Sensor Selection, Structure, Data Collection and Transmission

Version 2

Ronaldo Tsela **(August 2023)**

Contents

Purpose....6
Introduction....6
Objectives....7
Requirements....8
Methodology....10
Sensor selection....10
Programming the Sensors....11
Health Monitors....12
Programming the Monitors....14
Programming the Network Adapter....15
Power Supply....16
System Architecture....17

List of Figures

Figure 1 - The schematic diagram of the sensors 11
Figure 2 - The health monitoring schematic diagram 14
Figure 3 – The power management, storage and distribution system 16
Figure 4 – System architectural diagram 17
Figure 5 – Main control FSM 17
Figure 6 – The backend black-box architecture and the interconnection between the middleware and hardware with the backend components. 18

List of Tables

Table 1 - AM2315 device specifications 10
Table 2 - BMP280 device specifications 11
Table 3 - The R1 and R2 selected resistance values for 4 VDC threshold and different source voltage levels 13
Table 4 – AT commands utilized for transmitting data over GPRS/ GSM with HTTP application protocol 15

List of Equations

Equation 1.....13
Equation 2.....13
Equation 3.....13
Equation 4.....13
Equation 5.....14

Purpose

The purpose of this report is to present the preliminary design stage for the development of a ground-based weather station network and its associated Application Programming Interface (API) and User Interface (UI). This first stage involves selecting and programming various sensors, positioning them within the station structure, creating a data collection procedure, designing internal health monitors, and establishing a data transmission link for measurements and telemetry. The report outlines the fundamental requirements and objectives necessary to make the above feasible.

Introduction

Sensors play a crucial role in the weather monitoring system as they greatly influence the overall performance. Therefore, careful selection of sensors is vital for accurate and efficient data collection. The entire structure of the system is designed to support these sensors, so it is important to ensure compatibility between the sensors selected, the external system, the power supply, and the controller's motherboard. Typically, a weather monitoring system includes fundamental sensors such as temperature sensors, hygrometers, barometers, anemometers for measuring wind speed, wind direction sensors, and rain gauges for measuring rainfall.

Since these sensors are mostly electronical devices for controlling them, an onboard computer is required, equipped with proper control software. There is no need of high-performance computer systems, a small and simple microcontroller usually does the work very well since microcontrollers as opposed to the microprocessors are optimized for tasks that require accurate and low-level control with limited power consumption. The only requirement is that the microcontroller should be equipped with a series of standard interfaces such as SPI, I2C, UART, CAN etc and also with several general-purpose input and output ports in order to communicate with a wide range of devices.

The key parameter when designing outdoor projects such as this weather station network is autonomy. Autonomy refers to the capability for self-providing energy to the system for all its operations and to the existence of automated internal control logic for decision making and task execution. The selected controller here equipped with proper software plays a crucial role in achieving the half of control automation. The other half is achieved via remote control capabilities, where a network, preferably wireless, is required for data exchange.

A system can produce energy itself in multiple ways, one of which is the photovoltaic phenomenon, where the solar radiation is converted by photovoltaic arrays into electric power. This power is stored in a mass energy storage system such as a simple battery for continuous operation at night-time where the sun is not capable of providing the available power to support the system's continuous operations. Although in general solar energy is very efficient sometimes it is not applicable, therefore it is important to keep in mind that the main power source depends strongly in the field of interest.

Another important design parameter related to this report is the expendability of the system without significant interference to the existing components. Proper expendability obviously can refer to the addition of a new station to the network without significantly altering and interrupting the operation of the rest of devices. Moreover, expendability refers also to the station unit itself, where the addition of extra components and the removal or replacement of others must not become a significant, complex and time-consuming task or even impossible.

In general there are many design decisions to be made in order to create a proper system for supporting a wide range of weather monitoring sensors and operating according to the objectives set and discussed by Report *1*.

In this introduction we reminded that the core of the whole system relies on a set of sensor devices and all the other components must be chosen in order to accommodate these sensors. With that in mind the rest of this report describes some significant design decisions and implementation based on the *Proposed Design Flow – Stage 1.*

Objectives

Schematic capture: The primary objective of this stage is to create a schematic capture of the station's components. This schematic will illustrate the connections between different components, the communication protocols utilized, and the internal data transportation mechanism. This schematic will serve as a basis for designing an efficient circuitry and structural layout.

