BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

SYSTEM FOR AUTONOMOUS DATA COLLECTION FROM WEIGHTING SENSORS

SYSTÉM PRE AUTONÓMNY ZBER DÁT Z VÁH

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ADAM LUČANSKÝ**
AUTOR PRÁCE

SUPERVISOR **Ing. VÁCLAV ŠIMEK**
VEDOUCÍ PRÁCE

BRNO 2017

Brno University of Technology - Faculty of Information Technology

Department of Computer Systems Academic year 2016/2017

Bachelor's Thesis Specification

For: **Lučanský Adam**
Branch of study: Information Technology
Title: **System for Autonomous Data Collection from Weighting Sensors**
Category: Embedded Systems

Instructions for project work:

1. Perform in-depth study of sensoric technologies suitable for weight acquisition of the objects under observation.
2. Propose a solution for monitoring weight of beehives and export of the collected data via selected network technology.
3. Design the architecture of a system for data collection and their visualization that allows to monitor the weight of beehives.
4. Implement the proposed system and carefully verify its functionality.
5. Evaluate the achieved results and try to discuss possible extensions.

Basic references:

- According to the instructions of supervisor.

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Šimek Václav, Ing.**, DCSY FIT BUT
Beginning of work: November 1, 2016
Date of delivery: May 17, 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

LS

Lukáš Sekanina
Professor and Head of Department

Abstract

Thesis implements a system for the remote weight measurements of the beehives. Bus-unit board (STM32F0) samples two load cells (HX711 converter) and exposes data to the CAN bus. All units are chained through telephone cable, with a master unit (Raspberry Pi 3) with CAN shield at one end collecting and sending data to the web-server. Wireless data transmission is provided by integrated WiFi. System is battery-powered and charged by a solar panel large enough to accommodate continuous operation. Data are shown to the user in web-interface. Autonomous electronic weight measurement replaces current tedious manual mechanical scale reading and the need to visit beehives personally. System is installed on beehives for a local beekeeper.

Abstrakt

Práca sa zaoberá automómnym zaznamenávaním váhy včelích úlov. Zbernicová jednotka (STM32F0) zbiera dáta z dvoch prevodníkov HX711 a posiela ich na CAN zbernicu. Jednotky na zbernici sú prepojené telefónnym káblom ktorým su aj napájané. Na jednom z koncov zbernice sa nachádza Raspberry Pi 3 s rozšírením o CAN. Nazbierané dáta su posielané pomocou integrovaného WiFi modemu na webserver. Systém je napájaný z batérie nabíjanej solárnym panelom s dostatočným výkonom na nepretržitú prevádzku. Dáta su prezentované užívateľovi vo forme grafu na webovom rozhrani. Electronické váženie úlov nahradzuje aktuálne osobné meranie mechanickou váhou. Systém bol odtestovaný a nainštalovaný lokálnemu včelárovi.

Keywords

beehives, weighting, load cell, HX711, embedded systems, Raspberry Pi, STM32F042, Haskell, Ivory, Ivory Tower, Laravel, open-hardware

Klíčová slova

včelie úle, váženie, silomer, HX711, vstavané systémy, Raspberry Pi, STM32F042, Haskell, Ivory, Ivory Tower, Laravel, open-hardware

Reference

LUČANSKÝ, Adam. *System for Autonomous Data Collection from Weighting Sensors.* Brno, 2017. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Šimek Václav.

System for Autonomous Data Collection from Weighting Sensors

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Mr. Václav Šimek. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Adam Lučanský
May 18, 2017

Acknowledgements

I would like to express thanks to my supervisor for help during writing of this thesis as well as giving me possibility to have printed circuit boards made professionally. Special thanks goes to Richard Marko for showing me the way to the world of embedded Haskell, Hackerspace base48.cz for providing the space and tools for experimenting and Lojzík Kašpárek for the beehives to test on.

Contents

List of Figures

Chapter 1

Introduction

The need for measuring everything, even things we have not thought to measure emerges every year to many industries. One such industry is beekeeping. Health and performance of bees can be quantified by the weight of the beehive, giving insight about the beehives for the beekeeper without the need to open the beehive and disturb bees.

In many areas, millions of colonies of wild (or feral) honey bees have been wiped out by urbanization, pesticides, parasitic mites, and a recent phenomenon called „Colony Collapse Disorder“. Collectively, these challenges are devastating the honey bee population[6, p .32]. Adoption of electronic systems for an automated monitoring may provide early detection of a problem.

In the first chapters 2 and 3, general principles and selected technologies are discussed. Upcoming chapters are dedicated to details of implemented system. Final two chapters show completed device with mounted beehives as well as process of calibration and installation.

Hardware and software outputs of this thesis are published as an open-source. Crossroad repository for the project is located on URL: https://github.com/lucansky/hexamon.

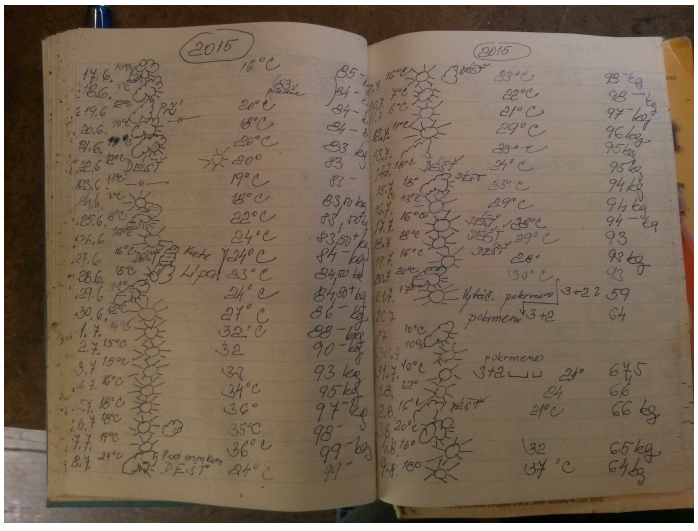

Figure 1.1: Traditional way of recording weight of beehive.

Chapter 2

Analysis

Beehive weighting provides an interesting source of data for a prediction of the health of bees. Weight fluctuation during the day indicates activity of bees, as well as overall performance. Weight also indicates underweight, which may occur during winter or rainy days if there is not enough honey to feed the colony.

Idea of the beehive weighting is nothing new, but is an extensive area for a further improvement, mainly gathering larger public audience for a feedback and hobby beekeepers for whom entry price of commercial weighting systems might be too high.

Figure 2.1: Mechanical scale

2.1 Goals and requirements

Clear definition of the requirements is necessary to set the scope of the project.

Accuracy of the weighting, declared by the beekeeper to make the system useful is $\pm 200g$ to be able to keep track of the honey supplies and occasional swarming.

Goal of the project is to make an open-source beehive monitoring system with all the necessary electronics for the real-time weight monitoring, as well as historical graphs – giving beekeeper precise information about the beehives. Location of the beehives may vary, depending on the needs, e.g. agriculture (pollinating trees or fields), commercial honey harvesting or hobby beekeeping. Because beekeeping may be operated far from conventional energy sources, off-grid solution is necessary to keep the whole electronical system running without the need for regular maintenance (battery replacement or recharge).

System is designed with a cooperation of a local hobby beekeeper to fit the real-world requirements. Selected beehives will be monitored for the purpose of testing and later for production deployment.

Reason for reinventing this system is based on an intention of making cheaper and open solution which scales more easily than available commercial alternatives. Reference beekeeper uses mechanical scale under one beehive and periodically notes the weight by hand into paper notebook. Weight of all other beehives is approximated by the selected one (see Fig. 2.1).

System shall use off-the-self components to make it possible for an experienced tinkerer to manufacture PCBs, solder necessary components and get the system running.

Summary of the goals:

- Accuracy not required as in commercial scales
- Wireless data transmission to server
- Off-grid power source
- Easily scalable
- Open source – Open hardware
- Deployment of the system for a local hobby beekeeper

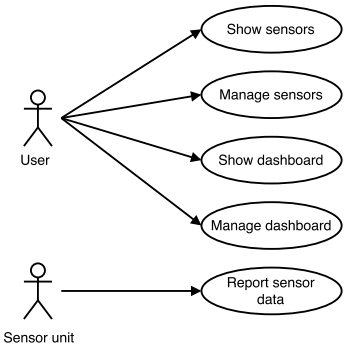

Figure 2.2: Use case diagram denoting required functionality of a web interface.


Figure 2.3: Left: Beehive placed on a weighting system of company ALYA s.r.o. [2]. Beehive placed on a weighting system Beewise [5]. (right)

2.2 Existing solutions

There are several existing solutions available.

