Beehive Health Monitoring with .NET Gadgeteer

# Introduction

## Overview

Honeybee colonies are an incredibly important part of agriculture in the UK and elsewhere, as they pollinate most of our crops and produce valuable honey. Unfortunately they are in decline, and beekeepers are facing difficulty keeping their hives alive from one season to the next. Through automated, more frequent and less intrusive monitoring, this project aims to aid beekeepers in improving the survival chances of their bee colonies.

The physical monitoring will be done using ‘Gadgeteer’ hardware provided by Microsoft, the external client for this project. This hardware consists of a microcontroller mainboard with a range of connected components such as a Wi-Fi transmitter, SD card reader/writer, and a range of sensors, for example those for detecting temperature and light. The mainboard, which hosts the .NET Micro Framework, will be programmed on top of the Gadgeteer platform in order to, amongst other functions, automate the monitoring and transmit sensor data to an internet service.

The second aspect of the project is the software to collect, display and visualise data coming from the monitoring device located in a beehive. Requirements from UK beekeepers will determine the nature and extent of this web-fronted application, though always ensuring extensibility is not compromised.

[Project may also involve using custom hardware interfaced with the Gadgeteer platform to expand the existing sensor base to measure original quantities. This may lead to development of an API for these custom sensors.]

## Aims and Goals

Bee-keeping is a widely-practiced hobby as well being involved in large amounts of research due to its great agricultural importance. The primary aim is therefore to develop a system that will be useful for other beekeepers wishing to implement and (if so experienced) further develop a hive health monitoring system. The ideal outcome is to better the survival chance of bee colonies, and perhaps learn more about the state of UK-based beehives such that the community can come closer to understanding the cause of declining colony numbers.

Thus, the goals for the system being developed are:

* Demonstrate what useful beehive properties can be monitored with off-the-shelf programmable hardware
* Demonstrate how bee keepers could be better-informed about the health of their hives than is possible at present with traditional tools, through the use of web services and user interfaces accessing internet-connected embedded devices
* Produce easily extensible open-source code with extensive documentation so that the bee keeping community can more easily make further progress in the field of automated hive health monitoring.

## Scope

In general, allowances need to be made for the fact that hive access will be difficult, so testing is mainly or entirely from lab-simulations. Assumptions about the type and location of the beehive can therefore be freely made (e.g. near mains power and a Wi-Fi hotspot); this is also to limit time wasting from getting potentially unreliable hardware to work (e.g. GSM, batteries).

* The Gadgeteer platform is to be used for the hardware, though some sensors will require interfacing with this platform as not all are currently available in pre-prepared form
* A web-facing application will be the single system front-end
* Reliability of data is a secondary hardware concern, whilst software robustness is a priority; In general, hardware sophistication (including enclosures and power supply) will be limited by time and cost
* The full system will be designed to work on a single hive, as testing multiple devices is beyond limitations of available time and funding.

## Structure and strategy

The project is externally supervised by Steven Johnston from Microsoft. He has kindly provided all the Gadgeteer hardware and offered advice on areas to explore, as well as providing contacts within the beekeeping community to discuss requirements and test the prototypes and final systems. The bulk of the work, however, is unsupervised and self-directed through my research of what is needed.

Due to the rather experimental nature of the project, and the lack of external constraint on the direction to be taken, it was important to decide on a good development strategy. The technique decided upon was the prototype model. By frequently building prototypes and testing them on users and in the laboratory, valuable early feedback can be introduced to better inform the project direction and the final system [Mythical Man Month][]. This iterative method is ideal given the rather broad areas to be researched and developed, from low-level hardware concerns up to high-level software design and implementation with a number of disparate technologies.

## Report outline

The next chapter will detail the necessary background theory and review existing research as well as any commercial or amateur software for beehive monitoring. Chapter three presents the requirements analysis, wherein the details of requirements capture and analysis are described. Following that is a discussion of the system design and implementation in chapter four, then by testing and an evaluation of the project in chapters five and six respectively.

# Context, Background and Research

## Bee colonies

The European honeybee is in trouble. In the past decade, colonies have been dying out a rate of about 30% a year [bee book], with no known cause [ibra.org.uk – look for research paper]. This has been termed Colony Collapse Disorder (CCD), and has sparked a great deal of research into what may be responsible []. Theories include climate change, human pesticide use, and parasitic disease []. What’s needed is better monitoring of these colonies so researchers and beekeepers can be better informed about what is happening in their hives.