Architecture: In addition to the schematic capture, another objective is to develop a comprehensive architecture for the overall weather monitoring system. The architecture provides a high-level description of the processes within the network. It outlines hardware, software, and user-related elements representing them as interconnected black-boxes.

Data collection: Once the layout of the ground-based weather monitoring system is defined, the next objective is to create the essential functionalities of the station. The main focus is on collecting data from sensors and telemetry monitors and transmitting it to an target end-point. Therefore, the objective is to develop the foundational firmware responsible for controlling the sensors, reading telemetry, and transmitting data over the network.

Component placement: The final objective of this initial design stage is to determine the placement of components within the station. Since the basic external structure has already been established from previous design attempts, this step primarily involves deciding the optimal positioning for each component and formulating subsystems to achieve a highly efficient design. Tasks such as power considerations and heat distribution, electronic placement, and sensor integration are necessary to accomplish this objective.

Requirements

Data packets: The collected data from the sensors include air temperature, air humidity, barometric pressure, wind speed and direction, and rainfall rate. Since the data is time-dependent, each measurement requires a timestamp to indicate when it was recorded together with a station unique identifier.

Controller: The weather station network utilizes the Arduino Mega 2560 microcontroller, which is implemented on the corresponding Arduino Mega motherboard as the main system controller. As the project has an academic nature, it is essential that all third-party components utilized, as software, and hardware are open-source and freely accessible for scientific study.

Network: The data transmission and reception in the network are facilitated by a wireless communication network adapter. The adapter in use is the GSM/GPRS Arduino R3 shield, which features an integrated SIM900 microprocessor. To support a broad range of end-users over the application stack, the network utilizes the TCP/IP protocol along with HTTP. While the current implementation relies on cellular network connectivity, the system is designed to be easily modifiable for alternative network adapters such as WiFi or LoRA. Additionally, for hardware component updates and debugging purposes, convenient access to the USB port is ensured.

Power: The electrical power required for system operations is supplied by solar panels. It is crucial that the entire system operates within a dynamic energy consumption limit of 5 W. Careful selection of voltage and current regulators is necessary to provide the appropriate operational level to the respective devices. Specifically, the selected network adapter (GPRS/GSM) requires a current of 1A during data transmission. To ensure uninterrupted operation throughout day and night, including extended periods of cloudy winter days, a rechargeable battery is employed. Additionally, a global hardware shutdown switch must be implemented to cut off power from any source to the main electronics when required without disconnecting the whole supply.

Sampling rate: The weather stations are designed to perform sampling on a real-time or near-real-time basis while considering limitations on transmission and storage load. For collecting valid and highly time-accurate data in an almost real-time monitoring, it is ideal to set a sampling period between 3 to 15 minutes when dealing with meteorological data. This ensures a balance between data frequency and the associated load on transmission and storage systems.

Telemetry packets: In addition to the sensor data, a separate link is used to transmit telemetry measurements. The telemetry data includes the battery voltage, solar panel voltage, adapter current intensity, internal temperature, and controller state. Each telemetry reading is accompanied by a timestamp and a unique station ID to ensure proper referencing.

Sensor placement: The temperature sensor should be placed outside the enclosure, ensuring it is protected from direct contact with water and sunlight. The anemometer and wind direction sensor need to be positioned at a significant height above the earth to obtain accurate measurements, away from potential interruptions. For the rainfall sensor, it should be positioned in an open area with the aperture pointing towards the sky. To achieve more accurate readings, it is recommended to place the barometer outside the enclosure. However, considering the device's sensitivity to environmental conditions, an alternative option is to install it carefully inside in a heat-free corner. This will also allow for measuring the internal system temperature without the need for additional equipment.

Software drivers: To program the microcontroller for autonomous tasks, it is recommended to utilize the respective libraries available for each sensor. Keeping the programming approach simple is a good practice, as the main focus of this project is not on efficient programming or high computational performance. Whenever possible, it is preferable to use specific libraries for each sensor or device designed by the sensor providers. Standard protocols like I2C and SPI provide a robust design perspective and allow for easy replacement or addition of other sensor devices. Additionally, incorporating a debugging mode is important for troubleshooting and fine-tuning the system. Object-oriented programming principles should also be considered for a well-organized codebase. The controller should follow a simple and well-defined Finite State Machine (FSM) approach. Every state must be carefully designed to avoid potential deadlocks and faulty behaviors. It is also recommended to implement safe-mode settings and soft-reset mechanisms for better reliability in case of faults.

