Project Acronym:	PhasmaFOOD
Grant Agreement number:	732541 (H2020-ICT-2016-1 - RIA)
Project Full Title:	Portable photonic miniaturised smart system for on-the-spot food quality sensing

DELIVERABLE

Deliverable Number	**D5.3**
Deliverable Name	**Implementation of embedded software for PhasmaFOOD smart food analysis platform**
Dissemination level	PUBLIC
Type of Document	REPORT
Contractual date of delivery	M16
Deliverable Leader	FUB
Status & version	V1.8
WP / Task responsible	WP5/T5.3
Keywords:	Embedded software, image processing, communication APIs
Abstract (few lines):	This deliverable reports on the implementation of embedded software for the PhasmaFOOD smart food analysis device. The communication APIs in the embedded system are described, as well as preprocessing functionalities on the embedded microprocessor.

Deliverable Leader:	Benedikt Groß (FUB)
Contributors:	Paraskevas Bourgos (WINGS), Konstantinos Tsoumanis (WINGS), Panagiotis Vlaheas (WINGS), Dimitris Kelaidonis (WINGS), Aimilia Bantouna (WINGS), Konstantinos Petsas (WINGS), Orestis Liakopoulos (WINGS), Nelly Giannopoulou (WINGS), Apostolos Voulkidis (WINGS), Konstantinos Trichias (WINGS), Ioannis Tzanetis (WINGS), Aspa Skalidi (WINGS), Evangelia Tzifa (WINGS), Milenko Tosic (VLF), Benedikt Groß

Reviewers:	(FUB), Annamaria Gerardino (CNR), Francesca Bertani (CNR), Susanne Hintschich (IPMS), Peter Reinig (IPMS), Ioannis Daskalopoulos (INTRA), George Tsoumanis (INTRA), Eugenio Martinelli (UTOV) Paraskevas Bourgos (WINGS), Konstantinos Tsoumanis (WINGS), Peter Reinig (IPMS), Susanne Hintschich (IPMS)
Approved by:	George Koutalieris (INTRA)

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

Executive Summary

The current deliverable, D5.3, reports on the embedded software for the PhasmaFOOD sensing device. It describes the communication APIs between the sensing and main electronic subsystems, the communication API between the embedded device and the mobile app, outlines the embedded data preprocessing functionalities and reports on the implementation of the first version of the embedded software and its integration into the first PhasmaFOOD software prototype.

Section 1 gives an overview over the PhasmaFOOD software architecture and the role of the embedded software within this architecture. The software components of the embedded system are described in section 2 together with the communication APIs between the sensing components and the main electronic subsystem, as well as from the embedded device to the mobile app. Section 3 reports on embedded data preprocessing for each of the PhasmaFOOD device's sensors. Section 4 deals with the implementation of the first software prototype and lists the requirements from deliverable D1.2 [1], together with a description of the current status of each requirement in the embedded software. This deliverable concludes with a brief summary and an outlook over the next development steps.

Document History			
Version	**Date**	**Contributor(s)**	**Description**
1.0	09/03/2018	FUB	ToC first draft
1.1	19/03/2018	FUB, WINGS, VLF	ToC updated
1.2	02/04/2018	WINGS	WINGS content
1.3	17/04/2018	VLF	VLF content
1.4	21/04/2018	FUB	First version for the internal review
1.5	26/04/2018	WINGS, IPMS	Internal review, section 3.2
1.6	27/04/2018	IPMS, CNR	Internal review 2, section 3.1
1.7	30/04/2018	INTRA	Update section 4
1.8	30/04/2018	WINGS	Format changes

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

Table of Contents

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

Table of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Acronym	Title
ADC	Analog-to-Digital Converter
API	Application Programming Interface
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
D	Deliverable report
DA	Data Analysis
GPIO	General Purpose Input-Output
I2C	Inter-Integrated Circuit protocol
IMU	Inertial Measurement System
JSON	Java Script Object Notation
ML	Machine Learning
Mb	Megabyte
NIR	Near Infra-Red light
PoC	Proof-of-Concept
RAM	Random Access Memory
RGB	Red Green Blue color space
SNR	Signal-to-Noise Ratio
SOP	Standard Operating Procedure
SPI	Serial Peripheral Interface
SSL	Secure Socket Layer
SW	SoftWare
UML	Unified Modeling Language
USB	Universal Serial Bus
UV	Ultra-Violet light
VIS	VISible light
WP	Work Package

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

1 Overview

1.1 PhasmaFOOD software architecture

The PhasmaFOOD system comprises three distinctive layers integrated together to provide scalable sensory and contextual data collection, as well as processing and analysis for food analysis related decision making. These three layers include: a portable sensing device, an end user mobile device (i.e. smartphone and tablet) and a cloud platform. The PhasmaFOOD software architecture is defined with the goal of taking full advantage of the envisioned system architecture. On the sensing device level, the embedded software integrates physical sensors, data preprocessing, control logic and a communication interface to the end user's mobile device. The PhasmaFOOD mobile application provides interface between the portable sensing device, as the primary source of sensory data, and the cloud platform which is the primary sink for all data streams. The mobile app also provides the interface towards the end user for configuring the food analysis process/measurements as well as visualizing and interpreting analysis results. The PhasmaFOOD mobile application may perform additional data processing in order to ensure that only relevant data is delivered to the cloud platform. Finally, the PhasmaFOOD cloud platform is the focal point of all sensory and contextual data coming from the portable sensing device, the mobile app and, if necessary, 3rd party data sources. The cloud platform is where the collected data is analyzed in depth. The platform hosts a data warehouse for all sensory and contextual data sets necessary for performing food analysis and deriving results. From this data warehouse data marts are derived for training different Machine Learning (ML)/Data Analysis (DA) models for realization of project use cases. Decision making algorithms (based on the trained ML models) are hosted on the cloud platform as well. Reactive (based on end user measurements) and proactive (strategic and based on data collected over time) decision making is performed based on the results of data analysis. **Error! Reference source not found.** shows a high level overview of the PhasmaFOOD software architecture. A detailed description of the PhasmaFOOD software architecture can be found in Deliverable Report D2.1 [2]. The software requirements and functionalities were collected in Deliverable Report D1.2 [1].

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

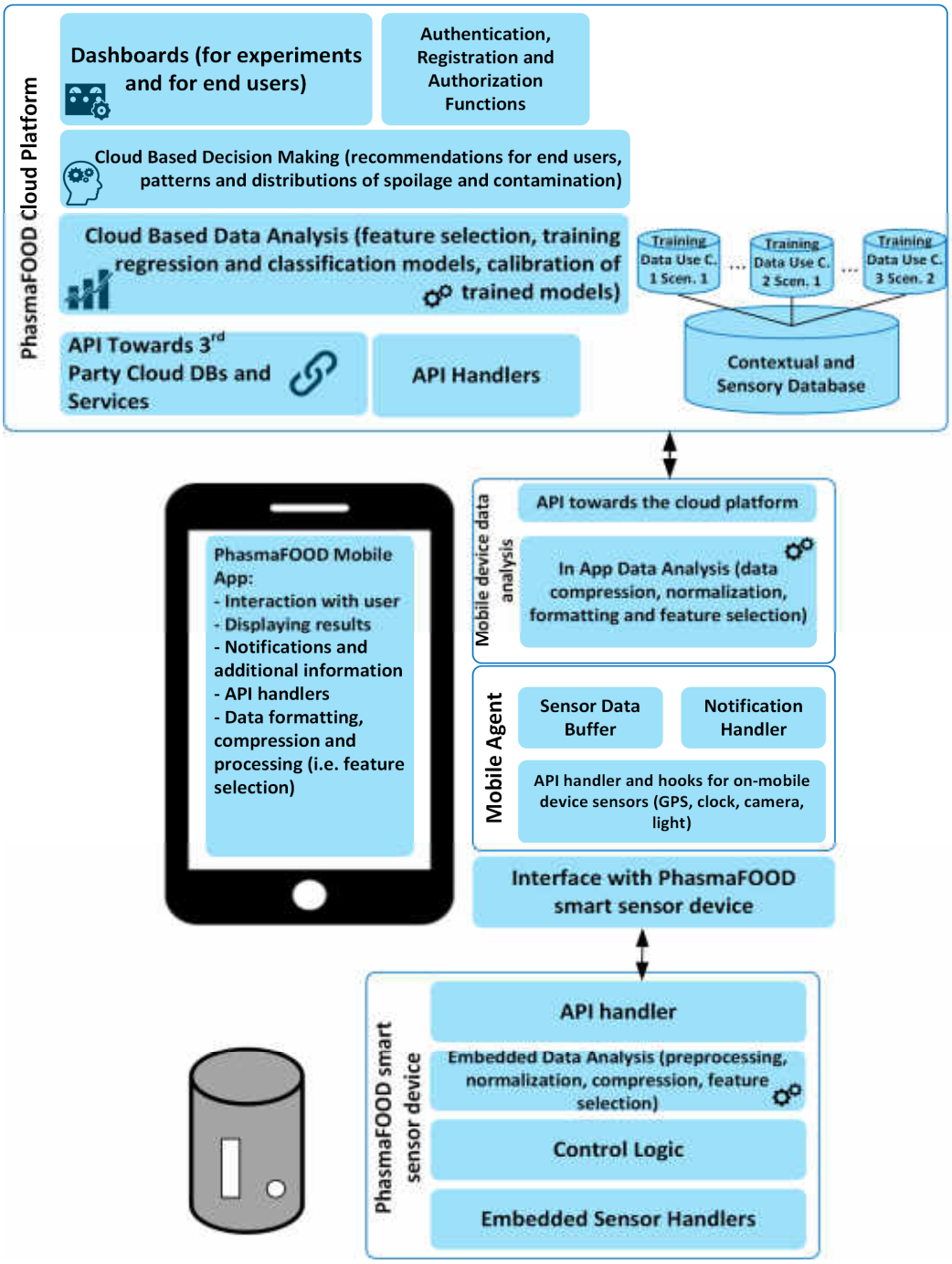

Figure 1 - PhasmaFOOD SW architecture overview

1.2 Embedded system as part of the PhasmaFOOD software architecture

The embedded software runs on the PhasmaFOOD sensing device and directly interfaces with the physical sensors on one side and prepares data for use case centric analysis performed throughout the PhasmaFOOD system architecture. The key modules, comprising the PhasmaFOOD embedded SW, are listed in Table 1.

