Unlike UART and SPI, I^2C uses one bi-directional line for communication and requires the addresses of the follower devices to initialize a connection with any of them. In addition, the leader needs to know the registers of the connected followers to read or write data from/to them. The wrapper included functions for sending and receiving data via the I^2C0 as a line leader was based on the TivaWare I2C driver and the driver provided for I^2C connectivity in “Solar-Jan-20”.

5.3.2.2.2. BME280 driver

Three BME280 libraries were evaluated for integrating the BME280 module to the current weather station. The first was provided by the producer Bosch Sensortec and provided options

for connecting the sensor via SPI or I^2C. The other two were a Github library for BMP280 [43] and a customized “Solar-Jan-20” driver, both of which extended the Bosch driver. A comparison table for the available drivers is listed in Table 15. The driver available from “Solar-Jan-20” provided temperature, humidity, and pressure readings, whereas the other did not provide a humidity reading. The “Solar-Jan-20” was chosen because it was considered to add more value if the driver and module continued to be used for the weather station project.

	Features
Bosch Sensortec	SPI and I^2C available, double-precision floating-point and integer versions, 32- and 64-bit implementations for pressure, results can be provided in decimal values for C°, Pa, and % relative humidity
Solar-Jan-20	Based on the Bosch driver, compensation for ambient temperature error, data is in held in bme280 structure
Github source	Based on the Bosch driver, available for SPI and I^2C on Tiva C boards, returns decimal values for C° and Pa, no humidity value available

TABLE 15: LIST OF DRIVERS EVALUATED FOR THE INTEGRATION OF A BME280 SENSOR

5.3.2.3. *Tests*

A set of tests was added to the test scenarios in Appendix G - Test Scenarios to accommodate the additional features.

6. Results and Evaluation

This chapter looks at the functionality delivered in the implementation phase after the testing with the defined test scenarios. The results are compared and evaluated with the defined requirements in 3.2.

6.1. Test Results

The tests from Appendix G - Test Scenarios were performed to confirm the functionality stated in the Implementation chapter.

6.1.1. Hardware

The hardware connections were proven via the functionality of the software for the UART5, UART7 and pin 5 of Port M of the LaunchPad board. The UART interfaces have been able to correctly deliver streams of data to and from the connected devices. The PM5 pin has been successfully detecting input from the Bluetooth status pin. The PN1 user-defined LED was lighting up according to the connection status received on PM5.

6.1.2. Debug Console

The Debug Console on UART0 was configured successfully to deliver messages from the microcontroller program to a personal computer connected via USB to the LaunchPad. The program used for displaying the serial output from the Debug Console was “PuTTY”. At the beginning of the microcontroller program, the Console will be cleared out and start at the leftmost upper corner of the serial application.

6.1.3. Wi-Fi module and AT Software

For the Wi-Fi module, the statuses of each command were set to be outputted on the already established Debug Console via a descriptive message with the format “WLAN:<Command Description>……<status>”.

A function including all the command definitions for connecting to the web server was written and set to execute at the setup phase of the microcontroller program. The function was defined in such a way that unless all steps are completed, the program cannot continue. The conditions for the successful execution were:

- the router with the access point “Wetterserver” and password “wetterdeluxe” to be in the range of the Wi-Fi module
- the web server with IP “192.168.178.82” and port “100” defined for data input is connected to the router.

The sequence tested was as described in Figure 34 in chapter 4.3.2.2.

Figure 42 and Figure 43 display the result of the test performed, proving that the data upload was successful.

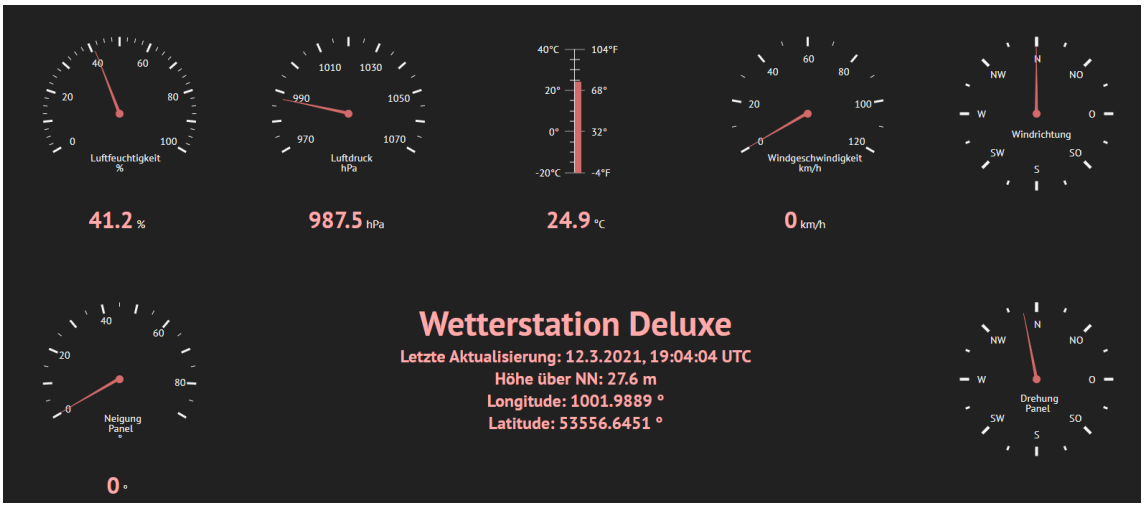


FIGURE 42: WEB PAGE GUI OF "PRO-JAN-17" WITH OLD DATA

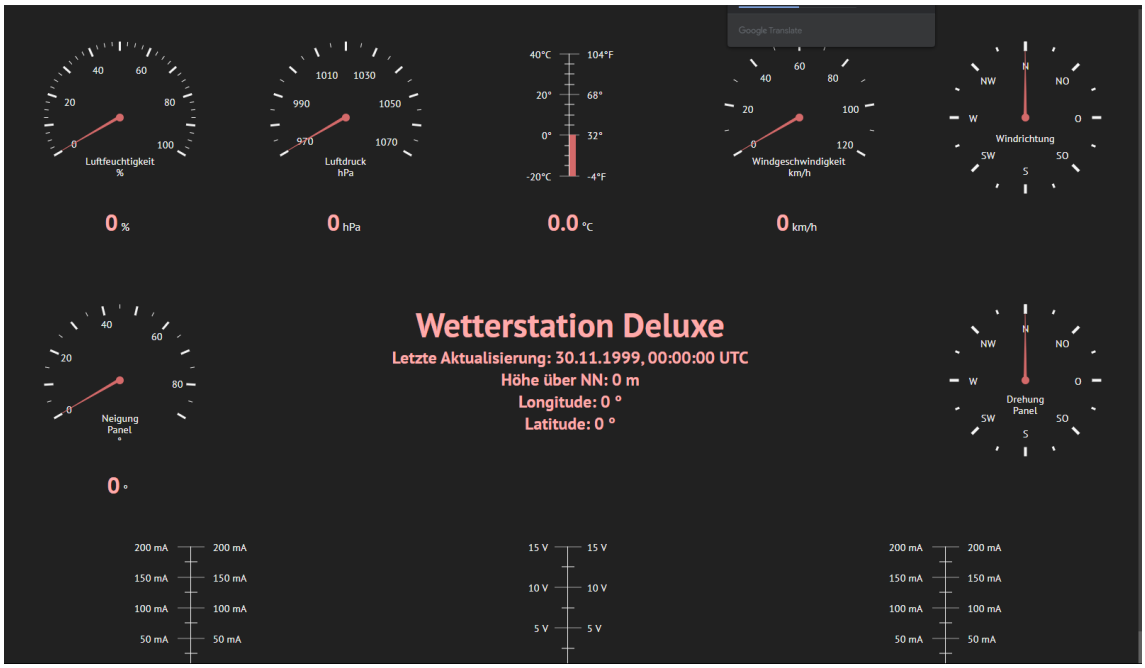


FIGURE 43: WEB PAGE GUI OF "PRO-JAN-17" WITH NEW DATA CONTAINING ONLY ZEROES

Additionally, a comparison between the PuTTY output and another serial console (Arduino IDE Serial Monitor) as indicated by TS5, TS6, TS7, TS8, TS9, TS10, TS11, TS12 was performed to confirm that the execution was successful. The PuTTY console is connected to the Debug Console, and the Arduino IDE Serial Monitor is connected to the serial USB output of the Wemos D1 mini module. This also confirms that the Debug Console operates as expected according to TS4 (see Figure 44).