Methodology

Sensor selection

AM2315: The AM2315 device integrates a thermistor and a polymeric capacitance humidity sensor into a single unit. It offers precise measurements of temperature and relative atmospheric humidity. The device is equipped with an embedded microcontroller that reads data from both sensors, performs analog-to-digital conversion, and communicates with the master controller through an I2C interface. The following Table 1 provides the basic specifications of the AM2315 device. More information can be found *here*.

Table 1 *- AM2315 device specifications*

Operational Voltage	5 VDC
Operational Current	10 mA
Output	Digital
Interface	I2C (0x5c)
Dimensions	98mm x 16 mm
Humidity Sensing Range	0 – 100% +- 2%
Temperature Sensing Range	-20 – 80 ºC +- 0.1 ºC

Cup Anemometer: The chosen anemometer features a rotating component with three cups mounted to it. These cups are designed to rotate the body of the anemometer when wind particles exert force on them. Every rotation of the sensor creates a circuit by closing a loop and therefore producing an electric pulse. A wind velocity of 2.4 km/hr corresponds to one pulse per second generated by the anemometer. To accurately capture this reading, it is necessary to implement an interrupt-based driver. This driver will ensure that the rotations of the anemometer are precisely counted and processed. More information about the sensor can be found *here.*

Wind direction sensor: The wind direction sensor comprises two main components: a wind-flap and a pointer. When the wind flows, the moving particles push the wind-flap, causing it to align with the direction of the airflow. Consequently, the pointer indicates the opposite direction of the air particle flow. The sensor is composed of a moving part and a static part, both equipped with internal magnets. These magnets align when the moving part responds to the airflow. As the magnets align, an electric circuit is created, generating a distinct voltage output for each direction the device points to. By observing the voltage level from the sensor, we can accurately determine the wind direction. More information about the sensor can be found *here.*

BMP280: The BMP280 is a versatile multisensory device that combines temperature and humidity sensing capabilities (humidity only by the BME280, BMP080) along with barometric pressure measurements. It is equipped with a high-performance embedded processor from BOCH, which ensures accurate readings. The device can be interfaced using either I2C or SPI, making it convenient for integration into various systems due to its compact size. In this project, the main focus is on utilizing the BMP280 for precise barometric pressure measurements. Additionally, the temperature sensor is employed to monitor the internal temperature of the system. Below are the basic specifications of the BMP280 (Table 2). More information about the device can be found *here*.

***Table 2** - BMP280 device specifications*

Operational Voltage	5 VDC
Operational Current	3.6 uA
Output	Digital
Interface	I2C (0x5c)
Dimensions	19mm x 18 mm
Temperature Sensing Range	-40 – 85 ºC +- 0.1 ºC
Pressure Sensing Range	0.3 – 110 kPa

Rainfall sensor: The rainfall rate sensor is of a box with a small aperture in top and a tray internally that empties when it fills. The movement of the tray when it empties triggers a hall sensor and produces a short voltage spick. This voltage spick when detected refers to water volume equal to the volume of the tray. For N number of spicks $N \cdot C_{tray}$ of rain has been collected. By also measuring the time for this amount to be collected the rainfall rate is obtained by simply dividing the amount with the time $N \cdot \frac{C_{tray}}{\Delta t}$. Therefore, an interrupt-based software driver is required for counting the spicks. Each spick for the selected gauge equals to 0.297 mm of water. More information can be found *here*.

Programming the Sensors

Once the sensors are selected, the next step consists of connecting them to the Arduino controller board. For this task a schematic capture for the sensory subsystem is formed. Figure 1 depicts the schematic capture for the sensors. All sensors are connected with the controller board by the use of an external connector. For the rainfall, the anemometer and the wind direction sensor a 10k resistor is utilized respectively (R1, R2, R3) as a pull-up or pull-down resistor. The connector also has two slots for the regulated power input.