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

Table 1 - PhasmaFOOD embedded software main modules

Embedded sensor handlers	Handler for UV-VIS sensor	Manages the operation of the UV-VIS spectrometer
	Handler for NIR sensor	Manages the operation of the NIR spectrometer
	CMOS camera Handler	Manages the operation of the integrated CMOS camera
	Communication modules	The I2C and SPI communication interfaces handlers and multiplexers that enable data readings from all sensors (i.e. spectrometers and environmental sensors) and interface with the camera
	Lighting source handler	Manages the lighting (selection of light source and intensity) based on selected measurement mode
	Handler for motion sensor	Collects readings from built-in motion sensor indicating high oscillations or unstable handling of the sensing device during measurements
	Handler for temperature sensor	Measures the temperature of the integrated environment
Control logic	Energy management	Manages battery charging, consumption and power saving
	Display management	Drives the built-in display providing device status, notifications and analysis results
	Measurement logic	The logic on how to configure and operate different combination of lighting sources and built-in sensors for each project use case and scenario
	Trigger management	The SW logic that measures different system status indicators and sets triggers for corrective actions (i.e. low battery, high oscillations, high temperature)
	Notifications	Sends notifications/triggers about high instability, high temperature, low battery level, finished measurement
	System update handler	Manages implementation of the updated embedded firmware
Embedded data preprocessing	Data compression	Performs data compression on CMOS camera output
	Data normalization	Performs normalization on spectroscopy measurements

	Data formatting	Formats preprocessed data in line with communication API specification
	Feature selection	Performs feature selection on spectroscopy measurements based on use case requirements
Communication/ API module	BT handler	Handles and operates utilization of the built in Bluetooth communication module (discovery, pairing, connection, data exchange, disconnection)
	BT LE handler	Same as above
	USB handler	Handles and operates utilization of the USB port for data exchange between the embedded system and the PhasmaFOOD mobile app
	API handler	Manages message exchange with the PhasmaFOOD mobile application and/or the cloud platform

This deliverable describes the way each embedded component is realized in the first prototype and validates the requirements from Deliverable Report D1.2 [1].

2 Embedded device software components

2.1 Embedded software architecture

The main electronic board, which is included inside the PhasmaFOOD sensing device, integrates a microprocessor environment. The embedded software is implemented and installed on the latter performing all the essential functionalities, which ensure the correct and defined operation of the PhasmaFOOD sensing device. Such functionalities include:

- Control of all communications with the sensing subsystem and the end-user's mobile device.
- Configuration of the sensing subsystem based on the values of a defined set of parameters received from the mobile application.
- Collection of sensory data.
- Preprocessing of the sensory data.
- Ensuring accepted levels of temperature and oscillations inside the sensing device for the correct operation of the sensing components.

2.1.1 High level description

The main modules included in the embedded software of the PhasmaFOOD sensing device are the following:

- Embedded control logic.
- Handlers for the sensors integrated on the electronic board and targeted to provide data related to the operational conditions, e.g., temperature of the integrated environment, mechanical oscillations.
- Handlers for the sensing and lighting components of the sensing subsystem.
- Handlers for the communication with the end-user's mobile device.
- Embedded data preprocessing and analysis.

The embedded control logic essentially comprises a set of operational states for the main electronic board. Each operational state defines a specific mode of operation for the board and runs one or more explicit services. Changing from one operational state to another takes place when an event is triggered either by the end-user of the sensing device or internally at the main electronic board due to violation of some specified thresholds.

The main electronic board integrates two sensors for monitoring the operational conditions: a high accuracy, low power, digital temperature sensor [3] and an Inertial Measurement Unit (IMU) [4], which includes an accelerometer and a gyroscope, targeted to be used in order to

detect any oscillations that can damage the sensing components or result to incorrect measurements. Both sensors communicate with the microprocessor via the Inter-Integrated Circuit (I2C) protocol [5] for on-board communication between mounted components.

Each of the three sensing components integrated in the sensing subsystem communicate with the main electronic subsystem via a driving board, which is placed near them inside the housing for the sensing subsystem. The driving board, which the UV-VIS spectrometer is mounted on, integrates an Analog-to-Digital Converter (ADC) as well. The analog output of the spectrometer is driven to the aforementioned ADC and the ADC's digital output is sent to the microprocessor unit of the main electronic board via the Serial Peripheral Interface (SPI) [6]. Some General Purpose Input-Output (GPIO) pins of the microprocessor are used for the control communication with the UV-VIS spectrometer. The communication with the driving boards of the NIR spectrometer and the camera is achieved via the Universal Serial Bus (USB) protocol [7].

The sensing device communicates wirelessly with the end-user's mobile device, on which the PhasmaFOOD mobile application is installed. The wireless communication is achieved via the Bluetooth/Bluetooth Low Energy (BLE) protocol [8].

The sensory data is preprocessed, before being sent wirelessly to the end-user's mobile device for further processing on the mobile application or direct forwarding to the PhasmaFOOD cloud platform. The preprocessing functions are presented in Section 3 of the current Deliverable Report. The details of the hardware architecture of the embedded device are presented in the Deliverable Reports D5.1 [9] and D5.2 [10].

2.2 Embedded control logic

The main electronic board operates at 10 different states: 1) Shut-down, 2) Communication, 3) Update, 4) Measurement, 5) Preprocessing, 6) Oscillations, 7) High temperature, 8) Overexposure, 9) Low battery, and 10) Energy save. Figure 2 illustrates all the operational states for the main electronic board. In Deliverable Report D2.3 [11], Section 2.3.1.1, each of these operational states was thoroughly described, providing details on the functionalities they integrate.

During the operation of the main electronic board, a number of events may be triggered either by the end-user of the sensing device or due to exceeding some specified thresholds. 2 depicts all the different triggered events and the transitions, which take place between two operational states according to a specific triggered event. The events, which are triggered by the end-user, are the two following: 1) UON (User pressed button to turn ON), and 2) UOF (User pressed button to turn OFF). The internally triggered events are named as following: 1) SUP (Start UPdate), 2)

UPC (UPdate Completed), 3) RSD (Read Sensors Data), 4) POR (Preprocessing Operations Required), 5) TSD (Transmit Sensory Data), 6) LBD (Low Battery Detected), 7) BCN (Battery Charging Needed), 8) HTD (High Temperature Detected), 9) HOD (High level of Oscillations Detected), 10) OED (OverExposure Detected), 11) GTS (Go To Shut-down), 12) RPP (Restore Previous Process), and 13) RCN (ReConfiguration Needed). In Deliverable Report D2.3 [11], Section 2.3.1.2, each of these triggered events is defined describing the cause of triggering and the actions and transitions between operational states that follow.

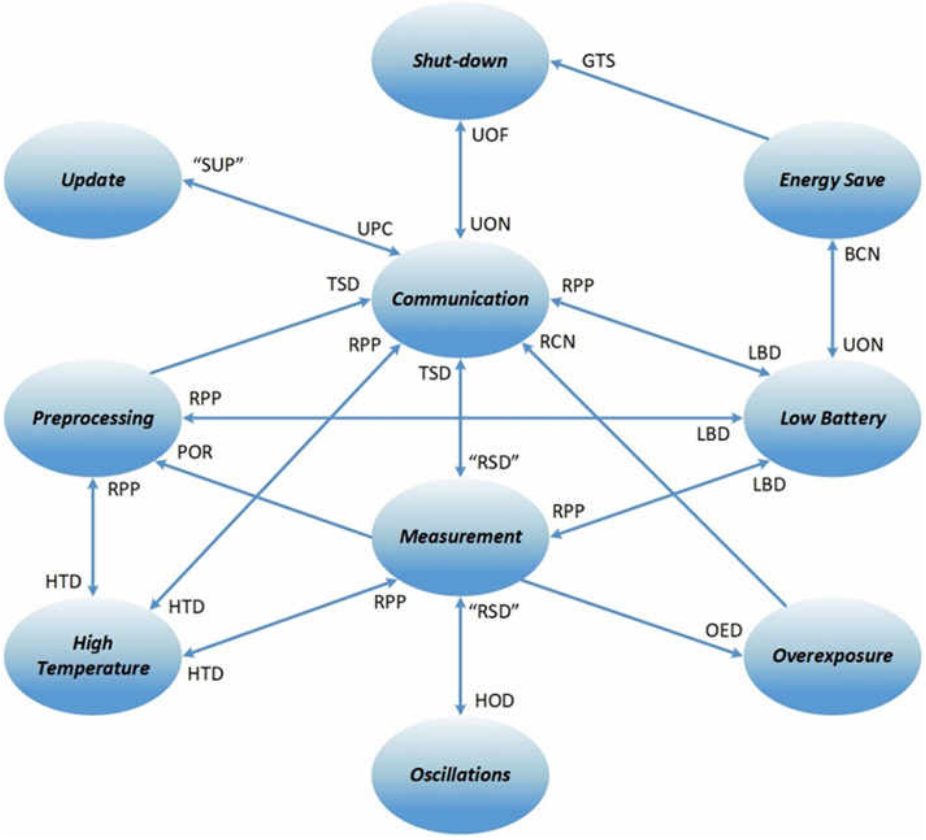

Figure 2 - Diagram of operational states for the embedded software

2.3 Communication between sensing and main electronic subsystems

2.3.1 Communication with the UV-VIS spectrometer

The UV-VIS driving board inside the sensing subsystem integrates the UV-VIS spectrometer and an ADC as well. The UV-VIS spectrometer, which the consortium have chosen to integrate inside the PhasmaFOOD sensing device, is the C12880MA from Hamamatsu Photonics [12]. The ADC is the AD4000 from Analog Devices [13]. The digital clock and control signals of the UV-VIS spectrometer are handled through GPIO pins of the microprocessor and the ADC communicates with the main electronic board using the SPI protocol [6].

The SPI bus is a synchronous serial communication interface. The SPI modules communicate in full duplex mode based on a master-slave architecture, which supports the existence of a single master. The SPI module of the ADC operates as a slave, while the SPI module on the embedded board operates as the master.