This chapter summarizes available commercial systems with similar functionality to the device set created as a result of this thesis. Provided functionality may vary.

- ALYA s.r.o
  Company ALYA s.r.o. offers product with name VILKO (Fig. 2.3), which contains electronics necessary for measuring beehive weight. According to manufacturer, weight is placed only under reference beehive which is located near the rest. Device is transmitting data through GSM network. Whole set includes weight and costs 660 EUR [1, p. 1].

- Beewise
  Company Beewise® offers similar system. This system is using reporting by SMS messages and can be also powered by solar panel. Interesting feature is availability of theft protection – stolen unit cannot be used because PIN code is hardcoded in device[5].

- Arnia
  Company Arnia offers wireless weighting system, where under each beehive, separate wireless module is transmitting data to central module, which aggregates data and sends them through 2G GPRS network. Units are battery powered without solar panels. Data are sent to a shared cloud service. [3].

- Beespy.cz Relatively recent product of the company CUTTER Systems spol. s r.o. called BeeSpy[1]. System is transmitting data through GSM network, optionally through WiFi. Interesting feature of this system is the counter of bees entering/leaving the beehive, as well as humidity/temperature sensor inside the beehive.

2.3 Weighting principles

The most common strain measurement sensor is a strain gauge. The strain gauge transduction principle is based on the relationship between the change in length and its resulting change in the resistance of a conductor [7, p. 373].

A Strain gauge is a sensor whose resistance varies with applied force and in commonly used for load, weight, and force detection. It is a foil resistor, whose line resistance is proportional to the length and inversely to the area of the cross section. It consists of a small diameter wire, that is attached to a backing material (usually made of plastic). The wire is looped back and forth several times to create an effectively longer wire. The longer the wire, the larger the resistance, and the larger the change in resistance. However, the change of the resistance is very small, so we need a good amplifier and measurement principle to detect such small differences. It is one of the most important tools of the electrical measurement technique applied to the measurement of mechanical quantities.

2.3.1 Wheatstone bridge

The Wheatstone Bridge circuit is series-parallel arrangements of resistors connected between a voltage supply terminal and the ground producing zero voltage difference, when the two parallel resistor legs are balanced. It has two input terminals and two output terminals consisting of four resistors configured in a diamond. You can see the typical drawing of Wheatstone bridge in the picture below. It is suitable for measuring small changes in resistance, making it good for strain gauges. [8]

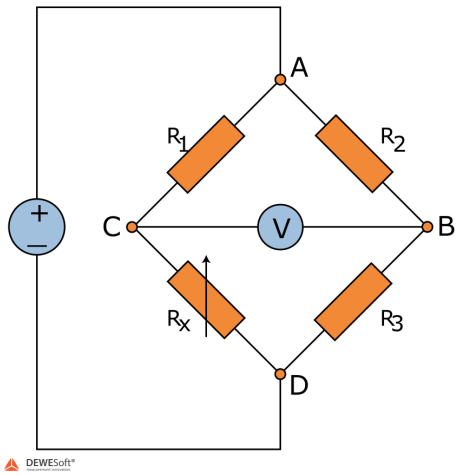

Figure 2.4: Wheatstone bridge [8].

[1] http://beespy.cz

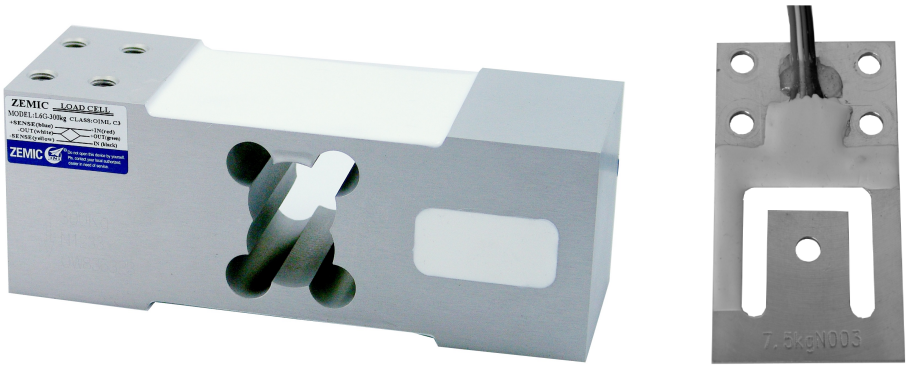

Figure 2.5: Single Point Load Cell (left), Planar Beam Load Cell (right), Source: [22]

2.3.2 Load cells

Load cell is an weighting instrument made from a metal, usually aluminium with strain gauges applied on selected surfaces embedded in some form of a gel or silica. There are several gauges to form a Wheatstone Bridge.

Load cells can be purchased from several manufacturers through resellers. Load cell itself comes without any other mechanical support and which has to be built by the end-user.

There are several types of load cells, as denoted in [22], such as:

- Single Point
- Planar Beam
- Shear Beam
- S-Type

For the purpose of measuring beehives, planar beam load cells can be used on all four corners, but comes with more complex cabling. Simplest solution is to use single point load cell per beehive.

Chapter 3

Used technologies

This chapter discusses necessary topics as a part of background for the forthcoming chapter.

3.1 Single-board computers

Single-board computers are credit-card sized computers with an integrated CPU and memory in form of a PCB without a case. Unlike full featured PC even in a small factor (MiniITX), these boards have variety of peripherals – header with GPIO pins, PWM outputs, camera connector, CAN controller, analog-digital converters. Power demands of these boards is in a range of a few watts, therefore they can be powered from a regular 5V USB port. Optionally these computers also contain USB ports, Ethernet connectivity, audio and HDMI output.

Operating system is located on a SD-Card but for some boards like Odroid CU2/XU4 and many other, eMMC memory can be used instead for higher throughput and reliability. Most of these board are based on an ARM™ architecture, various Linux distributions are therefore available, generally speaking, the more popular board – the better support and more options. Linux distributions are available in different variants – with a GUI or headless, depending on a particular use-case and the size of an SD-Card.

Pin headers available on the board make it possible to expand the functionality with an additional PCBs mounted on top – called shields. Shields are available from variety of manufacturers, providing additional functionality such as Stepper motor drivers[1], NFC card readers, displays, etc.

Some of the use-cases for these small-factor computers are: HTPC – Home Theater Personal Computer, sensor nodes, basic home automation, plant watering systems, weather stations, robotics and many other.

Care should be taken in selection of the SD card, as the consumer grade card may wear out quickly and fail without previous warning. Industrial grade SD cards shall be used for systems running 24/7.

3.1.1 Beaglebone Black / Beaglebone Black Industrial

Beaglebone Black[2] is a board with an AM335x 1GHz ARM Cortex-A8 processor and 512MB RAM. The board has an integrated 4GB eMMC flash non-volatile memory, replacing the

[1] https://www.adafruit.com/product/2348
[2] https://beagleboard.org/black

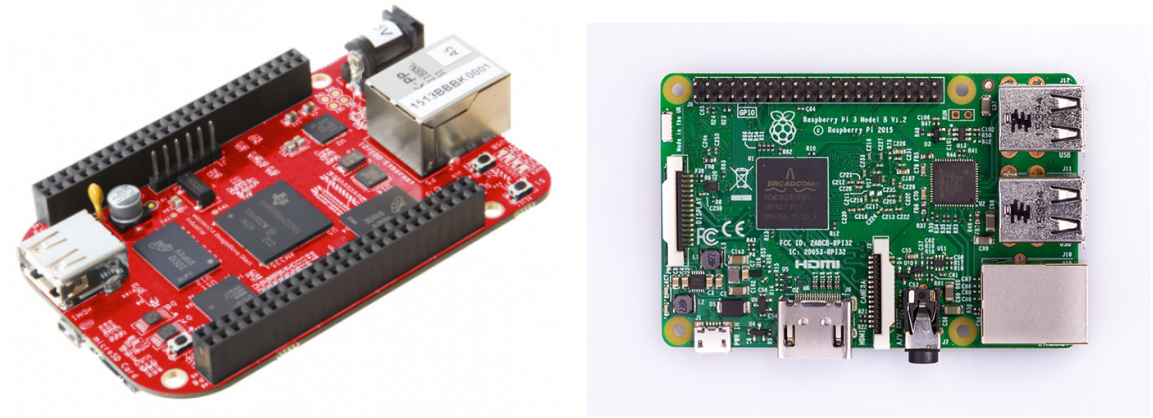

Figure 3.1: Left: Beaglebone Black Industrial, Right: Raspberry Pi 3

need for an external SD card. This might be useful in use-cases, where high reliability needs to be ensured. There is also a variant in an industrial grade, having an extended operating temperature range in -40C up to +85C – the only trade-off is a higher price.