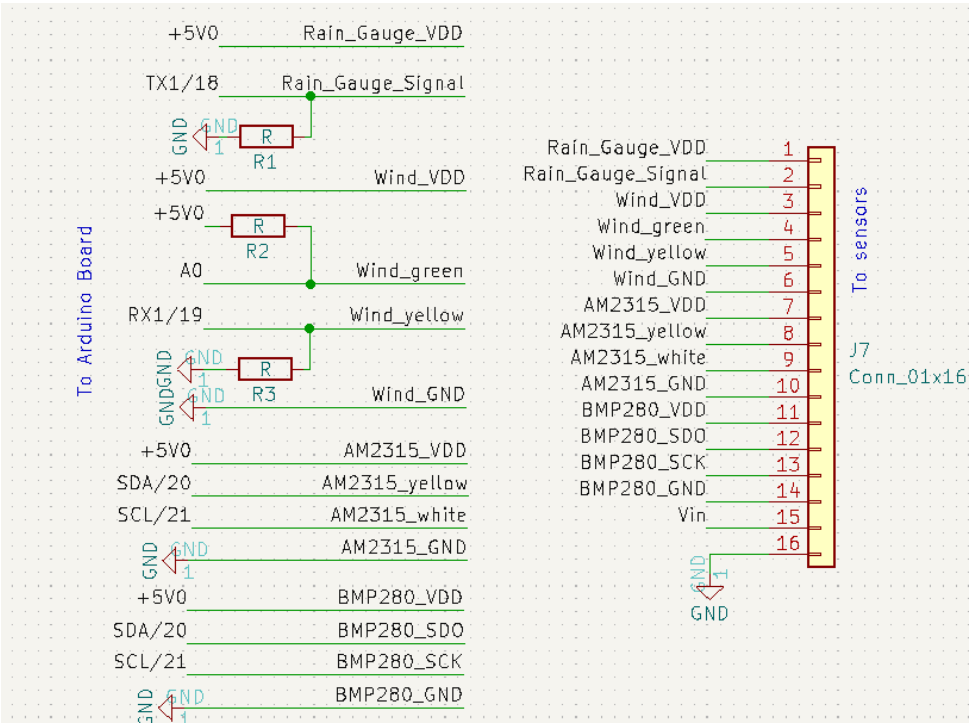

***Figure 1** - The schematic diagram of the sensors*

After the basic circuit is constructed the proper programming of the sensors follows. All functions are implemented as members of the $Weather\_Station\_Sensors$ class. The function description is presented below.

$void\ init()$: this function initializes the I2C bus, the sensors, sets the pin mode and attaches the interrupts to the relative pins.

$float\ get\_Temperature\_AM2315()$: this function returns the air temperature reading in Celsius as recorded by AM2315 sensor.

$float\ get\_Humidity\_AM2315()$: this function returns the relative air humidity reading as recorded by AM2315 sensor.

$float\ get\_Pressure\_BMP280()$: this function returns the barometric pressure reading in kPa as recorded by BMP280 sensor.

$float\ get\_WindSpeed\_AnemometerCup(double\ dt)$: this function returns the wind speed reading in m/s as recorded by the anemometer. The time interval parameter dt is in ms.

$float\ get\_WindDirection\_WindPointer()$: this function returns the wind direction reading in degrees as recorded by the wind direction sensor.

$float\ get\_RainRate\_RainGauge(double\ dt)$: this function returns the rain rate reading in mm/s as recorded by the rain gauge. The time interval parameter dt is in ms.

$void\ get\_AnemometerCupTick()$: this is the interrupt handler function for the interrupt pin related to the anemometer. Every time a new voltage spick is detected from the anemometer an internal counter is increased. The interrupt is activated at the falling edge of the voltage spick.

$void\ get\_RainGaugeTick()$: this is the interrupt handler function for the interrupt pin related to the rain gauge. Every time a new voltage spick is detected from the rain gauge an internal counter is increased. The interrupt is activated at the rising edge of the voltage spick.

Previous versions of the firmware used to have custom made drivers for most of the devices. In this new versions of the project, software optimization and high performance is not of the main target, therefore the already existing libraries for programing some of the devices are utilized. The libraries used for developing the sensor drivers are the *Adafruit_BMP085* for the BMP280 and the *Adafruit_AM2315* for the AM2315 respectively.