The SPI bus specifies four logic signals as following:

- SCLK (Serial CLocK): The SCLK signal is the clock signal with respect to which the data are transferred on the SPI bus. It is an output of the master module and each slave module receives the SCLK signal as its clock input.
- MOSI (Master Output Slave Input): The MOSI signal is used for data transmission out of an SPI master module and data reception at a slave one.
- MISO (Master Input Slave Output): The MISO signal is used for data transmission out of an SPI slave module and data reception at a master one.
- SS (Slave Select): When an SPI module operates as a master, in the independent slave configuration, one SS output signal from the master module drives a corresponding input signal at each slave module with which a data transfer may take place. When a specific slave module is selected to communicate with the master one, then the corresponding SS signal is driven low or high in order to establish a connection with the selected slave module. The SS signal is often driven low when the corresponding slave module is selected.

Based on the datasheets of the UV-VIS spectrometer [14] and the ADC [15], we have developed a UML sequence diagram for the communication between the main electronic subsystem and the UV-VIS driving board, which is illustrated in Figure 3.

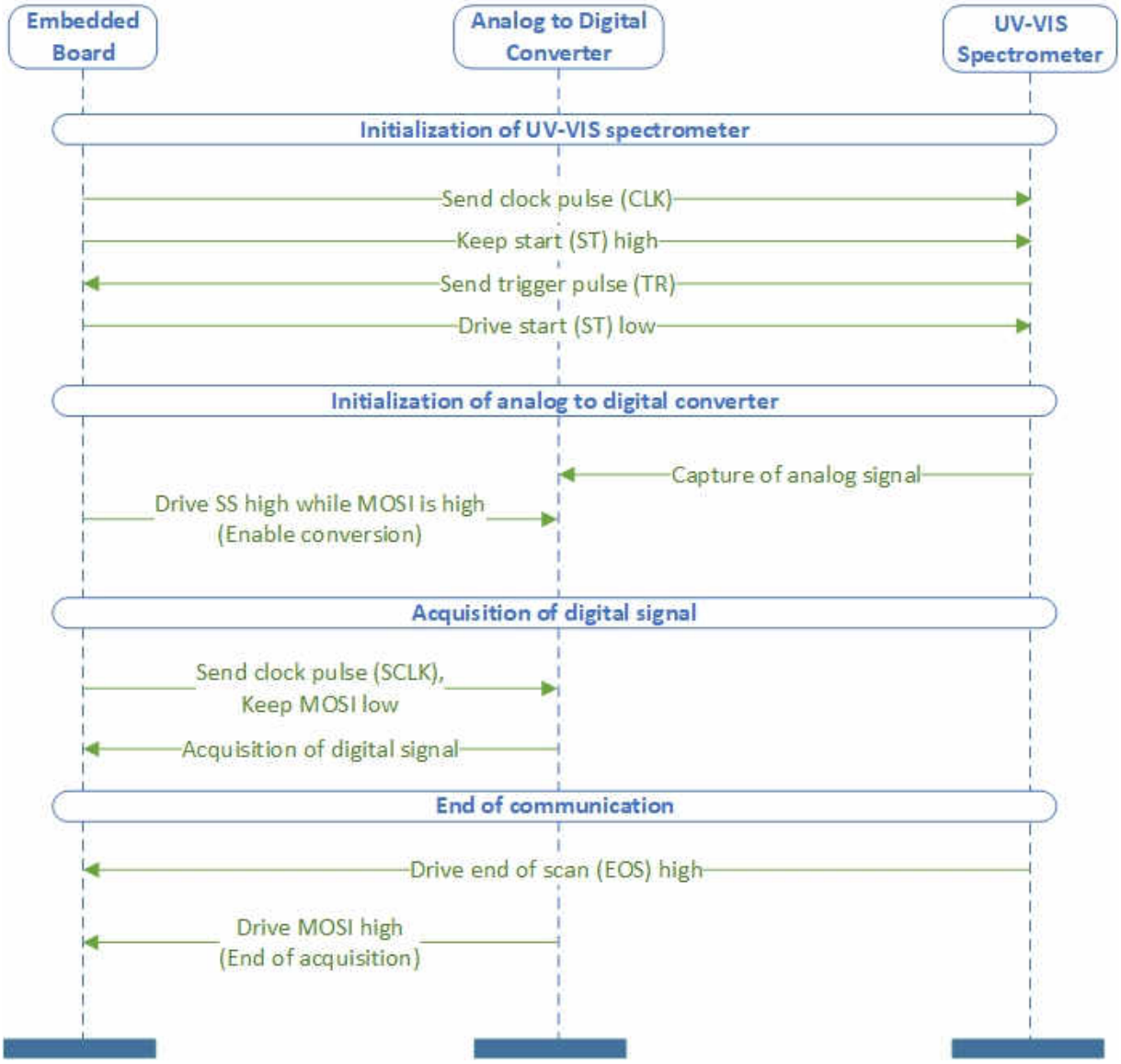

Figure 3 - UML sequence diagram for the communication between the main electronic subsystem and the UV-VIS driving board

2.3.2 Communication with the NIR-spectrometer

The communication of the main electronic board with the driving board of the NIR spectrometer is achieved via the USB protocol [7]. The USB is a cable bus that supports data exchange between a host and a wide range of simultaneously accessible peripherals. The USB connects USB devices

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

with the USB host. We should mention here that the main electronic board includes a USB hub module [16]. The USB hub device offers the capability of mounting two distinct USB host connectors on the main electronic board, each used for a connection to a USB device. The driving boards of the NIR spectrometer and the camera act as USB devices since they integrate the appropriate USB device interfaces and connectors. In the Deliverable Report D2.3 [11], Section 2.3.2.1, we have included some more details on the USB protocol and the USB hub component, which are integrated at the main electronic subsystem.

In Figure 4, we have developed and illustrated a UML sequence diagram for the communication between the main electronic subsystem and the processing unit of the NIR driving board.

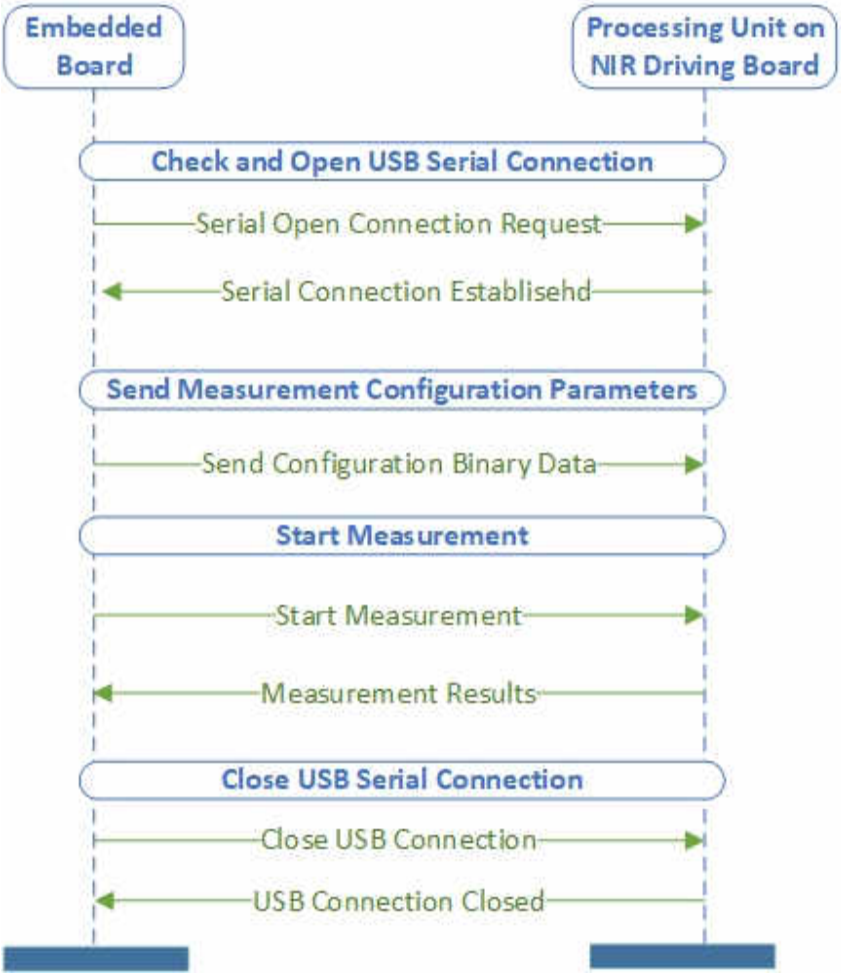

Figure 4 - UML sequence diagram for the communication between the main electronic subsystem and the NIR driving board

2.3.3 Communication with the camera

As previously mentioned in Section 2.3.2, the communication between the main electronic subsystem and the driving board of the camera is achieved via the USB protocol. In Section 2.3.2 of the current Deliverable Report and the Deliverable Report D2.3 [11], Section 2.3.2.1, more details and information are provided on the USB communication between the main electronic board and the driving board of the camera.

2.3.4 Handling of the lighting sources

The microprocessor of the main electronic board drives all the lighting sources inside the sensing subunit via the TLC5922 module of Texas Instruments [17]. This module is integrated on the main electronic board and communicate with the microprocessor via the SPI. The main functionality provided by the TLC5922 component is the adjustment of the brightness of each lighting source.

2.4 Communication with mobile device

The main electronic board is able to establish a wireless communication with the mobile device using the Bluetooth protocol [8]. The Bluetooth wireless technology is a short-range communications system providing robustness and low power consumption. The communication between the embedded and mobile devices should be achieved based on the BLE specification. In Deliverable Reports 2.1 [2] and 2.3 [11], more details and information are provided regarding the physical implementation of the Bluetooth protocol and the sequence of actions that take place during the establishment and data exchange of the Bluetooth communication.

The main electronic board is also able to communicate with the mobile device using the USB protocol. In this case, the embedded board has the peripheral role called USB device that is attached to the USB host, the mobile.

