**System Requirements  
Specification (SRS)  
  
PV wireless monitoring**

|  |
| --- |
| Using this Document – System Integration Benefits |
| The System Requirements Specification (SRS) document describes all data, functional and behavioral requirements of the hardware and software under production or development. |

**Table of Contents**

[1. Introduction 4](#_Toc475354427)

[1.1. Purpose 4](#_Toc475354428)

[1.2. Scope 4](#_Toc475354429)

[1.3. Context 4](#_Toc475354430)

[1.4. Organization 5](#_Toc475354431)

[3. Overall Description 7](#_Toc475354432)

[3.1. Product perspective 7](#_Toc475354433)

[3.2. Product features 9](#_Toc475354434)

[3.3. User classes and characteristics 9](#_Toc475354435)

[3.4. Operating environment 9](#_Toc475354436)

[3.5. Design and implementation constraints 9](#_Toc475354437)

[3.6. User documentation requirements 10](#_Toc475354438)

[3.7. Assumptions and dependencies 10](#_Toc475354439)

[3.8 System requirements 10](#_Toc475354440)

[4. Specific Requirements 11](#_Toc475354441)

[4.1. Operations 11](#_Toc475354442)

[4.2. Communication 11](#_Toc475354443)

[4.3. Safety/reliability 11](#_Toc475354444)

[5. Data Model and Description 12](#_Toc475354445)

[5.1. Use Case Diagram 12](#_Toc475354446)

[5.2. Domain Model 12](#_Toc475354447)

[5.3. Sequence Diagrams 12](#_Toc475354448)

[5.4. State Diagram 12](#_Toc475354449)

[5.5. Prototype 12](#_Toc475354450)

[Prototype Operation 12](#_Toc475354451)

[Sample Scenario 12](#_Toc475354452)

[6. References 13](#_Toc475354453)

# Introduction

This section presents an overview of the SRS to help the reader understand how the document is organized and how to read and interpret it. The System Requirements Specification (SRS) document describes all data, functional and behavioral requirements of the software.

Renewable Energy System development are inherent to the power electronics development. In the scope of the microgeneration project of SYSTEC-ENERGY – Smart Energy Systems and Technologies – a distributed PV harvesting with multiple power converters is in development. Despite of the standalone behavior of the PV power converters, a monitoring subsystem is expected to be implemented.

## Purpose

The purpose of this document is to describe the specific requirements for the Wireless PV monitoring system. This document is intended to provide an unambiguous, concise, and complete list of requirements to help design the Gateway and the nodes attached to each PV power converter. This document will include constraints and show how to use the system.

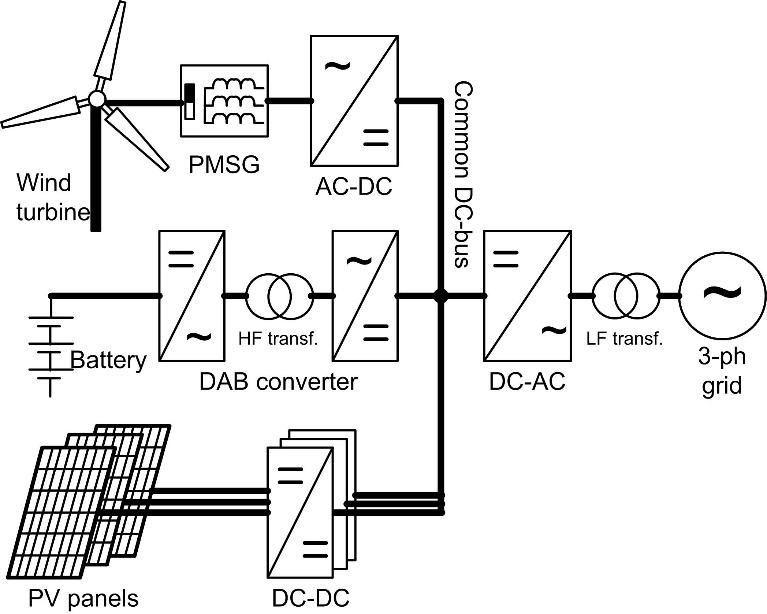
## Scope

The Wireless PV monitoring system is a system intended to acquire the data and behavior status of each PV power converter.

