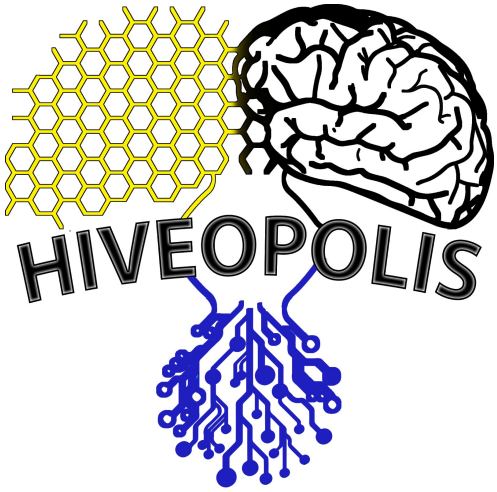


FUTURISTIC BEEHIVES FOR A SMART METROPOLIS

Deliverable D3.1

Hive architecture and core design

Lead Beneficiary	EPFL
Delivery date	30.11.2020
Dissemination Level	PU
Version	1.0
Project website	www.hiveopolis.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824069

DELIVERABLE SUMMARY SHEET

Project number	824069
Project Acronym	HIVEOPOLIS
Title	FUTURISTIC BEEHIVES FOR A SMART METROPOLIS
Deliverable No	D3.1
Due Date	Project month M20
Delivery Date	31.03.2020
Name	Hive architecture and core design
Description	This report gathers details of the performance requirements in communications, storage, computation and sensing, and evaluates identifies hardware, strategies and solutions that are appropriate for fulfilling these requirements. This phase builds on the functional requirements that were established by the consortium in the deliverable D2.1.
Lead Beneficiary	EPFL
Partners contributed	BST, UNIGRAZ
Dissemination Level	Public

Summary

Introduction

Purpose and scope of the document

The central core of each HIVEOPOLIS unit acts to connect the modules of each unit. It connects the modules to one another, issuing commands and collecting data, and also connects one unit to other units in the wider HIVEOPOLIS ecosystem. In addition, the central core provides some functions that directly relate to the functioning of the colony and to the support of the hive systems.

This report builds on the functional requirements that were established by the consortium in the deliverable D2.1, to develop details of performance requirements. This spans communications, storage, computation and sensing. Moreover, the report evaluates and identifies hardware, strategies and solutions that are appropriate for fulfilling these requirements.

Overview of the document

The first chapter contextualises the HIVEOPOLIS unit in the broader literature of technologically augmented honeybee hives, before giving a high-level overview of the proposed architecture. Chapter 2 presents collected information regarding data flows between hive elements, and Chapter 3 evaluates communication and interfacing appropriate for this data volume. Chapter 4 concentrates on software and computational aspects of the core. Chapters 5 and 6 respectively describe the biological and environmental sensing functionalities of the central core.

Acronyms and Abbreviations

Acronyms and Abbreviations	Definition	Acronyms and Abbreviations	Definition
GPIO	General-purpose input/output	GPU	Graphical processing unit
MCU	Micro-controller unit	COTS	Commercial off-the-shelf
CPU	Central processing unit	ABM	Agent based model
PCB	Printed circuit board	RFID	Radio frequency identifier
DSS	Decision-support system	RH	Relative humidity
SBC	Single board computer	RF	Radio frequency
API	Application programming interface	RTC	Real time clock
BLE	Bluetooth low energy	FPU	Floating point unit
ODE	Ordinary differential equation	IC	Integrated circuit
FLOPS	Floating point operations per sec	SD	Secure digital (memory)

Chapter 1: Central core functionality and context

1.1 The role of the central core

The central core of each HIVEOPOLIS unit has a role of interconnection: facilitating connections from the unit to other units in the ecosystem of HIVEOPOLIS entities and stakeholders; and enabling the modules of the unit to communicate with one another. Commands and control start in the core; responses and data collected are logged within the core. There are also several aspects provided by the central core regarding the functioning of the colony and that support the hive systems.

This report gathers details of the performance requirements in communications, storage, computation and sensing, and evaluates identifies hardware, strategies and solutions that are appropriate for fulfilling these requirements. This phase builds on the functional requirements that were established by the consortium in the deliverable D2.1.

In the remainder of this section we contextualise the HIVEOPOLIS unit in the broader literature of technologically augmented honeybee hives, before giving a high-level overview of the proposed architecture. Chapter 2 presents collected information regarding data flows between hive elements, and Chapter 3 evaluates communication and interfacing appropriate for this data volume. Chapter 4 concentrates on software and computational aspects of the core. Chapters 5 and 6 respectively describe the biological and environmental sensing functionalities of the central core.

1.2 Related work

In the expanding field of precision agriculture, and more specifically precision beekeeping (Zacepins et al., 2015) recent years have seen a dramatic increase in technology used in apiology and commercial beekeeping. Digital and continuous monitoring of weight, temperature and humidity is now commonplace (Anuar et al., 2019; Meikle et al., 2008), including for highly detailed (Becher & Moritz, 2009) and broad-scale study (Gil-Lebrero et al., 2017; Lecocq et al., 2015). Many of the smart-hive studies consider one (Ruan et al., 2017) or a small number of hives (Howard et al., 2018; Cecchi et al., 2019; Kridi et al., 2014). In broader studies the technology used in each hive, i.e., each element of the network, is typically identical to others (although note the study of Fitzgerald et al. (2015) includes a multi-node sensory system in which each node has the same architecture but differs in the sensors attached). This and other examples of multi-node networks that cover one or more apiaries are hierarchical (Dogan et al., 2017; Gil-Lebrero et al., 2017) and focus on transmitting data from sensory nodes up to a centralised server, with at most minor attention paid to data being transmitted in from the server outwards, as is natural in a sensor network.

Actuating, investigating honeybees is far more narrow. Although fitting into a field with roots in robo-ethology, swarm intelligence, and apiology, the number of studies using robots and honeybees is very few. One line of investigation explores artificially generated vibrations for

modulating honeybee behaviour, inducing stopping behaviour (Axel Michelsen et al., 1986) and see also (Mariano, 2018)), using a robotic “dancer” performing a waggle dance to recruit foragers to specific sites (Landgraf et al., 2018, 2010; A. Michelsen, 1992). A distinct approach has been developing interactions between mechatronic devices (Griparic et al., 2015) that emit heat to influence group decision-making (Stefanec et al., 2017; Bonnet et al., 2019). These studies with robots that influence the behaviour of honeybees have been limited to laboratory conditions and applied in experiments measured in minutes or hours. None of these works was attempted in the field; moreover, none comprise a hierarchical system of modules and components that are diverse or heterogeneous.
In swarm robotics, large groups of robots are sometimes programmed to investigate behavioural rules that lead to collective dynamics (Rubenstein et al., 2014) of insects (Wilson et al., 2004) including honeybees (Kernbach et al., 2009). Here, the majority of studies use homogeneous elements, and moreover limit communication to simple mechanisms between near neighbours (Gauci et al., 2014).

Thus, HIVEOPOLIS unit represents a unique position in terms of technological systems that interact with honeybees in terms of the target duration of application, the sophistication of elements, the presence of modulatory and sensory elements (that are coupled via models), and the architecture of heterogeneous elements.

1.3 Overview of the proposed core and system architecture

We open with some nomenclature for the sake of clarity:

Module - the HIVEOPOLIS system consists of several modules, including the storage modules, a dancefloor module, a gate module, and the brood nest modules. Each one is a sub-system that can be removed from the hive individually, some without affecting the overall hive functioning.

Component - various subsystems including the central core consist of multiple, well-defined lower-level entities. For the sake of avoiding re-use of the term module, we label these parts components.

The HIVEOPOLIS hive units are sophisticated bio-hybrid systems, comprising many modules and other components. The majority of components will be connected by a single communications bus, although some special cases exist - notably, the wireless devices such as the GUI and the weather station. See the topological overview (Fig 1.1).

Within the core, the internal organisation uses specialised interfaces for its own peripherals, including USB for the data link to the LTE module for mobile (off-hive) communications. The hive-internal monitoring includes wireless sensor nodes that connect over BLE; and the weather station connects via WiFi. The GUI will be designed as part of another task and reported in another deliverable, but the concept outlined is to use a wireless interface, either WiFi or BLE, to facilitate this.

Finally, the present evaluation of the computational load is that a single main CPU is sufficient. If future development in models or on-core analysis necessitates further resources, these will likely exploit an ethernet link and be managed by the primary CPU.

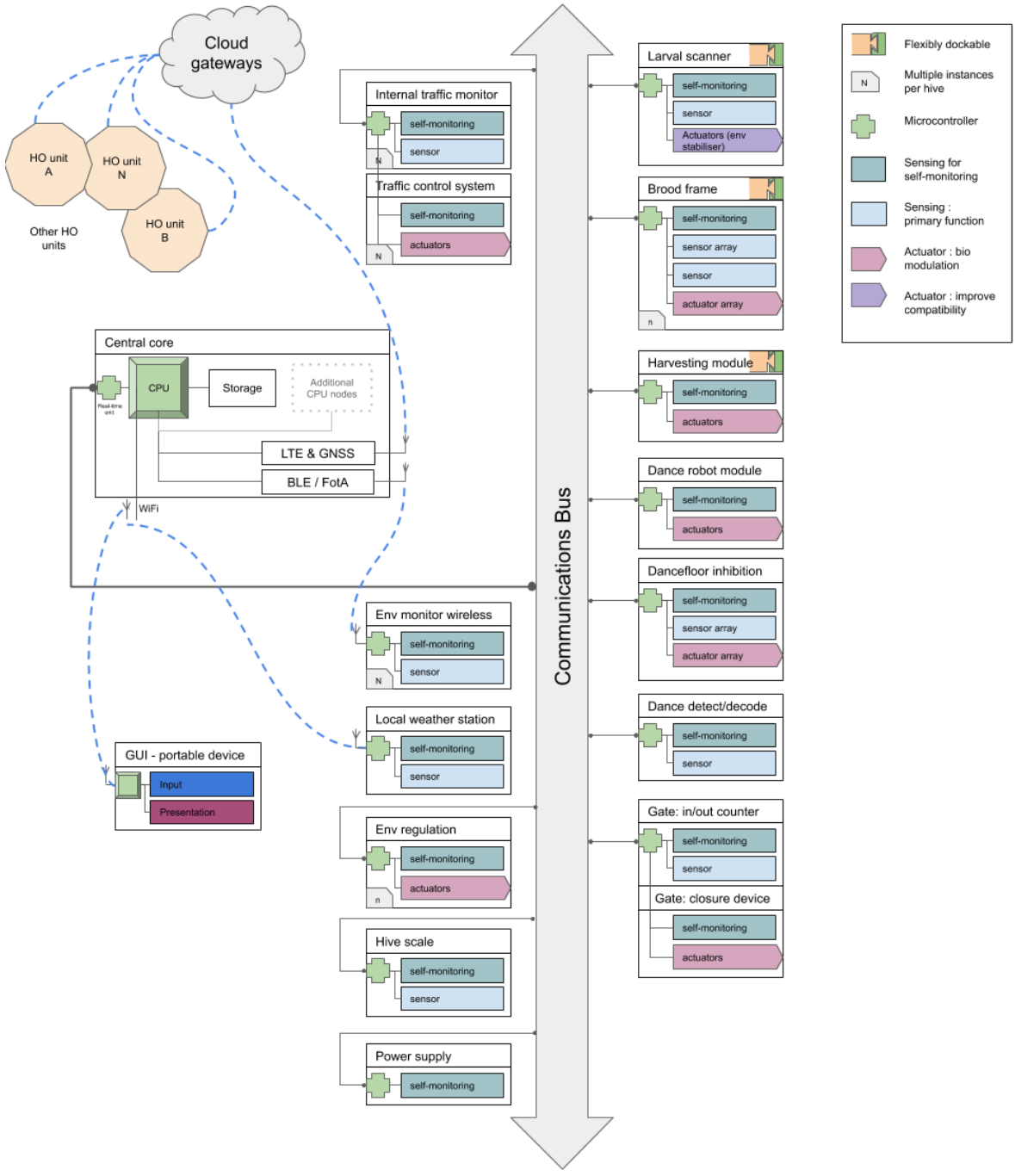


Figure 1.1 An overview of the modules and components within one HIVEOPOLIS unit.

Chapter 2: Data budget

We start with a detailed look at the data that must be handled within a HIVEOPOLIS unit, since many of the other core system components will be dimensioned according to what is transmitted, between which elements, how often, and when.

2.1 Data transfer patterns

The data exchanged between the components of the system fall into four categories:

1. **Instructions**
    a. *Control of operation*: e.g., To activate/disable actuators; to update parameters such as sampling rates; to enable/disable the element. An element may receive control instructions, generate control instructions for other elements, or both.
    b. *Request for data*: the sensing or analysis performed by some modules does not make sense to be periodically driven, and accordingly for another element to access those data, they must be solicited via a request.
2. **Data**
    a. *System health status*: including self-monitoring, such as current consumption, load levels, free memory levels; control state
    b. *Sensor or scientific data (including model outputs)*: when sensors are sampled; analysis is run that that extracts higher-level information from sensory data, e.g., dance parameters or centre of the brood nest; or models that produce forecasts, we have data corresponding to the primary function of the hive system.

    Whether relating to the bee colony or the system state, these data may be generated periodically, generated by events, or on demand from another element.

Each element of the system interacts with other elements by one or more of these forms.

We elaborate somewhat on the commands for *control of operation* and their origins. Although we envision a somewhat centralised system, it is not the case that all commands originate centrally. Rather, operational control can be generated in various ways:

- State changes are recommended by the DSS;
- Users can take manual executive control;
- For closed-loop subsystems that involve multiple components, the controlling component will generate and transmit commands to the actuating parts.

To facilitate distributed analysis and decision-making, the *requests for data* can also originate from components other than the central core:

- Some subsystems may request data (e.g. sensory data; current real time; status; but also higher-level information such as foraging exclusion zones) from another element.
- Some subsystems may request the predictions from a model, including higher-level models.
- Some subsystems may request data that is external to the hive (from other HIVEOPOLIS units, from online resources)

We can expect messages transferring instructions (both control and data requests) to be small (order 50 bytes or smaller). In addition, messages containing data on system health can also be assumed to be small. However, primary scientific data could be small (e.g. a single sensor reading plus metadata) but also large (e.g. image, video, multi-channel audio). We collected the expected volumes of data for modules/components that are already sufficiently mature, and estimated values for modules that are yet to be developed. This is elaborated in Sec. 2.2.

2.2 Hive-Internal data transfer

We gathered estimates for every data-producing element in the hive. We have a small number of elements that can produce high volumes of data (images, multi-channel vibration signals), while the majority of elements produce 1 kbit/s or less. In some elements the internal data rates are far higher than will be transmitted (e.g. traffic flow monitors sample at 30 Hz but could aggregate the number of bees passing to two counts per minute) from the element to the core.

In the original collection of requirements we had three elements that made up 98% of the entire data transfer, including one module generating 90% of the data. We took the opportunity to re-evaluate our estimates for these major generators, and identified one or two orders of magnitude overestimation. The revised estimates are shown in Table 2.1, showing an expected production of around 67 kbit/s or 700 MB/day from the modules.

We acknowledge that some of this data transfer will contain low-level information which could be conditioned or processed, e.g., by time averages or by signal distributions. However, we note that optimising too drastically at this point could lead to major system overhauls in the case of underestimate or changed methodology in one or more modules.

	typical data production	percentage of data	bit/s	constant / periodic	period	operation
WP5	brood nest thermal	1.93%	1315.87	constant	1s	24h
WP5	brood nest scanner	29.34%	20000.00	periodic	1h	24h
WP4	dance inhibition (monitor)	65.77%	44837.32	constant	100 Hz	daylight (?)
WP4	dance robot, in operation	1.76%	1200.00	sporadic	1s	daylight
WP4	dance detection system	0.33%	222.53	constant	1s	daylight
WP4	front gate in/out monitor (high detail mode)	0.02%	12.00	constant	1 min	daylight at higher rate, 24h
WP4	front gate door closure mech	0.00%	1.07	event-driven	5 min	event-driven, not periodic
WP6	honey harvester (5 devices)	0.00%	2.70	sporadic (v rare)	1h	not sure what limits
WP2	power supply	0.00%	0.47	constant + alarm even	15 min	24h
WP3	flow observation, debug mode (2 devices)	0.01%	7.47	constant	2 min	24h
	flow observation, full system in typical operation (20 devices)	0.30%	202.67	constant	2 min	24h
	hive scale (assumes per-module scale, 10x)	0.00%	0.36	constant	1h	24h
	weather station	0.02%	10.34	constant	12s	24h
	GNSS	0.00%	0.12	constant	15 min	24h
	internal env monitor, wireless (10 devices)	0.45%	304.00	constant	10s	24h
	traffic control system (20 devices)	0.08%	56.89	sporadic		24h
		100.0%	**68173.8**	**bit/s**		
			66.6	kbit/s		
			0.01	MB/s		
			702.2	MB/d		

Table 2.1 - Estimated quantity of data produced by each element of the hive (module, sub-module, or other component).

2.3 Data generated by models

A number of models for predicting the colony dynamics will be executed on the hive. There are models for various aspects of the colony dynamics, including: foraging, brood nest, storage, bee traffic; as well as a model that is informed by these detailed aspects.

In Sec 4.1 we elaborate further on the computational resources necessary; here we focus on the data production. These models are not currently developed with sufficient maturity to tightly define the data budget, but we at least attempt some loose estimations.

Given 1-10 MB of output data per each of 5 models, and between 1 and 24 runs per day, we estimate a production of 5x12x5 = O(300MByte) per day. There are simple strategies to avoid retaining all output (e.g. progressively smoothing or approximating the historical predictions), and apply lossless compression. We estimate a factor of 8 reduction overall is achievable, meaning 40-60 MB/day storage, although working space for the full amount would be prudent - especially in case of the reductions causing additional computational load.

2.4 Hive-external data exchange

There are two aspects to external exchange: from the hive (uplink requirement) and to the hive (downlink requirement).

For the **uplink requirement** we make the conservative assumption that all data generated on-hive and passed over the communications bus is to be transmitted to cloud storage. For models, there are two options to be considered: transmitting all model output for archiving, or transmitting the model parameterisation only, thus necessitating each model to be run again on a non-hive resource. It is too premature to compute the energetic outlays for the hive in each of these options, but since the models must run on embedded/constrained resources of the central core it is unlikely to be a large computational burden to replicate the model runs. The parameterisation for each model, beyond what is in the sensory/state data, likely fits in 1-10 kBytes, meaning 600 kB as an upper limit for the approximately 60 model runs per day without any effort to reduce duplication.

The target therefore would be to transmit O(1GB)/day in the worst case (equivalent to 95 kbit/s), and O(700MB)/day if model parameters are sufficient (equivalent to 67 kbit/s).

For the **downlink requirement** we consider what might be downloaded. Information falls into the categories C-DT12 to C-DT16, which cover broad air weather conditions, weather forecasts, community reports on foraging availability, and alert signals from other HIVEOPOLIS units. The weather conditions and forecasts are mature and thus form a good starting point. An example weather forecast contains 20 kB and is updated every 2h (see Sec 6.4). Assuming that reports on four topics are obtained at this same rate, a daily requirement would be O(1MB). Even if 10x more topics are subscribed to, the downlink usage would still be insignificant in comparison to the uplink usage.