Health Monitors

Monitoring the internal state of the device is very important for proper functionality and debugging. A series of internal health monitors are developed for collecting telemetry data. The telemetry data packet contains the bus DC voltage, the bus DC current intensity, the solar panel voltage production, the internal temperature, the current controller mode, and the heartbeat.

Voltage and current monitors: the voltage level monitors are implemented as simple voltage dividers. Each voltage level monitor is implemented to map the input voltage to be monitored to a voltage level that is possible to be read from the controller board. The voltage monitors operate according to the voltage divider principle described by Equation 1. The voltage drop V measured from the voltage source

V_{source} when utilizing a set of resistors R_1 and R_2 connected in series provides the linear voltage input function to the motherboard. Therefore, by inverting this function the V_{source} can be computed.

Equation 1

$$V = V_{source} \cdot \frac{R_2}{R_1 + R_2} \rightarrow V_{source} = \frac{R_1 + R_2}{R_2} \cdot V$$

The controller read these voltage values as analog signals representing them with 10 bits as digital signals utilizing the internal ADC. Therefore, a value in range 0 to 1023 is expected, with the highest value of 1023 representing the reference input voltage. The mapping between the input analog value from the ADC and the voltage measured is done by Equation 2.

Equation 2

$$V = V_{adc} \cdot \frac{V_{ref}}{1024} \rightarrow V_{source} = \frac{R_1 + R_2}{R_2} \cdot V_{adc} \cdot \frac{V_{ref}}{1024}$$

For a specific maximum source voltage, a maximum threshold is to be set in order to protect the controller board from a higher voltage level than is capable of reading through its ports. We set the limit to V_{lim} and with the Equation 3 we can calculate the values of the resistors to be used.

$$V_{lim} = V_{sourceMAX} \cdot \frac{R_2}{R_1 + R_2} \rightarrow \frac{R_2}{R_1 + R_2} = \frac{V_{lim}}{V_{sourceMAX}}$$

Equation 3

$$\frac{R_2}{R_1} = \frac{V_{lim}}{V_{sourceMAX} - V_{lim}}$$

Table 3 presents the resistors selected for a specific maximum voltage level provided from the source with respect to the voltage threshold set to 4 VDC. The results are verified with SPICE simulations.

Table 3 *- The R1 and R2 selected resistance values for 4 VDC threshold and different source voltage levels*

Source Maximum Voltage	R1	R2
24	10 kΩ	2 kΩ
12	10 kΩ	5 kΩ
7	1 kΩ	1.3 kΩ
5	500 Ω	2 kΩ

For measuring the current a simple resistor is utilized as the monitoring device. Specifically, the current I that passes through the resistor R will produce a voltage drop V. By measuring the voltage drop and utilizing the Ohms law the linear current intensity is obtained.

Equation 4

$$I = \frac{V}{R}$$

Voltage V is the result of the voltage divider utilized for measuring the voltage level that outputs the regulator. To utilize this simple method the sensor must be carefully calibrated for valid results.

Calibration consists of reading the raw ADC value of a known current input. If at I_0 the voltage value read from the ADC is, V_{ADC_0} then the current is determined by Equation 5**.**

Equation 5

$$I = \frac{V_{ADC} \cdot I_0}{V_{ADC_0}}$$

Internal temperature monitoring: The temperature of the internal components is obtained by utilizing the temperature sensor integrated to the BMP280 device, since this sensor is placed inside the enclosure.

Software monitors: In addition to the hardware health monitors a series of soft monitors are utilized providing complete monitoring of the stations. These virtual monitors collect information about the controller state and calculate the total operation time of the station from the last time a reset happened, which can also indicate the last time the system crashed, or a service was done.

Programming the Monitors

Once the monitors are designed, the next step consists of connecting them to the main controller board and creating the appropriate software for controlling them. Before delving into the software implementation, the schematic capture for these devices is required. **Figure 2**depicts the schematic for the hardware monitors utilized for the station's health monitoring system.