2.4.1 Bluetooth API description

Figure 5 shows the sequence of exchanging data and commands between the embedded software and the mobile application of the PhasmaFOOD system architecture. After the connection is established, the food analysis can start. The end-user selects a specific use-case and food type through the interface of the mobile application and enables sending a collection of configuration parameters for the sensing components of the sensing subunit to be sent from the mobile device to the embedded electronic board. The main electronic subsystem receives the values of the aforementioned parameters, proceeds with all the requested measurements and collects the sensory data. The details of the measurement protocol will be reported in Deliverable Report D4.1 [18]. The collected data are preprocessed and sent to the mobile application, over the established communication channel. The embedded software reports the

status of the embedded system and triggers alarms when a certain operational condition is met and requires corrective action (i.e., low battery, high oscillations, or high temperature). After the mobile application receives the analysis results and decisions, it may transfer the data to the embedded software to be presented on a display. Another stage in information exchange through this API is the embedded software update, which is transferred from the cloud platform to the sensing device through the mobile application.

The Bluetooth API employs a JavaScript Object Notation (JSON) data model for exchanging information between the embedded device and the mobile application. The structure of the JSON message, which is sent from the mobile application to the embedded device after the connection has been established and in order to initiate the food analysis procedure, is illustrated in Figure 6. It provides identification information on the selected by the end-user use-case and food sample along with some environmental information related to the food sample. Also, it includes configuration details regarding all sensing components and essentially requests for the sensory measurements related to the selected use-case and food sample.

Figure 7 depicts the structure of the JSON message, which is sent from the embedded device to the mobile application as a response to the request message that the mobile application has sent to the embedded device. The message includes information on the status and operation of the sensing components during the cycle of measurements conducted for the use-case that the end-user selected and the preprocessed sensory data.

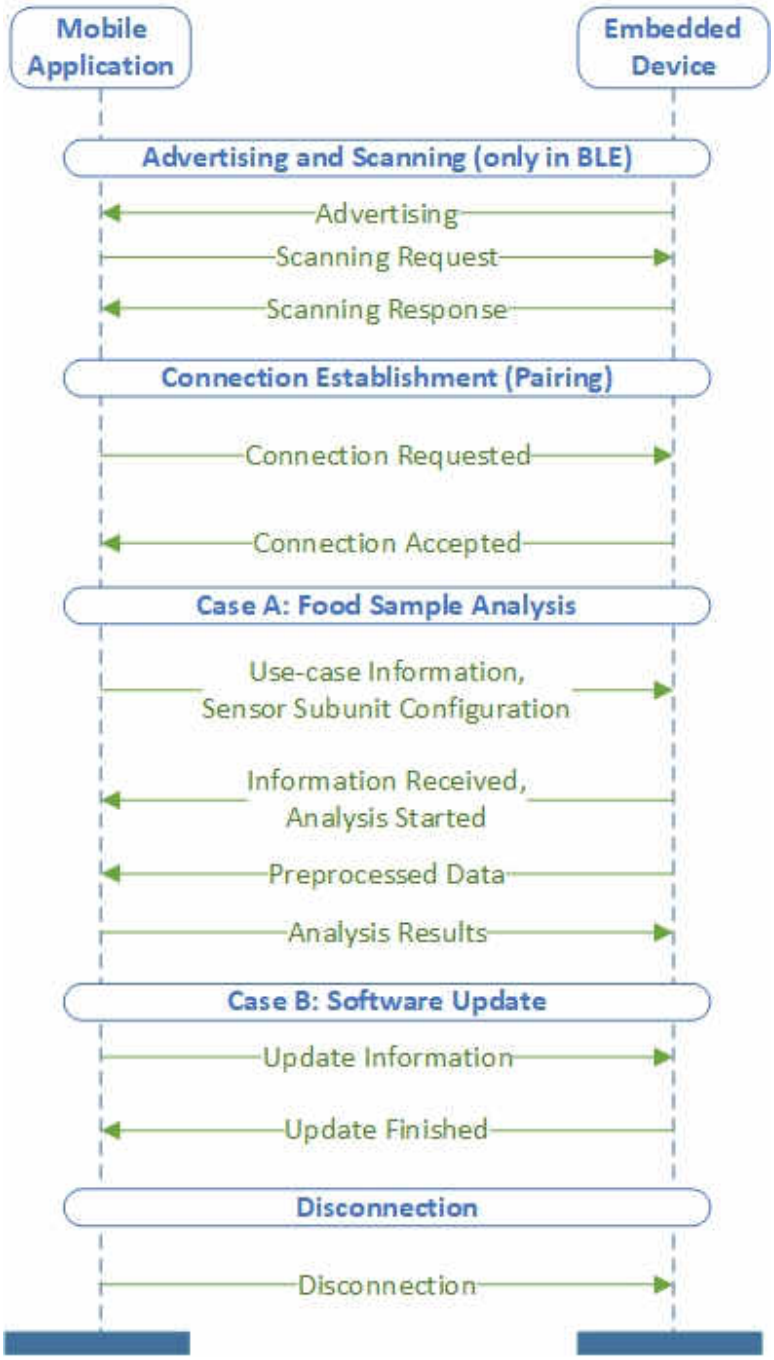

Figure 5 - UML sequence diagram for the wireless Bluetooth communication between the main electronic board and the end-user's mobile device

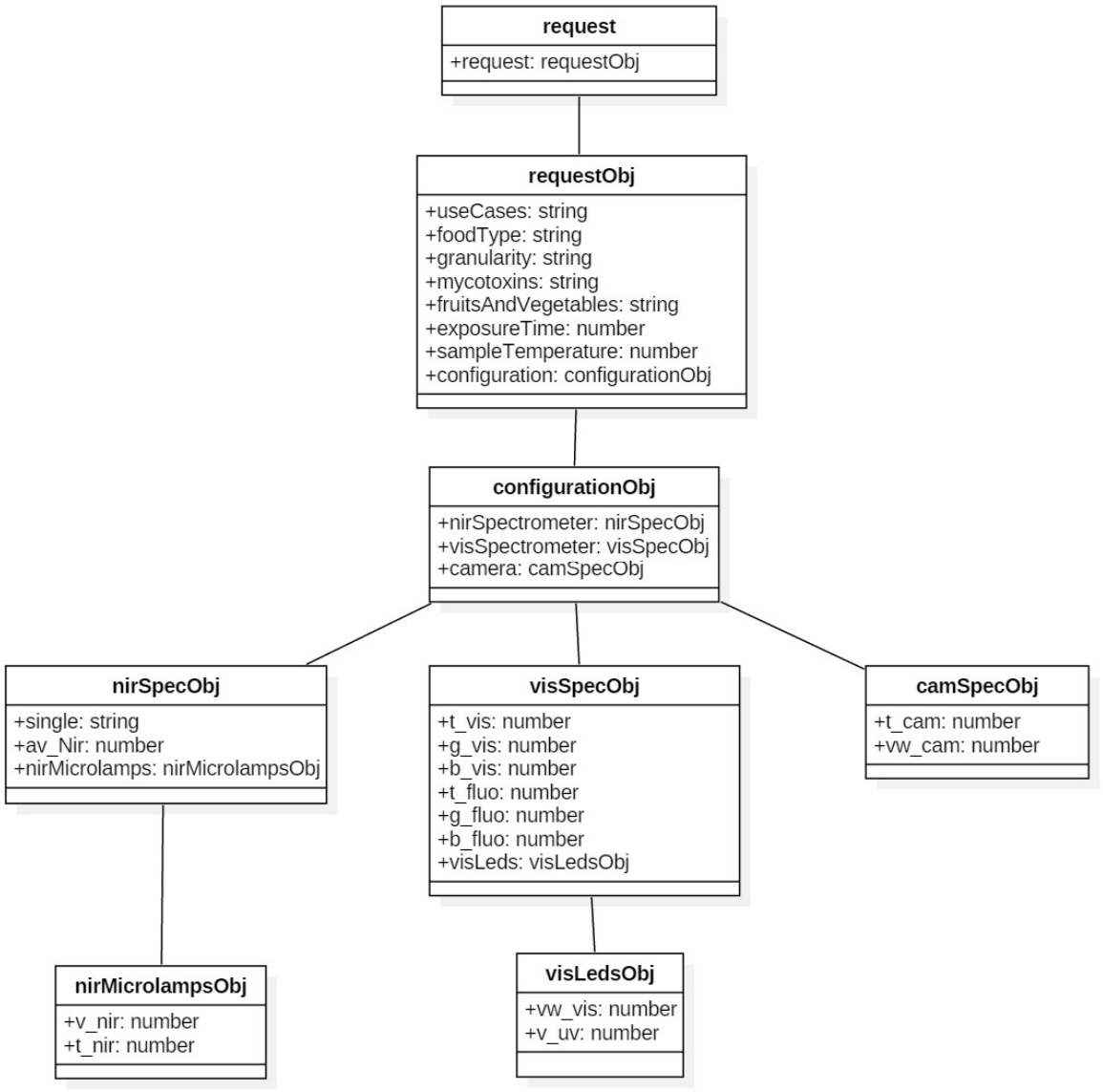

Figure 6 - Structure of the JSON message, which is sent from the mobile application to the embedded device