Note that these calculations are based on average requirements and do not take into account the temporal patterns of generation, e.g. burstiness, and accordingly a higher provision than the average should be targeted.

2.5 Storage

We identified the target duration of autonomous runtime (C-SI4) to be at least one full brood cycle, i.e., 21 days. However it is desirable to support 30 days data collection, not least since the onset of one brood cycle might not be known.

To store for a period of 21 | 30 days, we require:

- All messages generated by modules and transferred over the hive internal network
    - O(15GB) | O(21GB)
- All relevant data generated by the models
    - O(6GB) | O(9GB)

assuming that we do not make attempts to compress or summarise the data.

These calculations do not take into account storing all data received from external sources (Sec 2.4); however, we can assume that cloud-originating data can also be obtained and archived by off-hive resources. Accordingly, 1-2 days worth of storage is likely sufficient. In any case 30 MB additional storage is insignificant.

2.6 Hardware to fulfill requirements

External communications

From above we assumed the requirement to transmit *all data* from each system element to cloud storage. This estimate totals 100 kbit/sec. This order of data immediately renders long-range technologies such as SMS, LoRa and sigfox inappropriate. However, cellular technologies such as 3G/HSPA or 4G/LTE achieve these rates comfortably.[1]

An LTE device of CAT4+ has an upload capacity of 50 Mbit/sec, and hence will not be the bottleneck. A data plan at a reasonable cost is also available in Switzerland of under €10/mo, giving a limit of 10 Mbit/s uploads and downloads. Even assuming only 10% of the advertised bandwidth being achieved (e.g., due to signal strength in the apiary site), we still see that the target performance is easily within reach.

Based on the throughput analysis (Sec. 2.2-2.4) we narrowed the technologies to fulfill the downlink and uplink requirements for one HIVEOPOLIS unit and also its availability in EU countries. Table 2.2 summarises relevant specifications for the low-power wide area network (LPWAN) technologies considered in the device selection.

	Category			
	NB-IoT	**Cat-M1**	**Cat-1**	**Cat-4+**
Max. bandwidth	200 kHz	1.4 MHz	20 MHz	20 MHz
Downlink peak rate	~200 kbps	300 kbps	10 Mbps	150 Mbps
Uplink peak rate	~200 kbps	375 kbps	5 Mbps	50 Mbps
Latency	1.6-10 s	10-15 ms	10-15 ms	?
Duplex	half-duplex	half/full-duplex	full-duplex	full-duplex
# antennas	1	1	2	2
Max UE TX power	20/23 dBm	23 dBm	23 dBm	23 dBm
Relative power	very-low	low	medium	high

Table 2.2 - LPWAN cellular technologies based on 3GPP standards.

[1] https://kenstechtips.com/index.php/download-speeds-2g-3g-and-4g-actual-meaning is one useful source showing typical connection speeds from the various generations of 3G and 4G.

A survey of available COTS (commercial off-the-shelf) modules was performed in order to validate the compliance with the technical requirements of each HIVEOPOLIS unit (Table 2.3). To filter the modules to be evaluated we selected the devices that:

- Are compatible with the cellular network infrastructure availability in each of the consortium members countries.
- Have a high throughput, so during the investigation of different parts of the unit a comfortable amount of data can be exchanged between the HIVEOPOLIS units and external users/systems.
- Preferably has GNSS capabilities incorporated to the module, so no extra hardware will be required to fulfill the geolocation requirement (D2.1/C-SI10 and D2.1/C-DT6).

Therefore, we opted for modules that are LTE CAT4+ compatible and incorporated with GNSS capabilities, which pointed to two candidates for evaluation: SIMCom SIM7600SA and Quectel EC25.

			LTE						speed			interfaces						Voltage			
		GNSS	LTE-M1	LTE Cat 1	LTE Cat 4+	NB-Iot NB1	EDGE/GPRS	GSM / GPRS / EDGE bands	UL	DL	MCU	U(SIM)	AT cmds	UART	USB	I2C / SPI	Bluetooth	min	max	Dev kit?	PCIE module
Manufactu	**MPN**							MHZ	kpbs									V	V		
SIMCom	SIM7600SA	●	○	○	●	○	●	850/900/1800/1900	50M	150M	○	●	●	●	●	○	○	3.4	4.2	○	●
SIMCom	SIM7600G	○	○	●	○	○	●	850/900/1800/1900	5M	10M	○	●	●	●	●	●	○	3.4	4.2	●	●
SIMCom	SIM7600CE-T	●	○	○	●	○	●	900/1800	50M	150M	○	●	●	●	●	●	○	3.4	4.2	●	●
SIMCom	SIM7000E	●	●	○	○	●	●	900/1800	300	375	○	●	●	●	●	●	○	3	4.3	●	●
SIMCom	SIM808	●	○	○	○	○	●	850/900/1800/1900	85.6	85.6	○	●	●	●	●	●	●	3.4	4.4	●	○
uBlox	SARA-R410M-02B	◐	●	○	○	●	○	850/900/1800/1900	300	375	○	●	●	●	●	○	○	3.2	4.2	●	●
uBlox	SARA-R510M8S	●	●	○	○	●	○	850/900/1800/1900	1.2M	375	○	●	●	●	●	○	○	3	4.5	?	?
uBlox	LARA-R211	◐	○	●	○	○	●	900/1800	5M	10M	○	●	●	●	●	●	○	3.3	4.4	●	●
uBlox	TOBY-L210	◐	○	●	●	○	●	850/900/1800/1900	50M	150M	○	●	●	●	●	●	○	3.4	4.5	?	?
uBlox	MCPI-L2	◐	○	○	●	○	●	850/900/1800/1900	50M	150M	○	●	●	●	●	●	○	3	3.6	?	●
Quectel	EC25	●	○	○	●	○	●	850/900/1800/1900	50M	150M	○	●	●	●	●	●	●	3.3	4.3	?	●
Quectel	BG96	●	●	○	○	●	●	850/900/1800/1900	375	375	●	●	●	●	●	●	○	3.3	4.3	●	●
Quectel	MC60	●	○	○	○	○	●	850/900/1800/1900	85.6	85.6	○	●	●	●	○	○	●	3.3	4.6	?	?
Nordic	nRF9160	●	●	○	○	●	●	700-2200	300	375	●	●	●	●	○	●	○	3.3	5.5	●	○

Table 2.3 - Cellular modules to access LTE networks.

Storage media

From the storage estimates in Sec 2.5, we see a need for up to 30 GB data to be preserved in the medium term. Since the system will require temporary working space as well the overall requirement indicated is 64 GB removable storage (taking into account sizing of commercial storage). This provides a significant buffer, but we note that it is trivial to obtain industrial SD cards with 128 GB or 256 GB if these estimates turn out to be too small.

The longevity of SD cards could in principle be a concern if we have a large number of write operations. However, note that even budget SD cards with 1000 write cycles would last >>10

years if writing 1GB/day. Industrial SD cards are typically rated to 100,000 cycles or more (for example, the Sandisk XI range offers a 3K TB lifetime). Thus, the main considerations for hardware will be write speed and operating temperature ranges, rather than write cycles limit.

Chapter 3: Connectivity and interfacing

3.1 Communication protocols

The total data rates estimated are not prohibitively high for many existing protocols, such as UART, I2C, SPI, CAN, and Ethernet. So, we investigated protocols that would allow (a) the inter-connection of a variable number of devices, (b) that is capable of robust operation in noisy environments, (c) with a low number of conductors, and (d) facility to be integrated into future designs, taking into account different MCUs used by the various labs in the Hiveopolis consortium.

We provide a brief survey of candidate protocols that have the potential to be suitable (Table 3.1). To have a low number of conductors, the investigation was focused on serial buses.

Protocol	# wires	range	data rate	single ended or differential	duplex	Hot swapping
SPI	5	O(1m)	O(10Mb/s)	single	full	no
I2C ("TWI")	3	O(1m)	O(3Mb/s)	single	half	no
UART	3 (9)	O(1m)	O(1Mb/s)	single	half	no
USB 1.0	2	O(5m)	O(12Mb/s)	differential	half	yes
CAN	2	O(40m)	O(1Mb/s)	differential	half	yes
Ethernet	8	O(100m)	O(100Mb/s)	differential	full	yes
FlexRay	2 (4)	O(20m)	O(10Mb/s)	differential	half	yes
MIL-1553	2 (?)	O(30m)	O(1Mb/s)	differential	half	?

Table 3.1 - Selected technical specifications of serial communication protocols

SPI – The serial peripheral interface was created 40 years ago by Motorola with the objective of creating a communication channel between ICs on the same board. Although it is possible to achieve fast data transfers, we believe the SPI protocol has the following disadvantages: (a) 4 wires and common ground is required to establish a network, and (b) no addressing scheme, where the master controller needs to activate each slave device independently, requiring one extra line per device.

I2C – Like the SPI, the Inter-Integrated-Circuit was designed for communication between chips (i.e., short distance comms). But, it has some advantages over the SPI, like (a) the necessity of only 2 wires and (b) the possibility of addressing each device (therefore no need

for a separate chip-select). The I2C bus is half-duplex, meaning that data can move only in one direction at a time, and any device is capable of taking control over the bus. Like the SPI, the single-ended "construction" of the protocol, requiring the share of a common reference signal, imposes some challenges when trying to create a small network between off-board devices, especially when taking into account hot-insertion capabilities and bus control arbitration.

UART – It is an asynchronous serial link with some simple coding established between the transmitter and receiver to correctly send and receive the data (e.g., bit rate and data parity). Most modern microcontrollers are equipped with UART channels that can be translated to differential modes for increased ranges (e.g., RS-422) and multi-drop capabilities (e.g., RS-485). Data flow control can be achieved at the expense of extra cabling.

USB – The Universal Serial Bus was created in 1995 with the intention to simplify the connection between computers and peripherals, with relatively high speeds (12 Mb/s). Versions 1 and 2 are half-duplex point-to-point connections with 4 wires (1 differential pair, power and ground). The USB network should have a star topology with devices connected to active hubs. Also, USB connections can be hot-swappable.

CAN – It is a very robust protocol found in very diverse applications, like small instruments, all types of vehicles (robots, drones, cars, agricultural machines, airplanes, satellites) and plenty of industrial applications. Two important features that add to the robustness of a CAN bus is the use of a differential pair, to increase signal integrity, and its native capability of data validation built in the hardware, like CRC checking and collision detection. The CAN network is a multi-drop bus that can operate over distances of 40 m up to 1 Mb/s. Every device on the bus can broadcast messages that are limited to a payload of up to 8 bytes (if bigger data needs to be transmitted, it is required to fragment the data in blocks of 8 bytes. The new CAN FD allows packets sizes of up to 64 bytes).

FlexRay[2] – With the increasing demand of data in automotive applications a new bus was designed with thoughts of a possible future replacement for CAN buses. The FlexRay bus can achieve speed of 10 Mb/s through the use of one or two differential pairs as a transport medium connected between the nodes. Despite its fast data transmission and large data payloads capabilities, the number of devices supporting such technology is extremely limited.

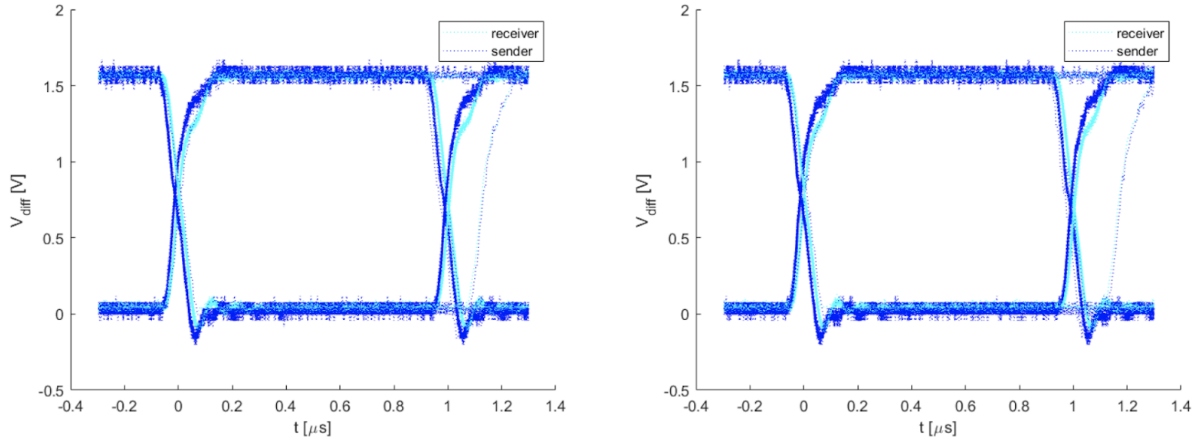
MIL1553[3] – Is a military standard developed in the 1970s for application requiring extreme robustness, like aerospace assets. The protocol utilizes Manchester encoding transmitting a clock signal with the data signal. The bus is limited to 32 devices that are coupled to the bus via a transformer. Since this technology is used mostly on military applications, the number of integrated devices implementing its functionalities are scarce.

[2] https://www.fujitsu.com/downloads/CN/fmc/lsi/FlexRay-EN.pdf
[3] https://www.ueidaq.com/mil-std-1553-tutorial-reference-guide

Ethernet – It is the technology used in most modern computer networks. It is a point-to-point serial bus requiring an active hub to connect multiple nodes. It is efficient when transmitting large data packets (an overhead of ~2% for 1500 bytes packets) but performs poorly when transmitting very small quantities of data since the minimum data packet size is 74 bytes. Also, Ethernet buses use 4 differential pairs, requiring 8 conductors (although it is possible to change medium, using two fiber-optic cables).

Presently, we are investigating the use of a CAN bus to transport all commands and data generated inside a HIVEOPOLIS unit. Despite the payload limitation of 8 bytes per message, the CAN protocol seems to be a good alternative for its robustness (differential signaling protecting against common-mode noise, hardware level data integrity validation, and usually requiring just one simple transceiver to allow a device to be connected to the bus) and capacity to achieve the data rates required by the modules.

In a first exploratory phase, we wanted to check if a 2 m long CAN bus with a different number of stubs would be able to handle all data communication of a HIVEOPOLIS unit. A simplistic experiment ("sanity check") was performed by adding multiple 1 m long open stubs to the bus and checking the signal integrity (e.g., distortions that could cause inter-symbol errors, crosstalks, jitter, ringing, etc.). One anecdotal result is presented in Fig. 3.1, where a 1 MHz data stream was injected into the bus and displayed in the form of an eye diagram. The "open eye" shape of the graph represents an example of a good result.



(a) 2 stubs (b) 5 stubs

Figure 3.1 - Eye diagram of multiple transmission in a CAN bus with (a) 2 stubs and (b) 5 stubs.

3.2 Docking

During the Hiveopolis project elaboration, a modular design was envisioned where diverse types of functional modules could be installed or swapped to different parts of the hive. Hence, in order to facilitate the module mobility, a docking interface will provide access to the hive infrastructure (e.g., comms bus, power rails, pneumatic lines), as required by the project technical specifications (see D2.1 C-DCK1 to C-DCK6).

One important aspect of the docking interface is to enable the central core and other modules to know exactly where each device is located in the hive and what is the type of that device. To accomplish such a requirement (D2.1 C-DCK6), each module is designed to have a persistent UUID (Universal Unique ID) that is combined with a docking slot ID, allowing the creation of an occupancy map of the hive. Currently, the docking slot identification is being thought of as a 5-bits hard-wired code (32 possible IDs) that is contact-transmitted to the module upon its installation to the hive. After power, the boot sequence of each module will read the slot ID and incorporate it in all exchanged messages. The final identifier, including module and slot IDs, should have a maximum size of 11 bits. Once a docking operation (insertion or removal of a module) is performed, it will be time-stamped and registered into the hive operations log (D2.1 C-DT4).

Also, another essential part of the docking interface is the provision of power to the modules. The consolidation of power supply lines is planned for 2021. Most present prototypes use 5V and 12V, however, some devices also use 3.7V and 24V. The power requirement for each module is being assessed by each group (D2.1 HO-PWR7) and will guide the design of the HIVEOPOLIS unit power supply (T2.3, D2.2), the selection of adequate wiring and connectors.

Chapter 4: Software infrastructure

This chapter considers the computational and software functionalities that the central core will provide. We first outline the goals before describing the collected performance requirements. We evaluate candidate solutions to facilitate these requirements including hardware and high-level approach to software, and then detail some of the components that have been prototyped.

The central core should provide computational facilities for a number of tasks, including execution of colony dynamics models, hosting of analysis for modules with constrained computing capacity, and the decision support system. Additionally, the central core should host a local database, a key element of the data warehouse solution (which will be elaborated in D3.3).

Besides these higher-level tasks, the core also provides some lower level functionalities including the orchestration of the module to module communications, the hive to external communications (through the archiving, external services, and remote access gateways), keeping track of attached modules, and providing a precise time for system synchronisation.

4.1 Computational requirements

In this section we analyse the requirements for computation that should be facilitated using central core resources. The tasks that require substantial computational resources include:

- Execution of predictive models
- Central analysis, including processing of larval image sequences
- Decision-support system and executive control
- Local data warehouse management

Predictive Models

We have a number of models that will predict different aspects of colony activity and how they will change over time. These range from foraging recruitments to the population growth, varying drastically in timescale of interest. The models will not be finalised until later stages in the project but we attempt to estimate the computational resources necessary to service these models.

A summary of the anticipated requirements is as follows:

- A higher-end MCU is likely to be sufficient to execute some of the simpler models; a posix-type processor with >= 1GB RAM will certainly be sufficient.
- For ABM-type models, the computational needs can be intensive (typically desktop up to compute cluster usage, see e.g. Abar et al., (2017)) although the time and spatial scales and number and complexity of agents all influence the needs.
- At least 32-bit architecture, supporting floating point in hardware, is strongly preferred. In the numerical integration of ODE-type models, the accuracy of small values might benefit from 64-bit variable support.
- The models will use parameterisation and variable values from hive measurements, from other models - depending on their position in the hierarchy, and from external sources via the services gateways (including other hives). The consumers of the model data will be other models, the decision-support system, and human users of the hive.
- The role of the models is to make predictions of the colony dynamics, and identify potential health issues, opportunities for certain objective maximisation, and the like. These are primarily slow-moving, i.e., they are unlikely to be useful in terms of guiding sudden responses such as attacks on the hive. Rather, such responses should be based on more direct sensory information. Accordingly, the acceptable timescales for requesting a prediction and having it ready can be measured in hours (although a hierarchical model would likely require the lower level parts to be executed more promptly). This indicates a possibility for enqueuing/scheduling the modelling tasks to specific windows, if need be.
- Some storage will be needed to store mode results, and although it is too early to define the precise size produced, from preliminary work we estimate O(1MB) to O(10MB) per model run.
- There exists the possibility that data conditioning or pre-processing will be required ahead of model runs. In principle the structure of the databases may be able to alleviate such loads, but it remains for specific data flows to be analysed.

Given this somewhat open set of requirements we propose to meet the needs from two different angles: 1) evaluation with a stand-in colony model; 2) structuring facilities to permit future expansion.

