1 Misc. Notes

Azure log live output: go to azure command prompt and type: “azure site log tail hivesensenodejs”

Output / convert PDF to black and white / greyscale:

PS C:\Program Files\gs\gs9.09\bin> .\gswin32.exe -o ouput.pdf -sDEVICE=pdfwrite -dPDFSETTINGS=/prepress -sColorConversionStrategy=Gray -sProcessColorModel=DeviceGray Notes.pdf

2 Logins and passwords

- WindowsAzureDB: uclbees, uclsenseDB2013
- AzureFTP: hivesense, uclsense2013
- AzureSQLDB: hiveSense, UCL2013hs
- Gmail: hiveSense.net, beekeepers2013
- AzureStorage: hiveSense, PAK: c2z/4TLAQSmb32ImEKocqmDax7PuJI2B+gZdAT2LufQmYdmGxSfTAJT6Z6XCqny19WLPGznSdU8Cs8k6ksEe1Q==, SAK: rCdiBjatyH4dQjUDLqsR+SRHvDNWkiPE0j40OW0lZh6wrMokgSwe7/3o8hF92OExtHXz0V1BHt3W//qmFWjvag==

3 Activity Log

3.1 Wed-29-May to Fri-7-Jun

- Read and made notes on BBKA book
- Briefing with Dean (internal supervisor) on general project structure
- Research on monitoring sensors and detection possibilities
- Briefing with Steven (external supervisor)

3.2 Mon-10-Jun

- Wrote-up hardware requirements from the clients’ emails
- Researched feasibility of having a camera over the hive entrance. Due to computational cost of ADC-ing a cam stream, ideal would be to use a wireless IP cam for streaming (£50). Cheaper alternative (£15) for prototype is to use USB webcam and stream using C# - http://stackoverflow.com/questions/233455/webcam-usage-in-c-sharp.
- Read research paper on load sensing. Conclusion: good set-up but expensive and requires extensive hardware work. More: http://www.comp.lancs.ac.uk/~albrecht/smart-its/platform/load/index.htm. Probably means load is unfeasible without significant funding, unless the hobby load sensor (£8) x4, with ADC, an op-amp and some other cheap stuff, could work, which it might but still need funds for *anything* non-Gadgeteer.
- Read about basic electronics. Not too helpful – need to get hands-on!

- Researched Xively. Found some sample code for getting data to it from netmf (http://www.ghielectronics.com/community/codeshare/entry/477). JS-library can be used for interfacing with my monitoring web app, though probably easier and more flexible to just do direct http GETs? Then again, see the JS tutorial for a nice subscribe feature to get live data into a DOM element. Xively can also do graphs natively (see API ref for single datastream).

3.3 Tue-11-Jun

- Spoke to Dean – found a workshop in 4.03 to do soldering
- Investigated domains – easy to add on to Clook in the client area -> order -> register domain (£5)
- Started mockup of web app in balsamiq
- Competitor analysis done (front-end web services)
- Load cell research – the sparkfun one would require a fair bit more circuitry (instrumentation amps, RC filter at min); plus, need 4 of them and the amps are £4 so total cost is near £50; basically they are pile of shite because they are only half of a wheatstone bridge. Alternative – how about hacking a bathroom scale? Still need amp but scale comes with 4 load cells (full bridge) so total is closer to £15 and will require less complex circuitry. Nerdkits has a good tutorial. Load cells can also be constructed from strain gauges – 4 per cell at £5 though is expensive, and the construction is prone to error for the novice. A full wheatstone bridge load cell (4 wires) is for sale here (http://proto-pic.co.uk/3135_0-micro-load-cell-0-50kg-czl635/), this could be an alternative to hacking the bathroom scale.

3.4 Wed-12-Jun

- Browsing the beek fora for ideas
- Found some more competitors
- Wrote the scope, v1
- Researched Wheatstones, and instrumentation amplifiers – go with the AD620 (A or B) DIP
- Wrote requirements v1
- Decided to work on the basis of scope being a single hive data stream and a single user, for now
- More mockup work
- Researched JavaScript frameworks (Backbone, ember). Need to know whether I should use any.