## Context

Place the software in a business or product line context with the aim for the reader to understand the 'big picture'. Strategic issues relevant to context are discussed.

This work is inserted in the scope of a microgeneration project of the SYSTEC-ENERGY Research Unit. Figure 1 presents the main architecture of this project. Currently, several work has been done in this project to implement a monitoring subsystem. This work focuses on data collection from each power converter and also on its control by sending references and actions. At the moment of this proposal, no work has been done to implement the monitoring feature in the PV converters. This way, a wireless network implementation is proposed to monitor the PV power converters.



*Figure 1 - Main architecture of microgeneration project.*

The PV converter uses a non-isolated high gain DC/DC topology and has a PIC32 microcontroller which implements a MPPT algorithm in the control loop. It is currently possible to connect a device with wireless capabilities to the PV power converter. The converter is able to provide power and data. The data collected from the monitoring device is sent to an aggregator node for post processing and data storage. Currently, several work has already been done to have a reliable wired data aggregator/concentrator. This subsystem is also responsible for sending user commands to the power converters and presenting the user interface in a webpage remotely accessed via a web browser.

## Organization

The rest of the document is broken down into six sections.

Section 2 contains the overall description. In this section, the document will provide information about the system functions. Also included in this section is a description of the constraints needed for safety, as well as any assumptions made in the user characteristics and dependencies.

Section 3 has a hierarchical enumerated list of the requirements the system must fulfill.

Section 4 contains various models that specify the bridge between application domain and machine domain. In this section, UML diagrams and their respective detailed explanations are used to visualize and demonstrate the expected behavior of the system.

Section 5 gives an overview of the prototype, describes how to access it and run it, as well as examples of sample scenarios.

Section 6 gives a list of all the references used to compile this document.

Section 7 gives a point of contact if more information is required.



# Overall Description

Present here a high-level overview of the product being specified and the environment in which it will be used, the anticipated users of the product, and the known constraints, assumptions, and dependencies.

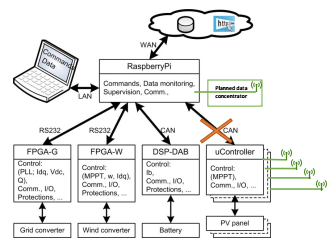
The system to be developed is a wireless monitoring system capable of acquiring data from several PV converters, supporting the mechanical and EMI influences from the environment and being integrated to the current monitoring infrastructure.

## Product perspective

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

The wireless feature is intended to be added to the monitoring infrastructure supported in a System-on-Chip computer – namely a Raspberry Pi 2 – and the PV power converters.

The diagram in the following figure presents the overall perspective of the development expected by this SRS



The major requirements of the microgeneration monitoring system are the data acquisition and storage, having a website interface that might be easily accessed from the internet (with all the security constrains guaranteed). The relevant data must be identified, collected and transmitted using regular communication channels (RS232, SPI, CAN, etc).

## Product features

Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.

This addition to the microgeneration project has two main objectives:

1. Extend the monitoring feature to the PV power converters
2. Implement and demonstrate a feasible wireless communication among the PV power converters and the head of the system

## 3.3. User classes and characteristics

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.

No user classes are expected in this communication module. The user interaction is defined by the upper layer of this system.

## 3.4. Operating environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

This system will be implemented in a cabinet with EMI constrains caused by the power converters. This SRS will include the specification of the hardware platform.

## 3.5. Design and implementation constraints

Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).

## 3.6. User documentation requirements

List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.

## 3.7. Assumptions and dependencies

List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change.

Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).

## 3.8 System requirements

Describe the minimum system requirements for this product (e.g. platform version, disk space, RAM, etc.)

# Specific Requirements

There are a large number of requirements that the wireless PV monitoring must meet. There requirements can be split up into requirements for operation, communication and safety/reliability of the data, as detailed below.

* 1. **Operations**
     1. The system must collect data from more than 1 PV converter (up to 10 PV converters)
     2. Each wireless node should be powered by 5V
     3. The wireless concentrator should be powered by 5V
  2. **Communication**
     1. Each wireless node should have a serial communication
     2. The wireless concentrator should have a serial communication
     3. The wireless concentrator should implement the microgen serial protocol
     4. A wireless concentrator must collect data from all the nodes in less than 10 seg
     5. All wireless nodes must send data to the wireless concentrator upon a request.
  3. **Safety/reliability**
     1. All data shall be sampled and hold upon a SYNC command
     2. The data jitter must be .