The honeybee has for millennia been of great importance to human agriculture, and today colonies are responsible for more than half of the pollination of UK crops [], generating an estimated £200 million [s7]. This huge impact makes CCD a great concern, and recently there have attempts to start monitoring beehives more closely through use of automated systems [] to replace intrusive, laborious manual checks. Colonies of domesticated honey bees are housed in man-made beehives, with around a quarter of a million existing in the UK alone []. Moreover, most apiaries (beehive ‘farms’) are run by amateurs []. These two facts mean that for effective, widespread monitoring, an inexpensive and easy-to-implement system is needed. Focused and sophisticated research is already underway at research laboratories, but the general market lacks a solution.

A typical beehive houses a colony of around 50,000 honey bees at its summer peak [], and is headed by a single queen. Typically, the colony will lie mostly dormant for the winter as the bees struggle to survive on supplies built up during the summer. In spring, activity resumes and the hive will begin raising new bees (the brood) and gathering honey. In early summer, as the population swells a swarming event may occur. This is how new colonies are formed, as the queen leaves the hive with a subset of her colony and tries to find a new nest [].

## Hive monitoring

Beekeeping mainly concerns monitoring the hive to ensure it is healthy so the benefits of crop pollination and the production of honey and wax can be realised. A secondary goal is to manage the swarming events by either moving the bees to a fresh hive when ready (rather than let them escape to possible death), or by preventing it all together. The monitoring currently done involves manual, intrusive checks – for disease, queen presence, food stores, and brood size amongst others []. It would be beneficial to the colony to reduce these checks as far as possible, and make it more convenient for the beekeepers themselves who often don’t live alongside their hives [].

It is very important to realise that beekeepers typically only check their hives on a weekly basis, and not at all in the winter [email – Mark Allan]. Thus, if anything undesirable were to happen between checks, such as swarming or foreign species attack, the beekeeper would not be able to respond to the threat in time, and the colony would likely die. Automation and configurable warning systems are therefore crucial.

## Properties to measure

### Temperature and Humidity

Temperature and humidity are important variables affecting colony survival, both for mature bees and for the raising of larvae in the spring and early summer. In the UK the problem is conditions that are too cold or wet; the colony thrives in a warm, dry environment, and can die when the temperature or humidity stray from a certain range [p12, p17]. In particular, it is the temperature differential between the hive core and the natural external environment that is the most important indicator [s3] of hive activity and health. This differential approaches zero when a colony dies [s4], so by observing this an intervention can staged to discover and rectify the cause. Present techniques often cannot detect this approaching death, as it is considered dangerous to open a hive in the winter so they tend to be left alone for many months in the cold season [].

### Mass

The hive’s mass is an indicator of honey storage and bee population. Although it is difficult to decouple this signal, along with variable contributions from debris, food and the brood [p64, p68], sharp changes are expected from bulk departure (swarming, a common problem for beekeepers), and long-term trends are expected to indicate honey storage []. At present, honey quantities are estimated by the practice of ‘hefting’, whereby a part of the hive is physically lifted from the base and jiggled around to gather an estimate [p70]. Given the shortcomings of this cumbersome technique, and the potential to discover other useful quantities and events from trends, automated electronic mass measurement would evidently be helpful.

### Disturbance and vandalism

Hives can be knocked over by common wild animals such as moles and badgers, and human vandals have been known to disrupt innocent beehives [p69]. By detecting movement with an accelerometer, and opening of the hive lid with a light sensor, it is expected that any form of disturbance can be detected.

### General activity

The entrance to a beehive is typically a thin strip that permits only a few bees at a time []. Consequently, the area around the entrance can be a good indicator of the activity of the foragers in the colony, as bees will tend to cluster there in large numbers when activity is high []. Additionally, the entrance is where foreign invaders such as wasps will stage an attack []. It is therefore desirable to be able to visually monitor the events in this area. It is expected that a camera will be perform this job admirably.

### Other

After extensive research on the possible properties that could be measured, the above are only the ones deemed feasible given the timeframe and project scope. Detection of foreign species (invaders and parasitic mites) through image analysis is one example of a useful property rejected on these grounds. Sound detection with a microphone, which has a multitude of potential uses including queen and disease-detection [], was thoroughly researched and partially implemented, but became too complex to proceed.

However, the system design should be such that integrating further sensors as they become available should be easier to achieve.

### Sound

<INCLUDE IF MICROPHONE IS USED>

Sounds in the hive could be important indicator of activity (drone ratio [p22], swarming [pxx], worker jobs (nurse vs. food collector) and queen presence [s5], disease detection/CCD [s1]). . Sensitive accelerometers embedded in the hive wall have been shown to pick up the ultrasound-induced vibrations of bees communication (swarm prediction for example) [s6], though Gadgeteer is likely not sensitive enough. Microphone could, however, provide a crude estimate, though analysis would be tricky.