3.5 Thu-13-Jun

- Hardware day – set up hive sense V1
- Temperature x2, acceleration, humidity, light, camera all set up and working at debug line
- Tried to set-up wifi but had some problems – work in progress
- Set up ankhSVN in Visual Studio and got a local repository going through ToroiseSVN
- Signed up for Codeplex but had problems connecting to vis studio to allow code upload
- Signed up to Xively –need to add some channels to the feed.

3.6 Fri-14-Jun

- Changed scope so focus is in one hive, one person, aim is to help others
- Got networking to work, finally
- Code clean-up and set to work on Xively-interfacing.

3.7 Mon-17-Jun

- Set up Xively – data is now pushed there by a timer
- Codeplex site now has source code (need to use TortoisSVN -> Import)
- Investigated Xively graphs – look good, e.g. https://api.xively.com/v2/feeds/1693757499/datastreams/AmbientTemp.png?w=750&h=400&c=8877aa&t=TempExternal&g=true&b=true
- Researching image upload(ftp/http possibilities)
- Try to get image data to post to nw3weather – works!
- Try to get camera bitmap to post… trouble!

3.8 Tue-18-Jun

Day off – trip to Oxford w/ L & M

3.9 Wed-19-Jun

- Got SD card working – just saving images atm
- Image POST working – trick was to read POST as $HTTP_RAW_POST_DATA (binary) in PHP
- Wrote first blog post.

3.10 Thu-20-Jun

- SD card writing sensor data in csv format
- Wifi now auto-checks for connectivity on a timer, so will re-join in case of downtime
- Wifi network up/down triggers working properly
- Reading from SD card, including line-ending detection for missed-upload lines which are only logged when no network exists.
- Set up a file system for the SD card – own directory and config file (will be XML).

3.11 Fri-21-Jun

- Set up automatic time-syncing – periodic and in case of network reconnection
- Wrote a very basic xml parser for nodes only
- Settings loaded from xml config file
- Buffered data from SD getting to Xively
- Auto-reboot configured – periodic to account for system hangs
- Started refactoring into design – class separation.

3.12 Mon-24-Jun

- Refactoring completed for this stage – classes for wifi, SD, config, and sensors made
- Updated Codeplex and blog
- Emailed Steve about progress

- Wrote potential new scope, given issues acquiring hardware
- Wrote extensive C# documentation.

3.13 Tue-25-Jun

- Got the device working with mains supply (9V adapter)
- Explored workshop; they have: multimeter, soldering, resistors, caps, PSU (see above). Nothing else, but I can make list for admin and he will try to get for me
- Tried to build html documentation using Sandcastle but it doesn't support NETMF (properly – managed to hack it to get the Program class done, but nothing more; have opened discussion on Codeplex so this issue is open)
- Worked out how to add Visual Studio class references (project -> Add ref) – now I can use the XML library! Also, I can now remove redundant libraries
- Dean group meeting –book-based references, work packages (experiments/testing – validation that the deliverables meet the project title/goals), Azure
- Re-wrote Goals and Scopes, tweaked requirements, added work packages (milestones).

3.14 Wed-26-Jun

- Discovered how to wire analogue inputs to Gadgeteer (internal 10-bit ADC)- http://technicalstuffhopefullyuseful.blogspot.co.uk/2013/01/using-analog-input-sensors-with-net.html
- Researched Microphone – found a cheap digital(!) MEMS mic that should work out the box hooked-up to a Gadgeteer digital pin as per adafruit blog post … NO! Very complicated stuff due to acceptable socket input types. Easier to use analogue and use the onboard ADC, then use RLP (https://www.ghielectronics.com/docs/50/rlp-getting-started) to process to WAV. Also, the GHI Music Module can record, but only through "line-in" – possible to convert microphone to line-in (try just using my phone?)– ask tech support about this once the module has arrived (I have just requested it from Microsoft, along with breadboard); new thought – the line-in solution is rubbish: not going to able to resolve audio finely enough, so I either get an analogue mic breakout module or accept the rubbishy-ness and try to explain it away as prototype-ery.
- Wrote and sent shopping list to Dean
- Researched some more background material for sensor justification
- Tried to determine whether accelerometers are feasible as vibration detectors – yes they are (as in s1 paper) but possibly not for simple ones like Gadgeteer version.