# Data Model and Description

This section presents various UML diagram to help showcase the functions of this PV wireless monitoring.

## Use Case Diagram

A Use Case Diagram visually captures functions of the system. It shows the interactions between external actors on internal functions of the system. The system boundary helps to define the external actors and the internal functions. The extends signify another use case that is similar, with added functionality. Moreover, use case templates are provided to give detailed information for each use case of the system in the diagram. These templates show which actors are involved for a specific use case and shows the system requirement it satisfies.

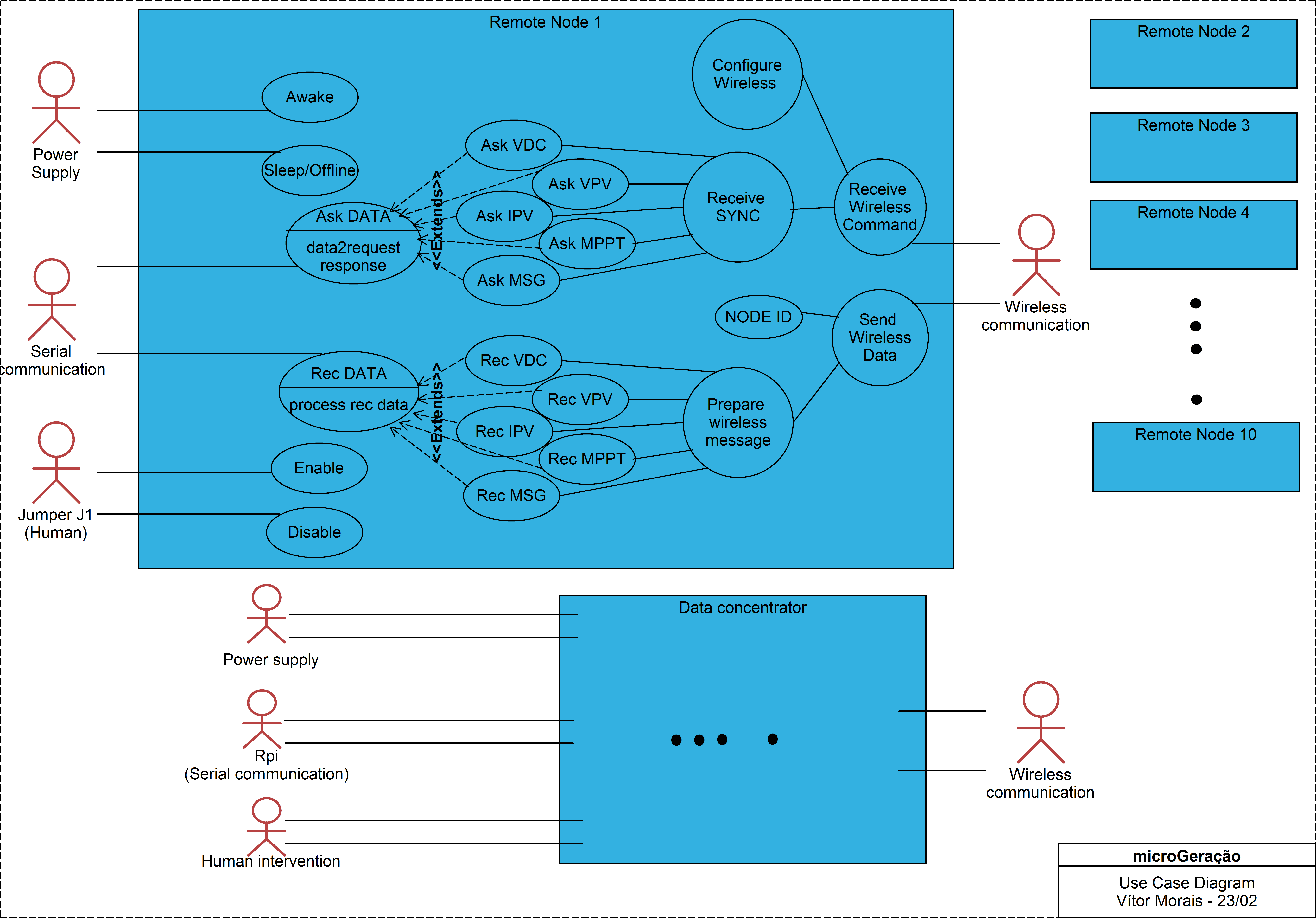


Figure 2 - Use Case Diagram – node

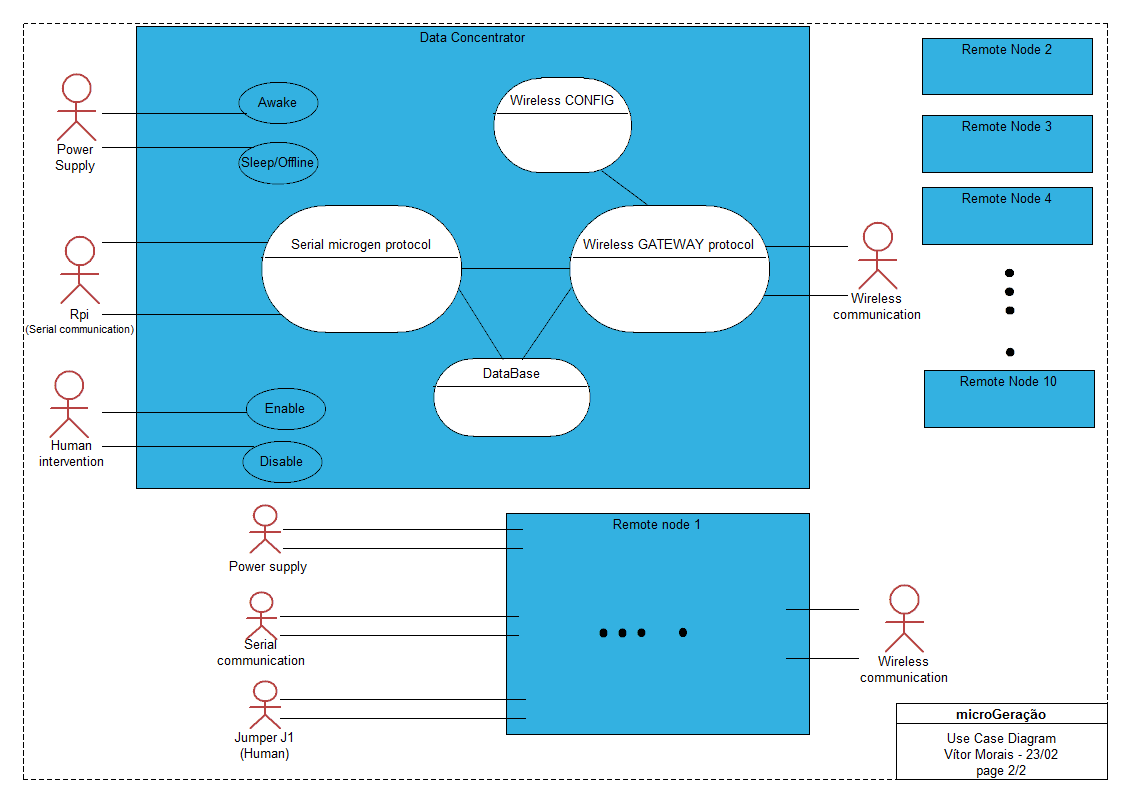


Figure 3 - Use Case Diagram - Data Concentrator

|  |  |
| --- | --- |
| **Use case** | Awake |
| **Actors** | Power Supply |
| **Description** | The system should start up when the system voltage threshold is  above 4.5 volts. |
| **Type** | Primary |
| **Cross-Refs** |  |
| **Use Cases** | All |

|  |  |
| --- | --- |
| **Use case** | Sleep/Offline |
| **Actors** | Power Supply |
| **Description** | The system should go into sleep mode or offline mode when the system voltage threshold is below 3 volts. |
| **Type** | Primary |
| **Cross-Refs** |  |
| **Use Cases** | All but awake |

|  |  |
| --- | --- |
| **Use case** | Enable |
| **Actors** | Human Intervention |
| **Description** | The system should restart and go to enable mode when the user presses a button (or jumper) |
| **Type** | Optional |
| **Cross-Refs** |  |
| **Use Cases** | All but disable |