<END INCLUDE>

## System components

### Gadgeteer platform

Microsoft’s Gadgeteer platform is a set of open-source hardware consisting of small modules which plug in to an ARM-powered mainboard - a microcontroller running on the .NET Micro Framework (NETMF), along with an open-source SDK for each module and mainboard. NETMF is essentially a subset of Microsoft’s .NET framework that can run on devices with limited memory and processing power, with some additional features specifically targeted to embedded applications []. Programs on NETMF are written in C# and developed in the Visual Studio IDE.

Gadgeteer’s plug-based architecture provides an abstraction of the underlying electronics, which makes it quick and simple to set-up and deploy working prototypes [], giving it an advantage over similar devices such as the Arduino and Raspberry Pi []. The downside is the limited availability of off-the-shelf modules, though it is possible to expose the inner mainboard connections and thus make a custom module, then program it using the core NETMF library. Programming the microcontroller is done using the Gadgeteer SDK [], which runs on top of NETMF.

For beehive monitoring, the useful Gadgeteer modules are mainly passive sensors like thermometers, though other utility modules will be needed, such as one for internet connection.

### RESTful web API

In early prototypes of the system, an Internet of Things web service called Xively [] was used. This provides the tools to connect an embedded device to the internet through a RESTful API and database backend. This service worked well for extracting current data from a fixed number of sensors. However, it has a number of major drawbacks which ultimately meant that a replacement had to be found. The principal problem is the limitation and difficulty in retrieval of historical data, though the less-than-desirable flexibility for changing the sensors base proved problematic too.

To this effect, it was decided that a custom RESTful API would be built with Node.js, and designed to better fit the flexibility requirement of this project. Node is a server-side software platform ideal for building networking applications such as RESTful APIs []. Writing a Node application is done in JavaScript, meaning seamless integration with the client-side web application – easy passing of JSON objects and no impedance mismatch as is common with rivals such as ASP.NET and PHP []. The networking model is single threaded asynchronous, in contrast with Apache’s synchronous model – when IO is required, Node continues running the thread and sets up a callback to execute when the IO is finished, rather than Apache’s technique of halting the thread on IO and using other threads. Apache is better for algorithmically intense applications such as multimedia processing, but Node performs better for lightweight RESTful services []. It is for all these reasons that Node.js was selected to perform all the server-side tasks.

The other component needed for the API is a database. Microsoft’s Azure Table service [] was chosen to fulfil this role because of its flexibility and scalability, important given that vast amounts of data can be generated by a continuously running monitoring device. Azure Tables offer structured storage of key-value pairs in a NoSQL database. The service is strongly oriented towards providing scalability by offering automatic load balancing [NEED TO REVIEW THIS SECTION].

### Web front-end

The system’s frontend GUI will be built using the industry-standard web technology stack: HTML5, CSS3, and JavaScript with JQuery []. Additionally, to speed up development the Bootstrap framework will be used []. Since frontend development is not part of this project’s scope, this is expected to be a justifiably valuable way to save time building a modern, responsive GUI. For similar reasons, a graphing library called flot will be used to power any charts; this is a popular, free, and well-maintained JavaScript library [].

### Version control

GitHub was chosen to host the source code []. This is a modern version control platform using Git. It has achieved widespread acceptance and integration with other service (see below), making it ideal over the alternatives []. Codeplex (running the SVN system) was trialled in an early prototype, but its poor integration with other services meant it was eventually disfavoured.

### Other tools

All the server-side tools mentioned above could be hosted on a local machine. However, in order to improve the visibility of the project so that beekeepers can provide the best possible feedback, it would be beneficial to host the entire project on the World Wide Web. Windows Azure’s Web Site service is ideal given its support for Node.js, and the useful feature of automatic deployment from GitHub each time the linked repository is updated. The current project is running on Azure with a free student trial license.

## Existing work

A number of research efforts have been conducted in the field of beehive health monitoring; typically involving custom-engineered (i.e. not off-the-shelf) hardware with a view to investigating a specific monitoring property. Notable examples include: using accelerometers to detect swarming [s1], swarm-detection with thermometers and microphones [s2], and detecting signs of colony development by monitoring temperature differentials [s3]. Various patents have also been submitted for acoustic monitoring devices [s5] [apidictor]. All these efforts lack the web connection and front-end aspect of this project’s intention, but have provided the author with useful ideas for monitoring (see §2.3).