3.15 Thu-27-Jun

- Fleshed out the research section of report
- Looked at a Gadgeteer testing and design patterns blog - http://possiblythemostboringblogever.blogspot.co.uk/2013/06/make-your-gadgeteer-app-testable-with.html
- Started work on the front-end GUI using Bootstrap responsive CSS.

3.16 Fri-28-Jun

- Continued work on HTML/CSS GUI
- Started to wire up the GUI with JS to display live data.

3.17 Mon-1-Jul

- Researched JavaScript object-oriented principles (public, private, ns)
- Continued on JS for the GUI
- Researched graphing libraries for JS – go for flot or highcharts.

3.18 Tue-2-Jul

- Produced dashboard graphs with flot
- More JS hook-ups for the dashboard.

3.19 Wed-3-Jul

- Div-based navigation working nicely
- Alarms working so rough draft (server-free) dashboard complete
- Met Dean and made notes on things for the report
- Started work on Graphs page.

3.20 Thu-4-Jul

Day off - Birthday trip to Wimbledon

3.21 Fri-5-Jul

Job interview prep most of the day

- Setup Azure started –see main bookmark to resume.

3.22 Mon-8-Jul

Job Interview most of the day

- Github working locally and remotely (Netbeans is best but can do through posh git)
- Azure website set up and running (Github connection)
- Git research so I can do it properly (sans Netbeans, and actually understand wtf is going on)
- Git set up properly now (command line)
- Wrote new blog post and updated Steven.

3.23 Tue-9-Jul

- Azure working with PHP and so server-side logic begun.

3.24 Wed-10-Jul

- Report writing – introduction complete, background started

3.25 Thu-11-Jul

- Report writing – finished background, added to intro, started requirements

- Minor GUI work
- More hardware arrived – mounted the kit onto a plate, rest untouched

3.26 Fri-12-Jul

- Researched hooking-up the load cell (now arrived) – AdvancedADC not working with firmware; possible fix on GHI forum but will proceed with native ADC for now
- Started implementing load cell code and breadboarding from drawn schematic.

3.27 Mon-15-Jul

- Tried to get load sensor working but failed
- Node.js research and started coding basic web server – locally working but not on Azure.

3.28 Tue-16-Jul

- Node – more webserver (POST, PUT and GET all working) and working on Azure
- Data model research – decided on relational db, made ER diagram, set up SQL server db on Azure, researched and started linking db with node.

3.29 Wed-17-Jul

- Failed to get SQL-server node module to build
- Node – serving static content, making GETs to other servers

3.30 Thu-18-Jul

- Node / Azure – set up Table storage for my schema and got live POSTing working
- Node – set up emailing using emailjs module
- Gadgeteer – modified code so data is pushed to Azure and Xively
- Azure – command line live logging working so I can see production console.log() output

3.31 Fri-19-Jul

- Finished Table set-up so REST queries can get new data efficiently
- Started implementing the REST service – getCurrent, postCurrent, error check
- Wrote Gadgeteer simulator - command line tool that auto-POSTs random data to the server, meaning no need for building custom sensors (just spoof them in this simulator)
- Set up front-end to grab current data from node REST service rather than Xively

3.32 Mon-22-Jul

- Prepared mid-project presentation
- Improved recentHistory GET to ignore huge data jumps

3.33 Tue-23-Jul

- Delivered presentation – feedback OK, specifics: money saved, data analysis?
- Report - finished background writing (tech stack)
- Node - Image POST saving to file and GET that image (POST using Gadgeteer simulator)

3.34 Wed-24-Jul

- Converted POST/PUT datapoints format from CSV to JSON
- Researched RESTful principles but yet to fully implement
- Refactored Gadgeteer code and formally dropped support for Xively
- Implemented node method to retrieve buffered historical data
- Tested PUT of 114 datapoints (x5 entities) into Azure tables from GadgeteerSimulator – successfully completed in ~5s => 100 lines PUT per time is reasonable
- Tested sending repeat data to ATS: gives error but nothing breaks – that'll do, Ben, that'll do.