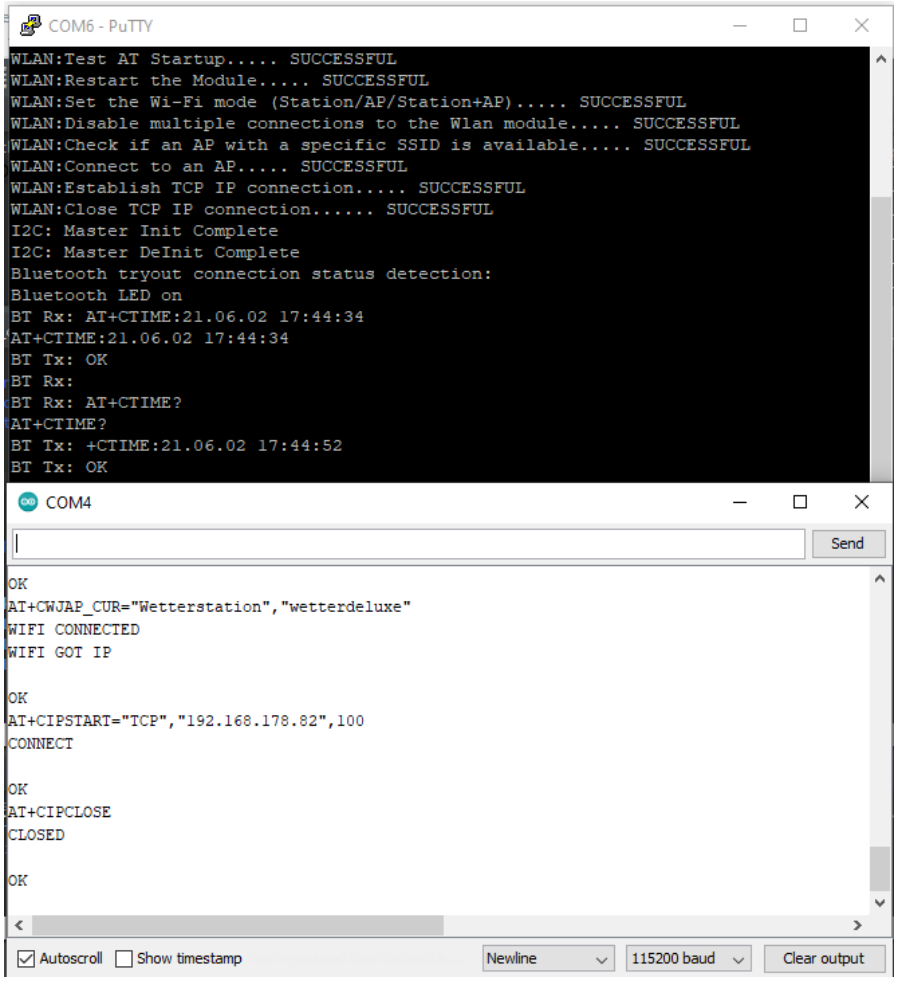


FIGURE 44: PUTTY AND ARDUINO IDE SERIAL MONITOR OUTPUTS

6.1.4. Bluetooth module and AT Software

The tests for the Bluetooth were performed using an application called “Serial Bluetooth Terminal” from the Google Play Store for Android [44]. The application was tested with two devices using it, successfully recognized the paired devices and connected them, sustained the connection, and sent and received data.

Figure 45 shows the main interface of the chosen application together with some of the performed command tests. All of the Bluetooth command tests performed as expected. The set commands always received a “FAILED” response, the “get” commands returned dummy hardcoded data, the RTC set and get functions were able to successfully update the real-time clock and acquire data from it. The Wi-Fi status was returned correctly, and data was successfully uploaded to the server using the Bluetooth command “AT+CWIFISEND!” as can be verified in Figure 46.

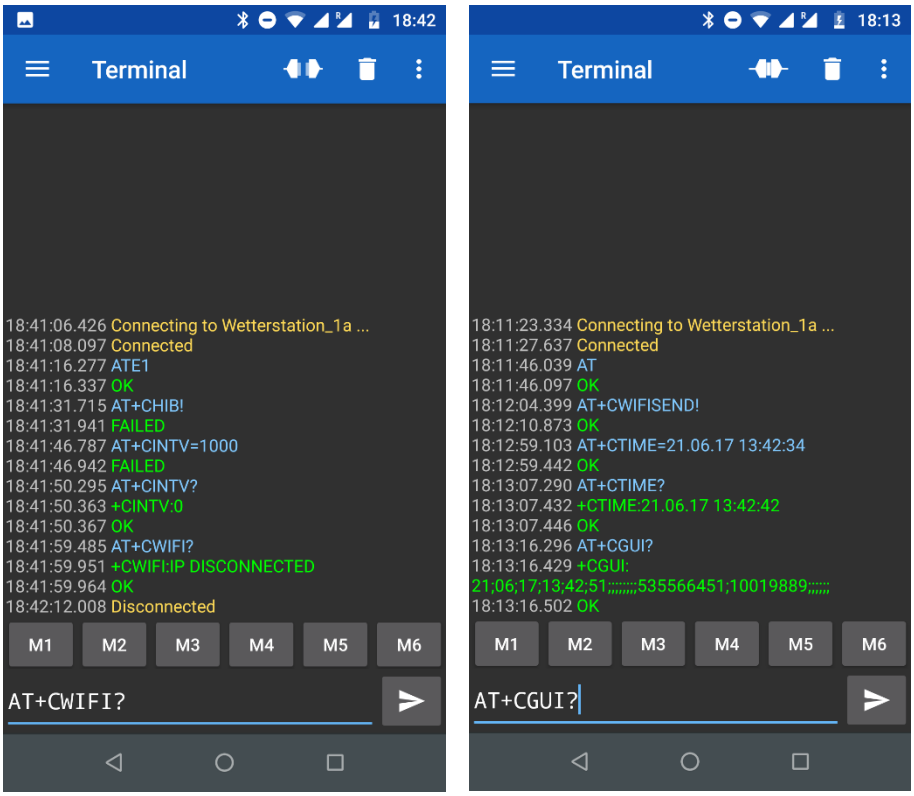


FIGURE 45: BLUETOOTH APPLICATION WITH COMMAND TESTS PERFORMED

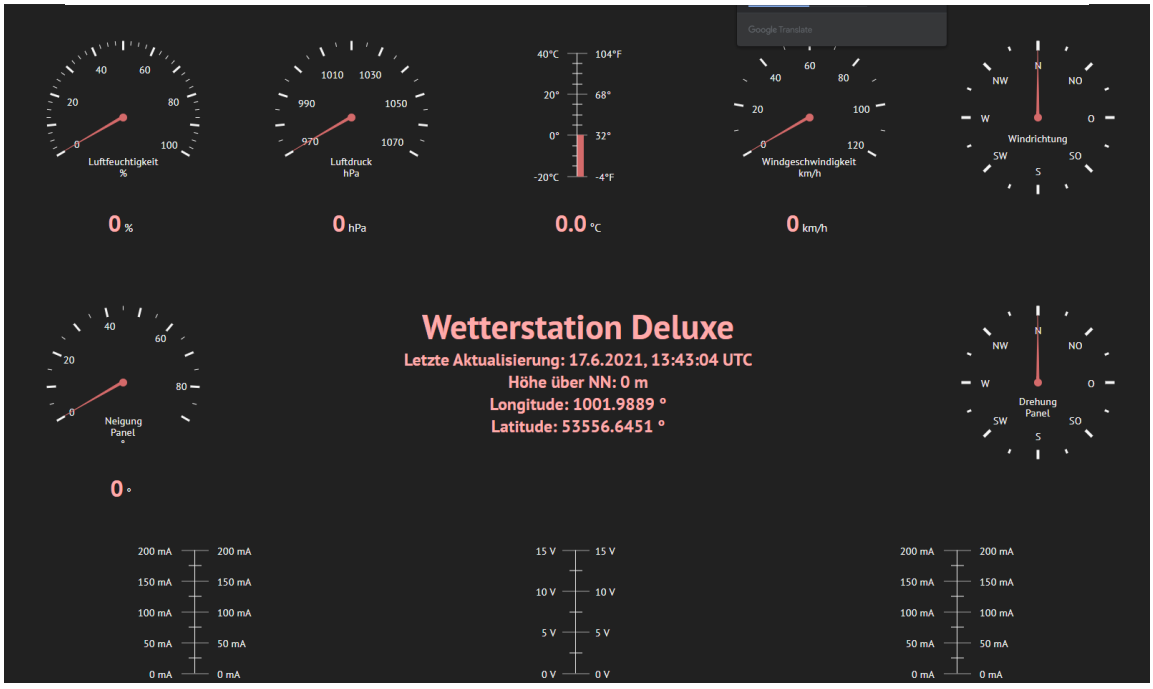


FIGURE 46: WEB PAGE GUI OF "PRO-JAN-17" AFTER RTC SETUP WITH TIME AND DATA FROM BLUETOOTH AND DATA UPLOAD VIA THE WI-FI LINK