|  |  |
| --- | --- |
| **Use case** | Disable |
| **Actors** | Human Intervention |
| **Description** | The system should restart and go to disable mode when the user presses a button (or jumper) |
| **Type** | Optional |
| **Cross-Refs** |  |
| **Use Cases** | All but enable |

|  |  |
| --- | --- |
| **Use case** | Serial microgen protocol |
| **Actors** | Rpi (serial communication) |
| **Description** | The system should execute the serial microgen protocol |
| **Type** | Essential |
| **Cross-Refs** |  |
| **Use Cases** | All |

|  |  |
| --- | --- |
| **Use case** | DataBase |
| **Actors** |  |
| **Description** | The system should stores the data and command between serial and wireless |
| **Type** | Essential |
| **Cross-Refs** |  |
| **Use Cases** | All |

|  |  |
| --- | --- |
| **Use case** | Wireless CONFIG |
| **Actors** |  |
| **Description** | The system should have stored all the configuration of remote nodes |
| **Type** | Primary |
| **Cross-Refs** |  |
| **Use Cases** | All but disable |

|  |  |
| --- | --- |
| **Use case** | Wireless GATEWAY protocol |
| **Actors** | Wireless communication (with nodes) |
| **Description** | The system should send a SYNC signal to remote nodes;  The system should receive all data from remote nodes |
| **Type** | Primary |
| **Cross-Refs** |  |
| **Use Cases** | All but disable |

## Domain Model

The following section contains a domain model and a data dictionary for the system.

In the domain model, boxes represent logical objects within the system. Within the boxes, lines preceded with a “­” represent individual pieces of information that that particular object knows, and lines preceded with a “+” indicate actions a given object can perform. Lines connecting the objects show relationships among the boxes, and include a descriptive name and numbers representing how many of each object is involved.