3.35 Thu-25-Jul

- Gadgeteer device successfully posting data (bin and utf8) to Azure (Node 0.8 upgrade)
- REST API – fully implemented recent history grab
- Worked on main graph to add more features.

3.36 Fri-26-Jul

- GUI improvements – mainly on Graph tab, also a few bug fixes and time synching
- REST API – added GET support for XML and CSV, and documented full GET API.

3.37 Mon-29-Jul

- Bug fixing for sending historical data to the API (bad date format)
- Added settings tab with setting for weather (prototype only)
- Updated blog
- Fixed numerous bugs on the time syncing feature
- Fixed bug on url query encoding
- Asked beek for feedback
- Improved GUI 'about' tab.

3.38 Tue-30-Jul

- Requirements gathering - finished emailing beeks, and updated Steven
- GUI/REST – alarms added dynamically from json settings file on server; file can be modified by PUT requests; settings tab has permanent commit option which saves settings on server – working for wxLocation.

3.39 Wed-31-Jul

- Requirements gathering - replied to beeks
- GUI - Improved documentation
- Git - branching research and setup of first hiveSense branch
- Azure – local emulated storage working (set environment var – see sec. 1), also figured no need for config.json file to load dev storage settings – will load from Azure management portal in tab 'configure' of the website
- API – implemented security header-checking for POST/PUT, and documented P/P
- API/Data-Model – revamped to be sensor-independent/flexible (schema-less db); server does not care at all what data is thrown at it, now only the GUI handles what sensors to use.

3.40 Thu-1-Aug

- GUI – non-default sensor blocks loaded from json settings, and their graphs too
- Recent data now only loaded along with other historic data, instead appending to local memory giving much performance boost
- Tested and debugged addition of new sensor successfully
- Made the dashboard graphs auto-resizable.

3.41 Fri-2-Aug

- Server – alarms monitored automatically each time current data is PUT, emails sent
- GUI – added HTML for alarms and general settings, hooked-up the alarms bit to JS so now possible to add and modify alarms dynamically (though local save doesn't yet affect DOM).

3.42 Mon-5-Aug

- GUI –pretty data tables (with datepicker – not hooked up to API yet, nor exporting)
- API – arbitrary historical range, support for .csv, .xml etc access to different formats.

3.43 Tue-6-Aug

- GUI – finished tables (loaded from datepickers, exportable, mean, re-ordered)
- GUI –finished alarms (deleting, affecting the live interface, validation).

3.44 Wed-7-Aug

- API - updated and prettified docs
- GUI - implemented hive-name and wx-place settings
- Production – merged flexisense with master and committed to Azure without problem
- Comms – new blog post, updated Steven and Rob
- Report – requirements analysis.

3.45 Thu-8-Aug

- Report – finished requirements analysis (terminology, requirements re-write, domain model, use case diagram and spec, work packages)
- Report – started Design/Implementation (system architecture).

3.46 Fri-9-Aug

- Report –Design (Device hardware)
- Report – manuals (Device)
- Refactoring – Device design improved.

3.47 Mon-12-Aug

- Device – bug fixes and refactored into better design
- Device – made class diagram (part VS auto-gen, minus a bunch of private stuff)
- Report – wrote-up design/imp for the Device.

3.48 Tue-13-Aug

- UML – state machines for Device (general), and Device data point
- Report – REST principles
- Refactoring – API + Web server/services.

3.49 Wed-14-Aug

- Refactoring, API – database logic extracted
- Report – design write-up for API and data model complete
- Report – system manual for API
- Manuals – API user docs improved and extended.

3.50 Thu-15-Aug

- Refactoring – major redesign of GUI code into packages with static classes.