Evaluating model costs

The first approach is an evaluation using an existing detailed colony model, BEEHAVE (Becher et al., 2014), to represent a moderate and realistic computational load. We measure the resources used by a typical simulation run using four different single board computers (Raspberry Pi Zero, 3B and 4B, and the Beagle Bone Green).

The runtimes for two NetLogo simulations are shown in Fig 4.1. For simple tasks, many different platforms are acceptable but the relatively complex model, BEEHAVE, is a very heavy load for the more resource- constrained devices. In terms of memory usage, the heap utilisation is between 2MB-4.5MB and the virtual memory footprint varies according to availability, (Zero: 600MB, BBGreen: 600MB, Pi4B: 1200MB, i7: 4000MB).

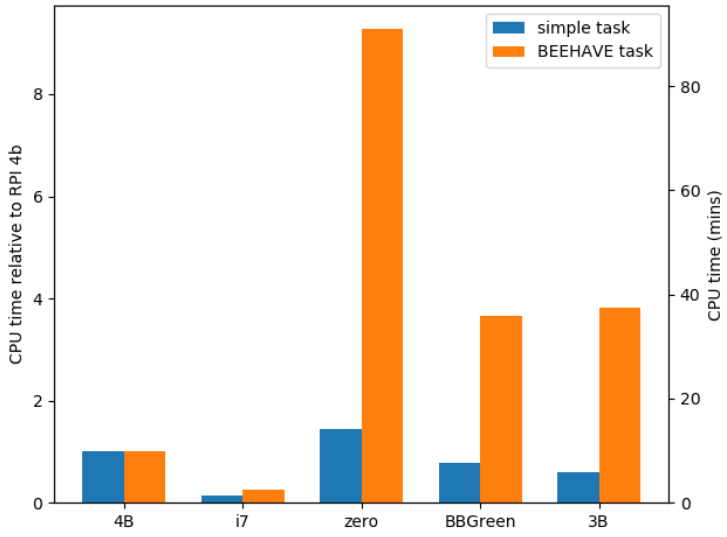


Fig 4.1 - Simulation model execution time, for two different NetLogo simulation models, each job containing 5 replicates. Hardware platforms include 4 SBCs and a typical laptop equipped with a 2.7 GHz i7 processor.

Future expansion of computational resources

The second approach is to examine structures for the core computational facilities that permit future expansion. For instance, by having an additional processor (or SBC) that could be enabled when the system is demanding more computational resources, empowering the hive with an energetically efficient dynamic load balancing system. Another possibility is for heavy computational tasks to be offloaded to cloud resources. This depends on an efficient transmission of the required parameterisation, input data, and computational results to be feasible, and is likely reasonably well suited to model runs but less well suited to data analysis.

Other computational tasks supported on the core

The **data warehouse** is being developed within task T3.4, and while this design will be reported in deliverable D3.3, at the level of defining requirements the following information is appropriate to outline.

- The architecture involves a local store on each hive unit as well as longer-term storage in cloud resources.
- The local store will involve some form of database on the hive, to appropriately organise the diversity of data (e.g., sensory, command logs, simulation outputs).
- It is likely to use an in-memory buffer combined with an in-file database, to avoid the bottlenecks associated with relatively slow storage media.
- The data budget established in Chap. 2 gives some framework to select an appropriate database, and moreover the anticipated connections will be primarily mediated through core software components for record insertions and queries. Specifically, the majority of record creation will originate in the modules, and the majority of query consumers will be feeding the predictive models.

These broad requirements can be serviced by many database management systems (DBMS) that fit within the resources of constrained devices, pointing to the likely need for a linux-type platform. Some systems could operate in <1 MB memory and <1 MB for libraries (e.g., SQLite[4]) while others minimal operations are plausible with 128 MB memory and recommend >1 GB (e.g., InfluxDB[5]). There do exist some options for more constrained devices (e.g., TinyDB (Madden et al., 2005) that is able to run on smaller devices like 8-bit microcontrollers).

The **decision support system** will be executed on the central core. This is also being developed in another task and will be reported elsewhere. For the purposes of the core infrastructure, we can note that the computational load of existing systems are typically quite light (although their development can be intensive if using machine learning for example, the runtime of a trained model is not typically heavy), but that the response time should be paid attention to. Some events in the hive could require rapid detection and action proposals (e.g. swarming events could be detected up to an hour ahead (Zacepins et al., 2016; Kviesis et al., 2020)).

Some **data analysis** will also be hosted by the core facilities e.g., processing of the larval scans. The programs are being investigated elsewhere and are not mature enough to evaluate the computational effort in detail, aside from the low frequency (3-6 hour intervals between snapshots are informative, see D5.1). There also exist some data pre-processing steps that might be best facilitated on the core, such as translating between absolute foraging target zones and relative coordinates/descriptors for waggle dances.

[4] https://www.sqlite.org/footprint.html
[5] https://docs.influxdata.com/influxdb/v1.8/guides/hardware_sizing/

4.2 Hardware

As a main processor to control a fairly complex field equipment, single-board computers (SBC) offer an interesting possibility to have relatively large computational power in a small package, with a small power budget and also a small cost. Moreover, the POSIX-type environment available and support for high level languages such as R and Python make them very attractive from the perspective of a wide user base. This general architecture can adequately service the requirements of complex predictive models, the local components of the data warehouse, and some signal processing or image analysis operations that are not handled within the modules directly.

In comparison to microcontrollers, SBCs have shortcomings in energy budgets and purchase price, but the primary issue is the poor guarantees on event execution timing. Because Linux operating systems orchestrate multiple programs simultaneously, most flavours of Linux do not offer strict guarantees on execution timing. However, our latency demands, for biological events, are rather slow in computing terms and a lag of 100 ms in operation servicing, for example, will not impact the ability to interact with the bee colony.

The communications must be treated with care, for high data rates combined with small buffer sizes in system nodes might cause deadlocks, data loss, or other issues. Thus, a possibility of real-time handling for the communications would be advantageous to simplify avoiding these issues.

Hardware survey

The market for single-board computers is large, and we provide details for a handful here (Table 4.1). The key notable features for devices are:

- Higher loads could be facilitated with GPU computing (particularly Nvidia Jetson, but also NanoPC)
- Real-time IO can be achieved with dedicated hardware (Beaglebone)
- General-purpose device with extremely wide community (Raspberry Pi family)

As the detailed performance requirements from several aspects are not currently defined, we still must defer the choice somewhat. Several subsystem prototypes have used RPi platforms (motivated in part by their support of image acquisition); however, most software is inter-compatible across the majority SBCs above. Note however that a physical implementation of a given interface is particularly valuable for achieving high service rates. In Sec. 3.1 we outlined a network using CAN bus to provide robust on-hive communications; accordingly a CAN interface is sought on the primary compute node.

An important consideration for what is required from the platform is not just how many FLOPS will suffice, but rather, how will these computational programs be developed - and moreover, by whom? The decision support system, for example, is being developed by a team experienced with constrained systems programming. However, the predictive models in particular are being developed by many members of the consortium including biologists

who are computer literate but not well versed in programming microcontrollers. This consideration puts considerable weight on useability of the platform.

Product	**Processor**	**I/O**	**O/S**	**Current (active)**	**Current (idle)**
Raspberry Pi 4B	Broadcom BCM2711 (4xARM Cortex A72) 64-bit	I2C, SPI, WiFi, Eth, BLE, USB, Camera	Linux (debian)	1200mA [1] 600-1000mA [4]	600mA [1] 460mA [4]
Raspberry Pi Zero W	Broadcom BCM2835 (ARM 11) 32-bit	I2C, SPI, WiFi, BLE, USB	Linux (debian)	150mA [1] 180mA[4]	80mA [1] 110mA[4]
Friendly elec NanoPC-T4	RK3399 2core+4core CPUs. 4 unit GPU 64 bit	2x camera, ETH, WIFI, BLE, I2C, SPI, I2S, USB	Android or linux (debian)	1100mA [2] up to 2000mA [3]	420mA [2]
Libre Renegade Elite	RK3399 2core+4core CPUs. GPU	USB, UART, ETH, 2x cameras	Android or linux (debian)	1200mA[2] up to 3000mA [3]	165mA [2]
Beaglebone black / green	AM3358 ARM Cortex-A8. ARM M3 co-processors for real-time. 32-bit	I2C, CAN, SPI, ADC, ETH, WiFi	Linux (debian)	450mA [2] 1200mA [3]	310mA [2]
Nvidia Jetson Nano	4xARM A57, 128-unit GPU. 64 bit.	I2C, I2S, SPI, UART, ETH, USB	Linux (debian)	2000mA [3]	
Beaglebone AI	AM5729 (2x ARM cortex A15 + 2x ARM M4). 2-unit GPU and DSP cores.	I2C, CAN, SPI, ADC, ETH, WiFi	Linux (debian)	2500mA [3]	

[1] from data sheets of manufacturer
[2] benchmarking data, various sources
- https://libre.computer/category/benchmark/
- https://learn.adafruit.com/embedded-linux-board-comparison/power-usage
- https://www.mickmake.com/post/nanopc-t4-no-room-for-anything-else-review/

[3] (no information available) power supply recommendations taken
[4] our measurements

Table 4.1 - Survey on single-board computers (SBC)

4.3 Orchestration and supporting functionalities

System orchestration and control

The central core has several needs as described above, whose functions are coordinated through the structure overviewed in Fig. 4.2. Specifically, the hive's modules and components interact via communications network, with low-latency queues for command and data handling; while non-urgent tasks including predictive models, analysis, and exchanging data from the external gateways are managed through scheduling. Besides these key orchestration functions, several supporting functionalities are also shown and described within this section.

Executive control should be feasible both by allowing the decision support system to run in an automatic mode and with human control (via the remote access API). With a general approach of separating low-level and high-level commands, either human or automatic sources of control can operate using simple algorithms, with details of communications remaining abstracted by the communication layer and modules interpreting the high level commands into low-level actions.

Two key **messaging patterns** enable the data transfer regimes outlined in Chapter 2: a publish/subscribe type pattern facilitates efficient sharing of new data that is relevant to multiple endpoints and inter-module closed loops, without the need for the core to explicitly request all data acquisition, while a request/response type pattern allows for straightforward organisation of long-running (or at least uncertain-duration) operations.

A defined message **serialisation format** enables structured and efficient communication between system elements. Many mature formats are available (e.g., CBOR, flatbuffers, protobuf), and we select the schema-based Google Protocol Buffers ("protobuf") due to the availability of compact libraries for both POSIX and embedded targets. This method is extensible to add new messages, even if it is not feasible to distribute the extended schema to all elements, and libraries for many common languages (e.g. C++, Java, Python, GO) add to the accessibility by the project stakeholders.

Scheduling There are several computational tasks on the core, including predictive models, fetching and pushing data to the cloud. These tasks are mostly low priority, with approximate timings that could be 1 hour or longer between request and requirement of the result. Accordingly, a high-level scheduler may assist with task execution. A purely time-based system is not sufficient since dependencies exist. Several lightweight open-source solutions that support periodic task launching, dependency chains and failed task handling. More detailed requirements regarding the timing of specific task chains will guide the specific choice (e.g. what can run at night, what are the execution horizons, etc).

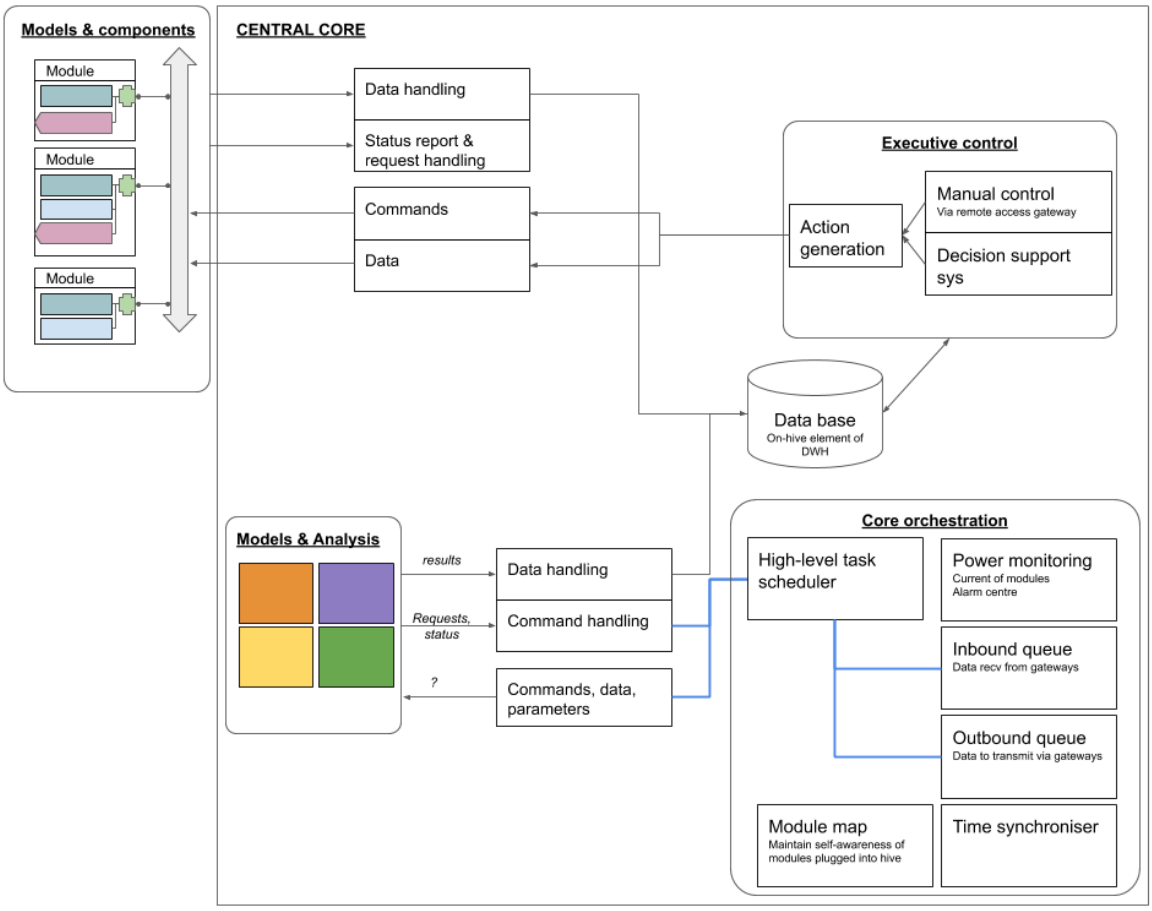


Fig 4.2 - Overview of central core data logic, indicating the key data transfer categories between core and modules, and between the components within the central core.

Micropython as a high-level interface to modules

The development of embedded firmware demands a very specialised skillset. Usually, programming microcontrollers is a non-trivial task due to the complexity of software tools, the constraints caused by the lack of user interfaces and the required acquaintance with fundamentals of electronics and the hardware that one wishes to program. In the Hiveopolis project, many different types of users will be creating code to interact with the developed instruments with all kinds of purposes (e.g., simulation of behaviours through mathematical models, collecting sensory data, injecting signals in the hive, etc.).

In order to facilitate the access to lower level hardware, we investigated the use of an abstraction layer to facilitate the access and control of the module using the python language. We took the brood nest electronics system as an example module, featuring arrays of sensors and actuators and some self-monitoring functionality. To achieve this objective, we ported the micropython[6] interpreter to our MCU running a RTOS, adding

[6] https://micropython.org/

functions to allow the reading of temperature sensors, the activation and configuration of thermal actuators, and the possibility of running algorithms locally. The RTOS (written in C) handles low-level operations including interaction with the sensors and actuators, while the micropython thread exposes access to these devices at a higher level (without needing to know about the communication protocols used or adhere to the strict timing requirements of the microcontroller, for example). The micropython terminal, exposed to the user, is very similar to a "regular" python shell and contains a subset of the python language. Any person familiar with the language is able to program and control the board after a few moments of experimentation.

Updating firmware

As a research platform, we must anticipate code updates in various modules, and accordingly we are interested in facilitating this as simply as possible - ideally, possible in the field. For the core to push updates to modules we use device firmware updating methods, for which we have three components with examples currently tested.

In the **wireless environment monitors**, the nRF52840 microcontrollers use firmware that implements a secure DFU service alongside its sensing and communications services. This service allows for firmware updates without button press (i.e., without physical access to the device). At present our implementation is functional with an Android device to transfer the new packaged firmware (see Fig 4.3). Since the process is reliable and straightforward for the user, it is of an acceptable duration with a small number of nodes. In the next implementation phase we will further investigate a command-line only setup that uses another nRF device physically connected to the core (e.g. via USB), and allows FOTA fully programmatically.

In the **brood nest thermal module**, we use a STM32F405 microcontroller, and in the current development we primarily use USB to interface with the module. Two additional wires are brought from the module to the hub (a RPi 4B), the boot-mode enable and the nreset line. When the boot line is brought high with an RPi standard digital GPIO, the USB interface exposes a DFU-ready mode. The command-line tool dfu-util is simple to transfer compiled .hex files to the microcontroller. The **dance inhibition module** uses a similar STM32F4 microcontroller and employs the same regime.

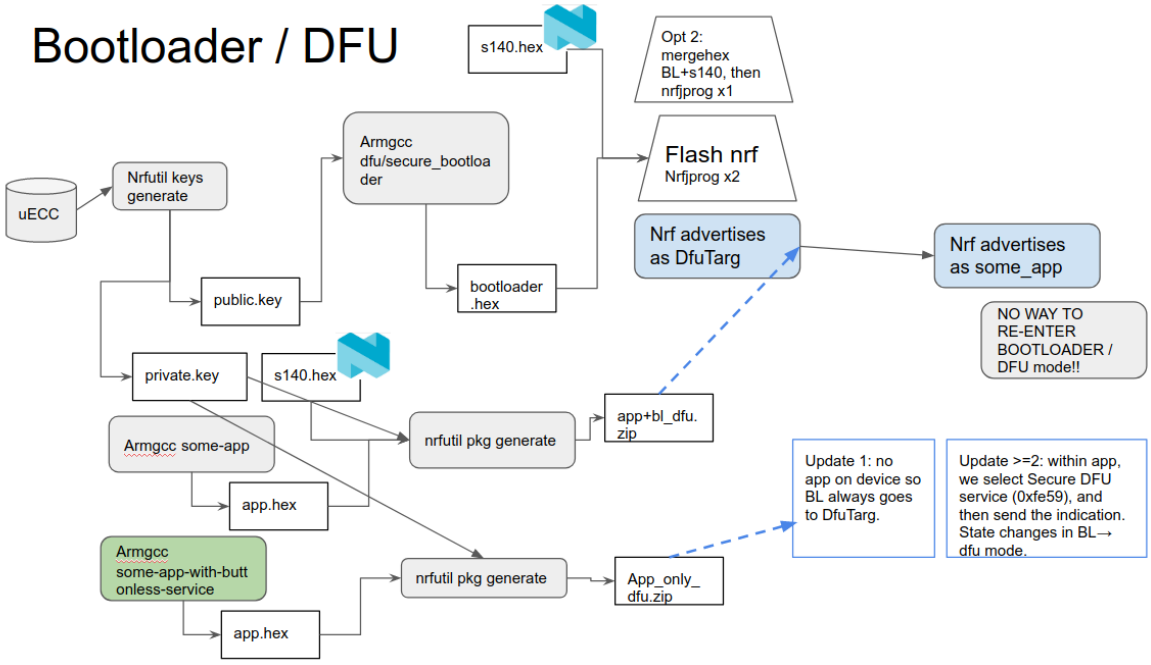


Figure 4.3 - Updating firmware on the nRF5x. Provided the Secure DFU service is implemented on the node, it is possible to transfer firmware updates without the need for any physical contact with the device (which is important when the device is deployed in a difficult-to-reach location inside the hive, for example).