6.1.5. I²C and BME280

The test for the BME280 and the associated I²C0 interface was to send and receive a packet successfully to and from the module. This was done via the driver functions for the initialization and update of the BME280. The initialization first performs a reset on the module and then reads data to compare the assigned module ID to the ID stored in the device memory. At most times, the initialization was completed, and the module ID was verified. However, due to the hard reset of the module, sporadically, the program would not be able to retrieve data from the BME280 immediately after reset. An implementation of a function waiting for the restart to be completed for the module should be implemented to ensure that it is ready before new access from the microcontroller is initiated. Because of this and probably other issues with writing/reading to/from the module, no calibrated data could be acquired for temperature, humidity and barometric pressure.

6.2. Evaluation

All of the tests performed, except for the BME280 and associated I²C0 interface, were successfully completed. The tests proved the fulfilment of the following requirements for the communication system: CF1, CF2, CF3, CNF1, CF4, CF5, CF9, CF11, CF12 , F5.

Additionally, in the concept design, the following were considered:

- the modules chosen are available as development boards with standardly spaced pins to allow for connection to a breadboard or additional connectors (CF8)
- the chosen Wi-Fi and Bluetooth modules can be directly powered by the weather station base controller via 3.3V and 5V accordingly. The LaunchPad board can be powered via a 5V USB source, thus powering the attached modules (CF8)
- the microcontroller base TM4C1294ncpdt allows three low-powered modes and a hibernation mode (partially F11) and provides sufficient other peripherals and features for the development of the fully equipped weather station
- a handheld device for connecting to the weather station via Bluetooth was chosen according to the specifications given by the stakeholders (CF10)
- the weather station can receive commands from the Bluetooth, recognize them and categorise them successfully and answer them accordingly
- the commands defined for the Bluetooth and the Wi-Fi communication were implemented using tables with definitions for the commands and their execution. Adding new commands would be easily feasible via adding a new command entry to the respective command table. No changes are needed on the processing of the commands (CF6)

In conclusion, all requirements set for developing the weather station’s communication system were met.

7. Conclusion

7.1. Summary

The work presented in this paper aimed to create a framework for the development of an automated solar-powered weather station and the implementation of its Wi-Fi and Bluetooth wireless communication channels. It was designed as a proof of concept where the Wi-Fi link is used for uploading data to a remote server, and the Bluetooth link is used for demonstration of the weather station features and maintenance.

Specifications for the weather station were derived together with ones for the communication subsystem based on stakeholder requirements and analysis of other weather stations developed by students or commercially available.

A development framework was created based on the EK-TM4C1294XL LaunchPad board as the core for the weather station. The software was built with Code Composer Studio and the C-written TivaWare SDK provided by Texas Instruments.

A handheld device for demonstration and maintenance of the weather station connecting to the Bluetooth channel was chosen for the later iterations of the weather station project. It has a rugged appearance and is Android-based to allow uncomplicated development of the weather station controlling application.

The specified proof of concept was delivered with both the Wi-Fi and Bluetooth links as hardware and software components. The software included an extendable set of AT style commands for both communication channels. The Wi-Fi medium was able to successfully connect to a specified access point and server and upload data to the server. The Bluetooth medium was able to recognize commands sent by an external Bluetooth connected device and execute the functionality associated with the given command on the weather station. Because of the nature of the Wi-Fi connection, the communication via this channel was set to be initiated by the weather station. On the other hand, the communication via Bluetooth was set to be initiated by the Bluetooth connected handheld device. Once a connection has been established, a user can query commands to the weather station.

Multiple tests were performed to ensure that the commands are received, processed and sent correctly for both channels. They were performed using a Debug Console connected to output informative messages about the current program status over the existing Debug USB of the LaunchPad board.

Furthermore, a user-defined LED on the LaunchPad was configured to light on when a device has been connected via Bluetooth to the weather station, and an interface for using the I^2C0 bus as a leader was provided and partially tested.

Deep comprehension of microcontrollers, C programming, bus systems, sensor technology, and software engineering was required to develop the communication subsystem of the weather station. Therefore the complexity of the project has been an excellent showcase for the knowledge and experience gained by students from computer science and electrical engineering programs of the Hamburg University of Applied Sciences.

7.2. Future Work

Future work should include incremental expansion of the existing source code.

Given that the functionality of the communication modules is similar, and all the modules have some common features, an object-oriented approach should be considered to allow a good level of abstraction. For the existing framework to continue to be used and the OOP approach to be integrated, the C++ programming language could be suggested for further development of the weather station.

An OOP approach would surely use additional program space. However, if the peripherals of the EK-TM4C1294XL are initialized and configured using the existing build-in TivaWare functions on the ROM memory, the overload to the program would be reduced considerably.

Furthermore, an LED indicating when the setup of the microcontroller has been completed would allow for visual help with identifying when a device can be connected to the weather station via Bluetooth. In addition, some error handling should be implemented, and universal status definitions should be introduced. These features could help with the automation of the weather station and the further development of the weather station framework.

Appendix A - Table with Detailed Overview of Features of Previously Developed Weather Stations

	Entw-Jan-13	Entw-Jan-14	Real-Jan-14	AWDAD-Jan-16	Real-Jan-16	Pro-Jan-17	Wett-Jan-18	Solar-Jan-20
Solution still available	No	No	No	No	No	Yes	No	Yes
Temperature	SHT15, BMP085	SHT15, BMP085	SHT15	SHT15	SHT 15, BMP085	HTU21D, BMP180	SHT 15, BMP085	BME280
Humidity	SHT15	SHT15	SHT15	SHT15	SHT 15	HTU21D	SHT 15	BME280
Air pressure	BMP085	BMP085	BMP085	BMP085	BMP085	BMP180	BMP085	BME280
Altitude	N/A	BMP085	Get via GPS	N/A	BMP085	Venus638FLP	BMP085, Fona 808	MPU6050
Direction/compass	HMC6352	HMC6352	CMPS03	HMC6352	HMC6352	HMC5883L	HMC5843	HMC5883L
Position/ GPS	N/A	Ultimate GPS breakout Board (MTK3339)	Ultimate GPS breakout Board (MTK3339)	Venus638FLPx	Venus838FLPx	Venus638FLP	Fona 808	NEO-6M
Wind direction and speed	N/A	WS2300-15	WS2300	WS2300	WS2300-15	WS2350	WS2300-15	SEN-08942
Real-time clock	DS1307	DS1307	Maxim OS1387	DS1307	RTC1307	DS3234	Fona 808	Get via GPS

Power data sent	Yes	Yes	No	Yes	No	Yes	No	Yes
SD card	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Communication module	XBee Pro S2	Xbee S1	Xbee Pro S1	Custom SIM using WLAN	XBee Pro S1	ESP-01	HC-05	HC-05
Communication protocol	Ember ZNet (a complete Zigbee protocol)	Zigbee stack + DigiMesh network protocol	Zigbee stack + DigiMesh network protocol	Wi-Fi	Zigbee stack + DigiMesh network protocol	Wi-Fi	Bluetooth	Blueto oth
Data output	N/A	N/A	Arduino COM	Raspberry Pi2, Server +Database	XBee attached + Windows /Linux app+ Microsoft SQL Server LocalDB	Raspberry Pi3, TCP Server + SQLite3 database	N/A	Windows app, no database
MCU	Arduino UNO	Arduino UNO	Arduino UNO	EK-LM4F120XL from TI	Arduino DUE	MSP-EXP432P401R from TI	Arduino DUE	STM8 L152