Advantage of this board is that MiniUSB connector used for power also acts as a USB OTG device, providing Ethernet adapter for the connected computer with a dedicated network subnet and a DHCP server, therefore user can connect through SSH without the need of anything other than a plain USB cable. Board has also regular 5V barrel jack connector, used in cases with higher power demand of the connected devices (mouse, keyboard, external audio sound card)[3].

3.1.2 Raspberry Pi

Raspberry Pi, first released in February 2012 – Raspberry Pi 1A/1B, single-board computer featuring 512MB of RAM and single core CPU. Later in July 2014 superseded by 1B+ and in February 2015 by 2B (900MHz quad-core CPU, 1GB RAM)[4].

Currently last revision is Raspberry Pi 3, with 64-bit quad core ARMv7, with an integrated 802.11n WiFi interface and Bluetooth Low Energy. [5]

Main advantage is its cheap price compared to the competition, as well as large community. There are many Linux distributions available – some of them designed just for one use-case.

3.2 Wireless technologies - comparison

Monitored beehives may be located in the garden or a few kilometers away from the city, therefore wireless technology used for the project must be properly selected to keep the system connected and to accommodate various needs of a beekeeper. There are several wireless technologies available, ranging from a magnitude of meters to kilometers to an aggregation point (wireless AP, GSM gateway).

[3] https://www.element14.com/community/docs/DOC-78671/l/element14-beaglebone-black-industrial-4g

[4] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[5] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

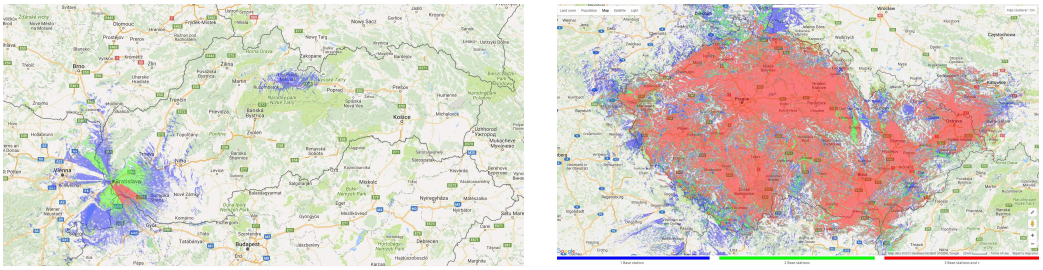

Figure 3.2: SIGFOX network coverage of the company SimpleCell in Slovakia (left), Czech Republic (right).

3.2.1 SIGFOX

SIGFOX is a proprietary Machine-To-Machine cellular network operating in Ultra Narrow Band frequencies around 868MHZ used for IoT applications like remote sensing, electricity meters. Network is based on one-hop star topology and consists of transmitting devices, base station and SIGFOX back-end.[12] By the design of the network, there is a set limit of messages a device can transmit to 140 messages per day.

Company SimpleCell [6] is building SIGFOX networks in Czech Republic (86% area coverage) and Slovak Republic (in testing phase, Fig. 3.2).

Advantage of this technology are cheap, low-power modems without the need for SIM cards. Provider defines monthly price up-to 26 czk (paid annually) depending on the amount of messages per day.[14]

3.2.2 GSM

GSM (Global System for Mobile communication) is a digital mobile telephony system that is widely used in Europe and other parts of the world. GSM uses a variation of time division multiple access (TDMA) and is the most widely used of the three digital wireless telephony technologies (TDMA, GSM, and CDMA). GSM digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its own time slot. It operates at either the 900 MHz or 1800 MHz frequency band[13].

On top of the regular telephony services provided by the GSM, this cellular network provides IP data connection (3G/GPRS/LTE). Advantage is almost complete country-wide coverage. Disadvantage is the need for an expensive modem as well as data plan with SIM card paid on monthly basis. In case of USB modem, most energy out of all presented wireless technologies is required for its operation. This may be partially solved by an integrated modem such as SIM808/SIM900 interfaced through serial port – this option is commonly used for alarm systems or car tracking devices.

3.2.3 WiFi

Wi-Fi is a technology that allows many electronic devices to exchange data or connect to the internet wirelessly using radio waves. The Wi-Fi Alliance defines Wi-Fi devices as any „Wireless Local Area Network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards“. The key advantage of IEEE

[6] http://simplecell.eu

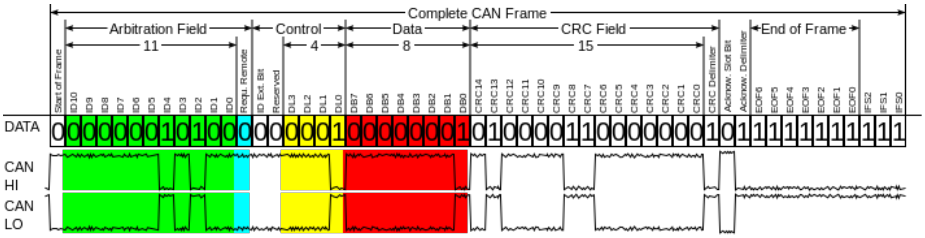

Figure 3.3: CAN bus frame.

802.11 devices is that they allow less expensive deployment of Local Area Networks (LANs). For places where running cables to every device is not practical, such as outdoor areas and airports, they can host wireless LANs. Products from every brand name can inter-operate at a basic level of service thanks to products being designated as „Wi-Fi Certified“ by the Wi-Fi Alliance[19]. Operating range of Wi-Fi is in range of 2.4GHz and 5.8GHz.

3.3 CAN Bus

CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for high immunity to electrical interference and the ability to self-diagnose and repair data errors. These features have led to CAN's popularity in a variety of industries including building automation, medical, and manufacturing.

The CAN communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP). CSMA means that each node on the bus must wait for a prescribed period of inactivity before attempting to send a message. CD+AMP means that collisions are resolved through a bit-wise arbitration, based on a preprogrammed priority of each message. The higher priority identifier always wins bus access. That is, the last logic-high in the identifier keeps on transmitting because it has the highest priority. Since every node on a bus takes part in writing every bit „as it is being written“, an arbitration node knows if it placed the logic-high bit on the bus.[15]

CAN bus is one of the five protocols used in the on-board diagnostics OBD-II vehicle diagnostics standard. The OBD-II standard has been mandatory for all cars and light trucks sold in the United States since 1996, and the EOBD standard has been mandatory for all petrol vehicles sold in the European Union since 2001 and all diesel vehicles since 2004.

3.4 Embedded systems

Embedded system is a computer system with small footprint, usually not programmed by an end-user. Processing power of such system is dimensioned for specific application. Most common embedded systems can be found in home appliances (washing machines, microwave oven), as well as cars (Engine Control Unit), switching supplies, remote controls and many other.

Small footprint of such systems and pressure for the cheapest possible solution forces developers to choose, in case of microcontrollers (MCU), the one with smallest computing power and memory that does the job. Such trade-off requires the program to be written in low-level language interacting with internal registers rather than in fully-featured language with external libraries.

Programming of the microcontrollers is usually done in C or Assembly language. Interaction with input/output peripherals is done through memory mapped registers of given peripherals. Each vendor of microcontroller may define own set of peripherals as well as used instruction set of the program itself. Common practice is to use standardized architecture of the core, such as ARM^{TM}, making it possible to reuse software tooling (compiler, debugger...) on all computer platforms, as well as easier migration between MCU vendors.

Vendors of MCUs provide application notes (implementation to reference applications), including examples, as well as Hardware Abstraction Layer (HAL) library, which contains unified interface usually written in C with peripherals across the family of MCUs of a given vendor.

After start, MCU expects program to be located in an internal Flash memory. Such program is flashed during the assembly of the device of which the MCU is part of (e.g. remote control). Program can be either very simple and use one infinite loop in `main()` function, or use operating system, such as FreeRTOS, giving developer ability to define tasks with periodic wakeup, queues for message passing as well as locking mechanisms for critical sections.

3.4.1 STM32 microcontroller

Company STMicroelectronics produces wide spectrum of integrated circuits as well as microcontrollers. For this thesis, STM32 32 bit MCU is selected and is further described.

STM32F0 family of MCU's consists of an ARM^{TM} Cortex-M0 core. MCU is not designed to be low-power. For the this project, STM32L0 may be more suitable, but comes without CAN controller.

3.4.2 Haskell

Haskell is a standardized, general-purpose purely functional programming language, with non-strict semantics and strong static typing. It is named after logician Haskell Curry.

Haskell features a type system with type inference and lazy evaluation. Its main implementation is the Glasgow Haskell Compiler.