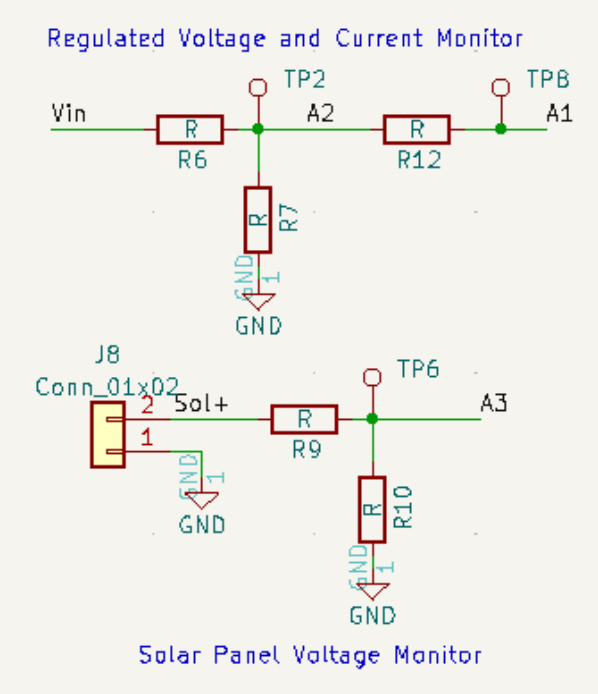

Figure 2 - The health monitoring schematic diagram

The class $Weather\_Station\_Monitors$ contains all the software routines required to read the stations telemetry data. The member functions that was implemented in this stage are presented below.

$void\ init()$: This function initializes the monitors and the variables related to them, sets the pin mode for all the monitor devices.

$float\ get\_PanelVoltage()$: This function is utilized for reading the solar panel voltage level.

$float\ get\_BusVoltage()$: This function is utilized for reading the bus voltage level.

$float\ get\_BusCurrent()$: This function is utilized for reading the bus's current intensity.

$float\ get\_Temperature\_BMP280()$: This function is utilized for reading the internal temperature of the system.

$int\ get\_ControllerState()$: This function is utilized for echoing the current controllers state (and later on the stations command mode).

Programming the Network Adapter

The last fundamental part of the stage 1 design is to program the network adapter for transmitting the registered data from both weather sensors and health monitors. The adapter utilized for the communication module that equips the stations is an Arduino Uno shield with the SIM900 microcontroller. For programming this device AT commands are utilized via a serial interface. The software-serial capabilities of the controller board are utilized for communicating with both Arduino board and adapter at the same time. The AT commands utilized for transmitting data over the network are described in the following Table 4. More information about the AT commands can be found *here*.

Table 4 *– AT commands utilized for transmitting data over GPRS/ GSM with HTTP application protocol*

Command	Description
`AT+CFUN=1`	Set the device with full functionalities
`AT+CGATT=1`	Attach the device to the closest cellular packet domain service
`AT+SAPBR=3,1,"Contype","GPRS"`	Enable the connection type settings on GPRS
`AT+SAPBR=3,1,"APN", "internet"`	Set the current APN. For Cosmote Hellas is "Internet", without password nor user-name
`AT+SAPBR=1,1`	Enable the GPRS module
`AT+SAPBR=2,1`	Get the public IP
`AT+CLTS=1`	Synchronize with the local network time
`AT+HTTPINIT`	Initialize the http transaction mode
`AT+HTTPPARA="CID",1`	Set the CID
`AT+HTTPPARA="URL", \<URL>\<path>\<values>\"`	Make a http request for access to the target web server
`AT+HTTPACTION=0`	Set http post action and establish connection with the target web server
`AT+HTTPTERM`	Terminate the http link
`AT+CIPSHUT`	Shut down the connection

All the control of the networking in each station is done via these AT commands. The class $Weather\_Station\_Network$ contains the member functions required for the control of the GPRS/ GSM shield module. These functions implemented in this first design stage are described below.

$void\ init()$: this function initializes the GPRS module. Sets the power pin and enables it in order to activate the internal network settings.

$void\ GPRS\_ATCommand(String\ AT\_cmd)$: this function is utilized to send the commands from the Arduino board to the adapter module via serial communication.

$void\ GPRS\_enable()$: this function sends the commands required for enabling the GPRS module.

$void\_transmitHTTP(String\ URL, String\ path, String\ value)$: this function is utilized in order to transmit HTTP data values from the stations to the web server described by their respective URL to the specific directory defined by the path parameter. The function sends the value of the parameter $value$ to the corresponding $URL/path$ directory.

In addition to the data transmission link the system must be also capable of receiving data over the network in the form of command packets. More information about the design and implementation of the command-related link will be given in later reports, since is the topic for another design procedure stage.