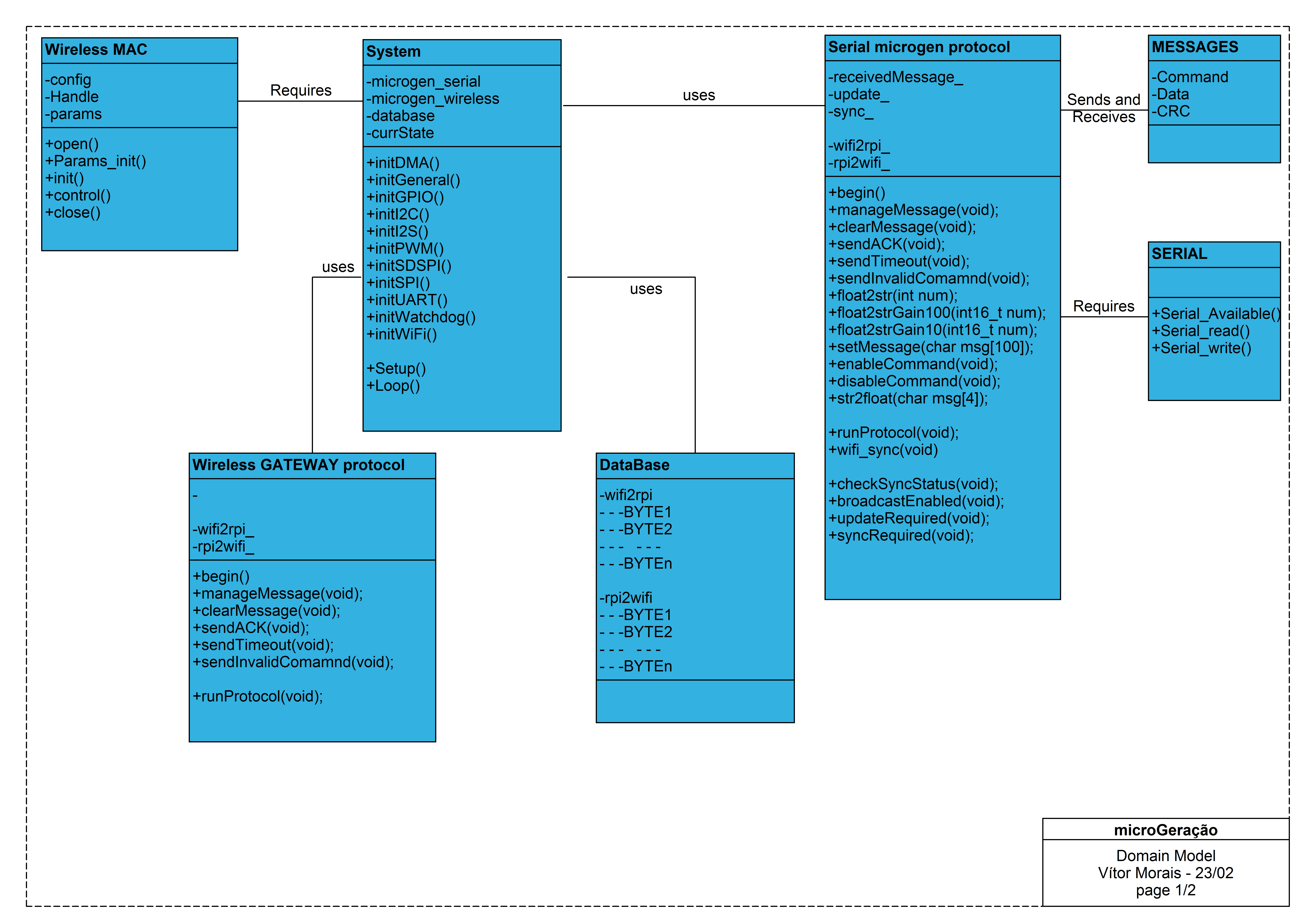


Figure 4 - Domain model diagram – Concentrator

**Data dictionary**

The data dictionary provides additional detail for every element in the domain model.

|  |  |  |
| --- | --- | --- |
| **Element Name** |  | **Description** |
| System |  | Contains the core functionality of the wireless concentrator node such as programming, protocol dependencies, peripheral and communication ports |
| **Attributes** |  |  |
|  | currState | Current state of the system: setup, loop\_waitForSerial, loop\_waitForWifi, loop\_updateDB |
|  | microgen\_serial | Object of serial communication protocol between the data concentrator and the raspberry pi |
|  | microgen\_wireless | Object of wireless communication protocol between data concentrator and wifi nodes |
|  | Database | Object of data storage |
| **Operations** |  |  |
|  | +initDMA() | Initialize board specific DMA settings |
|  | +initGeneral() | This function initializes the general board specific settings. |
|  | +initGPIO() | Initialize board specific GPIO settings |
|  | +initI2C() | Initialize board specific I2C settings |
|  | +initI2S() | Initialize board specific I2S settings |
|  | +initPWM() | Initialize board specific PWM settings |
|  | +initSDSPI() | Initialize board specific SDSPI settings |
|  | +initSPI() | Initialize board specific SPI settings |
|  | +initUART() | Initialize board specific UART settings |
|  | +initWatchdog() | Initialize board specific Watchdog settings |
|  | +initWiFi() | Initialize board specific WiFi settings |
|  |  |  |
|  | +Setup() | General setup of the system |
|  | +Loop() | General loop of the system |