3.4.3 Ivory

Ivory is a language that enforces memory safety and avoids most undefined behaviors while providing low-level control of memory-manipulation.[7] Ivory's implementation, however, is unique compared to safe C languages: Ivory is implemented as an embedded domain-specific language (EDSL) within Haskell. In addition to the benefits of rapid language development, this gives Ivory a powerful templating system–the language Haskell–allowing low-level programs to be written in a high-level style, despite some of the underlying languages restrictions[9].

[7] http://ivorylang.org/ivory-introduction.html

```
forever $ do
  datValue0 <- pinRead dat0
  datValue1 <- pinRead dat1
  when (datValue0 ==? false .&& datValue1 ==? false) breakOut

arrayMap $ \(i :: Ix 24) -> do
  pinSet clk

  val0 <- (pinRead dat0)
  val1 <- (pinRead dat1)
  aux_weight0 <- deref current_weight0
  aux_weight1 <- deref current_weight1
  store current_weight0 $ aux_weight0  .|
            ((safeCast val0) `iShiftL` (24 - signCast (fromIx i)))
  store current_weight1 $ aux_weight1  .|
            ((safeCast val1) `iShiftL` (24 - signCast (fromIx i)))

  pinClear clk
```

Figure 3.4: Snippet of Ivory language interacting with GPIO pins performing data reading from two HX711 weight converters.

3.4.4 Tower and port to STM32F0

Tower is a concurrency framework for the Ivory language. Tower composes Ivory programs into monitors which communicate with synchronous typed channels.

Originally for which was Ivory/Tower developed is a research project SMACCMPILOT[8], therefore memory maps, registers and peripherals are for STM32F4.

For the purpose of the thesis, port of the package `ivory-tower-stm32` for STM32F0 was developed[9]. Port consists mainly of modified register maps, peripherals, clock configs, allocated stack size by FreeRTOS (due to limited memory size) and other.

3.5 Tor

Tor (The Onion Router) is a network of nodes – servers running Tor software located worldwide forming circuit network. The client enables users to use SOCKS5 proxy to open an anonymous TCP connections through the circuit network and through an exit-node, while the complete path to the initiating user remains unknown, or at least hard to trace.

3.5.1 Hidden services

One of the Tor's functionality, other than anonymization network is its feature for hosting Hidden Service. Hidden Service consists of a generated onion address. Client's daemon can

[8] http://smaccmpilot.org/

[9] https://git.48.io/ivory-tower-stm32/c/cce669b7493452e8e7c00c2f2218e037e8dbb3b2?branch=f0proto

then forward received TCP connections to any service as needed. Address resembles base32 string (e.g. tz3fxw2oyklnnge6.onion) and is generated by the server. The address is a hash of of the public key, while private key remains only on the server. Address also functions as a validation of the origin.

Such feature of the Tor network can be used for remote management of the systems behind firewall or without public IP.

3.6 PHP

PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers. PHP, originally derived from Personal Home Page Tools, now stands for PHP: Hypertext Preprocessor, which the PHP FAQ describes as a „recursive acronym."

PHP is an alternative to Microsoft's Active Server Page (ASP) technology. As with ASP, the PHP script is embedded within a Web page along with its HTML. Before the page is sent to a user that has requested it, the Web server calls PHP to interpret and perform the operations called for in the PHP script.

An HTML page that includes a PHP script is typically given a file name suffix of „.php" or „.phtml". Like ASP, PHP can be thought of as „dynamic HTML pages," since content will vary based on the results of interpreting the script[18].

PHP interpreter can be installed on all standard Linux distributions through integrated package managers, such as `apt-get` or `yum`.

3.7 Laravel web framework

Laravel is a free, open-source PHP web framework. Framework follows MVC (Model-View-Controller) architecture, decoupling presentation layer, logic layer and database, although these may overlay. Major advantage of framework such as this are generators, e.g. CRUD (Create-Read-Update-Delete) scaffold generator[10].

3.8 Highcharts and Javascript

Javascript is a client-side scripting language that makes it possible to develop dynamic and interactive web applications. One such use-case for this functionality are charts.

HighSoft`https://shop.highsoft.com/` is a company developing charting software written in Javascript – such as HighStock, HighCharts and other.

HighStock library enables user to easily visualize time-series. Data can be either hard-coded in Javascript code attached with chart or loaded externally through AJAX request. Software is provided under paid license for commercial use-cases. Result of this thesis is a non-commercial system and the whole web interface is released as an open-source intended only for personal use, therefore may be used for free under Creative Commons (CC) Attribution-NonCommercial licence. HighCharts is a library similar to HighStock without extensive options for manipulation with time (grouping, navigation bar etc.).

[10] `https://github.com/appzcoder/crud-generator`

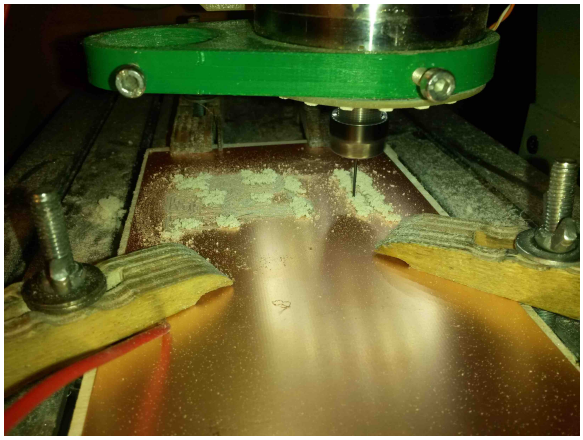

Figure 3.5: Drilling holes of prototype bus-unit board with 1mm drill on a CNC machine.

3.9 PCB design and manufacturing

Design of the boards used during the development and some for deployment involved drawing schema and layout of the board. Software used for the design is EAGLE 8.0.2[11]. Although EAGLE is commercial software, there is an option for a student's license without limitation of board size/sheet count.

Several iterations of printed circuit boards were made in hackerspace base48[12] in Brno in order to test and validate designed devices. Manufacturing method of prototype boards is milling the copper clad (FR4), which is glass-reinforced epoxy sheet covered with thin layer of copper.

Milling is done on modified CNC engraving machine (Fig. 3.5), running LinuxCNC software with support of probing the surface of the board to ensure correct milling depth across the whole surface. Tool used for engraving is 0.2mm 45 degree milling tool, drilling is done with 1mm drill/milling cutter and last step involves cutting the board out of the copper clad with 3.175mm tool.

On the finished board, layer of resin (FLUX 10 spray) is applied for easier soldering and protection of the board. Fast turnout of prototype boards made it possible to test the design before final boards were sent for manufacturing.

[11] http://cadsoft.io
[12] http://base48.cz

Chapter 4

Proposed solution and Implementation

Proposed solution of a beehive weighting system consists of a battery, charged by a solar panel, Raspberry Pi 3 with a CAN shield and sensor units – boards with a low-cost micro-controller with attached weight converters. Each board with an MCU has two instrumentation amplifiers, minimizing per-hive costs and to simplify cable management. Power management board provides charging and battery protection functionality. Bus-units are placed in a small standardized boxes. Raspberry Pi with attached CAN shield and Power management board is in one box. Bus-units are attached on a 3D printed mount with screws for better attachment in box.

Whole project is published as an open-hardware under MIT license on https://github.com/lucansky/hexamon – including all the source files, PCB schematics.

Measuring system consists of these parts, which are further described in detail:

- Sensor unit (synonymously bus unit) on a CAN bus regularly transmitting measured data with attached sensors (two weight converters in this case)
- Master unit (Raspberry Pi) with CAN shield, listening on a bus, processing received frames, repacking and sending them through IP network interface (WiFi or 3G modem)
- Power management unit with mounted solar battery charger, under-voltage protection
- 12V 7.2Ah Battery
- 20W 18V mono-crystal solar panel
- Linux server with web application and SQL database

All bus units are daisy-chained with telephone cable – with master unit at one end. For the purpose of this project, 4 middle wires in telephone cable (6 or 4 wire) are defined to be following signals:

- Black - Ground
- Red - 5V
- Yellow - CAN Low

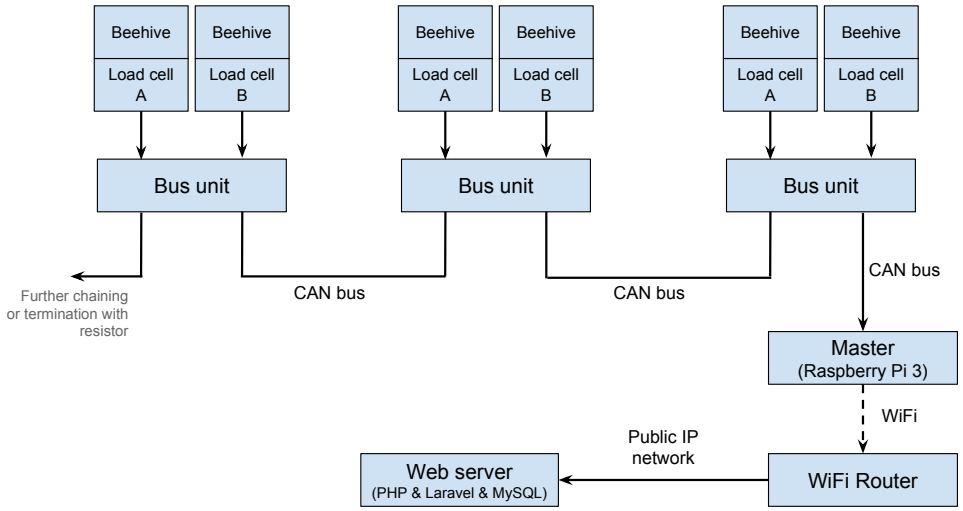

Figure 4.1: General schema of the system

- Green - CAN High

Printed circuit boards (PCBs) made for this project are designed to be single-side, suitable for etching or milling on a low-cost CNC machine, with as few vias and traces on the other side of the board as possible.