As a brief evaluation, we note that the use of a smartphone or a graphical tool is a weakness in the environmental sensor node updating, especially noticeable with a larger number of devices. However, a totally buttonless, wireless solution is highly beneficial, maintaining flexibility for each device's location. Conversely, the process for the STM-based modules is smooth and automatable, but currently uses additional data lines. This adds a burden to the physical interfaces, which aim to minimise connection count. We are investigating alternative methods that would be compatible with the networking design.

Reliable timestamping

For any data analysis that depends on results collected from multiple sources -- for example, modules across one hive unit -- the events should be annotated in the correct sequence. Thus, a reliable timestamp should be available for all modules (C-SI9). The timestamp, along with the unique and persistent identifiers for each module are key in the metadata that must annotate data (C-SI12) and commands (C-SI8) that are stored.

Several of the components represent time at an integer resolution, and accordingly do not require greater accuracy than ±0.5s. There are modules or components that internally consider far higher frequencies for sensor sampling or actuator control, but these are not sensitive to high precision alignment with other modules.

In Sec 2.6 we described solutions for data transfer that also included a GNSS unit. The LTE component provides a timestamp but in the case of problems with network connectivity, there is still a highly reliable time source in the GNSS.
In a prototype system used for the brood nest thermoregulation module, the single-board computer (SBC) that logs data is connected to NTP servers for time synchronisation. Approximately once per 30 mins the servers are verified and system time is adjusted automatically to global reference time. The adjustments are typically under 1ms which are performed on the majority of connections. The SBC clock is used as a local reference to set the internal time within the modules. Specifically, the brood nest module sends the acquired data to the SBC that compares the data timestamp (controlled by the module real-time clock) against its reference time. If the time difference between the two clocks is bigger than 1 second the SBC will send a command to resync the module time to the SBC one.

Chapter 5: Biological sensing

One of the roles of the central core is to collect data regarding the colony status and related data, and making it available externally (to the cloud, for bee-keepers, scientists and other stakeholders; and to other HIVEOPOLIS units), and internally (e.g., to the predictive models and decision support system). Besides the collation of information as detailed in Chaps. 2-4, the core also directly provides some sensing and monitoring functionalities.

This chapter concerns direct measurement of biological variables:

- Section 5.1 describes our efforts towards measuring the flow of bees inside the hive.
- Section 5.2 briefly summarises the goals and techniques on weighing beehives.

The subsequent chapter concentrates on measuring environmental variables with biological relevance.

5.1 Measuring the flow inside the hive

The objective for this segment of work is to obtain data on the mobility of bees inside their hives (C-DT10). Besides the direct activity rates being used in the control loop of the traffic control system, the data will be used as inputs to the analytical and predictive models on colony health.

Modalities of sensing for honey bee traffic

There exist prior works on measuring in-out traffic of the whole hive, which provide valuable information regarding candidate methods to sense honey bees in the setting of inside the hive.

The most common approaches are optical, including IR beams (Danka & Beaman, 2007), cameras (Chen et al., 2012; Kulyukin, 2017), and there have also been attempts to use other physical phenomena including capacitance (Campbell et al., 2005) or motion observed by

radar (Souza Cunha, 2019). Another, more invasive approach is to attach RFID tags (Klein et al., 2019) or passive barcodes (Boenisch et al., 2018), but such an approach is not appropriate for large scale or field-based deployments that we aim for in HIVEOPOLIS. Thermal measurements have been used to measure group activities at the hive entrance (e.g., Peters et al., 2019; Stabentheiner et al., 2007), but not to measure traffic to our knowledge.

Some of these techniques are not well suited to the internal hive environment of a field-based hive system and accordingly can be ruled out without empirical evaluation. Firstly, camera-based systems require too much power and require lighting that would introduce light pollution (although it is possible to use IR spectrum); moreover the computational power needed for the video processing is non trivial. LIDAR systems are too expensive. As mentioned, attaching any form of marker (e.g. RFID tag) to each individual does not suit the field setting that HIVEOPOLIS aims for. Elsewhere in the project, capacitance-based bee detection is being investigated; to avoid duplication of the exploration we leave capacitance modality aside for our study aiming at internal flow monitoring.

Accordingly, we focused our investigation on exploiting infra-red light, thermopile-based heat sensing, and laser-based time of flight sensing.

Design

After initial trials on the workbench, we selected the following devices (Fig. 5.1):

- IR light sensor: Vishay BPW85C (1st and 2nd iteration prototypes)
- Thermopile: Melexis MLX90614 (1st iteration prototype)
- Laser-time of flight sensor: ST VL6180X (2nd iteration prototype)

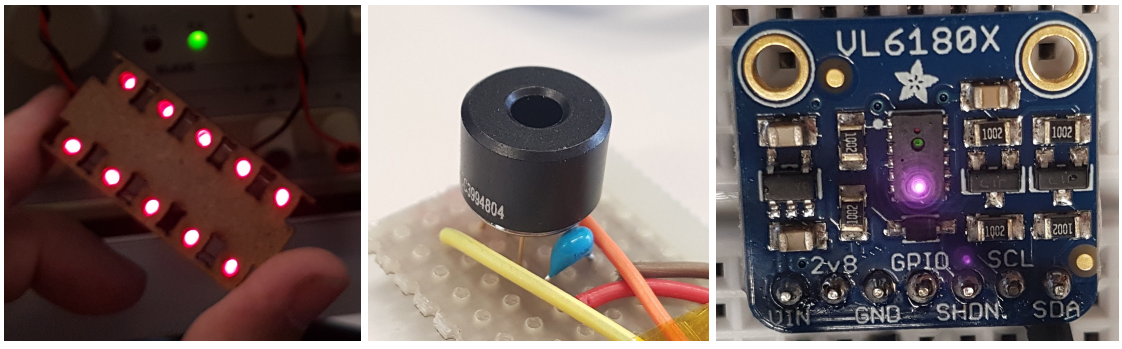


Figure 5.1 - Three analyzed sensor: IR light TX/RX, thermopile, and optical time of flight

The IR light sensors were coupled with IR LEDs OP165A (Optek), which emit light at 935 nm. This is beyond the sensitivity of honey bees, who are only sensitive to light with wavelengths under 650 nm. The thermopile does not need a second source since the bee body emits heat. These sensors were built into a housing with 5 channels, each of width 8 mm and height 6 mm (after Chen et al., 2012). Each channel had two IR sensors, and using dividing walls between the channels ensured that the pairs of sensors should be triggered by the same animal passing through (Fig. 5.2). Data was sampled at 150 Hz by an arduino

MEGA. The test harness for the flow monitor also included a Raspberry pi camera V2.1, connected to a Raspberry Pi 3B; this single-board computer captured the video data and also managed the arduino MCU, simultaneously gathering sensor data that was transmitted over the serial connection. An LED was used for synchronisation of the video and the sensor data streams.

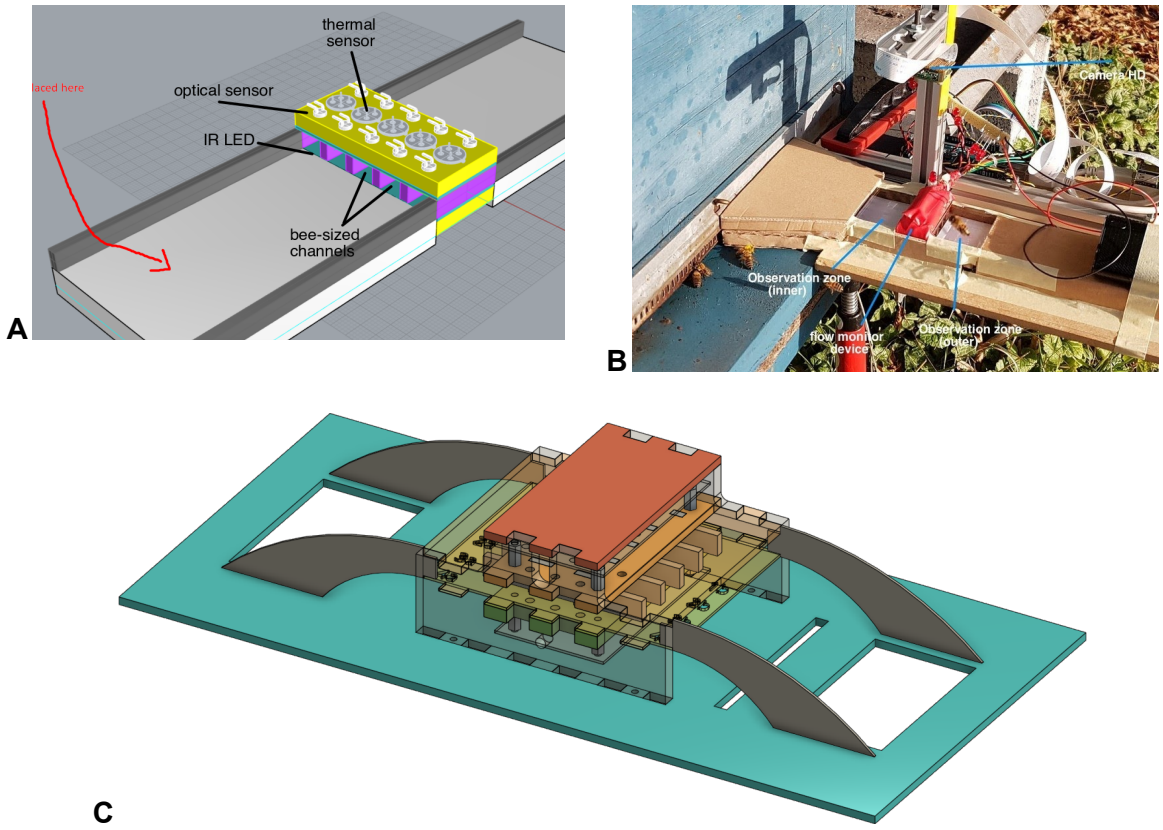


Fig 5.2 - Design of bee traffic flow monitor prototypes. A) First iteration design of multi-channel block with IR sensor and thermopile sensor. B) monitor in test harness, complete with funnel to catch bees from hive entrance, camera, and microprocessors for data acquisition. Note that the monitor is being tested at the hive entrance for initial data collection but the target is for internal flow monitoring. C) second iteration of design, including platform that allows for gravity-propelled inanimate objects (during winter).

Results

We performed several field tests to gather data with bees passing through the device in both directions. The events of bees entering and exiting each channel were annotated from the videos.

With 2 sensors per channel, we are able to detect bees moving through the device, and with high reliability, detect the correct direction of motion as well as bees that enter a channel but then turn around and leave (see Fig 5.3). These data are validated by comparison to ground-truth videos simultaneously captured.

The time it takes for a bee to pass through the channel can be measured; the distribution from n=82 events is shown in Fig 5.4. The fastest bee passing either sensor (i.e., when the bee's head first breaks the beam to when the abdomen stops breaking the beam) takes 170 ms. From this period we conclude that a sampling of 2x1/0.170 ~12 Hz should be sufficient to detect the passage of bees without missing any. Interestingly, bees pass the 2nd sensor faster than 1st one (Wilcoxon rank-sums, $p<0.001$). Thus, we conclude that bees accelerate through the channel, which indicates that the channel dimensions may be too restrictive and in a future iteration we will test larger dimensions.

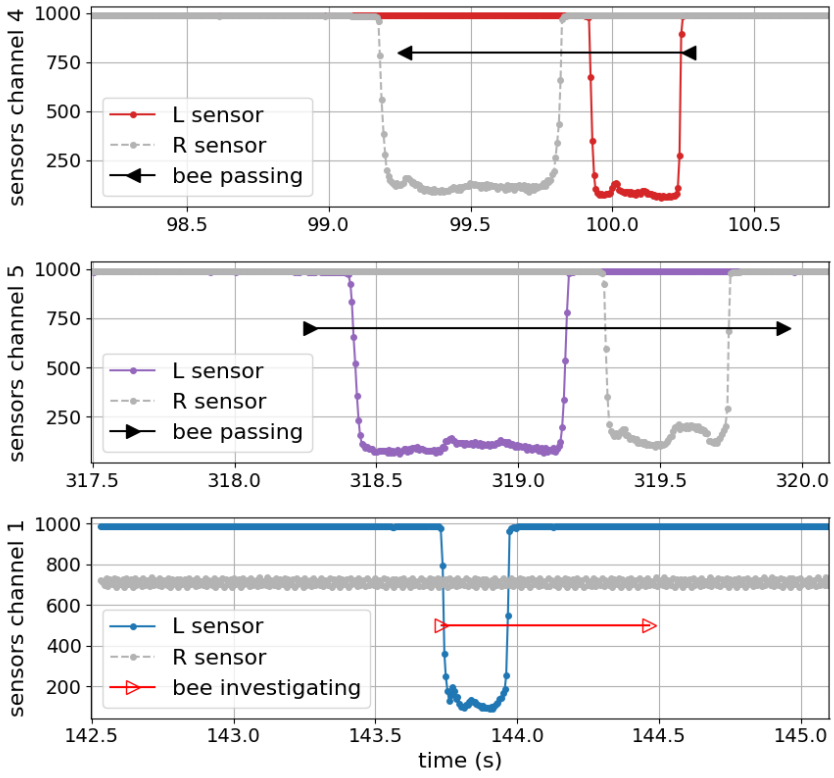


Fig 5.3: Examples of bees entering the channels with IR sensors, including annotations of the true events. Graphs show the raw sensor readings over time (high values reported when no opaque obstacles are present). Top row: a bee passing from right to left; middle row: a bee passing from left to right; bottom row: a bee who "investigated" the channel but ultimately turned around without passing through.

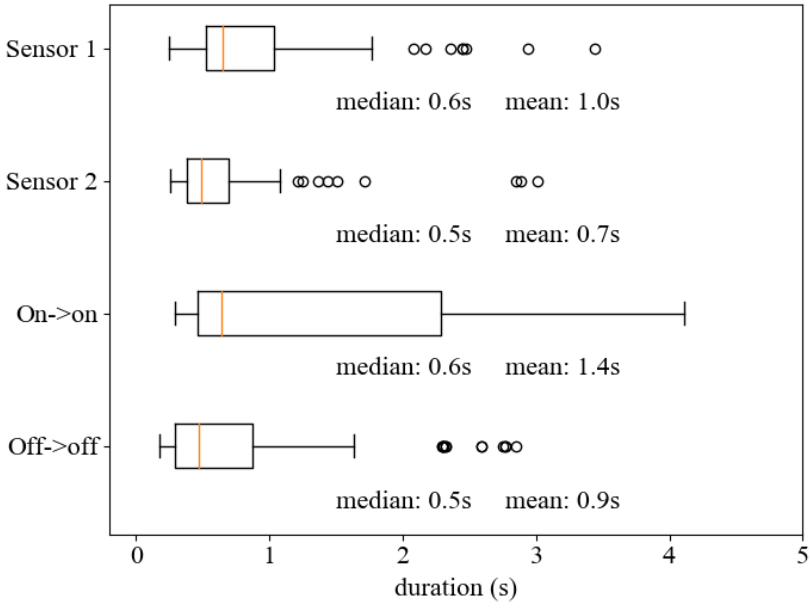


Fig 5.4: distribution of time sensors are triggered during an event of a bee passing through the channel. The sensors here are considered logically as 1st and 2nd, whether the bee went from left to right or right to left. The time difference between onset of 1st and 2nd signals are shown in the third row, and the final row shows the time difference between the falling edges.

Since the thermopile shares a channel with two IR sensors, it is possible to make an inter-comparison. When a bee passes through this channel, a significant deviation results in temperature. Interestingly, the deviation is not always an increase in temperature (see Fig 5.5). This may be due to the bees being in different states (Stabentheiner et al., 2010 measured ectothermic bees at 25 °C and endothermic bees at 32-36 °C). To use as a discrete event detector, the thermopile is sufficient but the maximum sampling rate of 10 Hz is too slow to obtain the profile of a passing bee.

The height of the head, thorax, and abdomen have a characteristic profile that could, in principle, reveal the direction of motion. However, the differences between these three points are only in the order 1-2 mm, and would require high precision measurements within the 170 ms passage time to be reliably informative of the honey bee's direction. At a sample interval of 15 ms, 12 samples could be taken. Idealising these to 4 each per head, thorax, abdomen is an (unrealistically) optimistic case. We found that the probability of misclassification is at least 30% even under these circumstances, and when we must avoid misclassification three times in a row to correctly match an idealised animal, it becomes clear that this route is infeasible.

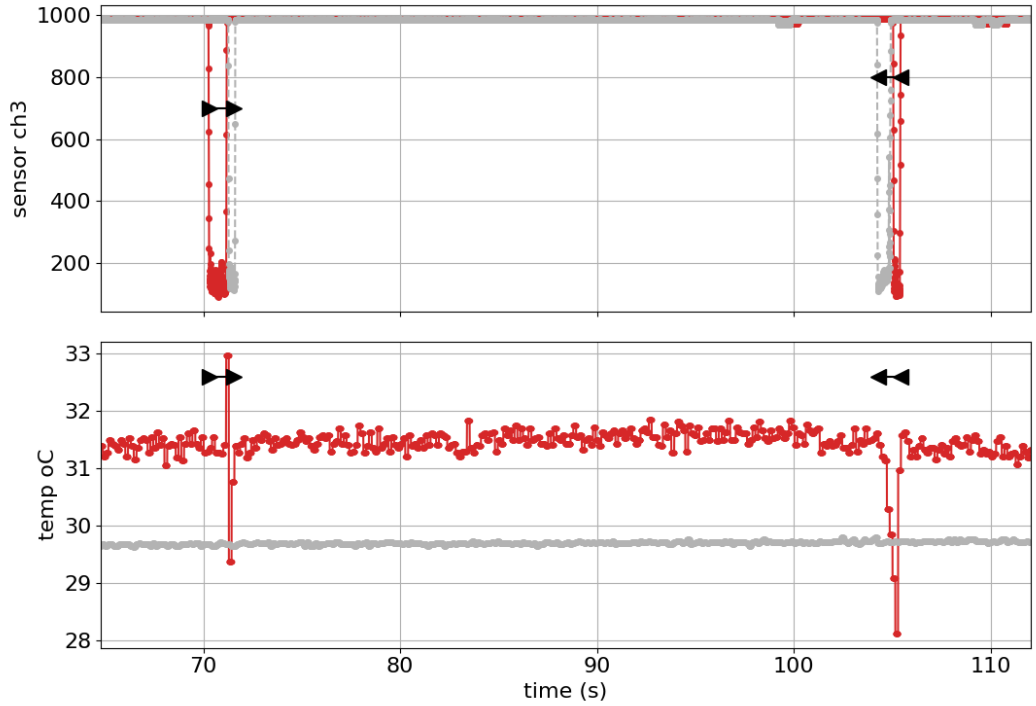


Fig 5.5 : Correspondence between bee-passing events in the IR sensors and the thermopile sensor in the same channel. The thermal signal drifts somewhat but the two events are clearly identifiable in temperature deltas.