TABLE 16: TABLE WITH DETAILED OVERVIEW OF FEATURES OF PREVIOUSLY DEVELOPED WEATHER STATIONS

Appendix B - Comparison Tables for Handheld Devices

	Short Description	System	Display and Graphics	Available ports	Communication
Alpha 712 by Blue Chip Technology Based in the UK [45]	Rugged aluminum front panel tablet	CPU: 1GHz Cortex A8 ARM RISC, 256MB SDRAM OS: Windows CE 6.0 R3 or Linux Ubuntu or Android Memory: 256 MB RAM, 512 MB Flash, MicroSD sloth	7.1" TFT touchscreen 800x400	External: 2x USB 2.0 1x USB 1.1 Analog Audio-in & Out plugs 2x RS232 1x RS485 HDMI Optional Video capture inputs Internal: 12x GPIO	Optional GPS Optional GSM Optional 2G Ethernet Bluetooth Wi-Fi
Starterkit-N2930 Baytrail (Pico) by Anders Electronics PLC Based in the UK [46]	Separate display and processor with visible cabling	CPU: 1.86 GHz x86 Intel Bay Trail 4-core N2930 OS: Linux or Windows 10 LTSB or Windows 7/8 Memory: 4 GB RAM, 16 GB SSD	7th Intel Graphics 7" or 10.1" TFT touchscreen800x480 or 1280x800	LVDS HDMI Optional VGA 7x USB 2.0 1x USB 3.0 SATA2 mSATA 2x PCIe 2x UART	Ethernet Optional Wi-Fi
Starterkit-iMX6 by Anders Electronics PLC Based in the UK [46]	Separate display and processor	CPU: 1.2 GHz ARM Cortex A9 i.MX6 4-/2-/1-core OS: Linux Memory: 4 GB RAM,	GPU3Dv4 7'' or 10.1'' TFT touchscreen800x480 or 1280x800,	2x LVDS HDMI 1x USB 2.0 OTG 4x USB 2.0 Host Optional SATA-II Optional PCIe	Up to 2x Ethernet Optional Bluetooth Optional Wi-Fi

		32 GB SSD	Parallel RGB	Up to 2x UART Up to 24 GPIO Optional CAN I^2C	
Starterkit-T335 by Anders Electronics PLC Based in the UK [46]	Separate display and processor	CPU: 600 MHz ARM Cortex A8 AM3354 Single Core OS: Linux Memory: 512 MB DDR, 1 GB NAND	PowerVR SGX530, 4.3" or 7" TFT touchscreen, 800x480, Parallel RGB	LVDS DVI 1x USB 2.0 OTG Up to 4x USB 2.0 Host Up to 8x GPIO 1x CAN 2x I^2C 1x SPI	Optional Ethernet Optional Wi-Fi

TABLE 17: COMPARISON LIST OF COMPUTER MODULES WITH DISPLAY

	System description	Features	Price
Scorpion 8 Slim by Bressner Technology GmbH Based in Germany [47]	CPU: 2.0 GHz MSM8953 Quad-Core OS: Android 9.0 Memory: 4GB RAM 64GB ROM Battery: 7500 mAh	Display: 8" 1280x800 Corning Gorilla Glass, 5-point multitouch Weight and dimensions: 647g, 220 x 143,4 x 15,7 mm Ports: 1x micro USB, 1x mini HDMI, 3.5mm audio jack, 1x SIM, 1x micro SD Communication: Wi-Fi, Bluetooth, GPS, NFC, 4G LTE Certifications: IP65, MIL-STD-810G	516 €*
TOUCAN Mobile 8.3 by BRESSNER Based in Germany [48]	CPU: 2.4 GHz Intel Atom Z3795 Quad-core OS: Windows 10 IoT or Android 6.0 Memory: 4GB RAM 64 or 128 GB eMMC(Flash) Battery life: 8-10 Wh	Display: 8.3" 1920×1200 Corning Gorilla Glass, 10-point multitouch Weight and dimensions: 758g, 227 x 150 x 12.5 mm Ports: 1x USB 2.0, 3.5mm audio jack, 1x micro SD Communication: Wi-Fi, Wi-Fi Direct, Bluetooth, GPS, NFC, Optional LTE Certifications: IP65	1020 €*

RT71 7" Rugged tablet PC by XANARC Direct Based in the USA [49]	CPU: 1.5 GHz ARM Cortex 53 Quad-core OS: Android 7.0 Memory: 3 GB RAM 32 GB ROM EMCP Battery: 9650mAh	Display: 7" 1280 x 720 Weight and dimensions: 662g, 202 x 123 x 22 mm Ports: 1x USB Type-C, 3.5mm audio jack, 1x RS232, 1x SIM, 1x micro SD Communication: Wi-Fi, Bluetooth, 2G/3G/4G LTE, GPS, NFC Certification: IP65, NEMA6P	478 €**
Apglos Armour by Apglos Based in the Netherlands [50]	CPU: 1.5 GHz ARM Cortex 53 Quad-core OS: Android 7.0 Memory: 3 GB RAM 32 GB ROM EMCP Battery: 10000mAh	Display: 7.9 " 1024x 768 capacitive Weight and dimensions: 860g, 159 x 233 x 21 mm Ports: 1x USB Type-C, 3.5mm audio jack, 1x SIM, 1x micro SD Communication: Wi-Fi, Bluetooth, 2G/3G/4G LTE, GPS Certification: IP67, MIL-STD-810G	490 €**
AIM-35AT-02307000 by Advancetech Based in Asia but with distributor in Germany [51]	CPU: 1.44 GHz Intel® Atom™ x5-Z8350 Quad-core OS: Windows 10 IoT or Android 6.0 Memory: 4 GB RAM 64 GB eMMC Battery: 4900mAh (18.6 Wh)	Display: 8" 1920 x 1200 Gorilla Glass 3, 10-point multitouch Weight and dimensions: 600g, 240 x 142 x 14.5 mm Ports: 1x micro HDMI, 1x micro USB, 3.5mm jack, 1x micro SIM, 1x micro SD, 1x AIM extension 14-pin pogo connector, 1x AIM dock 16-pin pogo connector Communication: Wi-Fi, Bluetooth, NFC Certification: IP65	707 €**

TABLE 18: COMPARISON TABLE OF INDUSTRIAL RUGGED TABLETS

* price available on dimedtec.de: last accessed on 22.04.2021
** price available on the original web page of the producer: last accessed on 22.04.2021

Appendix C - Bluetooth Commands to be Implemented and Command Parameter Definitions

	Description	Return
AT	Test the module	OK
ATE<n>	Activate/Deactivate remote echo	OK
ATI	Get the name of the weather station	<ID>
AT+CTIME=<YY>,<MM>, <DD>,<hh>,<mm>,<ss>	Set the date and time	OK
AT+CGPSPOS=<lat>,<lon>[,<alt>]	Set the GPS position	OK
AT+CGPSPWR=<n>	Turn GPS on/off	OK
AT+CMESINTV=<intvl>	Set the measurement interval	OK
AT+CUPINTV=<intvl>	Set the Bluetooth upload interval	OK
AT+CSPALIGN!	Request alignment of the Solar Panel	OK
AT+COPMODE=<opMode>	Set the operating mode of the weather station	OK
AT+CLED=<LED>,<n>	Set an LED on/off	OK
AT+CTEMP?	Get temperature reading	+CTEMP:<temp> OK
AT+CPRES?	Get pressure reading	+CPRES:<press> OK
AT+CHUM?	Get humidity reading	+CHUM:<hum> OK
AT+CTIME?	Get date and time reading	+CTIME:<YY>,<MM>,<DD>, <hh>,<mm>,<ss> OK
AT+CWIND?	Get wind speed and direction reading	+CWIND:<w_spd>,<w_dir> OK
AT+CGPSPOS?	Get GPS position and altitude	+CGNSPOS:<lat>,<lon>,<alt> OK
AT+CWSPWR?	Get the power statuses of the weather station	+CPWR:<pwr_stats> OK