|  |  |  |
| --- | --- | --- |
| **Element Name** |  | **Description** |
| DataBase |  | Contains the data to be exchanged between serial and wireless |
| **Attributes** |  |  |
|  | wifi2rpi : struct | Data exchanged from wifi nodes to rpi |
|  | rpi2wifi : struct | Data exchanged from rpi to wifi nodes |
| **Operations** |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Element Name** |  | **Description** |
| SERIAL |  | Contains the serial Hardware layer (HAL) |
| **Attributes** |  |  |
|  |  |  |
| **Operations** |  |  |
|  | Serial\_Available() | Checks if serial read buffer is available |
|  | Serial\_read() | Reads incoming byte |
|  | Serial\_write() | Writes message to serial port |
| **Element Name** |  | **Description** |
| Serial microgen protocol |  | Contains the class description of the serial protocol implemented in the microgen project |
| **Attributes** |  |  |
|  | currState | Current state of the protocol: setup, run |
|  | receivedMessage: array | Stores temporarily the received message |
|  | update : boolean | Flag: if enabled then updates the database for the new data |
|  | sync : boolean | Flag: if enabled then ask device for new messages |
|  |  |  |
|  | wifi2rpi\_ | Database: stores the data received via wifi to be sent to rpi |
|  | rpi2wifi\_ | Database: stores the data (commands) sent by rpi to the wifi |
| **Attributes** |  |  |
|  | begin() | Initializes the protocol |
|  | +manageMessage(void); | Processes the received message |
|  | +clearMessage(void); | Clears the received message |
|  | +sendACK(void); | Sends via serial a ACK message |
|  | +sendTimeout(void); | Sends via serial a TIMEOUT message |
|  | +sendInvalidComamnd(void); | Sends via serial a INVALID message |
|  | +float2str(int num); | Converts a number to ASCII and sends it via serial |
|  | +float2strGain100(int16\_t num); | Converts a number with 2 decimals to ASCII and sends it via serial |
|  | +float2strGain10(int16\_t num); | Converts a number with 1 decimals to ASCII and sends it via serial |
|  | +setMessage(char msg[100]); | Sets message of receivedMessage array |
|  | +enableCommand(void); | Sets command flag to COMMAND\_ENA |
|  | +disableCommand(void); | Sets command flag to COMMAND\_DIS |
|  | +str2float(char msg[4]); | Converts a received command value from ASCII to float |
|  |  |  |
|  | +runProtocol(void); | Waits and receives serial message |
|  | +wifi\_sync(void) | Enables system for SYNC (or broadcast with remote nodes) |
|  |  |  |
|  | +checkSyncStatus(void); | Disables Sync flag; returns SYNC flag status |
|  | +broadcastEnabled(void); | NOT USED in wifi node |
|  | +updateRequired(void); | Disables UPDATE flag; returns UPDATE flag status |

|  |  |  |
| --- | --- | --- |
| **Element Name** |  | **Description** |
| Messages |  | Data structure of the exchanged serial messages: #DATA#CRC# |
| **Attributes** |  |  |
|  | Data | Data exchanged |
|  | CRC | Number of bytes of DATA |
|  | Command | Optional: it assumes the format {COMMANDVALUE} with 4 ASCII number bytes for COMMAND and 4 ASCII number bytes for data |
| **Operations** |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Element Name** |  | **Description** |
| Wireless GATEWAY protocol |  | Contains the class description of the wireless protocol to be implemented |
| **Attributes** |  |  |
|  | currState | Current state of the protocol: setup, run |
|  |  |  |
|  | wifi2rpi\_ | Database: stores the data received via wifi to be sent to rpi |
|  | rpi2wifi\_ | Database: stores the data (commands) sent by rpi to the wifi |
| **Attributes** |  |  |
|  | begin() | Initializes the protocol |
|  | +manageMessage(void); | Processes the received wifi message |
|  | +clearMessage(void); | Clears the received message |
|  | +sendACK(void); | Sends via wifi to a node a ACK message |
|  | +sendTimeout(void); | Sends via wifi to a node a TIMEOUT message |
|  | +sendInvalidComamnd(void); | Sends via wifi to a node a INVALID message |
|  |  |  |
|  | +runProtocol(void); | Waits and receives serial message |

|  |  |  |
| --- | --- | --- |
| **Element Name** |  | **Description** |
| Wireless MAC |  | Contains the wifi Hardware layer (HAL) |
| **Attributes** |  |  |
|  | config | Configuration parameters |
|  | Handle | Object handler |
|  | Params | Object parameters |
| **Attributes** |  |  |
|  | +open() | Function to initialize a given WiFi peripheral |
|  | +Params\_init() | Function to initialize the WiFi\_Params structure to its defaults |
|  | +init() | Function to initialize the WiFi module |
|  | +control() | Function performs implementation specific features on a given WiFi\_Handle. |
|  | +close() | Function to close a WiFi peripheral specified by the WiFi handle |

**Remote node vs data concentrator**

The code implemented in each remote node should be similar to the data concentrator. On the serial communication, the microgen protocol should be implemented in both the PV converter and the remote node. On the wifi communication, the remote node must implement a Medium Access Control similar based on a producer-consumer mesh network.

## Sequence Diagrams

## State Diagram

## Prototype

### Prototype Operation

### Sample Scenario

DATA TRANSFER

REMAINING SUPPORTED SCENARIOS:

# References