Discussion

The aim of using the thermopile sensor was to detect a profile of bees passing, where the thorax (T) temperature would be higher than both the head (H) and abdomen (A) (in the vast majority of cases, in the rank order T>H>A, (Stabentheiner et al., 2010). Unfortunately, the sampling rate of 10 Hz was insufficient to identify a profile reliably, since the bees can pass through fairly quickly not allowing for a minimum of 3 measurements while the bee passes. Moreover, we observed both temperature increases and temperature decreases with different passing bees. These two factors make the thermopile solution infeasible.
In a similar vein, the heights of the thorax, head, and abdomen could be used to reveal the direction of motion using a time-of-flight sensor to measure the heights. However, the inter-sample variability was too high to obtain a reliable measurement, ruling out the possibility of obtaining a profile that could be trusted.

Although each of these sensors could be used to provide discrete signals regarding presence or absence of a bee, this would require the use of multiple sensors to identify directionality. Since the cost, the power consumption, and the space used by either of these sensors is higher than the IR phototransistors, these solutions will not be investigated further.

In the next period we will redesign the mechanical structure so to perform monitoring at the side of the frames, inside the evaluation hive (Fig. 5.6 and Fig. 5.7). We will also consider incorporating high-frequency multiplexing and sharing one IR source for many sensors as methods to reduce power consumption. The primary goal of this testing is to verify the reliability and biocompatibility, which will involve some detailed investigation of the channel size and also shape (see Fig 5.8). Once the hive's structural design is determined (WP2) the final mobility monitoring design can be tailored for integration.

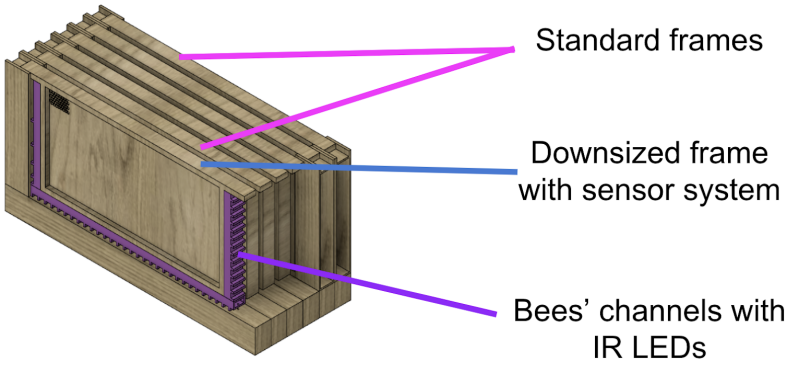


Figure 5.6 - Plans for improved traffic monitor sensing. The purple structure depicts bees' channels embedded with light sensors to detect the passage of bees.

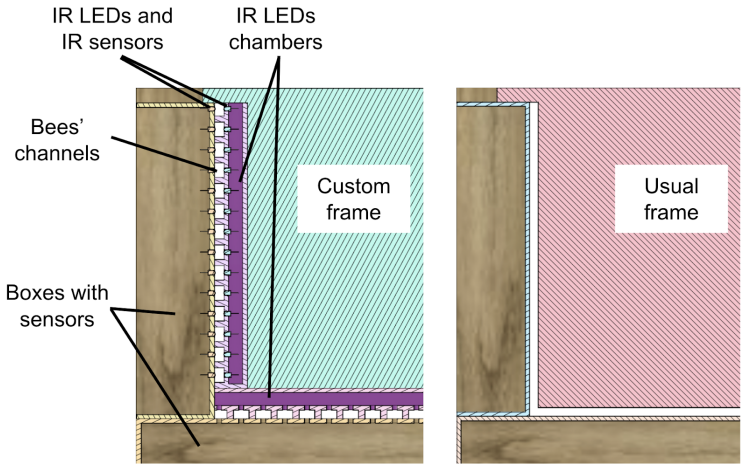


Figure 5.7 - Detailed view of the flow sensing channels (left) showing the space that will be occupied by the instrumentation, in purple, and the comparison against a standard frame, without any sensing capabilities (right).

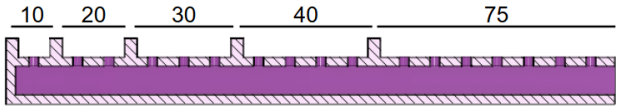


Figure 5.8 - Variable sized channels to examine the impact on motility of bees, thereby being able to select a more biocompatible design

5.2 Measuring the weight of honeybee hives

We use this section to highlight key information from the literature regarding the weighing of beehives as a measure for colony status, and the forms of technology that can provide this information.

Why measure the weight of a beehive?

Beekeepers and scientists have overlapping rationales for weighing beehives or their components. Practical uses for beekeepers include:

1. to identify the onset of honey flow (when to install the hives);
2. to identify whether honey flow is extensive in a given season (when to install more supers);
3. to identify when the flow is over (when to harvest honey) (Decourtye et al., 2018).

Besides these key uses, weight information can also be used for deciding when to apply varroa treatments (Human et al., 2013), to warn of a swarming event; for theft detection (see product description at SMSVaga.com); and for hive migration planning (Vidrascu et al., 2016).

Scientists' aims are either purely observational, or for measuring the consequences of different treatments, conditions or structures:

4. Investigating foraging activity, nectar acquisition, with daily changes (Meikle & Weiss, 2017) or longer term to infer changes in nectar stores (Meikle et al., 2008)
5. Predicting when swarms might occur (Howard et al., 2016)
6. Measuring the response to modulating dances (Dornhaus & Chittka, 2004; Okada et al., 2012; Sherman & Visscher, 2002)
7. Monitor brood production, if it is possible to distinguish from honey production.

Bee breeders also use weight gain as a proxy for honey production and thus as a colony fitness indicator (Calderone & Fondrk, 1991).

While the most direct measurement is the gross weight of a hive (including the "dead" weight and colony), it is not particularly informative of colony state (Meikle & Holst, 2015). The changes over time are informative, and typically help with estimation for the seven points above.

Technological approaches to obtaining hive weight

There exist a large number approaches to weighing beehives, as well as various products on the market. The vast majority fall into one of the following categories:

- pivot/tilt methods: half the hive is lifted (ideally with minimal rotation), with a strain gauge of some form, e.g. a torque wrench. The measurement is doubled for an estimate of the total hive weight.

- Multiple strain gauges: typically 4 strain gauges are used in parallel, their weight estimates summed. This method is more tolerant to asymmetric loading; the device can typically remain in place and thus opens the door to continuous monitoring (with suitable data loggers or transmission).
- Single strain gauge: a bridge structure ensures that the entire load of the hive passes through one measurement point. This design requires only a single load cell and an Analog to Digital converter (ADC), but more complex mechanical structure than using strain gauges on each corner.
- A “Roman balance”: a system that involves moving a counter-weight so that both sides of the scale are in balance. A motorised system that moves the counter-weight and measures the distance moved to estimate weight (see product description at openhivescale.org) permits automated data collection.

The pivot methods appear to be more common for apiary use, since one device can be used to measure many hives sequentially. However, the disruption of weighing seems to agitate the bee colonies (Stalidzans et al., 2017). Proponents of the Roman balance approach argue that load cell performance may degrade significantly if they remain loaded at all times. Nonetheless the single- or multiple- strain gauge approaches are well established and a number of commercial offerings are available from €200-€1500.

Factors affecting measurement

Measuring the weight of a hive has several issues that should be taken into account when designing the system and/or analysing the data.

- Humidity absorbed: wooden hives absorb additional mass when wet, leading to increased weight measurements. It is possible to compensate by measuring an empty hive in the same conditions (continuously or model-based), or using alternative materials.
- Creep of load cells: most load cells are designed to be ‘tared’ before use, and provide a difference during a short measurement. The cell reading may degrade over time if constantly loaded.
- Propolis, deposited by the bees and “glueing” the two sides of load cells together, thus taking some of the load instead of the measurement device. Isolation of the mechanism from the bees is important.
- Measurement can disrupt the colony if the method involves lifting each hive onto a scale (see Stalidzans et al., 2017), or lifting the hive with a scale; the detailed measurements of individual frames (e.g. as described by Meikle & Weiss, 2017) is highly invasive.

In terms of data, we must be careful to think about what the weight data corresponds to. We are not directly interested in the weight of a hive *per se*, but rather, we are interested in a way to discover resource income (nectar, pollen, water); animal mass (adult, brood); for the requirements C-DT8 and C-DT9. In some cases the dynamics of whole hive weight can be used to estimate these different factors without direct measurement. Data changes through daily and seasonal cycles, and measurements must be made at appropriate times of day,

and with measurement and understanding of critical covariates. Meikle et al (2008) propose a decomposition of the weight data into a long term component (moving average) and a within-day component (the difference between the moving average and the raw value for each measurement). The latter component can be used to estimate bee activity, e.g. foraging, while the former component can be used to estimate resource accumulation. Each component can have further analysis applied, e.g. modelling daily trends with sine waves (Meikle et al., 2008) or motifs (Holst & Meikle, 2018); or modelling seasonal trends with mixtures of Gaussians (Odoux et al., 2014). These forms of modeling would allow for comparison of activity and resource changes at a given site in comparison to other sites, or the same site between periods (days or seasons). These analyses forms may offer useful starting points, and the HIVEOPOLIS systems will feature a large diversity of measurements and off-hive information that will feed multi-modal analysis and predictive modelling.

Prototype hive scale

Partner BST has developed a hive scale utilising a multiple strain gauge approach, where two beams each contain two planar load cells at either end. This two-beam design allows for flexible deployment, which is important as the various hive physical architectures under investigation differ in dimensions and a fixed mechanical construction would require re-design for each of the physical architectures.

The load cells have a capacity of 75 kg and error ±0.05%, and are sampled by a 24-bit ADC close to the sensor to minimise interference. The four ADCs are read by an ARM M4 microprocessor in a separate enclosure that also measures temperature locally to compensate for changes in the load cell resistance. The scale can be queried, or it can be programmed to periodically sample and transmit the weight along with a timestamp.

Chapter 6: Weather and In-hive climate

As a complement to the measurement of biological variables (Chapter 5), this chapter considers the sensing of environmental variables. The HIVEOPOLIS unit makes use of environmental information at three scales:

- In-hive climate/environment
- Local weather information (e.g. outdoor humidity, local wind speed)
- Internet-based weather information including forecasts

The internal and local conditions are relevant for regulation, while all scales are useful in various models (from foraging to brood growth dynamics).

Sec 6.1 provides an overview of the candidate observables; 6.2 describes evaluation of devices to provide this sensing; 6.3 describes our development of a wireless node for flexibly sensing the internal environment; and Sec 6.4 describes the acquisition of weather information at broader and local scales.

6.1 What is relevant to measure?

We have direct objectives to measure certain environmental factors, including primarily temperature and humidity inside the hive (C-DT2, C-CL1-6), and obtaining weather forecasts and current conditions at the broader scale such as pollutants and air pressure (C-DT12, C-DT14). Besides the objectives that already have clear requirements for, there is a general desire to collect as many measurements of the environment -- internal and external to the hive -- as potential explanatory correlates for colony welfare and behavioural selection. To facilitate this latter goal takes some care, because we must also pay attention to power consumption, price, complexity, volume of data generated, and other engineering/human resource considerations. The importance of temperature to honeybees, for example, is well established, and thus the decision on whether to measure is trivial. However, making decisions towards less well-studied factors is more involved. If studies do not exist, is this a) because by theoretical reasoning the measurement will be too rarely relevant or never be relevant (gravity fields, light outside visible spectrum, lightning, earthquakes), b) simply because it is, or was historically, difficult to measure within the somewhat hostile environment of a bee hive (gases, traffic flow inside)? The absence of prior studies do not completely rule out the merits: Novel measurement techniques within the hive have found intriguing patterns, e.g., accelerometers revealing the phase of brood nest cycle(Bencsik et al., 2015).

Temperature

Temperature sensing and regulation is vital for the survivability of the colony. Following, some examples detailing how temperature can influence a honey bee society:

- Bee colonies can precisely control the microclimate within the hive. When they need to heat certain regions, they vigorously contract the thoracic muscles, used to control the wings, producing heat as a byproduct of the accelerated metabolic rate (Moritz & Southwick, 1992, pp. 86–87; Seeley, 1995, p. 26). And, when there is a need to cool parts of the colony, they use some strategies such as evaporative cooling (Human et al., 2006), or using their bodies to absorb heat from hot regions (Starks et al., 2005), or even organizing a process of collective ventilation (Peters et al., 2019).
- During the development phase of a new bee, from egg-laying to pupal metamorphosis, the temperature in the broodnest is precisely maintained between 32 ℃ and 36 ℃, with a slight preference for 35 ℃ (Kronenberg & Heller, 1982; Stabentheiner et al., 2010). This nest homeostasis behavior guarantees the healthy development of the next generation. When brood nest temperatures fall outside the comfort range it can cause several problems such as mortality, neuronal disorders, malformations, low foraging capacity, among others (Bonoan et al., 2014).
- In the cold season, when the ambient temperature reaches levels below 15 ℃, bees start an approximation movement, forming a cluster to decrease the heat loss by reducing the total surface area and increasing thermal insulation (Moritz & Southwick, 1992, pp. 88–89).

Humidity

The humidity is also a critical parameter for a healthy honey bee colony. Some findings relating to humidity and honey bees, collected from literature are:

- (Doull, 1976) studied the importance of relative humidity for honey bee egg hatching. They reported a very strong dependence of hatching on R.H.: at 70 %RH or below, less than 5% are viable (at lower RH values zero hatched). At 80 or 100 %RH around 60% hatch normally, while at 90 or 95 %RH, above 90% of eggs hatch. These results have been confirmed in more recent literature as well (Abou-Shaara et al., 2017; Aupinel et al., 2005; Kaftanoglu et al., 2011).
- (Human et al., 2006) shows a drastic difference in humidity levels between brood and nectar regions.
- (Z. Li et al., 2016) observed the micro-climate of both worker brood and drone brood, and found that the humidity is more tightly regulated in worker brood nests. They also found that the presence of honey does not have a large impact on humidity level within the hive on average, but the temporal dynamics are slightly different. Specifically, the rate of humidity increase in the mornings is slower in the presence of honeycombs when compared to an empty hive box.
- (Nazzi & Le Conte, 2016) indicate that the humidity levels influence how well varroa mites can reproduce: “Hygrometric conditions also play an important role, with optimum humidity for reproduction ranging from 55% to 70% and only limited reproduction taking place at higher humidity.” Since near-optimal conditions in both temperature and humidity are found in the honey bee brood nest, varroa can thrive.
- (Kraus & Velthuis, 1997) observed that temperatures in the brood nest of honey bees in temperate and hot climates do not differ a lot, but the varroa infection rates do.
- (Mitchell, 2016) studied the differences in temperature and also humidity of the internal environment of beehives. They argue that the high values for humidity that suppress varroa proliferation could be achieved by hive materials and shapes that are closer to tree trunks.
- (Ellis et al., 2010) showed that brood comb can act as a buffer of humidity.

Internal environment

Besides the two most studied parameters of temperature and humidity, their hive-internal environment is known to have physiological impacts and affect behaviours at the individual and collective levels.

Honey bees exhibit a fanning behaviour at the hive entrance to ventilate their hives. They are triggered into this behaviour according to high temperatures (Kronenberg & Heller, 1982); high CO_2 levels (Seeley, 1974; Southwick & Moritz, 1987); and high humidity levels (Human et al., 2006). Interestingly, high levels of O_2 do not appear to trigger bee-mediated ventilation (Southwick & Moritz, 1987).

Investigations measuring diverse gases and other airborne variables have been reported, e.g. the multi-gas “artificial nose” used by Szczurek et al (Szczurek et al., 2019), or the air quality sensing platform used by Edwards Murphy et al (Edwards Murphy et al., 2015). None of the studies that we found reported any direct relationship between colony health and any of the specific gases measured. However, the ability of machine learning tools to classify the Varroa load from the group of gases, shown in a very recent study, is intriguing (Szczurek et al., 2020). Conversely, studies have shown the impact of diesel fumes with high NOx concentrations on honey bee ability to locate flowers by their odour (Girling et al., 2013) but this finding relates to a behaviour exhibited away from the hive.

Overall, we aim to measure internal environmental variables whose link to colony health is established, rather than simply measuring quantities that can be measured. Our efforts at this point will focus on temperature, humidity, and CO_2.

Regarding CO_2 sensors there exist various classes of device, including IR-based sensors (with single and differential variants), those that approximate based on volatile compounds, and others that use heaters to make measurements. We have already installed one differential-IR type sensor in an observation hive (see D5.1) with initial success. We will evaluate the suitability of each class for use in beehives in the future period.

From phenomena to performance requirements

Here we consider the characteristics (e.g., range, rate of change, relevant resolution) of important variables (e.g., temperature, humidity, CO_2) to guide the design of in-hive instrumentation.

For the general climate regulation system, we explicitly exclude discussion of the broodnest thermal regulation systems - this is the most sensitive area of the hive and significant efforts address this within WP5 (see e.g., D5.1).

For each variable we performed a literature review to find the importance of each metric and its range. These numbers will steer parts of the design, and the selection of sensors and actuators.

Temperature

characteristic	value	unit	supporting references / notes
Range	[-20, +55]	℃	(min) Stalidzans et al., 2017 / (max)(X. Li et al., 2019)
Rate of change (max)	0.25	℃/min	Zhu 2019 (2.5 ℃ in 10 min)
Resolution	0.03 (0.2)	℃	(Bauer et al., 2018): brood cell varroa (bee sensitivity)

Humidity

characteristic	value	unit	supporting references / notes

Range	[20, 100]	%RH	(min) Human et al., 2006 / (max) Doull, 1976
Resolution	~5	%RH	Doull, 1976

Carbon Dioxide (CO_2)

characteristic	value	unit	supporting references / notes
Concentration	[0, 10]	$\%CO_2$	Seeley, 1974: Ideal [0.44, 0.78] $\%CO_2$
Resolution	0.45	$\%CO_2$	Lacher, 1966

Ventilation

characteristic	value	unit	supporting references / notes
Flow (avg.)	0.42	L/min	Southwick & Moritz, 1987
Air speed entrance (max)	3	m/s	Peters et al., 2019
Direction	outflow	--	Air should be sucked out of the hive and not injected

6.2 Sensing evaluation

Humidity sensors

Aiming to find a suitable sensor to monitor the humidity inside different hive compartments, we did a survey on the COTS parts that are easily available (Table 6.1).