3.51 Fri-16-Aug

- Refactoring – experimented with de-static-ising some classes. Successfully done on some, but seems pointless since entire design is so geared to static access
- Report – Design – web application. Design/Imp now complete
- GUI – added ability to modify complete settings file
- Manuals – web application including extensive settings description.

3.52 Mon-19-Aug

- Testing – Device/API communication
- Report – Testing – new sensors, device-api comms (max freq), user-acceptance
- Testing – Device sim UX improved with command-line parser library, and functionality extended to producing CSV historical data for given date start, interval and quantity.

3.53 Tue-20-Aug

- Testing/Implementation – Device buffering working properly as per requirements
- Report – Testing section draft complete.

3.54 Wed-21-Aug

- Bug fixing – API and front-end. Complete
- Report – Evaluation/Conclusions completed. Report draft therefore finished.

3.55 Thu-22-Aug

- Report – references found and filled-in with BibWord
- Report – minor rewrite of background, for clarity.

3.56 Fri-23-Aug

- Report – abstract
- Report – cover, contents, diagrams
- Misc – found cmd-line tool to convert pdf to greyscale.

3.57 Tue-27-Aug

- Yesterday was a bank holiday
- Report – editing.

3.58 Wed-28-Aug

- Report – editing
- Report – appendices (mass sensor, user manual API - reformat).

3.59 Thu-29-Aug

- Code commenting – Device and API
- Report – Source code appendix (Device and API).

3.60 Fri-30-Aug

- Report – finished source code addition (and commenting of GUI code)
- Report – all remaining content added (user manual screenshots were final pieces).

3.61 Mon-2-Sep

- Report – proof reading
- GitHub – finalising repositories and complete documentation
- Project hand-in.

4 Long-term plan

Five weeks remain. Rough allocation of time is to be as follows

~~1. **Implementation** – fulfilling beek GUI requirements, finishing API, making GUI responsive to sensors in the db~~

2. ~~**Implementation** finish-up~~, ~~**write**-up of Requirements~~
3. ~~Code **refactoring** into good design then **write** that up~~, ~~plus **manuals**~~
4. ~~**Testing** and its **write**-up, plus bug-fixing~~; ~~**Write**-up of conclusion/evaluation; bibliography (use Word’s built-in BibTeX-style sources manager);~~ ~~images~~; ~~abstract~~
5. ~~**Write**-up appendices~~; ~~proofing and editing~~; ~~code commenting and clean-up~~.

Penultimate day (Mon-2-Sep) is reserved for printing and binding.

Final day is for hand-in only.

5 Progress Diary (summary of activity log)

5.1 Week 0 (Wed-29-May to Fri-7-Jun)

- Read and made notes on BBKA bee-keeping book
- Briefing with Dean (internal supervisor) on general project structure
- Researched monitoring sensors and detection possibilities

- Briefing with Steven (external supervisor) on early specifics.

5.2 Week 1 (Mon-10 to Fri-14 June)

- Write-up
    - Hardware requirements from the clients' emails
    - Scope and rough requirements, v1
    - Competitor analysis (front-end web services).
- Research
    - Feasibility of having a camera over the hive entrance
    - Xively (internet-of-things REST service) for posting data to web from hive
    - Load cell – wheatstones, strain gauges, op-amps; hobby vs. commercial solutions
    - JavaScript frameworks (Backbone, ember etc.)
    - Browsed the bee keeping fora for sensing ideas.
- Mockup of web app (system front-end) in balsamiq
- Hardware arrived so set up device V1 - temp x2, acceleration, humidity, light, camera
- Set up ankhSVN in Visual Studio and got a local repository going through Tortoise SVN.