Power Supply

A very important subsystem for the stations to operate properly is the power generation, storage, and distribution subsystem. Energy generation is based on solar radiation, where a solar panel is utilized in order to provide the required energy by the day. The redundant energy is stored in a proper battery. This battery can provide continuous energy to the station for at least 2 days per full charge at 12VDC. A first level voltage regulator is utilized in order to move the energy from the solar array to the battery and from the battery to the station load respectively. The load output voltage passes through another regulator for enhancing the current and reducing the voltage level to no more than 7 VDC. The minimum current that the system requires in order to operate properly is 1A. Although 1A seems to have a high current intensity it is utilized only when the GPRS module transmits data. In idle state the whole structure does not drain more than 500 mA of current in total. A hardware switch is added to the stations electronic for cutting the power supply if needed. A led indicator lights up when the main electronics of the station are powered on. Figure 3 depicts the schematic capture for the power management, storage and distribution system.

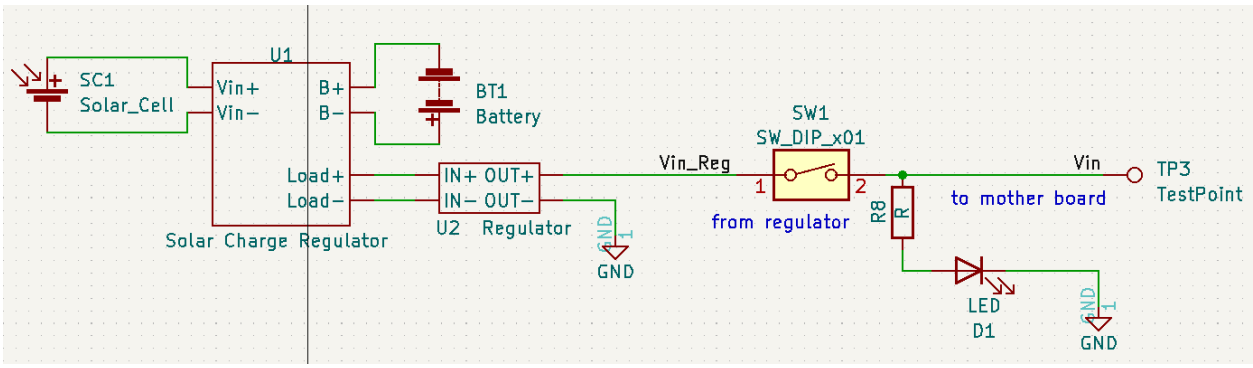

Figure 3 *– The power management, storage and distribution system*

System Architecture

Figure 4 depicts the high-level architecture diagram that describes the weather station devices as designed. The architecture approach can be divided into four main subsystems: the power related subsystem, the controller logic, the sensors, and the network adapter. It is important to note that protocols such as SPI, I2S and UART are utilized for most of the controller-peripheral connectivity. In addition to these protocols USB is utilized for programming and debugging the system, which is not visible in the diagram.

Data transmission is done via the TCP/IP data packet transfer protocol over the HTTP application protocol utilizing three different links. The first consists of the sensory data packets, the second for the telemetry data packets as described previously in this report and the third one consists of the telecommand data packets, which will be the topic for another development report.