Table 6.1 - Humidity sensor candidates list

	Humidity				Temperature				Consumption			Dimension			Price
	Range	Accu.	Res.	Heater	Range	Accu.	Res.	Supply	I_{DD}	I_{1Hz}	I_{sleep}	w	l	h	
Model	(%RH)				(°C)			(V)	(μA)	(μA)	(μA)	(mm)			(CHF/u)
TI HDC2010	**0–100**	**3**	**14b**	•	**-40–125**	**0.4**	**14b**	**1.6-3.6**	**1620**	**0.55**	**0.1**	**1.5**	**1.5**	**0.68**	**3.28**
TE HTU20D	0–100	7	0.04 (12b)	•	-40–125	0.4	0.01 (14b)	1.5–3.6	500	?	0.14	3	3	0.9	5.79
TE HTU21D	0–100	5	0.04 (12b)	•	-40–125	0.4	0.01 (14b)	1.5–3.6	500	?	0.14	3	3	0.9	6.19
TI HDC1080	0–100	4	14b	•	-40–125	0.4	14b	2.7–5.5	405	1.3	0.2	3	3	0.8	4.45
Sensirion SHT35	0–100	3	0.01	•	-40–125	0.3	0.01	2.2–5.5	1500	1.7	0.2^T	2.5	2.5	0.9	12.99
Sensirion SHT31	0–100	3.5	0.01	•	-40–125	0.4	0.01	2.2–5.5	1500	1.7	0.2^T	2.5	2.5	0.9	7.88
Sensirion SHT30	0–100	7.5	0.01	•	-40–125	0.4	0.01	2.2–5.5	1500	1.7	0.2^T	2.5	2.5	0.9	3.85
Sensirion SHTC3	0–100	6.5	0.01	○	-40–125	0.4	0.01	1.5–3.6	900	0.5^T	0.3^T	2	2	0.75	3.40
E+E EEH210	0–100	5	0.04 (12b)	•	-40-125	0.5	0.01 (14b)	2.1–3.6	450	3?	0.5	3.6	2.8	0.75	11.10
ST HTS221	0–100	5	0.004(16b)	•	-40–120	0.5	0.016(16b)	1.7–3.6	?	2	0.5	2	2	0.9	3.74
SiLabs SI7021	0–100	4.5	0.04 (12b)	○	-40–125	0.4	0.01 (14b)	1.9–3.6	300	?	0.62	3	3	1.21	3.30
Honeywell HIH8131	0–100	2^T	0.04 (14b)	○	-40–125	0.5	0.025 (14b)	2.3–5.5	1000	?	1	3.9	4.9	1.9	53.95
Sensirion SHTW2	0–100	7.5	0.01	○	-30–100	0.4	0.01	1.6–1.9	465	4.8	1.5	1.3	0.7	0.5	2.98
E+E EEH110	0–100	5	0.04 (12b)	○	-40–125	0.5	0.01	4.5–5.2	270	67?	65	3.6	2.8	0.75	11.10

[T] Used typical values when maximum was not available.

From the gathered list of contenders, we decided to perform tests with two devices. The TI HDC2010, described in Section 6.3, and the Sensirion SHT35, described below. The Sensirion part offers slightly better performance (typical accuracy of 1.5%, versus 2%), while the TI part has a lower power consumption and price, with a small penalty in accuracy. The power consumption is the more critical dimension for the wireless node (see Sec 1.3).

To evaluate the SHT35 humidity sensor, and also gain an initial picture regarding humidity gradients within the beehive, we designed a PCB with a long and narrow form. The PCB had three sensors on the board at a distance of 10 cm (Figure 6.1).

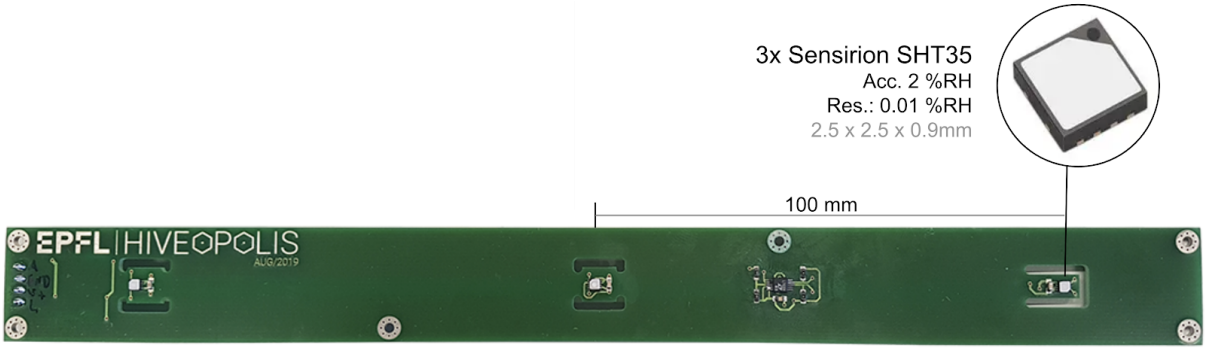


Figure 6.1 - Picture of the humidity sensor PCB. The white squares are the sensors (magnified in the circle).

The schematic of the circuitry is shown in Figure 6.2, and includes an address translator (chip U3) to facilitate a third device.

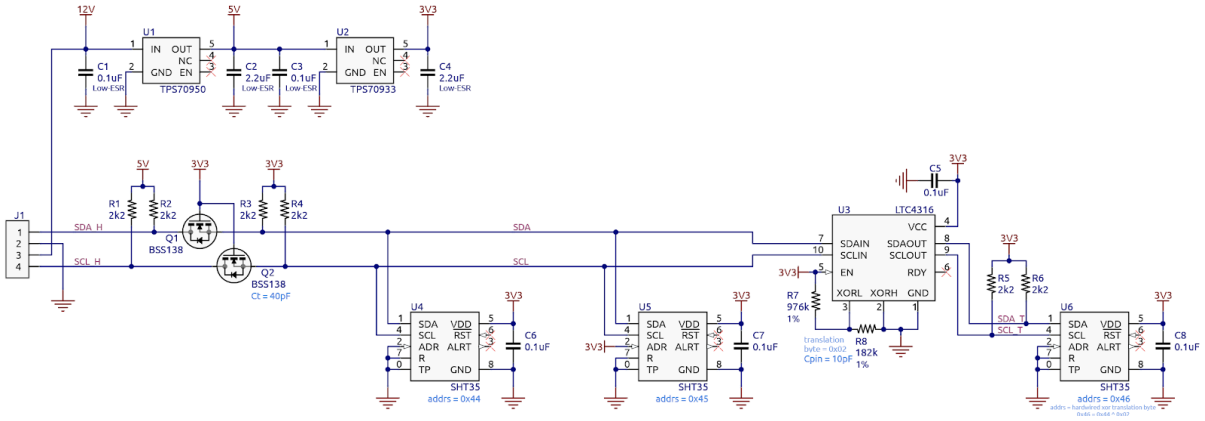


Figure 6.2 - Electrical schematic of the humidity sensor evaluation board. On the top left part is the power regulation and level translation circuits. In the bottom right, the three Sensirion SHT35 sensors.

We first did a test using a high-quality weather station as a reference (Gill GMX300) with an accuracy of 2 %RH and a resolution of 1 %RH, which is located on the EPFL campus. For comparison, we placed our sensors next to the reference weather station. The sensors were left overnight close to the reference station (d ~ 0.5 m). By conducting the experiment at night we avoided any influence that direct sunlight could have on our sensors.

In the beginning of the experiment, a 5 %RH deviation between the sensors and the reference station could be seen, but once the sensors thermallized, (after approx. 30

minutes), this difference decreased until the end of the experiment, 11.5 hours later (Figure 6.3).

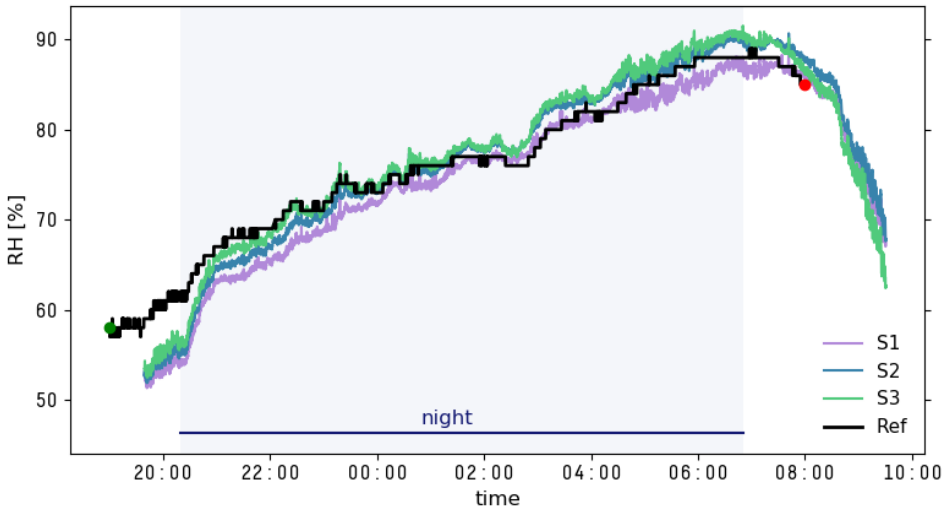


Figure 6.3 - Humidity data from three sensors (purple, blue and green curves) and a reference station (black curve).

For our research going forwards, we have secured access to the data from this and other instruments; these are located approximately 500 m away from where our hives are. The closest national weather station that provides data to which we also have secured access, is 8 km away.

We also tested the internal hive environment, in a vertical orientation. This revealed greater differences across the hive than would be expected without the presence of the bee colony, and the gradient was typically non-monotonic (i.e., highest humidity at the middle height) (Fig. 6.4) Importantly, we identified degradation of the circuitry over time and concluded that the board assembly needs better protection from the harsh hive environment.

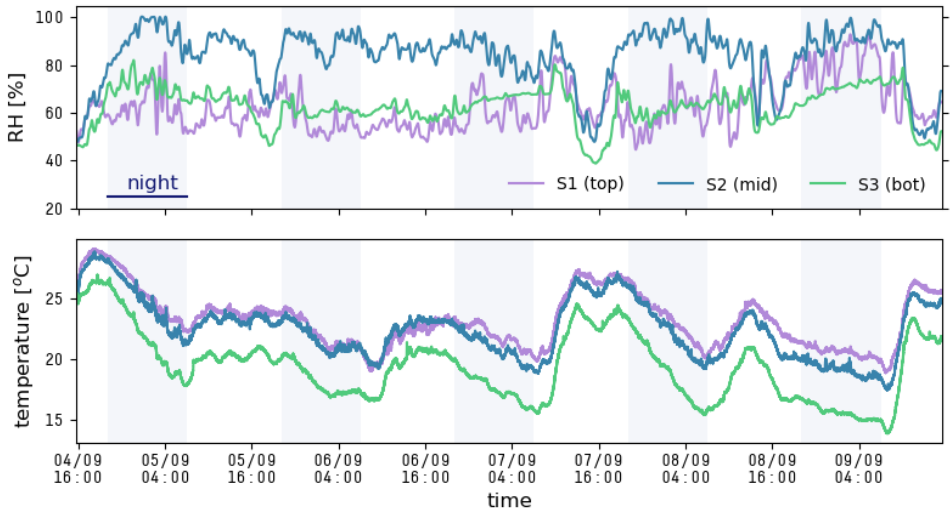


Figure 6.4 - 5-day long data series from humidity sensors installed at three different heights inside a beehive.

Temperature sensors

All of the humidity sensors considered here also measure temperature. Even without calibration (i.e. using factory tolerances) their performance is suitable for use within the beehive for all areas with the possible exception of the brood nest. In *Deliverable D5.1* we performed a detailed analysis of temperature sensing, and surveyed key temperature ranges and temperature gradients within the hive, focusing on the brood nest - the most sensitive and most tightly regulated area of the hive. In that work we selected the Texas Instruments TMP117 sensor for high precision, high resolution sensing at a modest price. If the need for higher precision temperature sensors in other areas of the hive becomes apparent, we will employ the same TMP117 device.

6.3 Prototyping a wireless sensor node

A recurring theme across many areas of HIVEOPOLIS is to better understand the in-hive environment, and two key measures whose importance is well established are the temperature and the humidity. While there have been studies measuring these dimensions at one or multiple positions in honey bee hives (Giammarini et al., 2015; Meitalovs et al., 2009; Zacepins et al., 2016), it has not been systematically studied -- although temperature more so than humidity (e.g., Owens, 1971 who studied the temperature of colonies during winter). Furthermore, our context is somewhat different since we aim to not only observe but also modulate the in-hive environment. Accordingly, a detailed understanding of where and when temperature and humidity fields might arise is of interest.

With a small wireless device that can be inserted into the hive at many locations easily, it will be possible to measure gradients of key environmental variables, or locations at which flux is significant and/or informative. Once relevant locations and densities of sensing have been established, the wireless devices can in principle be replaced with fixed components.

In the remainder of this section we describe our design of a wireless sensing node for use inside the honey bee hives.

System Architecture

The overall system architecture can be split in four different logic sections: a power supply module which includes the battery, the charging circuit and a voltage regulator; a processing module, integrated by a SoC for RF applications connected to an antenna; a data storage module with an external memory; and a sensing module with an integrated thermo-hygrometer, and the possibility to add a CO_2 sensor (Figure 6.5).

The basic working principle of this system is that the processing module activates the sensor(s) at regular intervals. After collecting the data, the SoC stores the collected data in the memory module. At the same time, the processing module broadcasts advertising packages for anyone to catch (BLE capable device scanning the vicinity). The whole system

is powered by a rechargeable Li-Ion battery that was selected for the sensor tag to operate for at least a month before it needs to be recharged.

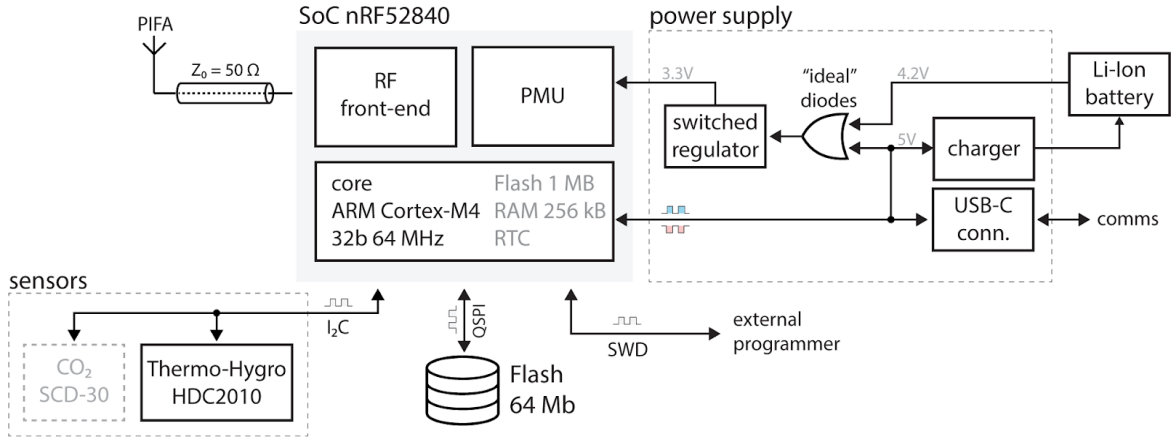


Figure 6.5 - Functional block diagram of the environmental wireless node.

System on Chip (SoC)

For the tag we chose the Nordic nRF52840 SoC, which is built around a 32-bit ARM Cortex-M4 CPU with floating point unit (FPU) with a clock frequency of 64 MHz. The SoC has also an integrated 2.4 GHz RF front-end with support for multiple wireless protocols like Bluetooth 5 and IEEE 802.15.4 based protocols like Zigbee and Thread. Some power characteristics of the device are presented and analyzed in the power budget section.

In the tag, the SoC is the heart of the device and it should perform the following tasks:

- Acquire and store the data from the humidity and temperature sensor
- Flash memory data management
- Track time and date through a Real-Time Clock (RTC)
- Establish and manage all wireless communication using Bluetooth protocols
- Monitor battery status

Sensor selection

We surveyed integrated humidity and temperature sensors to select the most suitable candidate for the system (see Sec 6.2). Based on technical requirements, we selected the sensor with the lowest sleep current, which is the HDC2010 (I_{sleep}= 0.1 μA).

A simple evaluation of the sensor self-heating can be made using the thermal resistance between the IC junction and the surrounding ambient temperature (θ_{JA}), and the dissipated power through the following expression (Horowitz & Hill, 2015, p624): ΔT [°C] = θ_{JA} [°C/W] · P_D [W]. For the HDC2010 sensor, we estimate a maximum temperature rise of ~0.0002 °C, when sampling at 1 Hz ($P_{D,1Hz}$ = 1.82 μW and θ_{JA} = 114.8 °C/W (Texas Instruments, 2017)). We concluded that the sensor will not “contaminate” or bias the temperature readings due to self-heating.

Communication protocol

The selected communication protocol was the wireless Bluetooth Low Energy (BLE) protocol. One of the advantages of BLE is its relatively high bandwidth and its suitability for power constrained devices.

The BLE communication protocol was first launched in 2010 jointly with Bluetooth 4.0 and is oriented towards IoT applications that focus more on lower amounts of data transferred at lower speeds. BLE operates in the 2.4 GHz bandwidth from 2400 MHz to 2483.5 MHz with a 2 MHz and a 3.5 MHz lower and upper band-guard respectively. Forty channels of 2 MHz width are available for use. Three of which (channels 37, 38 and 39) are advertising channels and the other 37 are data channels. The advertising channels are located at low interference frequencies to improve the reliability at which the packages are picked up because the 2.4 GHz bandwidth is shared with the WiFi technology.

Since the BLE 5.1 standard, it is also possible to send data over longer distances, by lowering the data rates to 125 kbps or 500 kbps. The maximum data rate is 2 Mbps (at up to ~50 m distance).

Comms range

In the following part, we used propagation models to roughly estimate the maximum distance that tags will be able to communicate with a mobile phone or another tag.

To establish a connection, the received signal energy must be sufficiently strong to allow the receiver to differentiate the signal from the noise. Depending on the transmission power, antenna gain, and the receiver sensitivity (and many other factors like antenna directivity, polarization, propagation medium, environmental conditions, obstacles, etc.) it is possible to estimate the radio communication range (Texas Instruments, 2018). A general method to calculate a radio link range is through Friis equation (Goldsmith, 2005):

$$P_r(R) = P_t \cdot G_t \cdot G_r \frac{\lambda^2}{(4\pi R)^2} \ [W]$$

where P_r is the power available from the receiving antenna, P_t is the transmitted power, G_t and G_r are the gains of the transmitting and receiving antennas respectively, λ is the wavelength (λ = c/f) of the signal, and R is the distance between the two antennas.