AT+CGUI?	Get readings in GUI format string	+CGUI:<YY>,<MM>,<DD>,<hh>, <mm>,<ss>,<temperature>, <pres>,<hum><w_dir>,<w_spd>, <zen>,<azm>,<lat>,<lon>,<alt>, <list_pwrStats> OK

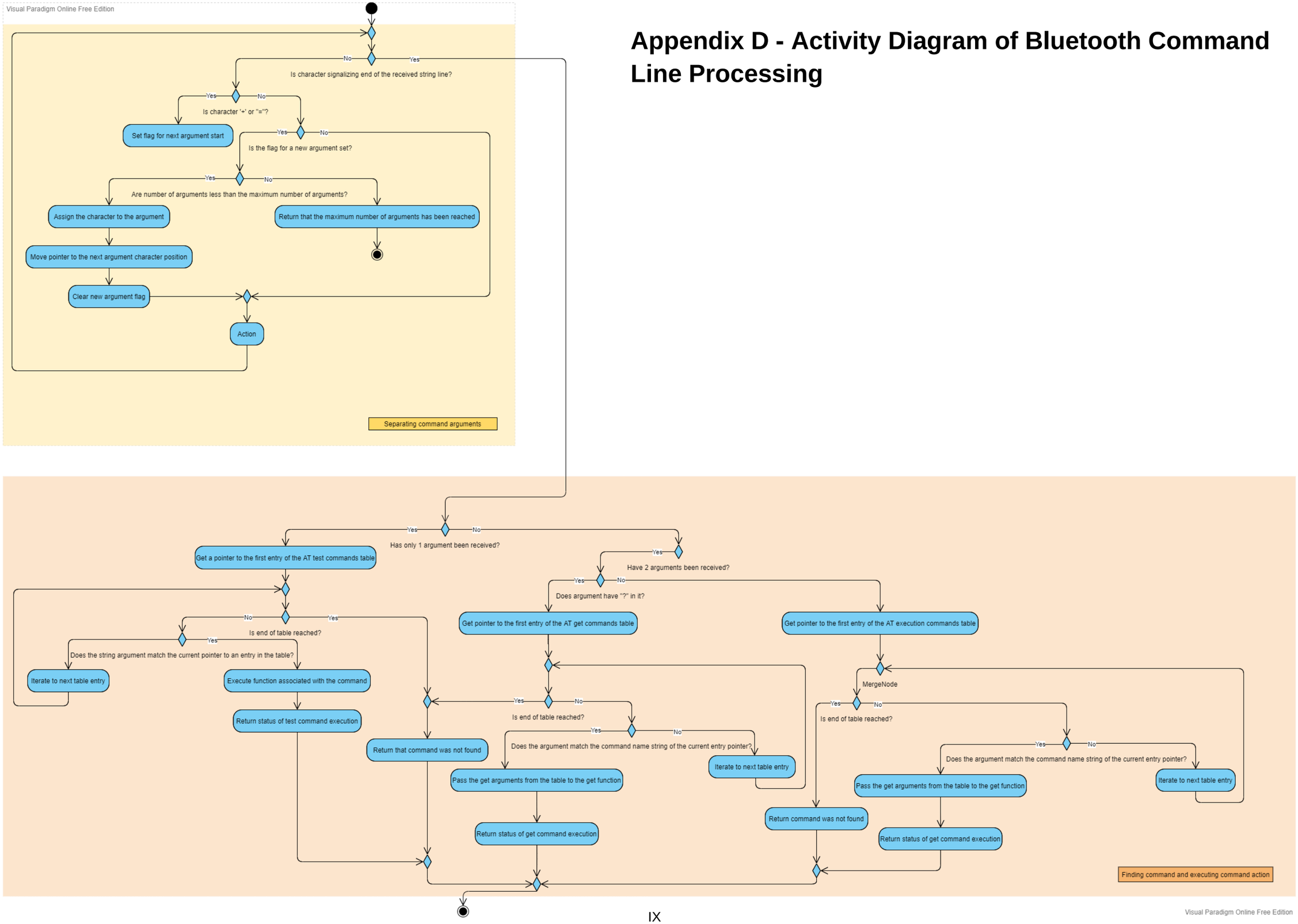
TABLE 19: LIST OF BLUETOOTH AT COMMANDS TO BE IMPLEMENTED AND THEIR EXPECTED ANSWERS

	Parameter meaning	Value type
<ID>	Name of the weather station	String with an artificial name for the weather station
<YY>	Year	Numerical from 0 to 99 relating for years after 2000
<MM>	Month	Numerical from 1 to 12
<DD>	Day	Numerical from 1 to 31
<hh>	Hour	Numerical from 0 to 23 (UTC)
<mm>	Minute	Numerical from 0 to 59
<ss>	Second	Numerical from 0 to 59
<zen>	Zenit	Numerical
<azm>	Azimuth	Numerical
<lat>	Latitude	Numerical
<lon>	Longitude	Numerical
<alt>	Altitude	Numerical
<hum>	Humidity	Numerical
<press>	Atmospheric pressure	Numerical
<temp>	Temperature	Numerical
<w_dir>	Wind direction	Numerical
<w_spd>	Wind speed	Numerical
<n>	On or Off status	Numerical 0 or 1
<intvl>	Interval	Numerical in seconds
<opMode>	Weather Station operating mode	Numerical
<LED>	LED number	Numerical
<i_bat>	Battery current	Numerical in mA
<v_bat>	Battery voltage	Numerical in mV
<i_solar>	Solar panel current	Numerical in mA

<v_solar>	Solar panel voltage	Numerical in mV
<v_sys>	System voltage	Numerical in mV
<list_pwrStats>	A list with the power statuses of the weather station	Numerical
<bme>	Temperature, Pressure or Humidity data from BME280 module	Numerical in 0.01°C, 1 Pa or 0.01%
<cpu>	Temperature of the CPU	Numerical in 0.01°C
<qmc>	Temperature from QMC5883L module	Numerical in 0.01°C
<mpu>	Temperature from MPU6050 module	Numerical in 0.01°C

TABLE 20: LIST OF BLUETOOTH COMMAND PARAMETERS AND THEIR MEANINGS

Appendix D - Activity Diagram of Bluetooth Command Line Processing



Appendix E - Bluetooth AT Commands Implemented

	Description
AT	Check Bluetooth connection
ATE0	Disable Bluetooth Echo
ATE1	Enable Bluetooth Echo

TABLE 21: LIST OF AVAILABLE BLUETOOTH AT TEST COMMANDS

	Description
AT+CWKUP!	Execute manual wakeup
AT+CHIB!	Manually put microcontroller in Hibernate mode
AT+CALIGN!	Request alignment of the solar panel
AT+CWIFISEND!	Send data from microcontroller to Server via Wi-Fi

TABLE 22: LIST OF AVAILABLE BLUETOOTH AT EXECUTION COMMANDS

	Description
AT+CTIME=<YY>.<MM>.<DD> <hh>:<mm>:<ss>	Set real time clock value
AT+CGNSPOS=<lat>,<lon>[,<alt>]	Set GPS coordinates
AT+CINTV=<intvl>	Set interval for taking measurements
AT+CTRACK=<n>	Set on/off the solar panel tracking
AT+CTURN=<dir>	Start manual calibration of the magnetometer
AT+CWFILE=<data to write>	Write the weather data received from BT to the SD card
AT+CBTINTVL=<intvl>	Set interval for sending measurements to BT connected device
AT+CGPSPWR=<n>	Set GPS power on/off
AT+CTESTOP=<n>	Set maintenance operating mode of the weather station on/off
AT+CLED=<LED_nr>,<n>	Set a given LED on or off

TABLE 23: LIST OF AVAILABLE BLUETOOTH AT SET COMMANDS

	Return command	Description
AT+CALIGN?	+CALIGN:<compound_parameter>	Get current alignment of solar panel
AT+CWSNAME?	+CWSNAME: <compound_parameter>	Get Station Name