5.3 Week 2 (Mon-17 to Fri-21 June)

- Gadgeteer software
    - Set up Wi-Fi module and Xively – data is now pushed there by a timer
    - Wi-Fi now periodically auto-checks for connectivity, so will re-join in case of downtime
    - Reading from SD card of config and buffered data, which is sent to web
    - Very basic XML parser for nodes only, so settings loaded from xml config file
    - Auto-reboot configured – periodic to account for system hangs.
    - SD card writing – saving csv data and images.
- Codeplex site setup with source code and basic description
- Researching image upload(ftp/http possibilities), eventually working through POST
- Day off on Tuesday – trip to Oxford
- Wrote first blog post on the Wordpress site (set up by Steven).

5.4 Week 3 (Mon-24 to Fri-28 June)

- Updated Codeplex and blog
- Re-wrote Goals and Scopes, tweaked requirements, added work packages (milestones)
- Dean group meeting –general advice and hardware promise
- Started work on the front-end GUI using responsive CSS, and begun to wire it up in JS.
- Gadgeteer software
    - Refactoring – classes for Wi-Fi, SD, config, and sensors made
    - Wrote extensive C# documentation but failed to convert xml to website (Sandcastle)
- Research
    - Discovered how to wire analogue inputs to Gadgeteer (internal 10-bit ADC)
    - Bee sound sensing: Microphone – analogue, digital, processing etc. – likely to be v. difficult, accelerometer: Gadgeteer one not sensitive enough for vibrations

    - Some more background material for sensor justification
    - Gadgeteer testing and design patterns.

5.5 Week 4 (Mon-1 to Fri-5 July)

- Research
    - JavaScript object-oriented principles (public, private, namespace, inheritance)
    - Graphing libraries for JS – opted for flot and produced dashboard graphs
- GUI
    - Continued on JS for hooking up the GUI to live data from Xively
    - More JS hook-ups for the dashboard (e.g. alarms), so this is mostly done for V1
    - Set-up Div-based navigation so application is a "single page" navigation experience
- Little progress on Thu/Fri due to Birthday, and interview prep.
- Met Dean and made notes on things to do for the project report

5.6 Week 5 (Mon-8 to Fri-12 July)

- Set up Github locally and remotely (to replace poorly-supported Codeplex)
- Wrote new blog post and updated Steven
- Server-side software
    - Azure website set up and running with direct Github connection
    - Azure working with PHP and so server-side logic begun.
- Report writing
    - Introduction
    - Background
    - Requirements started
- More hardware arrived – mounted the kit onto a plate, researched hooking-up the load cell then started implementing load cell code and breadboard prototyping.

5.7 Week 6 (Mon-15 to Fri-19 July)

- Tried to get load sensor working but failed for unknown reason
- Research
    - Node.js
    - Data model– wanted SQL server but won't work with Azure + Node, so went for Azure Tables, researched to make retrieval efficient for my use
- Webserver software on Node
    - started building web server and RESTful API, deployed locally and on Azure
    - Serving static content
    - Making GETs to other servers
    - Emailing using emailjs module and Gmail
- Gadgeteer and GUI modified so data is to/from my Node.js Azure RESTful API, not Xively
- Wrote Gadgeteer simulator - command line tool that auto-POSTs random data to the server, meaning no need for building custom sensors (just spoof them in this simulator)

5.8 Week 7 (Mon-22 to Fri-26 July)

- Prepared and presented mid-project presentation - feedback OK, think business and data
- Report - finished background writing (tech stack)
- Refactored Gadgeteer code and formally dropped support for Xively
- RESTful API
    - Image POST and GET implemented and working from Gadgeteer
    - Changed PUT/POST method to JSON and implemented this on Gadgeteer
    - Improved recentHistory GET to ignore huge data jumps
    - Researched RESTful principles but yet to fully implement
    - Fully implemented recent history grab
    - Added support for receiving buffered data and tested this from the simulator
    - Tested sending repeat data to ATS: gives error but nothing breaks
    - Added GET support for XML and CSV
- GUI
    - Worked on main graph to add more features – variable and period switching option
    - Some bug fixes and support for time synching