For example, having two Nordic nRF52840 SoCs operating at 2445 MHz, one transmitting at Pt = 8 dBm (6.31 mW), and both having an isotropic PIFA antenna with maximum gain Gt = Gr = 3.3 dBi (2.04 W/W). We can find the receiving level Pr for a distance of 100 m as follows:

$$P_r = 6.31\,mW \cdot 2.04 \cdot 2.04 \cdot \frac{(\frac{3\times10^8\,m/s}{2445\times10^6\,Hz})^2}{(4 \cdot \pi \cdot 100\,m)^2} = 2.50 \times 10^{-7}\,mW = -66\,dBm$$

We computed the received power curves (Figure 6.6), using the log distance path loss model, which is a variation of the Friis free-space model, but with the possibility to adjust the exponent *n*:

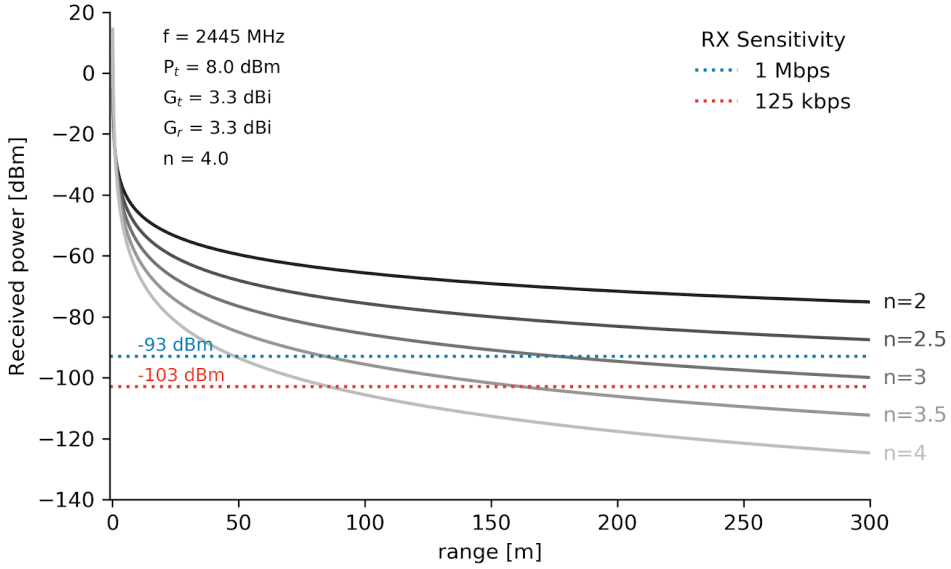


Figure 6.6 - Received power over distance between two tags. Free-space n=2.

Memory

Data can be immediately sent to any listening device or temporarily stored in a Macronix MX25R6435F low-power 8 MB flash memory. This part offers the following specifications:

- Minimum 100’000 erase/program cycles
- 20 years data retention
- 0.35 µA deep power-down current
- Four different packaging options (we used USON-8)

Power budget

To find the total energy usage, we calculated each part's power consumption. Some parts present a simple power profile since their current demand is reasonably constant. These parts alone require a constant current of 12.84 µA from the battery. During the development of the tag we also took dynamic loads from the SoC, the sensor and the memory into consideration.

In order to calculate the battery lifespan, we can create a simple scenario where the tag is configured as follows: BLE LR mode, TX power 8 dBm, BLE advertising of 10 s, and user hive inspection interval of 2 days. In that configuration, we analyzed the average current for three different sensor sampling rates of 1, 5, and 30 minutes. The calculated tag functioning time, in days, is presented in Table 6.2, for four different battery capacities. For the results presented, we included a battery self-discharge of 5.5% and a derating factor of 0.85, that accounts for the aging of the battery.

Table 6.2 - Battery endurance, in days, for 3 different sampling intervals.

Battery Part #	Capacity [mAh]	Sensor Interval 1 min 19.63 µA	5 min 17.11 µA	30 min 16.59 µA
Multicomp LIR2450	120	148	162	165
PKCELL 503035	500	306	320	323
Panasonic UF553436G	800	351	362	365
Panasonic UFU553443ZU	1000	369	379	381 days

With the selected architecture and components, the life expectancy of the tag with a single battery charge (C = 120 mAh) is estimated to be between 148 days in the case of a fast sampling rate of one measurement every minute to 165 days with measures every 30 minutes. Those results are satisfactory for a long-term monitoring device.

PCB implementation

The PCB was designed in a four-layer configuration with a total thickness of 0.8 mm. The dimension and position of major components are shown in Figure 6.7.

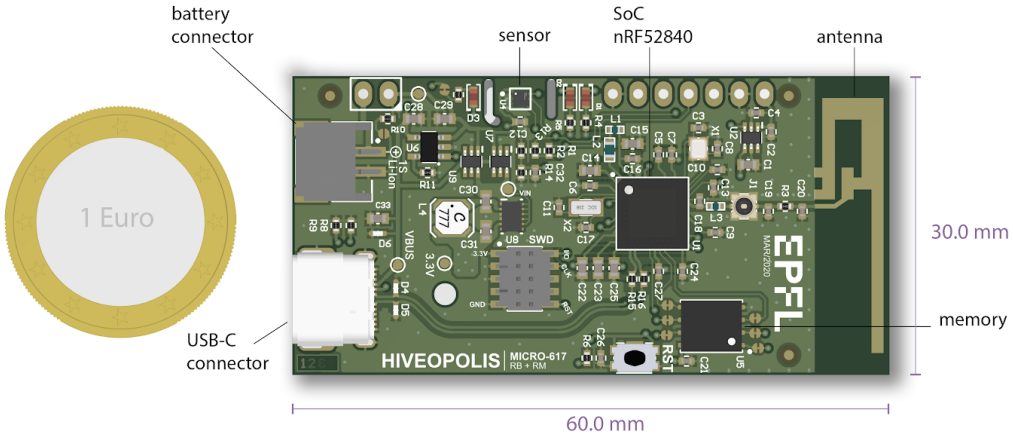


Figure 6.7 - Diagram of the tag PCB and its main parts.

Firmware

The firmware for the wireless node is written using the nRF5 SDK, a framework that gives access to the RF front end as well as low power modes of the device. The node is programmed as a BLE Peripheral, and includes services that a host (BLE Central role) can interact through. The implemented services are:

- Buttonless DFU service
- Sensor data (HDC2010, SCD30)
- System status (battery, clock)

The custom services include characteristics that can notify a peer of updated values for the sensory or system data. This requires the Central device to subscribe as desired; the less

efficient alternative is to poll for updates. The services also implement characteristics for reconfiguring parameters such as sampling intervals, setting and verifying the realtime clock. Each sensor is sampled on an independent timer, and data transmission for the main sensor is done in bulk via enqueuing the data that has been recently buffered. See Fig 6.8.

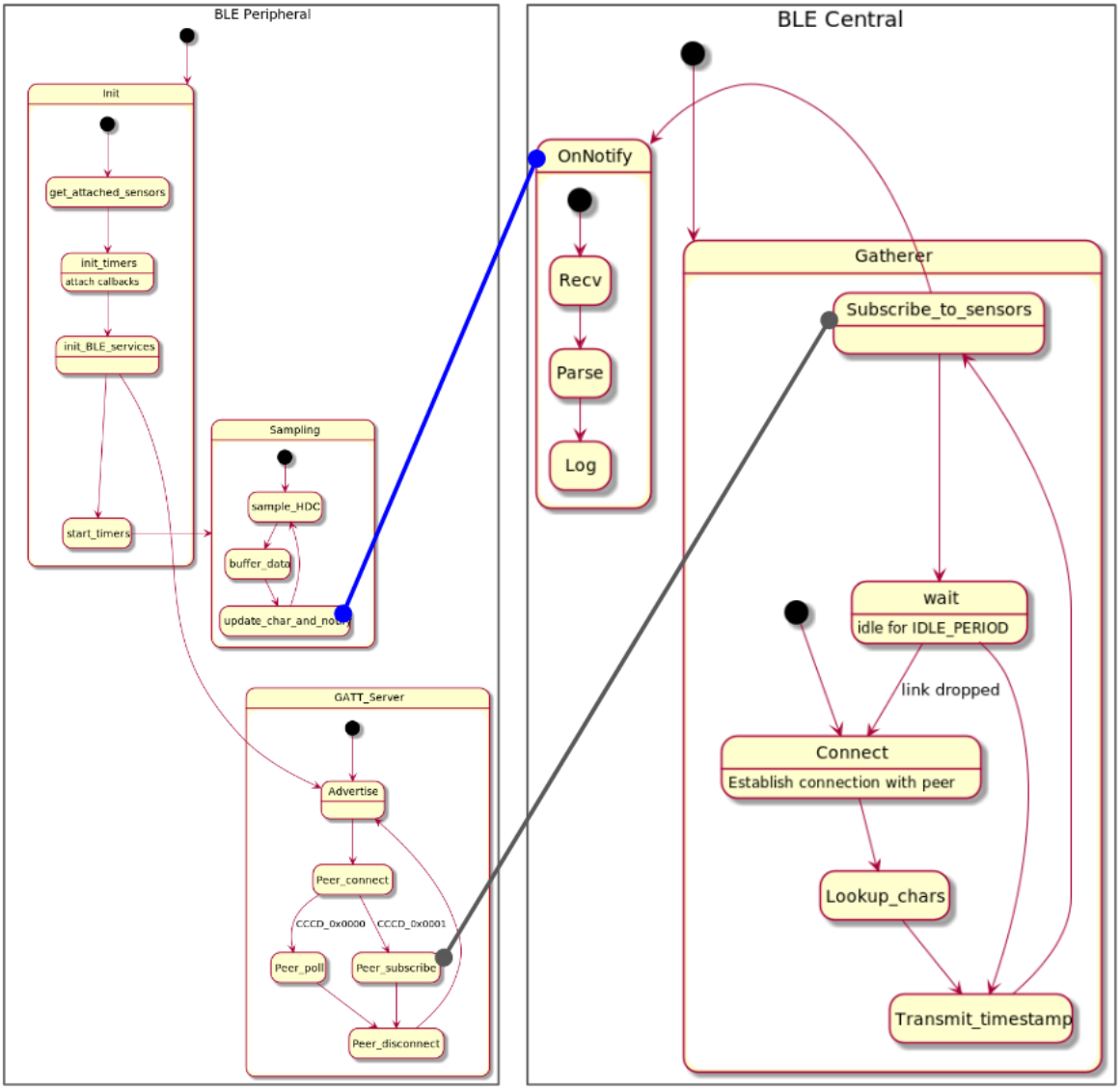


Fig 6.8 state machines in the nRF wireless node, and communication links in the BLE central peer that receives and logs the data. The sampling for each sensor and presentation over characteristics is independent but since the process is identical we show it in a simplified form.

Validation

The current drawn by the device is nonuniform due to the differing operations, primarily: sampling the on-board sensors (sensor and I2C bus active); sampling the battery (ADC peripheral active); sampling the CO_2 sensor (sensor idle or sensor sampling); bluetooth advertising; communicating; and a quiescent current. The multitude of different combinations are not of utmost interest, so we report the current measured in three different modes:

- With all sensors active, and connected to a peer:
    - 4-4.6 mA baseline, with short pulses of 65-70 mA while sampling CO_2

- With the CO_2 sensor disabled, and connected to a peer:
  - 40 µA baseline, with short pulses of 250 µA during sampling and 135 µA during communications
- With the CO_2 sensor disabled, and advertising:
  - 40 µA baseline, with short pulses of 100 µA while advertising, 250 µA during sampling, or 320 µA when both occur simultaneously.

With a 1500 mAh battery we estimate a lifespan of around 9 days with the power-hungry CO_2 sensor enabled, and >200 days when disabled (an idealised calculation that assumes no battery self-discharge, which is of course unrealistic for a period of 6 months). We will investigate methods to extend the battery life while also capturing an acceptable resolution of sensor data, but for the initial investigations of understanding whether important gradients are present in a hive, the duration is acceptable. In Fig. 6.9, we see a typical set of data recorded from in an office environment using a 550 mAh battery.

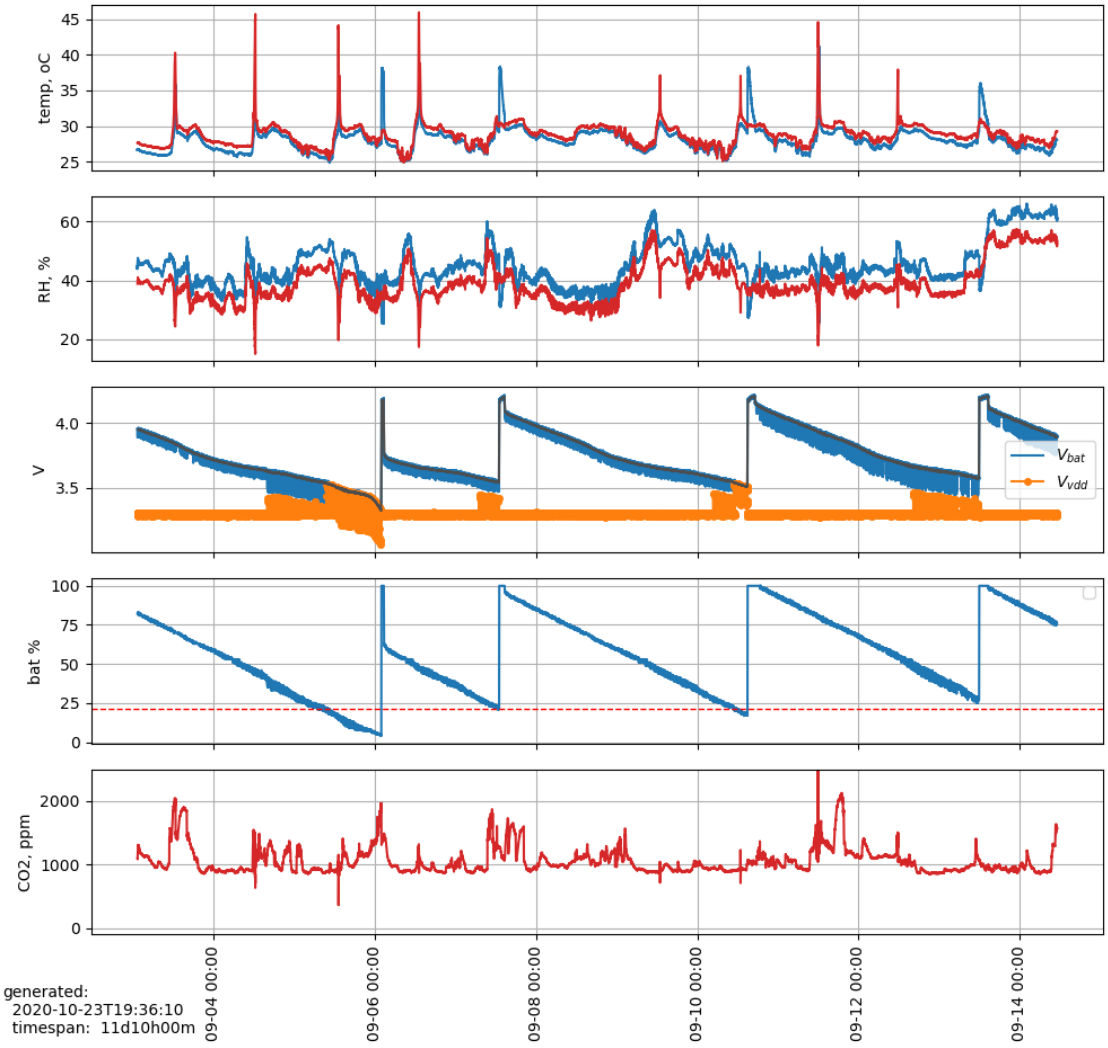


Fig 6.9 Typical data for sensor node with CO_2 device sampling at 15s interval, using a 550 mAh LiPo cell. The node was left indoors and received direct sunlight for a short period per day which caused the spikes in temperature. The node was recharged when the predicted battery level fell to around 20%.

6.4 Weather data

The external gateway design is part of T3.4 and will be reported in a future deliverable. To summarise, that design aims to serve a common form of data to hive units from various sources, weather services requirement C-DT12 and C-DT14, and form one exemplar of the external services gateway. Various sources for current conditions and forecasts exist, both specialised national services and others with global coverage. The gateways will be responsible for collecting and transforming the data into a common form that is usable by the hive data warehouse and subsequently the predictive models. What is important from the perspective of the central core is to estimate the volume of data that such services will generate.

- SRG/SSR national Swiss radio (CH only): 30 min update, 1 kB per observation
- BBC observations and forecasts (Global): 12-hour update interval, 3.5 kB per forecast
- OpenWeather observations and forecasts (Global): 2-hour update interval, 20 kB per fetch (combined observation and forecast)

Taking local measurements to complement national weather services is intended to have direct information about the very local environment (i.e., surrounding the hive area). This includes in particular: rainfall, light levels/insolation, and wind information (D2.1 C-DT12 and C-DT13).

We selected a COTS device, the Eurochron EWFS 2900, which has sensors for temperature, relative humidity, solar insolation, UV index, rainfall, wind speed and direction, relative and absolute air pressure (Fig. 6.10). Additionally, it derives wind gusts and aggregates rainfall over different periods. These variables are sampled and reported every 5 minutes. In terms of communication, the device transmits readings to weather servers, and besides using commercial services such as weather underground, it is compatible with an open-source EcoWitt server that can be hosted locally.

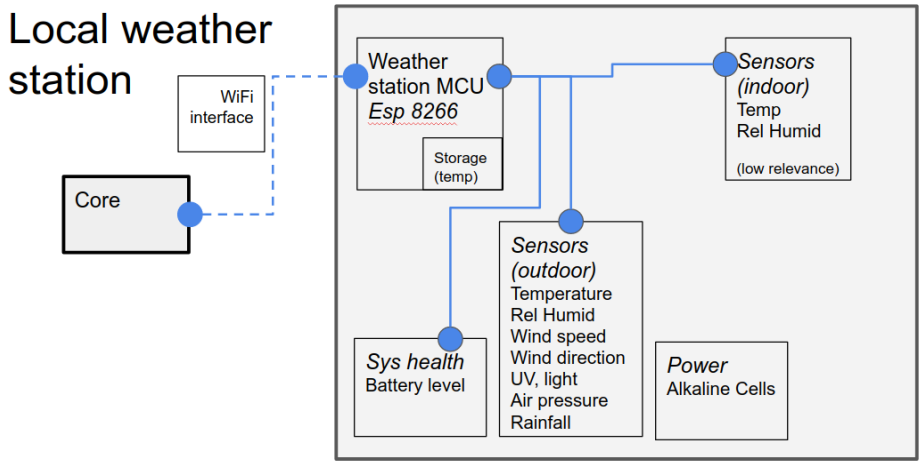


Fig 6.10 - Overview of the weather station sub-system. Wire connections are depicted as continuous blue lines and wireless links as a dashed blue line.

References

Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. P. (2017). Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review*, *24*, 13–33. https://doi.org/10.1016/j.cosrev.2017.03.001

Abou-Shaara, H. F., Owayss, A. A., Ibrahim, Y. Y., & Basuny, N. K. (2017). A review of impacts of temperature and relative humidity on various activities of honey bees. *Insectes Sociaux*, *64*(4), 455–463. https://doi.org/10.1007/s00040-017-0573-8

Anuar, N. H. K., Yunus, M. A. M., Baharuddin, M. A., Sahlan, S., Abid, A., Ramli, M. M., Amin, M. R. A., & Lotpi, Z. F. M. (2019). IoT Platform for Precision Stingless Bee Farming. *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 225–229.

Aupinel, P., Fortini, D., Dufour, H., Tasei, J.-N., Michaud, B., Odoux, F., & Pham-Delègue, M.-H. (2005). Improvement of artificial feeding in a standard in vitro method for rearing Apis mellifera larvae. *Bulletin of Insectology*, *58*(2), 107–111.