AT+CTEMP?	+CTEMP: <temp>	Check current temperature measurements
AT+CPRES?	+CPRES: <pres>	Check air pressure
AT+CHUM?	+CHUM: <hum>	Check relative humidity
AT+CTIME?	+CTIME:<YY>.<MM>.<DD> <hh>:<mm>:<ss>	Check real time clock value
AT+CWIND?	+CWIND:<compound_parameter>	Check wind direction and speed
AT+CGNSPOS?	+CGNSPOS:<compound_parameter>	Check GPS coordinates
AT+CPWR?	+CPWR:<compound_parameter>	Check current and voltage measurements
AT+CINTV?	+CINTV:<intvl>	Get interval for taking measurements
AT+CBTINTVL	+CBTINTVL:<intvl>	Get interval for sending measurements to BT connected device
AT+CGUI?	+CGUI:<compound_parameter>	Get data used for the GUI
AT+CTRACK?	+CTRACK:<n>	Get if the solar panel is being tracked or not
AT+CRFILE?	+CRFILE:<compound_parameter>	Read last line from SD card and return it
AT+CWIFI?	+CWIFI:<n>	Get Wi-Fi status - if available and if connection is established

TABLE 24: LIST OF AVAILABLE BLUETOOTH AT GET COMMANDS

Appendix F - Wi-Fi AT Commands Implemented

	Command type	Description
AT	Execute	Test AT Startup
ATE0	Execute	Switch echo off
ATE1	Execute	Switch echo on
AT+RST	Execute	Restart the Module
AT+GMR	Get	Check AT and SDK Version Information
AT+RESTORE	Execute	Reset all parameters saved in flash and restores the factory default settings of the module. The chip will be restarted when this command is executed.
AT+UART_CUR=<baudrate>,<databits>, <stopbits>,<parity>,<flowcontrol>	Set	Set the UART configuration of the Wi-Fi module
AT+UART_CUR?	Get	Get the UART configuration of the Wi-Fi module
AT+GSLP=<time>	Set_get	Put the Wi-Fi module in deep sleep mode for X ms
AT+SLEEP=<sleep mode>	Set	Set sleep mode of the Wi-Fi module
AT+SLEEP?	Get	Get the sleep mode of the Wi-Fi module
AT+WAKEUPGPIO=<enable>, <trigger_GPIO>,<trigger_level>	Set	Configure a GPIO to wake up the module from light-sleep mode
AT+SYSRAM?	Get	Get the remaining RAM of the Wi-Fi module
AT+CWMODE=<wifi_mode>	Set	Set the Wi-Fi mode (Station/AP/Station+AP)
AT+CWMODE_CUR?	Get	Get the Wi-Fi mode (Station/AP/Station+AP)
AT+CIFSR	Get	Check if the module is connected to an AP
AT+CIPMUX=0	Execute	Disable multiple connections to the Wi-Fi module
AT+CIPMUX=1	Execute	Enable multiple connections to the Wi-Fi module
AT+CIPMUX?	Get	Query the connection mode of the Wi-FI module
AT+CWJAP_CUR=<ssid>,<pwd>	Set	Connect to an AP
AT+CWJAP_CUR?	Get	Get the AP to which the Wi-Fi is connected to

AT+CWLAP	Get	List available APs
AT+CWLAP=<ssid>	Set	Check if an AP with a specific SSID is available
AT+CWQAP	Execute	Disconnect from an AP
AT+CIPSTA_CUR=<module_ip>	Set	Set the IP address of the Wi-Fi module
AT+CIPSTA_CUR?	Get	Get the current IP address of the Wi-Fi module
AT+CWHOSTNAME=<hostname>	Set	Set the host name of the Wi-Fi module
AT+CWHOSTNAME?	Get	Get the host name of the Wi-Fi module
AT+CIPSTATUS	Get	Get the Connection Status
AT+CIPSTART=<connection type>,<remote IP>,<remote port>	Set	Establish TCP IP connection
AT+CIPCLOSE	Execute	Close TCP IP connection
AT+CIPSEND=<length>	Set	Send data from the weather station to the Web server in single connection mode (CIPMUX=0)

TABLE 25: LIST OF AVAILABLE WI-FI AT COMMANDS BASED ON ESP8266 AT SOFTWARE

	Type [possible values]	Description
<baudrate>	Numerical, e.g. 9600,115200	The baudrate for the UART connection. Typically set to 115200.
<databits>	Numerical [5,6,7,8]	The number of data bits for the UART frame. Typically set to 8.
<stopbits>	Numerical [1,1.5,2]	The number of stop bits for the UART frame. Typically set to 1.
<parity>	Numerical [0,1,2]	The number of parity bits for the UART frame. Typically set to 0- no parity bit.
<flowcontrol>	Numerical [0,1,2,3]	The flow control for the UART connection. Typically set to 0- no flow control.
<time>	Numerical (>0)	The Time in ms for which the module is put to sleep mode
<sleep mode>	Numerical [0-disable sleep mode, 1-light sleep mode, 2-modem sleep mode]	Defines the sleep mode to use when put to sleep. It can be used only when the module is in Station mode.
<enable>	Numerical [0-disable,1-enable]	Defines if a given GPIO on the Wi-Fi module should wake it up from sleep mode
<trigger_GPIO>	Numerical [0 to15]	Trigger GPIO for wakeup of the Wi-Fi module
<trigger_level>	Numerical [0- wake on low level, 1- wake on high level]	The trigger level for the wakeup GPIO of the Wi-Fi module

<wifi_mode>	Numerical [1-Station mode, 2-SoftAP mode, 3-SoftAP+Station Mode]	The operating mode of the module. When operating in Station mode, the module can connect to remote servers.
<ssid>	String, e.g. “Wetterstation”	The name of the access point to connect to
<pwd>	String, e.g. “wetterdeluxe”	The password of the access point to connect to
<module_ip>	String, e.g. “192.168.178.1”	The IP address of the Wi-Fi module
<hostname>	String, e.g. “WS2021”	The hostname of the Wi-Fi module
<connection type>	String [“TCP”, “UDP”]	The type of the remote server connection
<remote IP>	String, e.g. “192.168.178.82”	The IP address of the remote server to connect to
<remote port>	String, e.g. “100”	The port of the remote server to connect to
<length>	Numerical (>0)	The length of the data to send in bytes

TABLE 26: LIST OF PARAMETERS USED IN THE WI-FI AT COMMANDS

Appendix G - Test Scenarios

Hardware

The UART and Hibernation module libraries as part of the supported by Texas Instrument's TivaWare SDK. It can be assumed that the drivers provided have already been tested and work as instructed as long as they are configured correctly.

For each hardware connection for this project, the configuration should be tested.

	Test name	Description
TS1	UART configuration for Wemos D1 mini module	The UART driver instance of the TM4C microcontroller for connecting to the Wi-Fi device shall be configured with the same frame format as the configuration of the UART interface of the Wi-Fi device itself. The receiving (Rx) and transmitting (Tx) wires should be correctly connected - the Rx line from the microcontroller is connected to Tx of the Wi-Fi device, and the Tx line from the microcontroller is connected to the Rx of the Wi-Fi device. This functionality could be proven either using a protocol analyser connected to the channel or by proof of the higher-level application tests, namely if a message is received in a readable ASCII-character format.
TS2	UART configuration for HC-05 module	Similar to TS1but the connection between the TM4C LaunchPad UART instance and the Bluetooth module are tested.
TS3	GPIO input pin configuration for HC-05 module status pin	The GPIO pin of the TM4C should be correctly configured as input for the State pin of the Bluetooth module. This can be proved via the correct detection of input from this pin, which should be part of the software tests.

Software

Debug console

Setup: A serial to USB connection is made between the microcontroller and a personal computer. The “PuTTY” serial console shall be used to display data received on the USB port of the computer. The same serial configuration should be setup on the microcontroller for the serial output and on the reading serial console on the personal computer.

	Test name	Description
TS4	Print out debug info	Hardcoded messages are printed out on the serial line of the associated microcontroller. The hardcoded messages

		are received and displayed in human-readable format on the serial console window.

Wi-Fi

Setup: At each command sending point an informative message shall be printed on the Debug console serial line. If the command was successful a “SUCCESSFUL” message shall be printed after the message, else a “FAILED” message shall be printed. Additionally, the USB connection of the Wemos D1 mini shall be used and connected to another serial console on the personal computer and setup with the same configuration as the serial port used on the microcontroller for the Wi-Fi module. The second console shall display all the commands received and sent to the Wi-Fi module from the microcontroller. A visual comparison between the two console outputs should be done to determine if the test was successful.