The HOBOS teaching project is a sophisticated and fairly complete all-round monitoring system with a live video web stream of the hive entrance, and live graphs of all the sensors []. It is intended as a one-off educational tool, costs many thousands of pounds, and is closed-source; these facts make HOBOS unsuitable for a widely-used hive monitoring system.

A few commercial integrated monitoring systems exist too, at least one of which has an online user interface for the live data []. Due to the proprietary nature of the software and hardware involved, these systems are not considered competitors, as a main goal of this project is to provide a platform for further development by anyone.

Various hobbyist attempts at a few of the sensors, some with web service connection, can be found in the literature [][][]. However, no serious integrated multi-sensor solution can be found, and the web interfaces that do exist leave much to be desired in functionality and modernity (see appendix?).

It is therefore evident that the work to be undertaken is sufficiently novel to justify its implementation, and it is hoped that it will become a standard for future beehive health monitoring projects.

# Requirements Analysis

## Problem statement

Save the bees. Aid the beeks.

## Monitoring device

Requirements for the monitoring device to be placed in beehives were gathered from a number of sources:

* Discussions with beekeepers, both hobbyists from the software development community and professionals from the British Beekeepers Association (see A2)
* Research of existing works from the available literature (§2.5)
* Results from prototyping and testing in laboratory conditions
* Limitations provided by the project scope (namely time and cost).

The physical hardware to be placed inside the beehive, and the program running on the microcontroller, shall be referred to as ‘the device’, an abbreviation of ‘hive health monitoring device’.

### Hardware requirements

|  |  |  |
| --- | --- | --- |
| ID | Description | Priority |
| RH01 | The device shall be capable of measuring the following basic properties of the beehive: internal temperature, external temperature, internal humidity, light level, and movement (Boolean). | 1 |
| RH02 | The device shall be capable of measuring the overall mass of the beehive. | 2 |
| RH03 | The device shall be capable of recording short (10s) samples of low frequency (>100 Hz) sound. | 3 |
| RH04 | The device shall be capable of capturing still images of the hive entrance. | 2 |
| RH05 | The device shall be capable of capturing a continuous video stream of the hive entrance. | 3 |
| RH06 | The device shall have the ability to connect to wireless internet. | 1 |
| RH07 | The device shall be able to read and write data from local storage. | 1 |
| RH08 | The device shall be continuously powered with 5-15 V. | 1 |

Table 1.1: Hardware requirements for the monitoring device. Priorities are generally influenced by availabilty, cost, complexity and time limitations, whilst some are essential for the rest of the project to run succesfully.

### Software requirements

|  |  |  |
| --- | --- | --- |
| ID | Description | Priority |
| RS01 | The device shall transmit live sensor data, including any binary data like images, to a web service over HTTP. | 1 |
| RS02 | The device shall additionally commit all numeric (non-binary) sensor data to local storage, in csv format for easy reading. | 2 |
| RS03 | The device shall buffer data to local storage when no internet connection is available, writing the full buffer on resumption. | 2 |
| RS04 | The device shall load all necessary configuration settings from local storage in xml file format, so the device does not to be reprogrammed. | 1 |
| RS05 | The device shall read sensor data and take pictures/audio/video at intervals specified in the configuration file. | 2 |
| RS06 | The device shall be able to operate with any network connection through specification in the configuration file. | 1 |
| RS07 | The device shall commit to the web service specified in the configuration file. | 3 |

Table 1.2: Software requirements for the monitoring device. The emphasis is on configurability and flexibility, reducing the need to reprogram the device if the hive is relocated or front-end requirements change.

## Front-end application

|  |  |  |
| --- | --- | --- |
| ID | Description | Priority |
| RA01 | The application shall prominently display all live sensor data, including any binary data (pictures/audio/video). | 1 |
| RA02 | The application shall graphically display recent (~3hr) trends of the numeric sensors. | 1 |
| RA03 | The application shall display an alarm for each sensor indicating whether it has breached a configurable threshold. | 1 |
| RA04 | The application shall display a report on the past few threshold breaches. | 2 |
| RA05 | The application shall show graphs of longer-term sensor trends, configurable on the period, period length, and variables to display. | 1 |
| RA06 | The application shall allow any numeric data to be exported to CSV, JSON and XML format, with the configurability specified in RA05. | 1 |
| RA07 | The application shall automatically send alerts of threshold breaches to a configurable email at a configurable rate. | 1 |
| RA08 | The application shall display reports on simple statistical analysis of the collected data such as monthly min/max/mean. | 3 |
| RA09 | The application shall display a local weather report from a nearby station using a weather API. | 3 |
| RA10 | The application shall display numeric data in configurable units. | 2 |
| RA11 | The application shall store and display recent historical (24 hrs) binary data such as images. | 2 |
| RA12 | The application shall connect to a RESTful API to get and store live data. | 1 |
| RA13 | The application shall use responsive design to be suitable for use on desktop and mobile devices, and run in a standard modern browser. | 2 |
| RA14 | The application shall be customisable on the sensors displayed based on what is available in the database. Customisation is to be via a JSON settings file passed through the RESTful API, with some features showcased through the GUI. | 1 |