4.1 Bus unit - weight converter

Bus unit is a PCB with an MCU – STM32F0, CAN transceiver (SN65HVD230/SN65HVD233), two HX711 ADC converters, voltage regulator (LM1117 3.3V or package compatible), debug LED and necessary connectors (2x RJ12, 2x 4 pin 2.54mm screw terminals). Two RJ12 sockets are used to chain bus units through telephone cable. Two screw terminals are used to connect load cells, following color schema:

- E+ - Red
- E- - Black
- A+ - White
- A- - Green

In the case load cell has 6 wires (+ grounding), these wires are left floating.

Bus unit with connected load cells performs periodic measurements of weight (10 second interval), packing measured data into CAN frame structure further sent to CAN bus. Transmission is done regardless the presence of the master unit on bus.

4.1.1 PCB

Designed PCB is a double-sided board with single trace on bottom side. Six board were manufactured with a pool service in company Gatema s.r.o. (Fig. 4.3, 4.4). PCB can

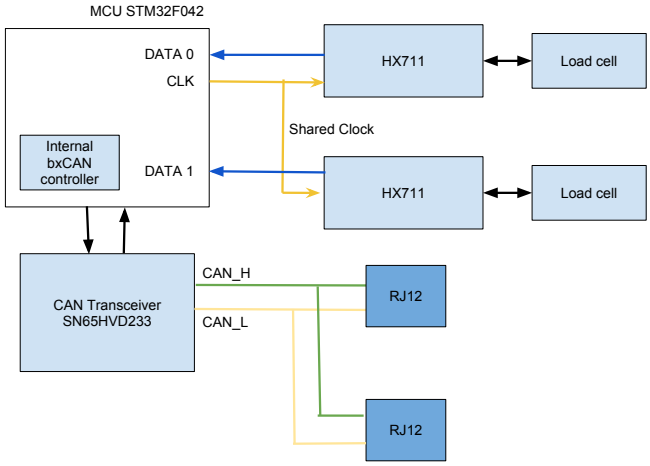

Figure 4.2: Busunit overview schema

be manufactured also by etching or milling. 4 prototype board were created during development phase. Component with the smallest pitch of the pins is the MCU in a package TSSOP20 – milling traces for such packages require 0.2mm tool.

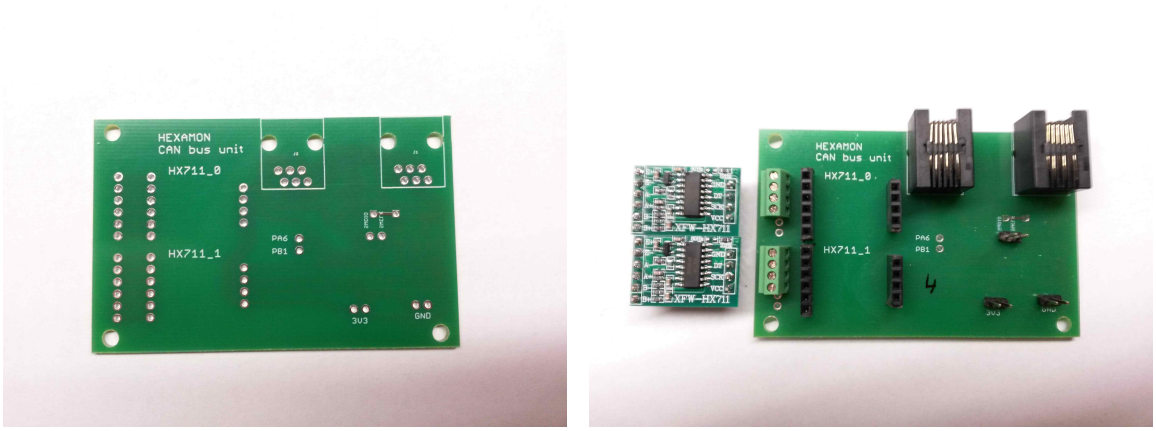

Figure 4.3: Bus-unit unsoldered (left), bus-unit with all components soldered (right)

4.1.2 Microcontroller

Microcontroller selected for the processing is an ST-Micro STM32F042K6 with Cortex-M0 in package TSSOP20, featuring 32kB of Flash, 6kB of SRAM. Selection is based on the presence of the CAN controller, integrated 48MHz oscillator and cheap price.

Flashing (programming) of the firmware to the MCU is done through ST-Link v2 USB programmer. ST-Link programming device is obtainable from eBay[1] for price up to US $3. Pin-headers on the board provide 2 pins for the programming (SWDIO, SWCLK) and two pins for the power (3.3V) and ground (GND). If the board is connected to powered CAN bus, power pin (3.3V) is not necessary.

Flashing is done through ARM GDB from an official `arm-none-eabi-gdb` package.

[1] eBay search keywords: ST-Link V2 USB

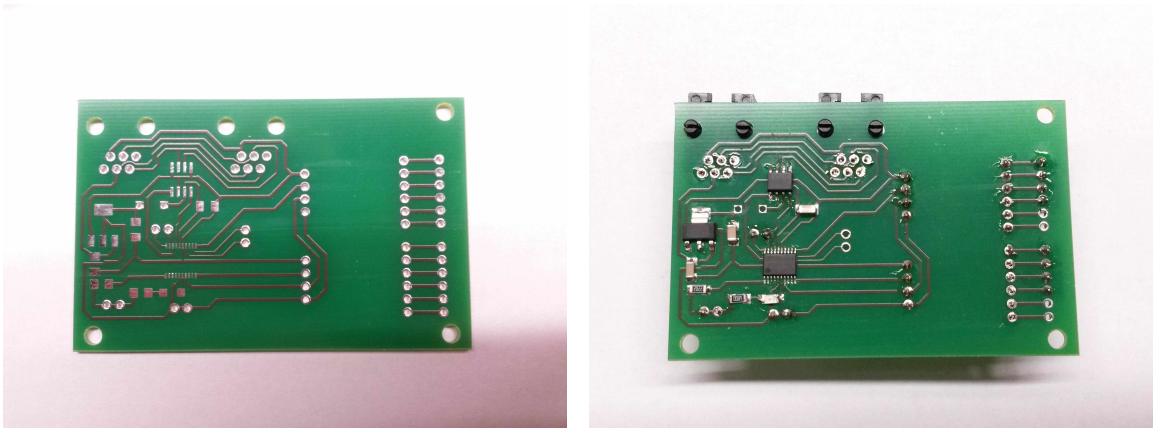

Figure 4.4: Bus-unit unsoldered (left), bus-unit with all components soldered (right)

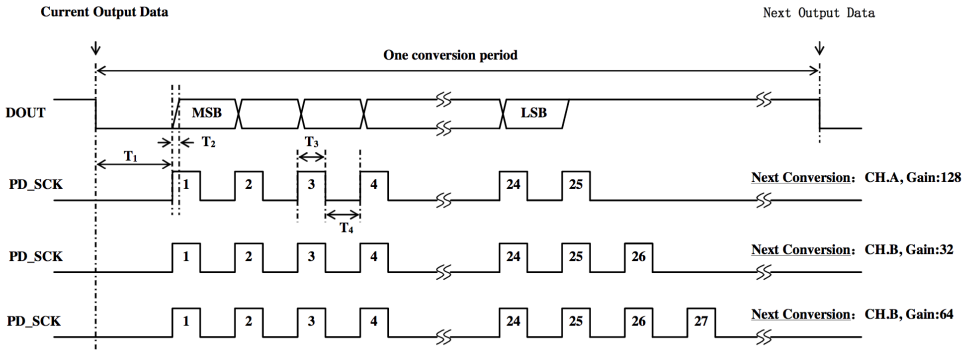

Figure 4.5: Timing and control of HX711. Source: [4]

4.1.3 Weight acquisition – HX711

HX711 is a differential ADC combined with an amplifier with programmable gain (128x, 64x or 32x). IC can be purchased separately or integrated on a „breakout“ board with additional necessary components (DC coupling capacitors, resistors, sampling speed selector).