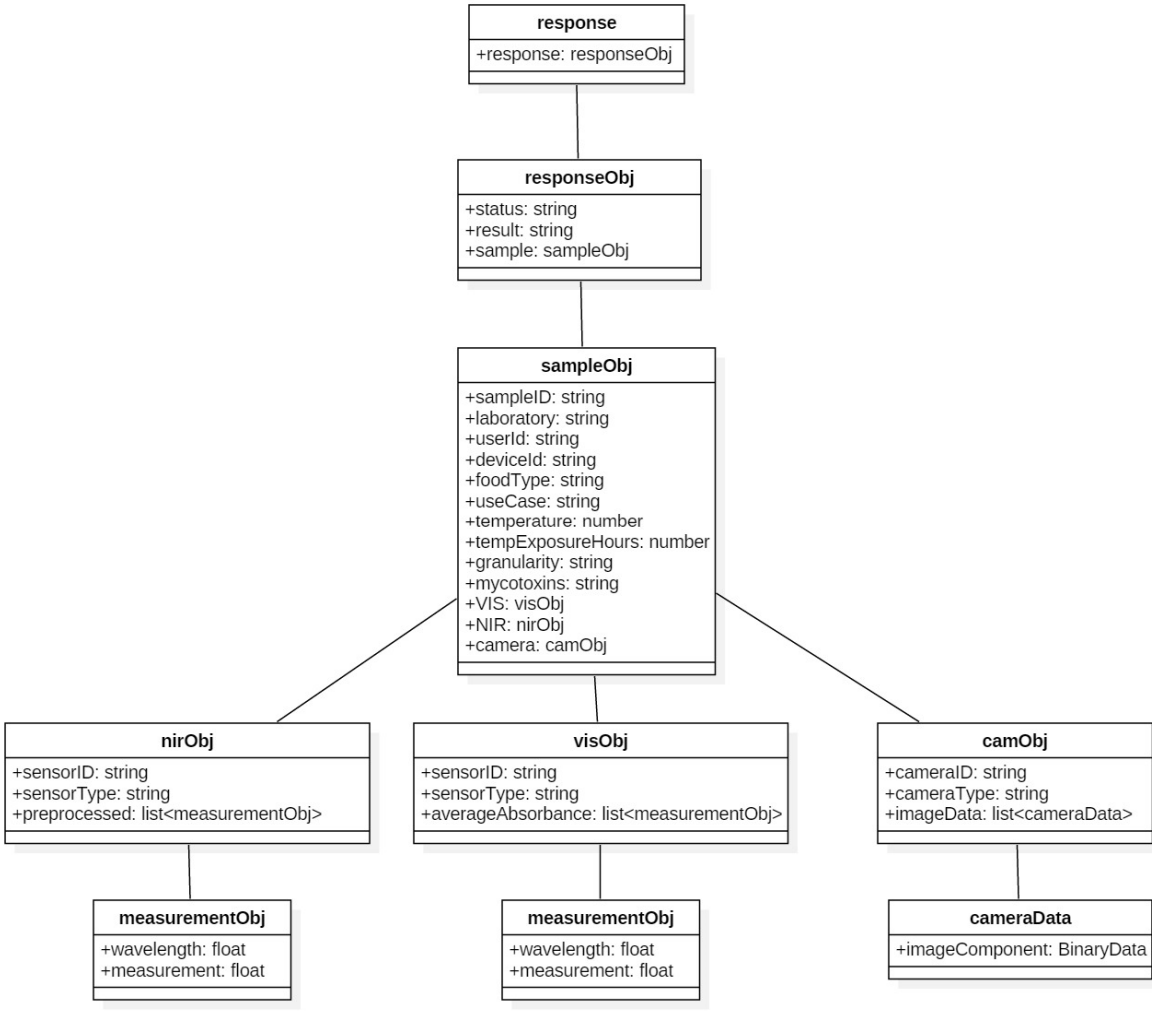

Figure 7 - Structure of the JSON message, which is sent from the embedded device to the mobile application

3 Embedded data preprocessing

The embedded software is able to apply preprocessing steps on the raw data in order to reduce the amount of data to be transferred via Bluetooth to the mobile application and to the compute engine on the cloud platform. The detailed data acquisition procedures and measurement processes will be presented in Deliverable Report D4.1 [18].

3.1 UV-VIS data

The flow of acquisition and storage control is currently performed with Hamamatsu evaluation board connected to a PC by USB cable. A scheme of driving circuit and of the timing chart provided by Hamamatsu are shown in Figure 8 and Figure 9.

Since VIS spectrometer will work in two different modalities, the raw data recorded from the VIS spectrometer in either reflectance or fluorescence mode is accompanied by previous dark and reference measurements relative to reflectance and fluorescence mode. These references are used to calibrate the sensor and, hence, ensure correct and reliable measurements, but also to perform data preprocessing. Wavelength conversion factors, provided with each sensor device, will be implemented in embedded software.

Both for fluorescence and reflectance modality, integration times will be selected automatically by the app according to sample type: the building of calibrated database is allowing to set the optimal integration time for each case. In the reflectance mode the VIS sensor records 10 measurements for the dark and white reference, and stores them in local memory. These measurements are averaged for improving the SNR of the reference measurements. Similarly, the VIS sensor performs 10 internal scans during each measurement operation, which are then averaged for improved SNR. After that, the dark reference is subtracted from the averaged scans and divided pointwise by the white reference (after dark subtraction from reference). For database-building, the embedded device is also able to send the internal scans without averaging. In the same way, in fluorescence modality the embedded software records 5 measurement both for dark, reference and sample measurement and similarly averaged dark is subtracted from averaged measure that will be then divided (or normalized) to the fluorescence measurement. For database-building the embedded device is also able to send the internal scans without averaging.

A SOP of the visible reflectance spectroscopy and Fluorescence spectroscopy measurements can be found in D 3.1 [19].

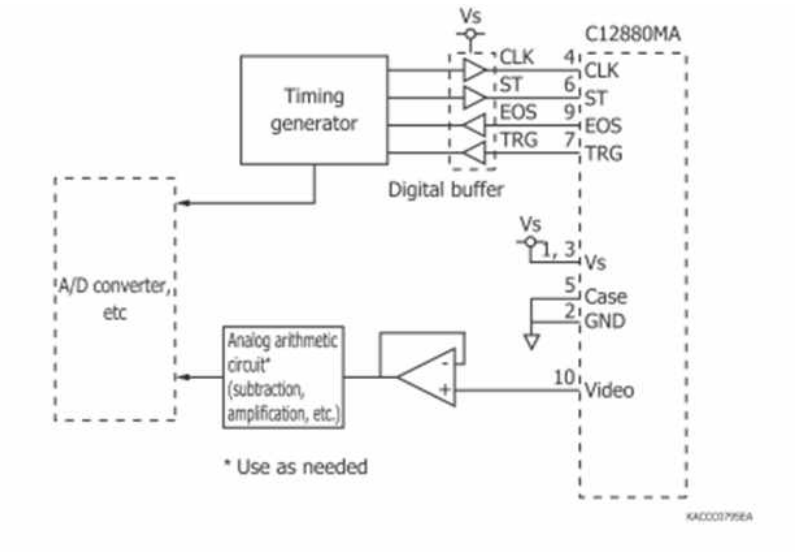

Figure 8 - Driving circuit suggested by Hamamatsu

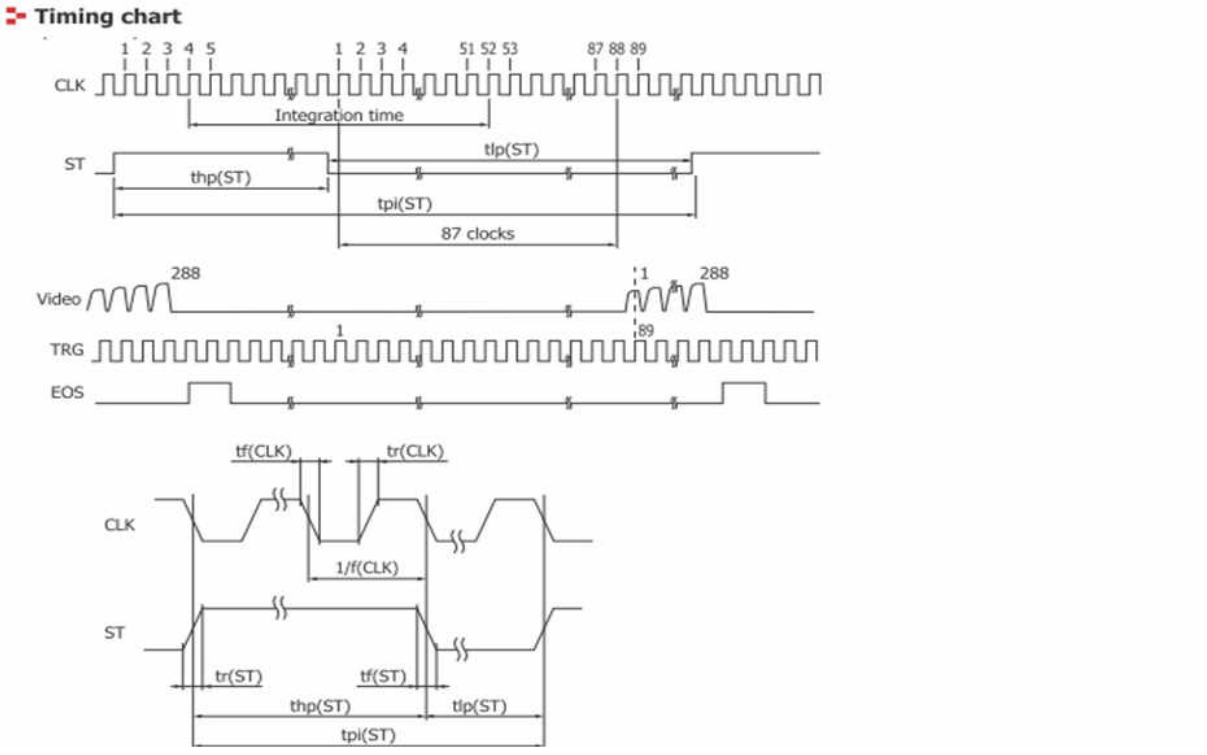

Figure 9 - Timing chart of VIS spectrometer acquisition

3.2 NIR Data

The raw data recorded from the NIR spectrometer are transferred to the main board electronics sub-unit via a standard USB protocol (shown in Figure 10) to the embedded software.