	Test name	Description
TS5	Connection to access point	Sending the command “AT+ CWJAP_CUR= <ssid>,<pwd>” where the ssid is “Wetterstation” and the pwd is “wetterdeluxe” and receiving an “OK” in return.
TS6	Connection to web server	Sending the command “AT+CIPSTART=<type>,<ip_addr>,<port>” where the type is “TCP”, the ip_addr is the IP address “192.168.178.82” and the port is “100”. The expected response is “OK”.
TS7	Data send to web server	Sending the command “AT+CIPSEND=<byte_length>”, where the parameter byte_length is the byte length of the data to be sent. The server should respond with a “>” character, but there is no need to wait for it. Next the data string as is, is sent to the server, if the data byte length is as expected a “SEND OK” string will be received on the UART line.
TS8	Test AT command driver	Sending the command “AT”, the driver responds with “OK”.
TS9	Restart the Wi-Fi module	Sending the command “AT+RST”, the driver responds with “ready” and then with “OK”.
TS10	Disconnect from the AP	Sending the command “AT+CWQAP”, the driver responds with “OK”.
TS11	Disconnect from the web server	Sending the command “AT+CIPCLOSE”, the driver responds with “OK”.
TS12	Data upload to the web server	Sending 21 hardcoded values in semi-colon separated format where the time and date data are accurately written, and all the other values are assumed to be zero. Upon check on the server web page, the time and date are those of the hardcoded time and date values.

Bluetooth General

Setup: For testing the Bluetooth capabilities, an application allowing Bluetooth to serial console shall be used. The application could be mobile based and it itself needs to be tested that it

accurately transmits and receives data over a Bluetooth channel via using two devices who have installed this same application.

	Test name	Description
TS13	Bluetooth application testing device	Connecting two devices which have installed the same application, interconnecting them via a Bluetooth link and sending a simple “Hello” message from one to the other results in a “Hello” message received on the receiving device application.
TS14	Bluetooth connected/disconnected recognition	Continuously polling the Bluetooth status pin associated GPIO connected to the LaunchPad, when a connection is initiated by the testing BT device, a recognised connection is indicated by printing an informative message on the Debug Console.
TS15	Bluetooth command receiving	A command is sent from the already connected BT device application to the BT of the weather station. On command received, it is printed on the Debug Console via an informative message.
TS16	Bluetooth command processing	Categories are assigned to the commands - execute, set, get. Once a message is received, it should be assigned a category according to a recognition pattern. If the pattern works correctly, the informative message displaying the type of command will display the correct assignment.
TS17	Bluetooth test, set or execute command execution	For the test command “AT”, if it was received and processed correctly, the microcontroller should send an “OK” message back to the BT connected device followed by <CR> and <LF> characters. The message shall be displayed on the BT connected device application console. An informative message indicating the string sent shall also be printed on the Debug Console.
TS18	Bluetooth get command execution	Similar to TS17, however the message returned should have “+<command_name>=<value1>,<value2>…”, followed by <CR> and <LF> characters.

Bluetooth Commands with Their Expected Response

Setup: A command is sent from a Bluetooth enabled handheld device. The Settings involve a <CR><LF> combination of characters at the end of each command. The actual response is evaluated if it matches the expected response.

	BT Command	Expected Response
TS19	AT	OK
TS20	ATE0	OK
TS21	ATE1	OK
TS22	AT+CWKUP!	FAILED

TS23	AT+CHIB!	FAILED
TS24	AT+CALIGN!	FAILED
TS25	AT+CWIFISEND!	OK
TS26	AT+CTIME=<YY>.<MM>.<DD> <hh>:<mm>:<ss>	OK
TS27	AT+CGNSPOS=<lat>,<lon>[,<alt>]	FAILED
TS28	AT+CINTV=<intvl>	FAILED
TS29	AT+CTRACK=<n>	FAILED
TS30	AT+CTURN=<dir>	FAILED
TS31	AT+CWFILE=<data to write>	FAILED
TS32	AT+CBTINTVL=<intvl>	FAILED
TS33	AT+CGPSPWR=<n>	FAILED
TS34	AT+CTESTOP=<n>	FAILED
TS35	AT+CLED=<LED_nr>,<n>	FAILED
TS36	AT+CALIGN?	FAILED
TS37	AT+CWSNAME?	"+CWSNAME:STP_1A WETTERSTATION" OK
TS38	AT+CTEMP?	"+CTEMP:0" OK
TS39	AT+CPRES?	"+CPRES:0" OK
TS40	AT+CHUM?	"+CHUM:0" OK
TS41	AT+CTIME?	"+CTIME:21,06,17 13:09:51" OK
TS42	AT+CWIND?	"+CWING:0,0" OK
TS43	AT+CGNSPOS?	"+CGNSPOS:100198,535566" OK
TS44	AT+CPWR?	"+CPWR:0,0,0" OK
TS45	AT+CINTV?	"+CINTV:0" OK
TS46	AT+CBTINTVL	"+CBTINTVL:0" OK
TS47	AT+CGUI?	"+CGUI:21;06;17;13;38;17;;;;;;;;535566451;10019889;;;;;;" where the first 6 values will be the differnet OK
TS48	AT+CTRACK?	FAILED
TS49	AT+CRFILE?	"+CRFILE:No values yet”

TS50	AT+CWIFI?	"+CWIFI:IP DISCONNECTED" or “CWIFI:TCP DISCONNECTED” OK

Bluetooth LED

	Test name	Description
TS51	Bluetooth Status LED	Visual inspection on Bluetooth LED and comparison with the HC-05 embedded red LED shall be done. When the HC-05 LED is continuously blinking, the Bluetooth LED should be turned off. When the HC-05 LED is blinking twice with an interval of 5 seconds, the Bluetooth LED should be turned on.

I^2C

	Test name	Description
TS52	Operation of I^2C0 with the microcontroller as the initiator of the bus	Assume that the chosen driver works correctly. The BME280 is initialized in the main routine after the I^2C0 master connection has been enabled. If the BME280 initialization is successful, then the I^2C0 master interface has been configured correctly. This would mean that , that a packet was sent and received on the $I2^C0$ bus line from the microcontroller.

Acknowledgement

I would like to express my gratitude to my primary supervisor and mentor Prof. Dr.-Ing. Lutz Leutelt from the Faculty of Engineering and Computer Science, Hamburg University of Applied Sciences. His expertise, guidance and encouragement played a vital role in making this thesis a success. An additional acknowledgement is due to Prof. Dr. Marc Hensel and Detmar Rüdiger from HAW Hamburg for providing technical support and knowledge, which have helped me tremendously for the completion of the project described in this thesis paper.
I would also like to thank my friends for the support and understanding they showed during my studies and during the writing of this paper.

Bibliography

[26] A. J. Spolsky, User Interface Design for Programmers, illustrated ed., Apress, 2001, p. 144.

[46] Anders Electronics PLC, "Embedded Displays," Anders Electronics PLC, [Online]. Available: https://www.andersdx.com/embedded-display-touch-systems/. [Accessed 26 April 2021].

[50] Apglos, "Rugged tablet Apglos Armour," Apglos B.V., [Online]. Available: https://www.apglos.eu/shop2/hardware/handhelds/apglos/rugged-tablet-apglos-armor/. [Accessed 26 April 2021].

[42] AZ-DELIVERY, "GY-BME280 Barometric sensor for temperature, humidity and air pressure," [Online]. Available: https://www.az-delivery.de/en/products/gy-bme280. [Accessed 12 Jun 2021].

[45] Blue Chip Technology, "ALPHA 712," Blue Chip Technology, [Online]. Available: https://www.bluechiptechnology.com/products/alpha-712/. [Accessed 26 April 2021].

[41] Bosch Sensortec, "BME280 Combined humidity and pressure sensor - Data sheet," 2020.

[48] Bressner, "TOUCAN Mobile 8.3" - Windows," Bressner, [Online]. Available: https://www.bressner.de/shop/mobile-computing/industrial-tablets/toucan-mobile-8-3-windows/. [Accessed 26 April 2021].

[36] C. David, "ESP8266 Pinout Overview [ESP-01, NodeMCU, WeMos D1 Mini]," [Online]. Available: https://diyi0t.com/what-is-the-esp8266-pinout-for-different-boards/. [Accessed 23 April 2021].