HX711 is interfaced through two pins by bit-shifting acquired value, similar to SPI with modification – there is no data signal from MCU. Gain on the programmable amplifier is set with count of the pulses after 24 bits of data are read (Fig. 4.5).

4.1.4 Firmware

Firmware is written in Ivory language on top of Tower framework based on scaffolding project `ivory-tower-helloworld`[2]. `DualHX711` tower periodically triggers data acquisition from two HX711 converters. Output from the `DualHX711` are two typed output channels – emitting sensor samples. Data acquisition from the HX711 is implemented by bit-banging CLK signal as defined in datasheet[4] and stuffing low 24 bits of a 32-bit integer bit-by-bit as they come after each `CLK` pulse (example of the implementation in Fig. 3.4). One more `CLK` pulse is sent to converters to set gain for next transmission to 128x.

[2] https://git.48.io/ivory-tower-helloworld

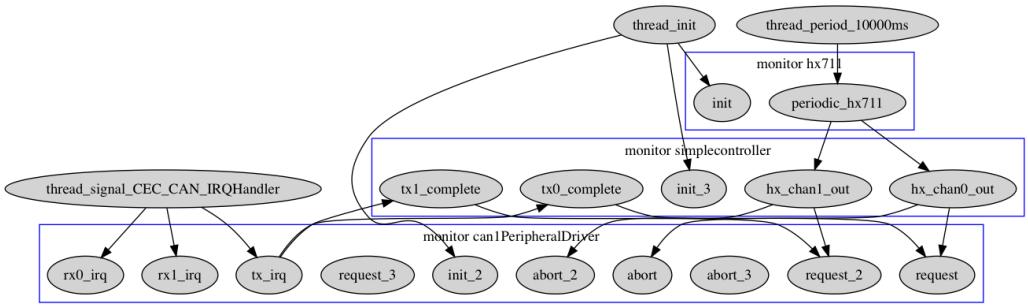

Figure 4.6: Controller diagram. Arrows denote typed channels. Generated by Ivory Tower.

Whole data-flow through the channels can be seen in Fig. 4.6.

Functionality for both converters is included in one tower due to shared clock pin. After the measurement is done for both converters, sensor sample is emitted to the respective channel for each converter.

Main application tower, gluing together `App` tower a `DualHX711` tower is managing conversion from sensor sample to CAN message format and filling bxCAN[17, p. 814] transmit mailboxes. Scheduling is not handled by the software, due to limited size of the SRAM on the selected MCU, but instead, three output mailboxes in integrated CAN controller are used – keeping the arbitration in hardware. In this case, two mailboxes are used.

Source code of the firmware can be found on `https://github.com/lucansky/hexamon-firmware`. Compilation process of the firmware consists of building the program generating whole project in C along with Makefile and then compilation with `arm-none-eabi-gcc`. Generated source files and compiled firmware can be found in `/build/hexamon` folder of the firmware project after issuing `make hexamon-run` command.

Resulting ELF binary file ready for flashing is located on path `/build/hexamon/image`.

4.1.5 CAN message format

CAN frames sent from the MCU have a predefined format. One bus-unit can accommodate multiple sensors therefore frames have to contain identification of the sensor, type of the transmitted data as well as the data itself. For this purpose, simple 6 byte frame format was established, and is as follows:

- 8 bits - sensor index – arbitrary offset of the sensor
- 8 bits - type of the sensor – predefined list of constant values with given meaning (weight, temperature) – current implementation uses only 0: amplified differential voltage weight acquired from the HX711 converter
- 32 bits - data, MSB first

Extended identifier (11+18bits) is a sequence number with fixed offset per each sensor unit deployed, hard-coded in firmware during build-time located in `/src/Hexamon/Platforms.hs` at the end of file.

```
pi@raspberrypi:~ $ candump can0
  can0  00043210   [6]  00 00 00 01 38 78
  can0  00043210   [6]  01 00 00 02 2B 6A
  can0  00043210   [6]  00 00 00 01 38 44
  can0  00043210   [6]  01 00 00 02 2B E2
  can0  00043210   [6]  00 00 00 01 38 7C
  can0  00043210   [6]  01 00 00 02 2B E6
  can0  00043210   [6]  00 00 00 01 38 A6
```

Figure 4.7: CAN frames as seen by a Raspberry Pi with utility `candump`. One bus unit is connected on the CAN bus.

4.2 Master – RaspberryPi with CAN shield

Designated signaling rate of the CAN bus is 125kbps. Based on [16, p. 5], theoretical maximum length of the bus at selected speed is 200 meters. 5V power is injected to the bus, therefore maximum length may be limited by the voltage drop on the selected cable.

Beehives in chain share one Master unit – Raspberry Pi 3 with a CAN shield. Selection of the Raspberry Pi is based on the cheap price and availability in local stores. Drawback is that there is no integrated CAN controller, unlike in Beaglebone Black.

To expand the functionality of a Raspberry Pi for CAN bus, custom CAN shield (Fig. 4.8) was developed. CAN shield consists of an integrated SPI CAN controller – Microchip MCP2515, CAN transceiver – SN65HVD233, as well as external power input and connectors for the bus. To use available space on the board, DB9 connector, ESD protection and split-termination[16, p. 8] is added.

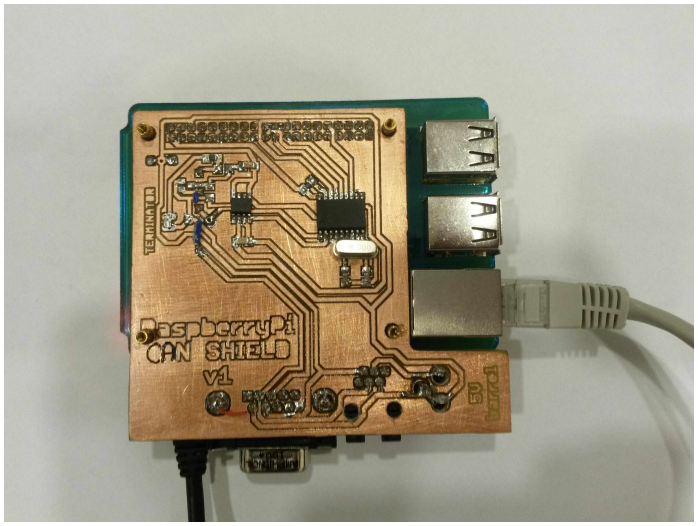

Figure 4.8: Raspberry Pi 3 in a plastic case with mounted CAN shield. Left IC – SN65HVD23 CAN Transceiver, Right IC – MCP2515 CAN Controller

Wireless communication with the Internet is either through a USB GSM modem or WiFi. In case of hobby beekeeping in the garden, integrated WiFi module in a Raspberry Pi 3 can be used for connection to nearby WiFi AP. For the case of distant measurement, 3G/GSM modem with a SIM card can used, but comes with the burden of paying for it monthly.

Connection of the CAN controller (MCP2515) with the 40 pin header of the Raspberry Pi, as well as setup instructions for the SPI overlay is based off a tutorial found on an official Raspberry Pi forum[21].

4.2.1 Operating system

Raspberry Pi is running Linux, for this use-case, Raspbian[3] Jessie Lite distribution is chosen for it's similarity with Debian. After booting, system has a `can0` network interface initialized, receiving CAN frames off the bus.

CAN controller interfaced through SPI is managed by a kernel module `mcp251x`. After interface setup, CAN interface `can0` shows up in list of network interfaces. Command `ifconfig` can used to verify functionality (dropped frames) as well as status of the interface.

4.2.2 Software

Program `hexamond` running on the system is receiving frames in infinite loop from the CAN interface `can0` and repacks them to HTTP protocol and sends them to the web-server for storage. Current implementation drops measured data in case of unavailability of the internet connection. Program is written in Haskell. Implementation can be found in repository `https://github.com/lucansky/hexamond`. To ensure that the program running on the master unit is as simple as possible, CAN frames are not given semantics for a better future compatibility in case new sensors are introduced.

Due to complexity of the subsystems needed to setup whole Haskell build system as well as CAN overlay, complete image of the SD-card is provided on attached DVD.