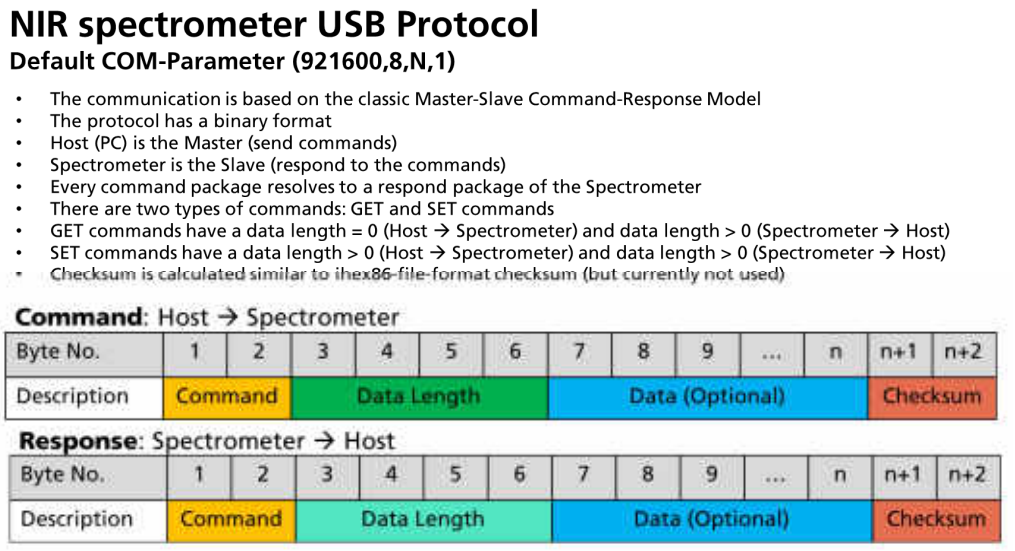

Figure 10 - USB protocol of the NIR spectrometer

The raw data transferred via the shown USB protocol can be classified into status commands, configuration commands and spectral commands. These specific commands including data length, type, format and typical default values are shown in Figure 11.

Command	Send data length (in Byte)	Respond data length (in Byte)	Command Type	Data format	Default value(s)	Description
Status commands						
0x0000	0	4	GET	ASCII	SPEC	Get identification word
0x0001	0	4	GET	Hex	00.00.00.01	Get Revision Number
0x0002	0	4	GET	HEX	XX.YY.nn.nn	Get Status of spectrometer
						XX= 5 or 6 MEMS works correct
						XX= 0 or 3 MEMS error
						XX= 2 or 4 MEMS in startup
						YY = 2 or 3 in acquisition
0x0003	1	1	SET			Set Software start of acquisition state; 0x01 = single
0x0003	0	1	GET		0	Get Software start of acquisition state
0x0004	0	4	GET	Hex		Get MEMS ChipID
Configuration commands						
0x0100	1	0	SET	uint8		Set number of averging
0x0100	0	1	GET	uint8	40	Get number of averaging
0x0102	0	4	GET	float	20.0	Get MEMS Voltage
0x0103	0	4	GET	float	140.0	Get MEMS Frequency
0x0104	0	4	GET	float		Get MEMS Amplitude
0x0105	0	4	GET	float		Get MEMS Phase
Spectral commands						
0x0200	0	7	GET			Spectral data scaling:
				uint16	0	1 Byte sampling status
				uint16	1001	2 Byte sampling points
				uint16	900	2 Byte start wavelength
				uint16	1900	2 Byte stop wavelength
0x0201		s x 4	GET	float		(sampling points) x float (4 Byte) spectral data

Figure 11 - Commands for data transfer via USB protocol of the NIR spectrometer

Depending on the sample of interest the number of averages might have to be adjusted via command 0x0100 to optimize the signal-to-noise ratio (SNR) prior further data processing. The general measurement procedure for the NIR spectrometer follows the Standard Operating Procedure (SOP) routine described in D3.1 [19] for the commercial device SGS1900. However, it has to be mentioned that the embedded software implementation of the SOP routine for the NIR spectrometer is still work in progress. With the commands in Figure 11 the UML sequence (Figure 5, Section 2.3.2) for the communication between the main electronic subsystem and the NIR driving board can be realized. Output data for the NIR device is a standard table containing spectral data points (corresponding to wavelengths) with associated intensities (counts) and a header with status and configuration settings.

3.3 Camera data

3.3.1 Image compression algorithm requirements

The RGB-camera obtains high-resolution color images from the measured samples. A raw 5 megapixel image amounts to 5-10Mb of data. To reduce the traffic from the embedded device to the mobile application via Bluetooth, and later from the mobile application to the cloud platform, these images need to be compressed at an early stage. The compression must not destroy the features possibly present in the images that will be used later for classification/regression in the cloud. Hence, the compression algorithm shall be designed to be data-adaptive and label-aware, i.e. the class information present in the labeled training data shall be exploited by the compression algorithm to preserve the most distinctive features, while still achieving a high compression rate. Since the compression step should take place on the embedded device, which has only limited resources, it is mandatory to design a fast, low-complexity algorithm that is able to operate with the given RAM and CPU. Below, such a fast, adaptive compression algorithm is proposed, which takes advantage of the distributed architecture of the PhasmaFOOD system. The actual implementation of this image compression algorithm is still in development and not yet part of the embedded software for the first prototype. The details of the implementation and a thorough evaluation of its compression rate and achievable classification accuracy will be subject to Deliverable Report D6.1 (M18).

3.3.2 Image compression using adaptive overcomplete dictionaries

The main idea behind the proposed compression algorithm is to take advantage of the redundancy inherent in natural images, and especially in the images that will be collected with the PhasmaFOOD device. Since the camera will operate at a distance of 30mm to the sample, the images obtained will give a zoomed-in view of the measured sample, centered at the focal point of the spectrometers. The resulting images will look very similar for each instance of a food type, but with possibly subtle variations in color and surface structure in case of a spoiled sample.

For each food type, an overcomplete dictionary will be learned offline in the cloud from labeled training data sets. The goal of this dictionary learning approach is to enable incoming images to be sparse coded efficiently, which will efficiently reduce the amount of data to be transferred without losing important information content. After a quantization of the non-zero values, a lossless compression technique will be employed to further compress the sparse code that now represents the image. In this way, the algorithm consists of two separate parts, i.e. the learning stage, which is time consuming and resource-intensive, but will be performed on the powerful PhasmaFOOD cloud engine before-hand, and the compression step, which will be employed on the embedded system and is designed to be fast, despite the limited resources. An overview of the proposed scheme is given in Figure 12.

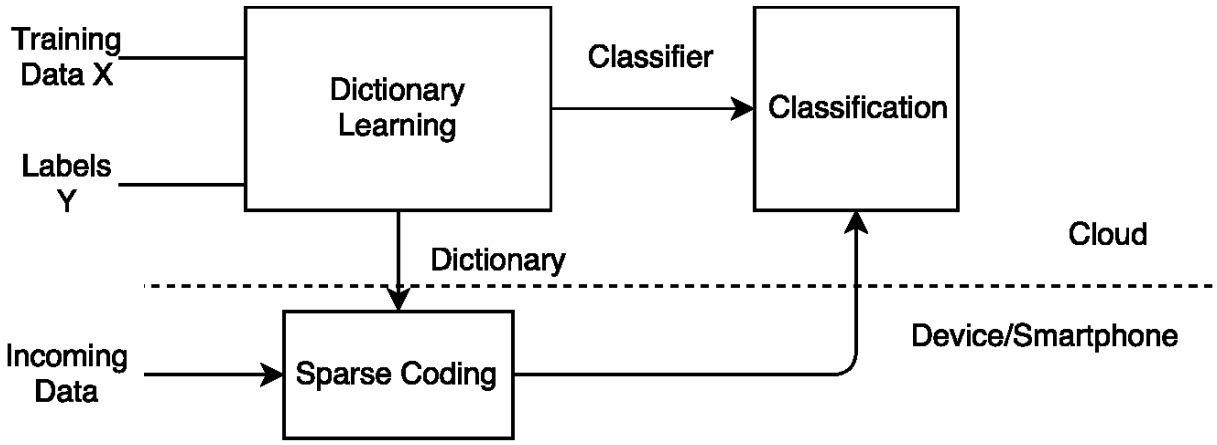

Figure 12 - Compression scheme

The compression step consists of cutting the incoming raw image into non-overlapping patches, the size of which is a free parameter that can be chosen before-hand according to the computing capabilities of the embedded device. Then, each patch is sparse-coded with the pre-computed dictionary for the current food type using orthogonal matching pursuit (OMP) [20]. OMP is a greedy search algorithm, which at each step seeks for the dictionary atom that is most correlated to the residual of the previous step. The algorithm produces an approximation of each image patch, consisting of the weighted sum of only few dictionary atoms. Since the dictionary is learned in order to well represent images from a certain food type, this sparse coding step greatly reduces the description length of the image without losing much information. The obtained non-zero values will be quantized uniformly, e.g. into 256 bins so that the values can be represented as 8-bit integers. Finally, the sparse matrix will be represented in an efficient format. There exist several methods for efficiently storing sparse matrices, the simplest being to only store the non-zero values together with their indices. Additionally, other preprocessing steps like a wavelet transform or some color transformation can be performed before applying this compression. The most efficient preprocessing options will be evaluated once complete image datasets are available. A high-level description of the compression step can be found in Figure 13.

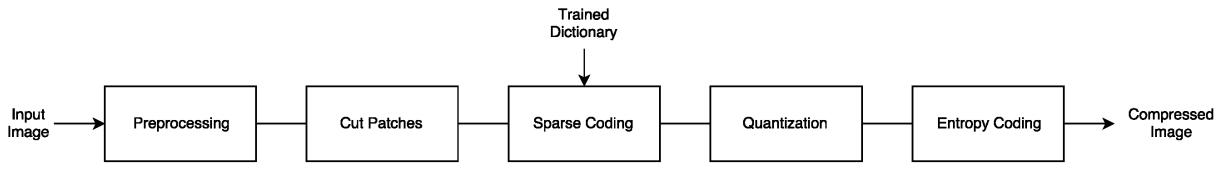

Figure 13 - Compression steps on the embedded device

After transferring the compressed image to the cloud, the compression steps can be reversed and the sparse codes can be fed into a classifier to obtain a prediction of the use-case specific dependent variable. It is noteworthy that recent advances in machine learning provide a multitude of tailored algorithms dealing with sparse data. Hence, the compression algorithm also serves the need for meaningful feature extraction from high-dimensional image data, as will be elaborated on in the next section.