Bauer, D., Wegener, J., & Bienefeld, K. (2018). Recognition of mite-infested brood by honeybee (Apis mellifera) workers may involve thermal sensing. *Journal of Thermal Biology*, *74*, 311–316. https://doi.org/10.1016/j.jtherbio.2018.04.012

Becher, M. A., Grimm, V., Thorbek, P., Horn, J., Kennedy, P. J., & Osborne, J. L. (2014). BEEHAVE: A systems model of honeybee colony dynamics and foraging to explore multifactorial causes of colony failure. *Journal of Applied Ecology*, *51*(2), 470–482. https://doi.org/10.1111/1365-2664.12222

Becher, M. A., & Moritz, R. F. A. (2009). A new device for continuous temperature measurement in brood cells of honeybees (Apis mellifera). *Apidologie*, *40*(5), 577–584. https://doi.org/10.1051/apido/2009031

Bencsik, M., Le Conte, Y., Reyes, M., Pioz, M., Whittaker, D., Crauser, D., Simon Delso, N., & Newton, M. I. (2015). Honeybee Colony Vibrational Measurements to Highlight the Brood Cycle. *PLOS ONE*, *10*(11), e0141926. https://doi.org/10.1371/journal.pone.0141926

Boenisch, F., Rosemann, B., Wild, B., Dormagen, D., Wario, F., & Landgraf, T. (2018). Tracking All Members of a Honey Bee Colony Over Their Lifetime Using Learned Models of Correspondence. *Frontiers in Robotics and AI*, *5*, 35. https://doi.org/10.3389/frobt.2018.00035

Bonnet, F., Mills, R., Szopek, M., Schönwetter-Fuchs, S., Halloy, J., Bogdan, S., Correia, L., Mondada, F., & Schmickl, T. (2019). Robots mediating interactions between animals for interspecies collective behaviors. *Science Robotics*, *4*(28), eaau7897. https://doi.org/10.1126/scirobotics.aau7897

Bonoan, R. E., Goldman, R. R., Wong, P. Y., & Starks, P. T. (2014). Vasculature of the hive: Heat dissipation in the honey bee (Apis mellifera) hive. *Naturwissenschaften*, *101*(6), 459–465. https://doi.org/10.1007/s00114-014-1174-2

Calderone, N., & Fondrk, M. (1991). Selection for high and low, colony weight gain in the honey bee, Apis mellifera, using selected queens and random males. *Apidologie*, *22*(1), 49–60.

Campbell, J. M., Dahn, D. C., & Ryan, D. A. J. (2005). Capacitance-based sensor for monitoring bees passing through a tunnel. *Measurement Science and Technology*, *16*(12), 2503–2510. https://doi.org/10.1088/0957-0233/16/12/015

Cecchi, S., Terenzi, A., Orcioni, S., Spinsante, S., Primiani, V. M., Moglie, F., Ruschioni, S., Mattei, C.,

Riolo, P., & Isidoro, N. (2019). Multi-sensor platform for real time measurements of honey bee hive parameters. *IOP Conference Series: Earth and Environmental Science*, *275*, 012016.

Chen, C., Yang, E.-C., Jiang, J.-A., & Lin, T.-T. (2012). An imaging system for monitoring the in-and-out activity of honey bees. *Computers and Electronics in Agriculture*, *89*, 100–109. https://doi.org/10.1016/j.compag.2012.08.006

Danka, R. G., & Beaman, L. D. (2007). Flight Activity of USDA–ARS Russian Honey Bees (Hymenoptera: Apidae) During Pollination of Lowbush Blueberries in Maine. *Journal of Economic Entomology*, *100*(2), 267–272. https://doi.org/10.1603/0022-0493(2007)100[267:FAOURH]2.0.CO;2

Decourtye, A., Dangléant, A., Allier, F., & Alaux, C. (2018). *La ruche connectée: Objet de surveillance environnementale, de zootechnie ou de découverte récréative. Innovations Agronomiques 67, 77-87*. https://doi.org/10.15454/4LSDMM

Dogan, S., Akbal, E., Koca, G. O., & Balta, A. (2017). Design of a Remote Controlled Beehive for Improving Efficiency of Beekeeping Activities. *8th International Advanced Technologies Symposium*.

Dornhaus, A., & Chittka, L. (2004). Why do honey bees dance? *Behavioral Ecology and Sociobiology*, *55*(4), 395–401.

Doull, K. M. (1976). The Effects Of Different Humidities On The Hatching Of The Eggs Of Honeybees. *Apidologie*, *7*(1), 61–66. https://doi.org/10.1051/apido:19760104

Edwards Murphy, F., Popovici, E., Whelan, P., & Magno, M. (2015). Development of an Heterogeneous Wireless Sensor Network for Instrumentation and Analysis of Beehives. *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, 346–351. https://doi.org/10.1109/I2MTC.2015.7151292

Ellis, M. B., Nicolson, S. W., Crewe, R. M., & Dietemann, V. (2010). Brood comb as a humidity buffer in honeybee nests. *Naturwissenschaften*, *97*(4), 429–433. https://doi.org/10.1007/s00114-010-0655-1

Fitzgerald, D. W., Murphy, F. E., Wright, W. M., Whelan, P. M., & Popovici, E. M. (2015). Design and development of a smart weighing scale for beehive monitoring. *2015 26th Irish Signals and Systems Conference (ISSC)*, 1–6.

Gauci, M., Chen, J., Li, W., Dodd, T. J., & Groß, R. (2014). Self-organized aggregation without computation. *The International Journal of Robotics Research*, *33*(8), 1145–1161. https://doi.org/10.1177/0278364914525244

Giammarini, M., Concettoni, E., Zazzarini, C. C., Orlandini, N., Albanesi, M., & Cristalli, C. (2015). BeeHive Lab project—Sensorized hive for bee colonies life study. *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 121–126.

Gil-Lebrero, S., Quiles-Latorre, F., Ortiz-López, M., Sánchez-Ruiz, V., Gámiz-López, V., & Luna-Rodríguez, J. (2017). Honey bee colonies remote monitoring system. *Sensors*, *17*(1), 55.

Girling, R. D., Lusebrink, I., Farthing, E., Newman, T. A., & Poppy, G. M. (2013). Diesel exhaust rapidly degrades floral odours used by honeybees. *Scientific Reports*, *3*(1), 1–5. https://doi.org/10.1038/srep02779

Goldsmith, A. (2005). *Wireless Communications*.

Griparic, K., Haus, T., Miklic, D., & Bogdan, S. (2015). Combined actuator sensor unit for interaction with honeybees. *2015 IEEE Sensors Applications Symposium (SAS)*, 1–5.

https://doi.org/10.1109/SAS.2015.7133604

Holst, N., & Meikle, W. (2018). Breakfast canyon discovered in honeybee hive weight curves. *Insects*, *9*(4), 176.

Horowitz, P., & Hill, W. (2015). *The art of electronics* (3rd ed.). Cambridge University Press.

Howard, D., Duran, O., & Hunter, G. (2018). *A low-cost multi-modal sensor network for the monitoring of honeybee colonies/hives*. 69–78. https://doi.org/10.3233/978-1-61499-874-7-69

Howard, D., Hunter, G., Duran, O., & Venetsanos, D. (2016). Progress Towards an Intelligent Beehive: Building an Intelligent Environment to promote the Well-being of Honeybees. *2016 12th International Conference on Intelligent Environments (IE)*, 262–265.

Human, H., Brodschneider, R., Dietemann, V., Dively, G., Ellis, J. D., Forsgren, E., Fries, I., Hatjina, F., Hu, F.-L., Jaffé, R., & others. (2013). Miscellaneous standard methods for Apis mellifera research. *Journal of Apicultural Research*, *52*(4), 1–53.

Human, H., Nicolson, S. W., & Dietemann, V. (2006). Do honeybees, Apis mellifera scutellata, regulate humidity in their nest? *Naturwissenschaften*, *93*(8), 397–401.

Kaftanoglu, O., Linksvayer, T. A., & Page, R. (2011). Rearing honey bees, apis mellifera, in vitro 1: Effects of sugar concentrations on survival and development. *Journal of Insect Science*, *11*, 96. https://doi.org/10.1673/031.011.9601

Kernbach, S., Thenius, R., Kernbach, O., & Schmickl, T. (2009). Re-embodiment of Honeybee Aggregation Behavior in an Artificial Micro-Robotic System. *Adaptive Behavior*, *17*(3), 237–259. https://doi.org/10.1177/1059712309104966

Klein, S., Pasquaretta, C., He, X. J., Perry, C., Søvik, E., Devaud, J.-M., Barron, A. B., & Lihoreau, M. (2019). Honey bees increase their foraging performance and frequency of pollen trips through experience. *Scientific Reports*, *9*(1), 1–10. https://doi.org/10.1038/s41598-019-42677-x

Kraus, B., & Velthuis, H. H. W. (1997). High Humidity in the Honey Bee (Apis mellifera L.) Brood Nest Limits Reproduction of the Parasitic Mite Varroa jacobsoni Oud. *Naturwissenschaften*, *84*(5), 217–218. https://doi.org/10.1007/s001140050382

Kridi, D. S., Carvalho, C. G. N. de, & Gomes, D. G. (2014). A predictive algorithm for mitigate swarming bees through proactive monitoring via wireless sensor networks. *Proceedings of the 11th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks - PE-WASUN '14*, 41–47. https://doi.org/10.1145/2653481.2653482

Kronenberg, F., & Heller, H. C. (1982). Colonial thermoregulation in honey bees (Apis mellifera). *Journal of Comparative Physiology*, *148*(1), 65–76. https://doi.org/10.1007/BF00688889

Kulyukin, V. A. (2017). In Situ Omnidirectional Vision-Based Bee Counting Using 1D Haar Wavelet Spikes. *Hong Kong*, 6.

Kviesis, A., Komasilovs, V., Komasilova, O., & Zacepins, A. (2020). Application of fuzzy logic for honey bee colony state detection based on temperature data. *Biosystems Engineering*, *193*, 90–100. https://doi.org/10.1016/j.biosystemseng.2020.02.010

Lacher, V. (1966). Verhaltensreaktionen der Bienenarbeiterin bei Dressur auf Kohlendioxid. *Zeitschrift für Vergleichende Physiologie*, *54*(1), 75–84. https://doi.org/10.1007/BF00298210

Landgraf, T., Bierbach, D., Kirbach, A., Cusing, R., Oertel, M., Lehmann, K., Greggers, U., Menzel, R., & Rojas, R. (2018). Dancing Honey bee Robot Elicits Dance-Following and Recruits Foragers. *ArXiv:1803.07126 [Cs]*. http://arxiv.org/abs/1803.07126

Landgraf, T., Oertel, M., Rhiel, D., & Rojas, R. (2010). A biomimetic honeybee robot for the analysis of the honeybee dance communication system. *Intelligent Robots and Systems (IROS), 2010*

IEEE/RSJ International Conference On, 3097–3102.

Lecocq, A., Kryger, P., Vejsnæs, F., & Bruun Jensen, A. (2015). Weight Watching and the Effect of Landscape on Honeybee Colony Productivity: Investigating the Value of Colony Weight Monitoring for the Beekeeping Industry. *PLOS ONE*, *10*(7), e0132473. https://doi.org/10.1371/journal.pone.0132473

Li, X., Ma, W., Shen, J., Long, D., Feng, Y., Su, W., Xu, K., Du, Y., & Jiang, Y. (2019). Tolerance and response of two honeybee species Apis cerana and Apis mellifera to high temperature and relative humidity. *PLOS ONE*, *14*(6), e0217921. https://doi.org/10.1371/journal.pone.0217921

Li, Z., Huang, Z. Y., Sharma, D. B., Xue, Y., Wang, Z., & Ren, B. (2016). Drone and Worker Brood Microclimates Are Regulated Differentially in Honey Bees, Apis mellifera. *PLOS ONE*, *11*(2), e0148740. https://doi.org/10.1371/journal.pone.0148740

Madden, S. R., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2005). TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, *30*(1), 122–173. https://doi.org/10.1145/1061318.1061322

Mariano. (2018). Evolving robot controllers for a bio-hybrid system. *The 2018 Conference on Artificial Life*. https://doi.org/10.1162/isal_a_00036

Meikle, W. G., & Holst, N. (2015). Application of continuous monitoring of honeybee colonies. *Apidologie*, *46*(1), 10–22. https://doi.org/10.1007/s13592-014-0298-x

Meikle, W., & Weiss, M. (2017). Monitoring colony-level effects of sublethal pesticide exposure on honey bees. *JoVE (Journal of Visualized Experiments)*, *129*, e56355.

Meikle, William G., Rector, B. G., Mercadier, G., & Holst, N. (2008). Within-day variation in continuous hive weight data as a measure of honey bee colony activity. *Apidologie*, *39*(6), 694–707. https://doi.org/10.1051/apido:2008055

Meitalovs, J., Histjajevs, A., & Stalidz, E. (2009). Automatic microclimate controlled beehive observation system. *Engineering for Rural Development*, *8*, 265–271.

Michelsen, A. (1992). How honeybees perceive communication dances, studied by means of a mechanical model. *Behav. Ecol. Sociobiol.* https://doi.org/10.1007/BF00166696

Michelsen, Axel, Kirchner, W. H., & Lindauer, M. (1986). Sound and vibrational signals in the dance language of the honeybee, Apis mellifera. *Behavioral Ecology and Sociobiology*, *18*(3), 207–212. https://doi.org/10.1007/BF00290824

Mitchell, D. (2016). Ratios of colony mass to thermal conductance of tree and man-made nest enclosures of Apis mellifera: Implications for survival, clustering, humidity regulation and Varroa destructor. *International Journal of Biometeorology*, *60*(5), 629–638. https://doi.org/10.1007/s00484-015-1057-z

Moritz, R. F. A., & Southwick, E. E. (1992). *Bees as superorganisms*. Springer.

Nazzi, F., & Le Conte, Y. (2016). Ecology of Varroa destructor, the Major Ectoparasite of the Western Honey Bee, Apis mellifera. *Annual Review of Entomology*, *61*(1), 417–432. https://doi.org/10.1146/annurev-ento-010715-023731

Odoux, J.-F., Aupinel, P., Gateff, S., Requier, F., Henry, M., & Bretagnolle, V. (2014). ECOBEE: a tool for long-term honey bee colony monitoring at the landscape scale in West European intensive agroecosystems. *Journal of Apicultural Research*, *53*(1), 57–66.

Okada, R., Akamatsu, T., Iwata, K., Ikeno, H., Kimura, T., Ohashi, M., Aonuma, H., & Ito, E. (2012). Waggle dance effect: Dancing in autumn reduces the mass loss of a honeybee colony. *Journal of Experimental Biology*, *215*(10), 1633–1641.

Owens, C. D. (1971). *The thermology of wintering honey bee colonies* (No. 1429; p. 32). Agricultural research service, US Dept Agriculture.

Peters, J. M., Peleg, O., & Mahadevan, L. (2019). Collective ventilation in honeybee nests. *Journal of The Royal Society Interface*, *16*(150), 20180561. https://doi.org/10.1098/rsif.2018.0561

Ruan, Z.-Y., Wang, C.-H., Lin, H.-J., Huang, C.-P., Chen, Y.-H., Yang, E.-C., Tseng, C.-L., & Jiang, J.-A. (2017). An internet of things-based weight monitoring system for honey. *World Academy of Science, Engineering and Technology, International Journal of Biological, Biomolecular, Agricultural, Food and Biotechnological Engineering*, *11*(6), 478–482.

Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., & Nagpal, R. (2014). Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, *62*(7), 966–975. https://doi.org/10.1016/j.robot.2013.08.006

Seeley, T. D. (1974). Atmospheric carbon dioxide regulation in honey-bee (Apis mellifera) colonies. *Journal of Insect Physiology*, *20*(11), 2301–2305. https://doi.org/10.1016/0022-1910(74)90052-3

Seeley, T. D. (1995). *The wisdom of the hive: The social physiology of honey bee colonies*. Harvard University Press.

Sherman, G., & Visscher, P. K. (2002). Honeybee colonies achieve fitness through dancing. *Nature*, *419*(6910), 920.

Southwick, E. E., & Moritz, R. F. A. (1987). Social control of air ventilation in colonies of honey bees, Apis mellifera. *Journal of Insect Physiology*, *33*(9), 623–626. https://doi.org/10.1016/0022-1910(87)90130-2

Souza Cunha, E. A. (2019). *Evaluating a Doppler Radar Monitor for Assessing Honey Bee Colony Health*. https://digitalcommons.library.umaine.edu/honors/537

Stabentheiner, A., Kovac, H., & Brodschneider, R. (2010). Honeybee Colony Thermoregulation – Regulatory Mechanisms and Contribution of Individuals in Dependence on Age, Location and Thermal Stress. *PLoS ONE*, *5*(1), e8967. https://doi.org/10.1371/journal.pone.0008967

Stabentheiner, A., Kovac, H., & Schmaranzer, S. (2007). Thermal Behaviour of Honeybees During Aggressive Interactions: Thermal Behaviour of Aggressive Bees. *Ethology*, *113*(10), 995–1006. https://doi.org/10.1111/j.1439-0310.2007.01403.x

Stalidzans, E., Zacepins, A., Kviesis, A., Brusbardis, V., Meitalovs, J., Paura, L., Bulipopa, N., & Liepniece, M. (2017). Dynamics of weight change and temperature of Apis mellifera (Hymenoptera: Apidae) colonies in a wintering building with controlled temperature. *Journal of Economic Entomology*, *110*(1), 13–23.

Starks, P. T., Johnson, R. N., Siegel, A. J., & Decelle, M. M. (2005). Heat shielding: A task for youngsters. *Behavioral Ecology*, *16*(1), 128–132. https://doi.org/10.1093/beheco/arh124

Stefanec, M., Szopek, M., Schmickl, T., & Mills, R. (2017). Governing the swarm: Controlling a bio-hybrid society of bees robots with computational feedback loops. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8. https://doi.org/10.1109/SSCI.2017.8285346

Szczurek, A., Maciejewska, M., Bąk, B., Wilk, J., Wilde, J., & Siuda, M. (2019). Gas Sensor Array and Classifiers as a Means of Varroosis Detection. *Sensors*, *20*(1), 117. https://doi.org/10.3390/s20010117

Szczurek, A., Maciejewska, M., Zajiczek, Ż., Bąk, B., Wilk, J., Wilde, J., & Siuda, M. (2020). The Effectiveness of Varroa destructor Infestation Classification Using an E-Nose Depending on the Time of Day. *Sensors*, *20*(9), 2532. https://doi.org/10.3390/s20092532

Texas Instruments. (2017). *Low-power humidity and temperature digital sensors, HDC2010, SNAS693C*.

Texas Instruments. (2018). *Range measurements in an open field environment* (No. DN018).

Vidrascu, M., Svasta, P., & Vladescu, M. (2016). High reliability wireless sensor node for bee hive monitoring. *2016 IEEE 22nd International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 134–138.

Wilson, M., Melhuish, C., Sendova-Franks, A. B., & Scholes, S. (2004). Algorithms for Building Annular Structures with Minimalist Robots Inspired by Brood Sorting in Ant Colonies. *Autonomous Robots*, *17*(2), 115–136. https://doi.org/10.1023/B:AURO.0000033969.52486.3d

Zacepins, A., Brusbardis, V., Meitalovs, J., & Stalidzans, E. (2015). Challenges in the development of Precision Beekeeping. *Biosystems Engineering*, *130*, 60–71. https://doi.org/10.1016/j.biosystemseng.2014.12.001

Zacepins, A., Kviesis, A., Stalidzans, E., Liepniece, M., & Meitalovs, J. (2016). Remote detection of the swarming of honey bee colonies by single-point temperature monitoring. *Biosystems Engineering*, *148*, 76–80.