5.9 Week 8 (Mon-29-Jul to Fri-2-Aug)

- API
    - Settings can be retrieved, and modified by PUT requests
    - Implemented security header-checking for POST/PUT,
    - Documented API for users
    - Revamped to be sensor-independent/flexible (schema-less db); server does not care at all what data is thrown at it, now only the GUI handles what sensors to use; tested and debugged addition of new sensor successfully.
- GUI
    - Large parts are now loaded dynamically from settings.json retrieved over API – alarms, sensors, wxLocation, graphs etc.
    - Some settings (e.g. alarms) have been made GUI-modifiable for user-friendly access
    - Made the dashboards graphs auto-resizable
    - Recent data now only loaded along with other historic data, instead appending to local memory, giving much performance boost
    - Improved documentation and 'about' section.
- Asked and received useful feedback from three beekeepers with validation and comments on which sections are most important
- Alarms monitored automatically on server, each time current data is PUT, with emails sent in response to breaches
- Updated blog.

5.10 Week 9 (Mon-5 to Fri-9 Aug)

- GUI
    - Pretty data tables (with datepicker and mean of values, and exportable)
    - Finished alarms (deleting, affecting the live interface, validation)

  - Implemented hive-name and wx-place settings
- API
  - Arbitrary historical range, support for .csv, .xml etc access to different formats.
  - Updated and prettified docs
- Report
  - Requirements analysis finished (terminology, requirements re-write, domain model, use case diagram and spec, work packages)
  - Design/Implementation (system architecture, Device design)
  - Manuals - Device
- Comms – new blog post, updated Steven and Rob
- Refactoring – Device design improved.

5.11 Week 10 (Mon-12 to Fri-16 Aug)

- Device
  - Bug fixes
  - Refactored into better design
  - Made class diagram (part VS auto-gen, minus a bunch of private stuff)
  - UML – state machines for general operation, and for a data point.
- GUI
  - Added ability to modify complete settings file
  - Major refactor of code into packages with static classes
  - Experimented with de-static-ising some classes. Successfully done on some, but seems pointless since entire design is so geared to static access.
- API
  - Refactor - database logic extracted
  - Refactoring – general redesign.
- Report
  - Wrote-up design and implementation
  - Wrote-up manuals.

5.12 Week 11 (Mon-19 to Fri-23 Aug)

- Testing
  - Device/API communication
  - Device buffering working properly as per requirements
  - Device sim UX improved with command-line parser library, and functionality extended to producing CSV historical data for given date start, interval and quantity
  - Bug fixing – API and front-end.
- Report
  - Testing
  - Evaluation/Conclusions completed. Report draft therefore finished.
  - References found and filled-in with BibWord
  - Abstract
  - Cover, contents, diagrams.

5.13 Week 12 (Mon-26 to Fri-30 Aug)

- Code commenting –all
- Report
    - Abstract
    - Cover, contents, diagrams
    - Editing
    - Appendices.

5.14 Week 13 (Mon-2-Sep)

- Report – proofing
- Hand-in.

6 Email to beekeepers

Dear beek

I am a Masters student at UCL, doing a project with external supervision by Steven Johnston.
I understand there was some communication between you two about requirements for a beehive monitoring device.

My project is a continuation of the work Steven and co. started in April, and I am looking to consolidate requirements for the data visualisation. I have configured a device monitoring a number of properties and linked it to the web.

In addition, I have developed an application for viewing live data, but not being a beekeeper myself, I don't know how useful it is!

The front-end application prototype is here
http://hivesensenodejs.azurewebsites.net/

My question is, how suitable is this application to you, a beekeeper, and what would you like to see in addition?

Bear in mind that the application will be customisable by the beekeeper (in terms of adding new sensors etc.); the prototype shown just demonstrates the visual aspects.
However, I am open to suggestions and comments on aspects that are in nature both visual (e.g. graphs) and non-visual (e.g. email alerts).

Many thanks for your time.

7 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). No concrete platform – a confused and unrelated collection of hardware and software. Good points: deployed in about 10-15 hives, open-source.

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]

8 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.*

Round Two

Mr Allan, Mr Mcguire, and Mr Fatland were shown the prototype, and thus evaluated the GUI and specified key requirements for it.