3.3.3 Adaptive, label-aware dictionary learning

The input for the dictionary learning algorithm in the cloud are labeled training images, i.e. datasets consisting of many examples for every possible class of samples (e.g. different levels of spoilage). These will be cut into overlapping patches to obtain a huge, shift-invariant training set. Then a dictionary will be learned by alternatingly updating the dictionary atoms and sparse-coding the training samples with the updated dictionary. In order to obtain good classification results on the sparse codes, the dictionary is composed of a common sub-dictionary and class-specific sub-dictionaries [21] [22]. Each training sample is encoded using only the common dictionary and the class-specific dictionary belonging to its class. Since the images of various degrees of spoilage are expected to look quite similar, and only exhibit small differences, the atoms of the common sub-dictionary will be used to encode most of the image, but won't be able to capture the small variations due to spoilage of the sample under consideration. These will be captured only by the class-specific sub-dictionary that was trained from training images of a similar level or type of degradation.

In the dictionary update step, the updated dictionary is obtained by minimizing a cost function that consists of a data fit term and penalties that encourage the sub-dictionaries to be incoherent, i.e. it is made sure that the atoms in one sub-dictionary are different from the atoms in the other sub-dictionaries. These updates can be made in small batches, the size of which will be chosen according to hardware constraints.

After one sweep over all sub-dictionaries, the sparse codes of each training sample will be updated using OMP. The compression rate of the sparse coding step can be controlled by fixing the number of atoms to be used in this step or bounding the reconstruction error.

The algorithm alternates between these two steps until a stopping criterion is reached. A detailed mathematical and numerical analysis of the algorithm will be presented once complete image datasets will be available.

4 Implementation of the first prototype

The PhasmaFOOD embedded software is an integral part of the PhasmaFOOD software platform that spans the smart sensing device, mobile device and cloud layers of the system architecture. It is driving the PhasmaFOOD sensing device operations as the main source of sensory data on which the data analysis models are trained and decisions are made in line with the project use cases. The embedded software is hosting the data formatting, compression and preprocessing modules, which are the first step in every data analysis chain necessary for realization of the project use cases. The full PhasmaFOOD software platform will be documented in details in D4.1 on "First implementation for software for PhasmaFOOD smart food analysis system" [18].

The consortium decided to develop the first PhasmaFOOD software platform prototype in two phases. The first phase was the preparation of a proof of concept (PoC) software platform, which was used for implementation, configuration and validation of the communication channels and APIs, as well as decision making mechanism implemented in the platform. In the PoC PhasmaFOOD software platform, the embedded software was hosted on a BeagleBone microprocessor board [23]. The role of the embedded software in this PoC implementation was to store test/validation parts of the use case data sets. Each available dataset collection was divided into training and test portion. This was achieved by employing a random splitting approach based on a configurable percentage value that splits the available data in test and training sets when they are loaded in the cloud database. While for prototype tests a 70/30 ratio was used resulting in 70% training data and 30% test data, choosing other split ratios can easily be achieved by the extraction tools. The training portion was used in the PhasmaFOOD cloud platform to train the machine learning models for classification and regression, which are part of the decision making mechanisms for project use cases. The test/validation portion of these data sets were used to emulate real measurements coming from the integrated PhasmaFOOD smart sensing device. For the purpose of extracting the test data in a suitable format and in order to ensure in the process that training data will never be used in subsequent tests, a set of scripts was developed that serves the exact purpose of extracting the test data from the cloud database (30% of the data sets) and generating files identical to the ones that would have been produced by the actual sensor hardware, for all sensor types. In addition to the test data, the extraction scripts also provide the classification information of the samples in a suitable format together with relative information (contamination levels, aflatoxins, microbiological data), providing adequate information to simulate a measurement and verify that the correct classification response was subsequently received. It is also worth mentioning that specifically for the cases that averaging and/or preprocessing is to be performed on the embedded device (e.g. average across measurements, Dark subtraction and White division), the averaged and preprocessed

values are also calculated and available in the cloud. This provides a mechanism to further ensure, if necessary, the fact that preprocessed and averaged values originating from the embedded device are matching to the ones that were calculated by a different implementation on a different platform. These validation data sets were stored onto the BeagleBone board and queried based on measurement configuration (selection of use case, food type and other measurement parameters which are used in building and labeling the collected data sets). Since the validation dataset comprise labeled measurement data, we were able to compare the results obtained through data analysis chains performed in the cloud platform and real data label annotated in the laboratory which collected the data. The data analysis chains that were used in the cloud platform have been previously trained by using the remaining 70% of the training data sets. This is a validation procedure that we will follow in the next iterations of the software platform as well.

The PoC embedded software hosted a Bluetooth Low Energy (BLE) API handler and was responsible for establishing Bluetooth communication with the PoC mobile application to receive measurement configuration and send randomly selected measurements from a corresponding validation data set. The PoC embedded software also performed data preprocessing (normalization and application of the white and dark references) on the raw data stored in the emulated test data sets. To enable all this, the first version of the embedded software control logic was implemented.

After the successful validation of the PoC PhasmaFOOD software platform, a roadmap for developing the first system prototype was setup. The embedded software is deployed on the microprocessor of the electronic board integrated inside the smart sensing device, and enhanced with additional software modules and procedures necessary for addressing all requirements of the first prototype.

4.1 Requirements

As integral part of the PhasmaFOOD software platform and based on the role of driving the PhasmaFOOD smart sensing device, the PhasmaFOOD embedded software needs to address the following functional requirements [1]:

Table 2 - Requirements from D1.2

Ref. number	Description	Embedded SW 1st prototype
DATA-1	The PhasmaFOOD device MUST be able to compress data before communication	The first prototype of the embedded software will include data compression methodologies on camera images and also dimensionality reduction procedures (see section 3). These

		methods will be further refined and optimized during preparation of the final prototype.
DATA-2	Data compression SHOULD be performed on the PhasmaFOOD scanner and on the PhasmaFOOD mobile apps.	The embedded software is employing data preprocessing procedures including image compression, normalization and dimensionality reduction.
DATA-CALIB-1	The system MUST be able to give a standard protocol for each specific application.	The first prototype will include software procedures on the embedded system which will be responsible for enabling calibration of individual sensing components and saving optimal calibration parameters for each experiment in a configuration file. The system can collect white and dark references before each measurement session.
DATA-CALIB-2	The drift of the system during PhasmaFOOD device lifetime MUST be known and compensated	The procedures for enabling system recalibration (automatic or manual) will be developed for the final prototype.
DATA-COMM1	The PhasmaFOOD device MUST be able to communicate data to the PhasmaFOOD cloud through the smartphone in low latency, possibly secured (e.g. SSL, see Section 5.7).	The Bluetooth Low Energy interface is utilized on the embedded system and API handler is prepared which enables low latency two way communication between the scanning device and the PhasmaFOOD mobile application. Encryption of the communication channel will be tested for the final prototype.
DATA-COMM3	The PhasmaFOOD device MUST be able to communicate data to a connected smartphone using low-power and shortrange wireless and wired communication.	This is addressed with the BLE communication API described in the section 2.4. This communication module is part of the PoC and the first prototype. It will be further developed and optimized for the final prototype. In addition, the PhasmaFOOD sensing device is able to establish a wired connection to the mobile device using the USB interface. Further information can be found in section 2.4 and the Deliverable Report 5.2 [10].
DATA-COMM4	The PhasmaFOOD system MUST offer bidirectional	After the two devices discover each other and finalize the pairing process, the two way

	communication between PhasmaFOOD device and the connected smartphone	communication channel is established. Measurement configuration parameters (including configuration of individual spectrometers and camera) are transferred from the mobile app to the embedded system. Also, a command for initializing the conduction of measurements is sent. Measurement data for the current measurement configuration are transferred in embedded sw – mobile app direction. In the first PoC, the measurement data (randomly selected from a corresponding test data set) are accompanied with a correct label (measurement classification), which enables validation of data analysis models trained in the cloud.
DATA-COMM5	Communication APIs MUST be defined to specify the communications between the PhasmaFOOD device, the PhasmaFOOD smartphone and the PhasmaFOOD cloud.	The embedded system – mobile application communication API is presented in the section 2.4.
DATA-COMPRESS-1	Embedded µ-controller software (and related hardware specification, i.e. µ-controller, memory etc., specifications) on PhasmaFOOD device SHOULD implement compression tools for 'on-board' preprocessing of sensor data.	The first prototype will include image compression algorithms, perform data normalization and apply white and dark references on spectrometer measurements. Further development and optimization of these procedures will be performed during preparation of the final prototype.
DATA-COMPRESS-2	NIR sensor data rates (expected below Mb/s range) SHOULD be pre-processed ('filtered') in conjunction with the sensor device capability to reduce noise margin and/or possibly extract relevant signal components exploiting	The first prototype includes procedures for calibrating individual spectrometers and further testing their output. Averaging of the acquired internal scans reduces measurement noise. Further online noise reduction procedures will be developed for the final prototype.

	learned or known prior information	
DATA-COMPRESS-3	Embedded software components MAY provide in addition feature extraction capability.	This will be explored for the final prototype, the envisioned image processing algorithm foresees sophisticated feature extraction.
DATA-COMPRESS-4	UV-VIS data rates (expected in Kb/s range) SHOULD be pre-processed the same way as NIR sensor data.	Preprocessing procedures for the two spectrometers slightly differ to accommodate different modes of operation.
DATA-COMPRESS-5	Camera data SHOULD be compressed using standard image compression tools.	The embedded software utilizes image compression tools as presented in the section 3. Further optimization and performance improvements will be conducted on a larger set of images during preparation of the final prototype. The inclusion of an additional JPEG2000 encoder will be discussed as well.
DATA-COMPRESS-6	Camera data MAY be preprocessed (e.g., filtering RGB signal components) or even more advanced feature.	Camera image preprocessing on embedded system will be taken care by the compression algorithm and will be implemented for the final prototype.
DATA-COMPRESS-7	Embedded µ-controller software components in general and compression tools specifically MUST support energy efficient operation in the measurement process.	Energy efficiency of the embedded reprocessing procedures will be investigated during preparation of the final prototype and after conducting excessive energy measurements and tests using the first prototype.
SW-ARCH-3	PhasmaFOOD software MUST be modular so as to enable easy updates of only parts of the system to enable new functionalities or update existing ones.	The embedded system is developed and organized so as to enable easy implementation of new functionalities (i.e. preprocessing algorithms and models), API attributes and commands and states for the control logic.
SW-ARCH-4	Software modules MUST communicate through well-defined and extensible APIs (i.e. REST) supporting future	We have added new features to the established communication API over BLE during preparation of the first prototype and the process of updating the API will be documented for system integrators.

	add-ons, upgrades and new features.	
SW-ARCH-5	The PhasmaFOOD system SHOULD offer hybrid processing and data analysis capabilities between the PhasmaFOOD device (accelerators, microcontrollers), mobile application on a smartphone/tablet and the PhasmaFOOD cloud system.	Embedded software performs first preprocessing of sensory data in each data analysis chain for all project use cases.
SW-ARCH-6	The PhasmaFOOD APIs MUST be defined and implemented so as to enable communication of configuration parameters between the cloud platform, mobile apps and the sensing device.	Configuration of the measurements process and configuration of individual spectrometers and the camera are enabled through the BLE API.
SW-EMBED-1	The PhasmaFOOD device MUST provide the PhasmaFOOD system with the ability to support several operation modes of the PhasmaFOOD device through an operational system software	The embedded software runs control logic based on the state machine presented in the section 2.2.
SW-EMBED-2	An analog front-end API MUST be provided to PhasmaFOOD device for abstraction of the complexity of the analog front-end.	The API for integration of the sensing and control unit is presented in the section 2.3.
SW-EMBED-3	The PhasmaFOOD device MUST provide a trade-off between energy consumption and processing overhead.	Performance optimization and energy efficiency improvements of software platform will be investigated during preparation of the final prototype.