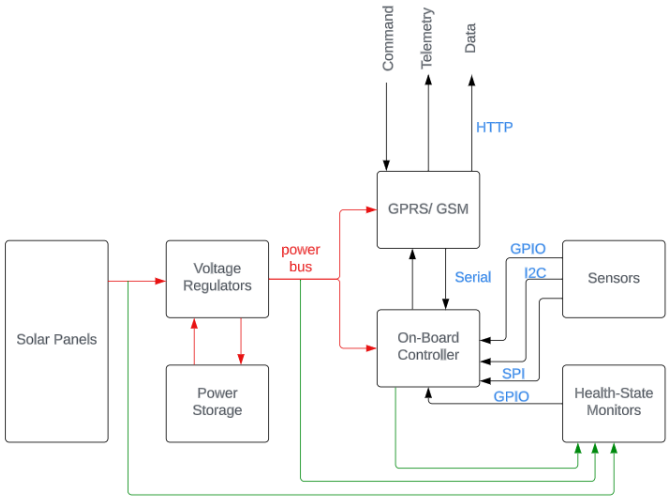

Figure 4 *– System architectural diagram*

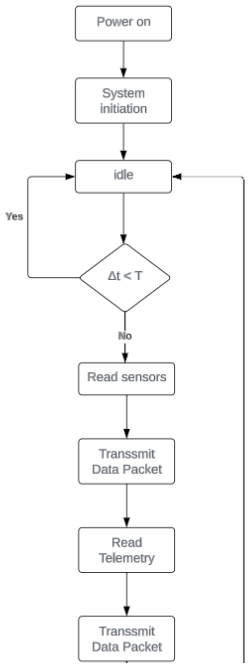

Figure 5 *– Main control FSM*

Another significant part of the system is the operation logic for data collection and transmission. The following Figure 5 depicts the station's internal FSM high-level logic for the aforementioned task. When the station is powered on and all the modules are initiated properly then it records weather data over constant time intervals (controlled) together with telemetry and transmits these data through the appropriate link to an end data-handler.

Specifically, every T time intervals which is set as one of the three modes: FULL MODE where T = 3 min, NORMAL MODE where T = 5 min, SLOW MODE where T = 10 min and POWER SAVING MODE where T = 30 min.

The same principle is followed also for telemetry data recording where the same modes exist respectively for the telemetry data. In addition, the weather station controller holds a register for recording the heartbeat. Every minute the registers content is increased by one and in that way, we obtain information about the operation time from the last reset that happened to the system.

The data packets collected are transmitted to separate data paths over HTTP. Therefore, the backend subsystem must be able to provide two different (and one more for the telecommands) directories/ absolute paths / URL's in order to handle differently the generated data. The following lists present the data contained in a data backet concatenated to the respective URL.

Sensor Data : [station_id, temperature, humidity, pressure, wind speed, wind direction, rainfall]

Telemetry Data : [station_id, internal temperature, bus voltage, bus current, solar array voltage, heartbeat, mode]

The design and implementation of the backend data handling subsystem is one of the topics described by a later progress report. However, Figure 6 depicts the abstract architectural approach of that system and sets a baseline for choosing between different available design tools and layouts. Three main components are depicted in that diagram:

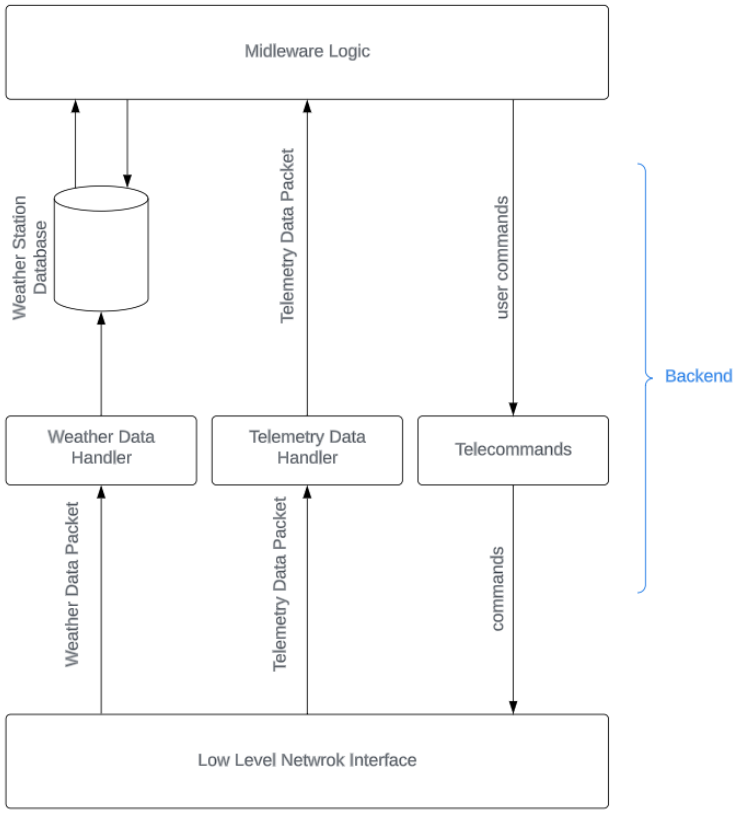

Figure 6 *– The backend black-box architecture and the interconnection between the middleware and hardware with the backend components.*