[37] Conrad, "HC-05 Wireless Bluetooth Transceiver Modul mit Adapter Board für MCU (AVR/ARM/PIC)," Conrad, [Online]. Available: https://www.conrad.de/de/p/hc-05-wireless-bluetooth-transceiver-modul-mit-adapter-board-fuer-mcu-avr-arm-pic-802247514.html. [Accessed 23 April 2021].

[34] e-Gizmo Mechatronix Central, "HC-05 Bluetooth Module Breakout Board Technical Manual Rev1r0," 2016. [Online]. Available: https://www.e-gizmo.net/oc/index.php?route=product/product&product_id=1258&search=hc05&description=true. [Accessed 14 Jun 2021].

[25] Energia, "Guide to the TM4C129 Connected LaunchPad (EK-TM4C1294XL)," [Online]. Available: https://energia.nu/pinmaps/ek-tm4c1294xl/. [Accessed 22 April 2021].

[38] Espressif Systems, "ESP8266 AT command examples," 2017.

[31] Espressif Systems, "ESP8266 AT Instruction Set," Espressif Systems, 2020.

[32] Espressif Systems, "ESP8266 Low-Power Solutions," 2019.

[35] Espressif Systems, "ESP8266 Technical Reference," 2020.

[13] European Telecommunications Standards Institute, "Human Factors (HF); AT Commands for Assistive Mobile device Interfaces," European Telecommunications Standards Institute (ETSI), 2007.

[7] G. Gridling and B. Weiss, "Introduction to Microcontrollers," Vienna University of Technology, Vienna, 2007.

[15] Hayes Microcomputer Products, Inc., "Technical Reference for Hayes Modem Users," Hayes Microcomputer Products, Inc., Atlanta, 1993.

[39] I. Sommerville, Software Engineering, Ninth ed., M. Horton and M. Hirsch, Eds., Boston, Massachusetts: Addison-Wisley, 2011, p. 773.

[9] I. Susnea and M. Mitescu, Microcontrollers in Practice, Heidelberg: Springer, 2005, p. 257.

[21] IBM Redbooks, TCP/IP Tutorial and Technical Overview, Eight Edition ed., International Business Machines Corporation (IBM) Redbooks, 2006, p. 998.

[16] IEEE Computer Society, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, 2021.

[20] IEEE Computer Society, "Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)," 2005.

[14] ITU Telecommunication Standardization Sector (ITU-T), "Series V: Data Communication over the Telephone Network - Serial asynchronous automatic dialling and control," International Telecommunication Union (ITU), 2003.

[2] J. D. Exline, A. S. Levine and J. S. Levine, "Meteorology: An Educator's Resource for Inquery-Based Learning for Grades 5-9: Introduction," National Aeronautics and Space Administration (NASA), August 2006. [Online]. Available: https://www.nasa.gov/centers/langley/pdf/245901main_MeteorologyTeacherRes-Intro-Ch1.r3.pdf. [Accessed 17 May 2021].

[43] K. D. Trinh, "BMP280 Driver using TivaC," [Online]. Available: https://github.com/khoitd1997/BMP280_driver_TivaC. [Accessed 10 June 2021].

[44] K. Morich, "Serial Bluetooth Terminal," 6 Mar 2021. [Online]. Available: https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=en&gl=US. [Accessed 18 Jun 2021].

[18] K. Sharma and N. Dhir, "A Study of Wireless Networks: WLANs, WPANs, WMANs, and WWANs with Comparison," International Journal of Computer Science and Information Technologies (IJCSIT), vol. 5, no. 7810-7813, p. 4, 2014.

[4] L. Leutelt, "Basic Microcontroller Architecture" slideset, lecture material of "Microcontrollers", Hamburg: HAW Hamburg, 2020.

[11] L. Leutelt, "Introduction to Embedded C" slideset, lecture material of "Microcontrollers", Hamburg: HAW Hamburg, 2020.

[6] L. Leutelt, "Serial Interface: UART" slideset, lecture material of "Microcontrollers", Hamburg: HAW Hamburg, 2020.

[12] L. Leutelt, "Software Design with Interrupts" slideset, lecture material of "Microcontrollers", Hamburg: HAW Hamburg, 2020.

[5] M. Mitescu and I. Susnea, Advanced Microelectronics: Microcontrollers in Practice, Berlin Heidelberg: Springer, 2005, p. 250.

[51] Mouser, "Advantech AIM-35AT-02307000," Advancetech, [Online]. Available: https://www.mouser.de/ProductDetail/Advantech/AIM-35AT-02307000?qs=%2Fha2pyFadujyiqQuuZxN7sXjzvlh7QhA%252B9i6wZznkpJRYlNPjP1Q%2FR6iI0w1xYn%252B. [Accessed 26 April 2021].

[10] S. Campbell, "Basics of the I2C Communication Protocol," Circuit Basics, [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed 11 May 2021].

[8] S. Campbell, "Basics of UART Communication," Circuit Basics, [Online]. Available: https://www.circuitbasics.com/basics-uart-communication/. [Accessed 11 May 2021].

[19] S. Driver, "Wi-Fi 6 and the Legacy of Wi-Fi Standards," Business News Daily, 20 Mar 2020. [Online]. Available: https://www.businessnewsdaily.com/10570-wi-fi-standards-explained.html#:~:text=The%20802.11%20standards%20establish%20how,between%20the%20router%20and%20devices.. [Accessed 21 May 2021].

[17] S. Rackley, Wireless Nerworking Technology, Jordan Hill, Oxford: Elsevier, 2007, p. 413.

[47] Scorpion, "Scorpion 8" Slim - Industrial Rugged Tablet with 8.0 inch Display," Scorpion Rugged Tablet & Handheld Solutions, [Online]. Available: https://www.scorpion-rugged.de/en/products/scorpion-8-slim/. [Accessed 26 April 2021].

[27] Skymet Weather Team, "How to choose a home weather station," Skymetweather, 30 Mar 2021. [Online]. Available: https://www.skymetweather.com/content/weather-news-and-analysis/how-to-choose-a-home-weather-station/. [Accessed 20 May 2021].

[3] STMicroelectronics Microcontroller Division Applications, "Microcontrollers made easy," [Online]. Available: https://www.st.com/resource/en/application_note/cd00003980-microcontrollers-made-easy-stmicroelectronics.pdf. [Accessed 31 March 2021].

[30] Texas Instruments, "CC3100 SimpleLink™ Wi-Fi® and IoT Solution BoosterPack Hardware User's Guide," 2015.

[23] Texas Instruments, "Tiva™ C Series TM4C1294 Connected Launchpad Evaluation Kit - User's Guide," Texas Instruments Incorporated, Austin, 2014.

[24] Texas Instruments, "Tiva™ TM4C1294NCPDT Microcontroller - Data sheet," Texas Instruments Incorporated, Austin, 2014.

[29] Texas instruments, "Tiva™ TM4C129X Development Board - User's Guide," Texas Instruments Incorporated, Austin, 2016.

[40] Texas Instruments, "TivaWare Peripheral Driver Library User's Guide," Texas Instruments Incorporated, Austin, 2020.

[28] Texas Instruments, “TM4C Microcontrollers Product Selection Guide,” Texas Instruments Incorporated, Austin, 2021.

[1] University of Waikato, "Measuring the weather – a timeline," University of Waikato, 19 April 2018. [Online]. Available: https://www.sciencelearn.org.nz/interactive_timeline/9-measuring-the-weather-a-timeline. [Accessed 17 May 2021].

[33] W. Ewald, "Wemos D1 Mini Boards," Wolles Elektronikkiste, 8 Jan 2021. [Online]. Available: https://wolles-elektronikkiste.de/en/wemos-d1-mini-boards. [Accessed 15 Mar 2021].

[22] W. Goralski, The Illustrated Network: How TCP/IP Works in a Modern Network, Second Edition ed., Cambridge, Massachusetts: Morgan Kaufmann, 2017, 2009, p. 936.

[26] A. J. Spolsky, User Interface Design for Programmers, illustrated ed., Apress, 2001, p. 144.

Declaration

I declare within the meaning of section 25(4) of the Examination and Study Regulations of the International Degree Course Information Engineering that this Bachelor report has been completed by myself independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Hamburg, 21 June 2021
City, Date

Sign