SW-EMBED-4	The PhasmaFOOD device MAY be able to perform image processing.	See section 3.
SW-EMBED-5	The PhasmaFOOD device MAY be able to run smart application software algorithms (e.g. detection algorithms and feature selection).	This feature will be tested during preparation of the final prototype. For the first prototype, the embedded software performs procedures for data preprocessing and compression and leaves the machine learning to the cloud platform.
SW-EMBED-6	The PhasmaFOOD device SHOULD be able to send the data to a specific software (i.e. algorithm) on the cloud and/or mobile device to perform an additional analysis (e.g. detection algorithms).	The BLE communication API enables exchange of raw and preprocessed data for further analysis in the upper layers of the PhasmaFOOD software platform.
SW-EMBED-7	The PhasmaFOOD device MUST support firmware upgrade.	The firmware upgrade state is part of the embedded software control logic. However, the complete logic from including APIs, cloud and mobile application will be addressed in the final prototype.
SW-EMBED-8	The embedded software SHOULD perform data preprocessing operations.	See section 3 for data preprocessing and analysis performed as part of the embedded system.
SW-EMBED-9	The PhasmaFOOD device MUST contain embedded control logic that manages sensory data streams, communication interface and data preprocessing.	The embedded software control logic is presented in the section 2.2 and D2.1 [2]. It will be further upgraded with new states in the state machine and transition triggers for the final prototype.
SW-SEC-7	Encryption mechanisms MAY be integrated in the PhasmaFOOD system in order to ensure data privacy policies.	Encryption of the stored data and communication channels will be considered for the final prototype.
SAF_5	The mobile app/user interface software MUST indicate which kind of failure is happening.	The BLE communication API provides the mobile application with status of the measurement process and indicates types of errors that have occurred.

SAF_6	In this regard, the three sensors MUST be able to work independently to overcome possible failures of one of them.	The embedded sw control logic is capable of utilizing any and all sensors independently to collect data sets. Also, configuration of individual data sources is enabled.
PAR_6	The PhasmaFOOD device MAY have the additional capability to detect all autofluorescent materials like most of the organic substances (urine, body fluids, etc) and fluorescent marker in banknotes.	This feature will be considered for the final prototype.

4.2 Embedded system in the first PhasmaFOOD software platform prototype

The goal of the PhasmaFOOD software platform in the first prototype stage is to integrate all the system architecture layers (smart sensing device, end user mobile device and cloud platform), enable exchange of data and commands between the layers, enable realization of data analysis and decision making procedures for each use case and support collection of new data sets and realization of the project use cases. The first prototype of the software platform targets primarily internal usage within the project and demonstrations on events and workshops. The functionalities developed in software at this stage address the need of advanced/expert users who need to be able to configure the measurement processes, analysis chains, decision making engine and sensing device with detailed parameters which ensure complete control over the measurement process and enable precise troubleshooting and calibration in laboratory usage for collecting new datasets. The final prototype will take this extensive set of features and come up with the set of functionalities which enable non expert users to use the PhasmaFOOD system, but still be able to obtain all the necessary information on the scanned food and build their own scenarios and applications.

The role of the embedded software in the first prototype of the PhasmaFOOD software platform is to:

- Drive the PhasmaFOOD smart sensing device
    - The embedded software system includes the main control logic in a form of a software state machine with predefined states and triggers.

  - The control logic integrates all other components and enables exchange of data within the embedded software modules and outside with the rest of the PhasmaFOOD software platform.
- Enable integration and calibration of the sensing components
  - The embedded software integrates drivers for the sensing unit and individual sensors and light sources.
  - It enables calibration of key parameters of the sensing unit.
- Enable measurement processes for all project use cases
  - Based on optimal calibration of individual sensors and lighting sources the embedded software is capable of executing measurement procedures which employ corresponding sets of sensors with calibration parameters, illumination parameters and sequence and timing.
  - The measurement process parameters are defined in measurement protocols in WP2 and WP3.
- Provide communication channel with the PhasmaFOOD mobile application
  - The embedded software hosts software module and driver for the BLE interface.
  - It also hosts API handler for establishing communication channel with the PhasmaFOOD mobile application. This API enables exchange of measurement configuration and calibration parameters in one direction and measurement status indicators and sensory data in the other direction.
- Preprocess collected data
  - The embedded software integrates a collection of algorithms for data compression, normalization and other preprocessing necessary for preparing data for further analysis performed in the cloud platform.
- Ensure precision and correctness of the measurement process
  - The control logic employs procedures which monitor built in sensors to ensure that the device is used correctly during the measurement process.

5 Conclusion and next steps

In this deliverable report, the embedded software for the PhasmaFOOD smart food analysis platform was described, together with its role in the first PhasmaFOOD software prototype. The current deliverable also verified that the PhasmaFOOD embedded software is designed according to the requirements posed in D2.1 and pinpoints the functionalities that are not yet implemented.

The communication APIs between the sensing and main electronic subsystems and between the embedded device and the mobile app were described in detail and it was checked that the implementation meets the requirements from D1.2. First preprocessing steps are realized on the embedded device. The advanced task of embedded image compression is still under development. A description of the envisioned dictionary learning algorithm was given. The image compression functionality will be fully integrated into the software once the algorithm is implemented and tested thoroughly with complete, labeled camera datasets, which are expected to be available soon. Finally, the embedded software implementation in the first prototype of the PhasmaFOOD software platform was described.

The next steps of embedded software development will be the integration of image compression tools and the migration of the embedded software from the beaglebone to the first version of the PhasmaFOOD sensing device. Furthermore, the embedded functionalities will be refined in accordance to the development in the other work packages and the user experiences gained during recording of new datasets by the consortium. Since an AGILE development strategy is employed by the consortium, frequent small updates and additions of functionality are expected, the outcome of which will be reported in Deliverable Report 6.1.

References

[1] PhasmaFOOD, "D1.2 - System specification and functional requirements," May 2017.

[2] PhasmFOOD, "D2.1 - Sofware specifications for PhasmaFOOD smart food analysis system," July 2017.

[3] [Online]. Available: http://www.ti.com/product/TMP116. [Accessed April 2018].

[4] [Online]. Available: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/. [Accessed April 2018].

[5] [Online]. Available: https://www.nxp.com/docs/en/application-note/AN10216.pdf. [Accessed April 2018].

[6] F. Leens, "An introduction to I2C and SPI protocols," *IEEE Instrumentation and Measurement Magazine 12(1),* pp. 8-13, 2009.

[7] [Online]. Available: http://www.usb.org/developers/docs/usb20_docs/. [Accessed April 2018].

[8] [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification. [Accessed April 2018].

[9] PhasmaFOOD, "D5.1 - Sensing front-end sub-system implementation," June 2018.

[10] PhasmaFOOD, "D5.2 - Hardware Processing Platform, Storage and Communication units integration," June 2018.

[11] PhasmaFOOD, "D2.3 - First version of smart system design," November 2017.

[12] [Online]. Available: https://www.hamamatsu.com/eu/en/C12880MA.html. [Accessed April 2018].

[13] [Online]. Available: http://www.analog.com/en/products/analog-to-digital-converters/precision-adc-20msps/single-channel-ad-converters/ad4000.html. [Accessed April 2018].

[14] [Online]. Available: https://www.hamamatsu.com/resources/pdf/ssd/c12880ma_kacc1226e.pdf. [Accessed April 2018].

[15] [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD4000-4004-4008.pdf. [Accessed April 2018].

[16] [Online]. Available: http://www.ti.com/product/tusb4020bi/description. [Accessed April 2018].

[17] [Online]. Available: http://www.ti.com/product/TLC5922. [Accessed April 2018].

[18] "D4.1 - First implementation fo software for PhasmaFOOD smart food analysis system," May 2018.

[19] PhasmaFOOD, "D3.1 - Feasibility results and use case benchmarking," June 2018.

[20] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory 53(12),* pp. 4665-4666, 2007.

[21] S. Gao, I. W. H. Tsang and Y. Ma, "Learning category-specific dictionary and shared dictionary for fine-grained image categorization," *IEEE Transactions on Image Processing, 23(2),* pp. 623-634, 2014.

[22] M. Nejati, S. Samavi, N. Karimi, S. M. R. Soroushmehr and K. Narajarian, "Boosted dictionary learning for image compression," *IEEE Transactions on Image Processing, 25(10),* pp. 4900-4915, 2016.

[23] "https://beagleboard.org/blue".

[24] "USB 2.0 Specification," November 2017. [Online]. Available: http://www.usb.org/developers/docs/usb20_docs/.

[25] "TUSB4020BI - TI," [Online]. Available: http://www.ti.com/product/TUSB4020BI. [Accessed November 2017].