Table 1.3: Front-end Application requirements. Priorities are mostly based on aggregated responses from target users.

CSV support is important because many beeks are not technically skilled in JSON/XML (can be artists, scientists, techies etc [email]), so this gives easy and custom-queryable access to their data from the web.

## Other (raw)

* All code to be well-documented so it is easy for novices to pick up
* All processes well-documented so bee keepers can implement on their own
* Core Gadgeteer system to be easily extended when new sensors come along

## Work Packages

* Core system prototype (initial sensors + Xively pump)
* Full stack prototype (own JS front-end, REST & db back-end– Azure + Node.js)
* Flexible system (‘API’ of sorts) that is customisable on the sensors (data and GUI); this will be a fully tested system and be extensible, robust, documented etc. For testing the ability to use new sensors, a Gadgeteer hiveSense simulator (command line) will be used, negating the need to build sensors

# Design and Implementation

## Monitoring device

Basic sensors, pump, buffer, config.

## Data model

## RESTful API and other server logic

## Web application

GUI for the live data and graphs.

## Complete system

Bringing it all together.

# Testing

Field tests, user-acceptance for the web app.

# Evaluation

It worked.

Current cost of device - ~£200 []. Expected drop to £50 in specialised production, at most. As mention in sec.2, beehives’ value is £200m across 250,000 hives in UK due to pollination alone (a few 10s more from honey [s8]) => > £800 per hive => saving the life of a colony is very valuable.

# Bibliography

[1] x

[2] y

[3] z

NodeJS start-up: https://leanpub.com/nodebeginner

# Appendices

###### Competitor Analysis

Beewatch.biz

Source: <http://www.beewatch.biz:8080/Basic/> (username GUEST, password guest, then select scale).

A commercial service for mass and environ monitoring using their sensor. Data transmission is via GSM.

Good graphs but the table is useless and buggy. Decent export function but little else can be done.

[img1]

Other commercial

Some behind a pay wall – e.g. Hivemind (<http://hivemind.co.nz/>). Data transmission is via GSM.

Arnia (<http://www.arnia.co.uk>) claim to have sensors for: environ, mass, sound, vibration, CO2; all sent to web server with a variety of analytics (queen, swarm, brood). Pictures exist of attractive and comprehensive web app, but no details on how to get it as the product is in RnD stage and only being tested in research field (see BBC article).

Hivetool.org

Source: <http://hivetool.org/>

Environ and mass data in research/education situations. Data sent to local web server (power hungry).

Atrocious organisation – a real mess of info material and data; shocking graphics, and slow.

[img2]

OpenEnergyMonitor

Source: <http://openenergymonitor.org/emon/beehive/v2>

Environ only (but x4 temp in different parts of the hive). Data is wifi-ed to local PC and relayed over http to simple gauges and graphs on blog posting. Battery operated.

[img3]

###### Requirements Gathering

Mr Allan (email) primary requirements: camera covering the entrance, *external*temperature, hive/supers mass, movement detection. Secondary detections: disease, stores, pollen, laying pattern, mites, full supers. Not interested: internal temp/hum. Power: battery (on-board or nearby). Data: live broadcast to web.

Mr McGuire (email) primary requirements: internal temp, brood core-edge temp difference, swarm detection (suggests detecting: queen pheromone, drone level, queen cells being tended!), movement, supers’ mass. Secondary: vibration, in/out rate. Power: solar or on-board. Data: n/a.

Romsey & District BKA (email) primary requirements: internal temp/hum, queen detection, swarm detection, identify non-bees at entrance (wasps mainly). Secondary: hive mass. Power: n/a. Data: n/a/.

Mr Flottum, Senior Editor of Bee Culture Magazine (<http://colonymonitoring.com/cmwp/for-entrepreneurs/>) primaries: Varroa level, queen detection. Secondary: weight, environ. Power: battery. Data: n/a.

###### Source Code

Public static void main();

###### System Manual

Plug and play.

###### User Manual

Hack away.