4.2.3 Remote administration

External administration of the Master for the case of troubleshooting is assured by the Tor hidden service running on the background, mapping port 22 to SSH. Hostname of the hidden service can be found in file `/var/lib/tor/hidden_service/hostname`.

4.3 Power management board

Raspberry Pi 3 consumes on average 2.05W (0.41A@5V) with CAN shield and WiFi connected (bus units are powered through bypass on the CAN shield, therefore not measured). For an embedded system, such power usage is quite high, therefore stable source of energy is necessary. To solve this problem, 20W solar panel is used to charge 7.2Ah 12V battery. This makes it possible to run indefinitely until battery wears-out. Few days without sun may make the system unavailable, but selected solar panel provides 10% of energy even on a cloudy day.

Solar battery charger used on the power management board is a BQ24650 controller on a prefabricated board with necessary support components **(F)**, purchasable through eBay[4]. Maximum Power Point Tracking voltage, as defined by the solar panel manufacturer as well as battery voltage is settable by the user.

[3] https://www.raspberrypi.org/downloads/raspbian/

[4] Search keywords: BQ24650 5A MPPT Solar Panel Battery Charging Board

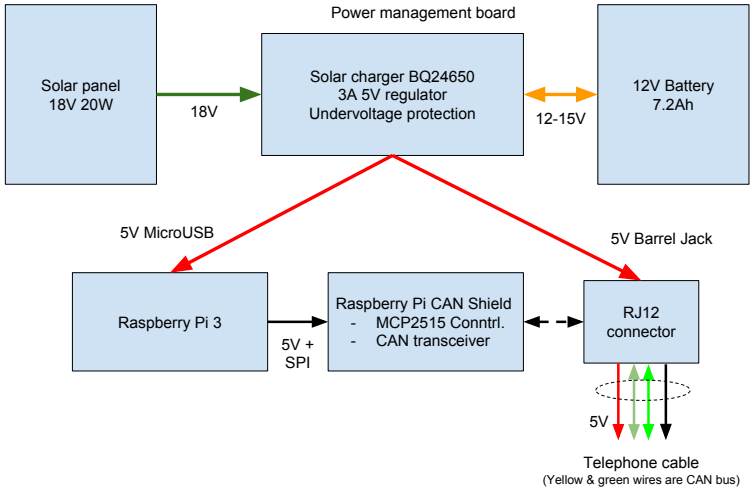

Figure 4.9: Schema of the power distribution for the whole system.

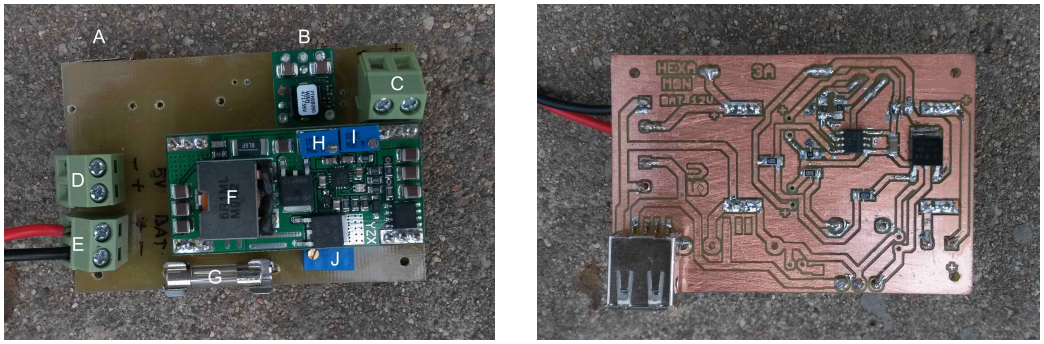

Figure 4.10: Assembled power management board. Top (left), bottom side (right).

Shutdown of the system in case of battery under-voltage is ensured by a low-power voltage supervisory circuit MAX8212 enabling P-MOSFET transistor. Threshold is set by the trimmer mounted on board **(J)**.

Battery voltage protected from an under-voltage is ran through 5V voltage regulator PTH08080 **(B)** for supplying Raspberry Pi and power for the units connected on the CAN bus. Board also contains a fuse holder with a 3A **(G)** fuse for a battery protection in case of accidental or intended cable damage/cut.

During the design, decision was made to use 5V for a bus voltage instead of raw 12V from the battery due to 3,3V linear voltage regulators on the units, therefore unnecessary heat dissipation is avoided. Bus units can withstand any voltage in range from 5V to 12V as defined in the datasheet of the `LM1117` regulator[20] – up-to 15V, giving an option for future compatibility.

USB female connector **(A)** is used for powering the Raspberry Pi. Board contains three 5.08mm screw terminals, and are used for following functionality:

- 5V output for the bus **(D)** – connects to barrel jack on the CAN shield

- 12V battery input **(E)**
- 18V solar panel input **(C)**

4.3.1 Solar panel and solar charger

To ensure correct battery charging, breakout board with BQ24650 solar battery charger is used. Two trimmers on board define the voltage at which current shall be drawn from the solar panel **(F)** and voltage of fully charged battery **(C)**.

Selected solar panel (GWL/Sunny Poly 20 Wp 36 cells) can be purchased from local distributors[5] for a reasonable price.

4.3.2 Voltage regulator

Current drawn from the Raspberry Pi 3 is 0.41A@5V and 45mA@5V for each bus-unit with weight converters. To make sure the system sustains spikes (e.g. system booting), switching regulator PTH08080W 2.25A is selected.

Output voltage 4.97V of the switching regulator is selected by the 360 ohm resistor between V_o `ADJUST` and `GND` pins.

4.3.3 Undervoltage protection

In order to protect the battery from under-voltage which may occur during rainy days or when there is snow on the solar panel, shutdown functionality is required. Voltage regulator PTH08080W has `INHIBIT` pin functioning as an enable pin. Decision was made to use MAX8212 voltage monitor with internal 1.5V reference voltage[11] and IRFR5305 P-Channel MOSFET as a switch turning-off output from the battery. Enable pin on the regulator is not used, as the voltage from the battery may be used in future. Shutdown circuit is based on [10].

4.3.4 Battery

Selected battery is Shimastu 12V 7.2Ah lead-acid battery, due to occasional freezing outdoor temperature, other types of batteries are not considered. Battery will be used in cycle mode, therefore charging voltage is set to 14,7V and lower limit is set to 10,8V.

4.4 Web interface

Interaction of the system with the user is through the web interface, providing basic means of analysis of data, such as charting from different sensors, sensor calibration, management of dashboards and real-time information from the sensors. All use-cases provided by the interface are shown in Fig. 2.2.

Back-end of the web is written in PHP framework Laravel 5. This selection is based on familiarity with Ruby On Rails, mainly ActiveRecord for an ORM interface. Web application uses Bootstrap for a basic template with HighStock Javascript library for charts. Data from sensors are parsed (converted from hexadecimal string) and stored in database as they arrive, user has then ability to choose which beehives are shown on which dashboard.

[5] https://www.i4wifi.cz/Solarni-panely-1/Solarni-panely/Solarni-panel-GWL-Sunny-Poly-20-Wp-36-cells-MPPT-18V.html

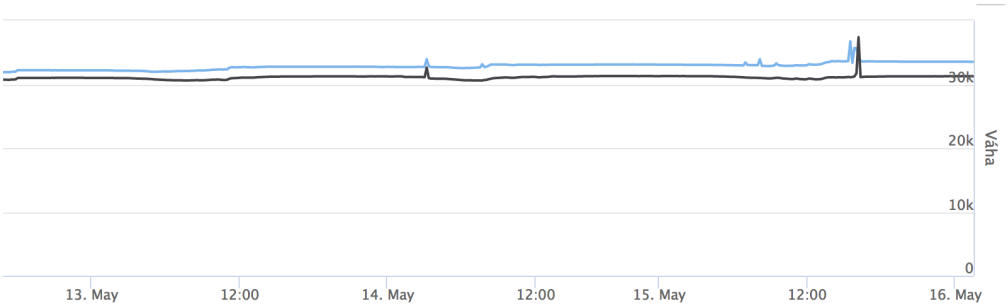

Figure 4.11: Dashboard chart with weight of two beehives under the test. Spikes in chart corellate with a thunderstorm in area.

After the master unit sends CAN frame in a HTTP request to an URL not protected by a password, CAN ID and sensor index is used to find particular sensor in the database. If sensor is found, based on the type of the sensor (current only weight is supported), measurement is logged, otherwise HTTP error 404 is raised.

User has the ability to define sensors shown in different charts in dashboard, as well as list all sensors in the system.

Each sensor provides two features necessary for weighting:

- Tare – setting zero
- Factor setting – known weight is placed on the scale, rise of the raw value is scaled to known value and setting the inclination rate

Data are stored in an SQL database MySQL. ER diagram (Fig. 4.12) describes tables necessary for the functionality of the information system.

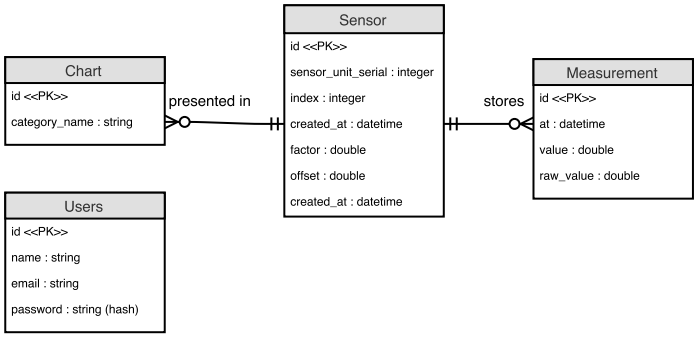

Figure 4.12: ER diagram of the database.

4.4.1 API

Currently only one API call for reporting measurements to the webserver is necessary, and is as follows:

- GET /report/raw/{CAN_ID}/{CAN_MESSAGE}

where:

- `CAN_ID` – Extended CAN ID as received in frame in decimal base (e.g. 1450001).
- `CAN_MESSAGE` – CAN message encoded in hexadecimal format (e.g. 000100AABBCC).

Example: `http://192.168.0.105:8000/sensors/report/raw/1450051/010000110f8e`

Chapter 5

Testing and deployment

This chapter discusses and describes testing of individual parts and deployment of the system on real beehives.

5.1 Deployment site

First production set of devices (4 bus-units, power management board, Raspberry Pi 3 CAN shield, along with battery and solar panel) created by this project is deployed on beehives of a local beekeeper, living in a village near Brno. Beehives are in the garden, approximately 30 meters from the house, therefore Internet connection can be easily acquired through WiFi (Fig. 5.1).

Figure 5.1: Housing of the beehives with mounted solar panel on the roof.

The beekeeper made his own welded metal platform. Metal ground bars are properly fixed, with mounts for eight Zemic L6G load cells. On top of the mounted load cell, welded cross is fixed with screws (Fig. 5.2).

Figure 5.2: Metal platform for eight beehives with Zemic L6G load cells.

5.2 Load cell calibration

In order to calibrate the load cells, object with known weight are required. Certified set of weights can be purchased[1], although prices may vary and for a hobby project or non-commercial grade weighting, such weights are not suitable in terms of price. Simpler solution is to use certified scales in the local super-markets in the vegetable/fruit section of the shop, where customer places plastic bag with products on scale, presses button and attaches the printed sticker with precise weight and price on the bag for further scanning at the checkout. This scale can be used for weighting items such as bottled water – printed sticker marks precise weight just for the price of the bottle.

Load cells intended for beehive weighting can be loaded up to 120kg, with limit of 300kg for Zemic L6G. Expected average weight of a beehive is 60kg before honey extraction.

Three Zemic L6G Load cells were calibrated (out of eight) on-site. For the calibration purpose, two 1.5L water bottles were used. These bottles have a sticker with weight on them – previously measured in the store, totaling $3066g$. Rest of the load cells are left uncalibrated due to existing beehives placed on them without the bees and without the possibility to move them out for the purpose of calibration, therefore all load-cells will be recalibrated during autumn without load.

[1] https://www.profivahy.cz/profi-vahy/eshop/37-1-Zavazi/318-3-Jednotlive-kusove

Figure 5.3: Scales for calibration with bottled water.

5.3 Battery powered system

First testing of the solar panel, solar charger along with battery and its self-sustainability has been tested upon milling prototype PCB of the power management board (Fig. 5.4).

Under the bright sun (no clouds blocking the sun) at 15:00, half-charged battery is charging with rate of 0,93A at voltage 14,5V. Solar panel has 21,0V without any load. Battery designated for deployment has 7,2Ah at 12V, in other terms 86Wh. At the charging rate of 13.49W, approximately 6 hours of the sun are necessary to fully charge the battery. Raspberry Pi 3 with its power consumption 2,05W (0,45A at 5V) can therefore run 43 hours continuously without the need of the sun. As there is no feedback from the power management board, power consumption as well as the voltage of the battery/solar panel cannot be remotely monitored.

Under-voltage threshold is set to 10.8V by the trimmer on board, 0,3V higher than lowest safe voltage as defined by the battery manufacturer[2].

5.4 Deployment

After the metal platform (Fig. 5.2) was installed under the beehives, plastic boxes with bus-units were mounted on the wooden wall. Cables from the load cells were grouped by two, having PVC insulation removed and attached to the screw terminals on the bus-unit. Interconnection between bus-units is as previously mentioned – telephone cable cut to the necessary length to make all the cabling as short as possible.

Master unit (Raspberry Pi 3 with CAN shield) along with power-management board placed in plastic box KP22 on appropriate place for easy access for the case of troubleshooting.

12V battery is placed right under the master unit. Solar panel is mounted on the roof.

[2] http://www.shimastu.com/upfile/NP7-12.doc

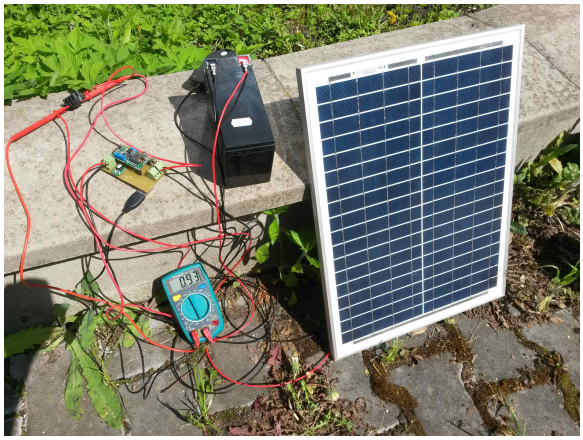

Figure 5.4: Test of the 20W 18V solar panel, 12V battery and power management board. USB mouse in the background indicating presence of the voltage on the USB port. Ampermeter connected in series with battery reading 0,93A.

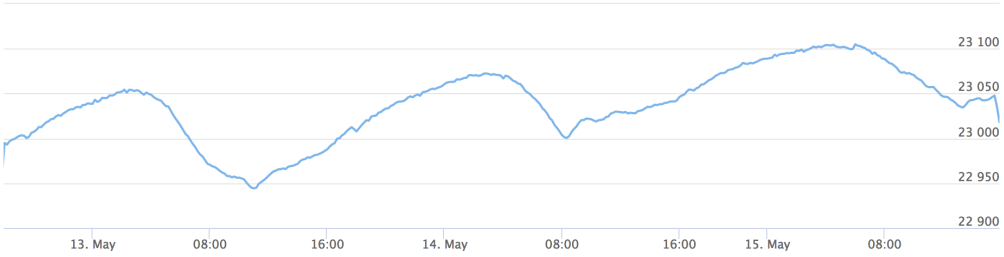

Figure 5.5: Weight measured by reference calibrated load cell.

Only first and last bus unit are connected. Two beehives have active bee colony, third one only honey for extraction.

5.5 Achieved results

Three day test of the system deployed on three beehives (one only with honey) was conducted. Collected data show that there are sudden spikes happening during thunderstorms on two load cells (See Fig. 4.11). Such events are suspected to be caused by missing grounding of the metal platform.

Temperature compensation of the acquired weight is not performed and as there is one load cell that is loaded with a constant weight, error caused by the temperature change can be estimated.

Average weight measured by the reference load cell is $(23037,33 \pm 40.54)g$ on 24786 weight samples. Current implementation of the web interface loads complete data-set for each sensor, therefore aggregation of the data will be required after some time. For the testing, responsiveness of the interface is sufficient.

Chapter 6

Conclusion

Designed system has been deployed as defined in the goals on designated beehives and tested. Several tweaks are necessary to make the system complete, and that is:

- Properly ground the metal platform with load cells
- Recalibrate all load cells when the beehives are ready for honey extraction and can be placed out of the scale
- Mount solar panel on a pole on the roof at the right angle to harvest energy throughout the whole day
- Place all cabling to a hardened plastic conduits for protection against rats and weather

During installation of the system on beehives, conclusion was made that even though wired CAN bus with power does the necessary job, it is not something that is easy to work with. Cabling between nodes comes with burden of mounting cables as high as possible to eliminate getting them chewed by rats.

If this system were designed once again after the experience of deployment, surely SIGFOX technology would be chosen, where each beehive has separate unit with battery pack, measuring and sending data independently. Also logging in to the Raspberry Pi with SSH client as well as flashing the MCU firmware is not an ability that general beekeeper

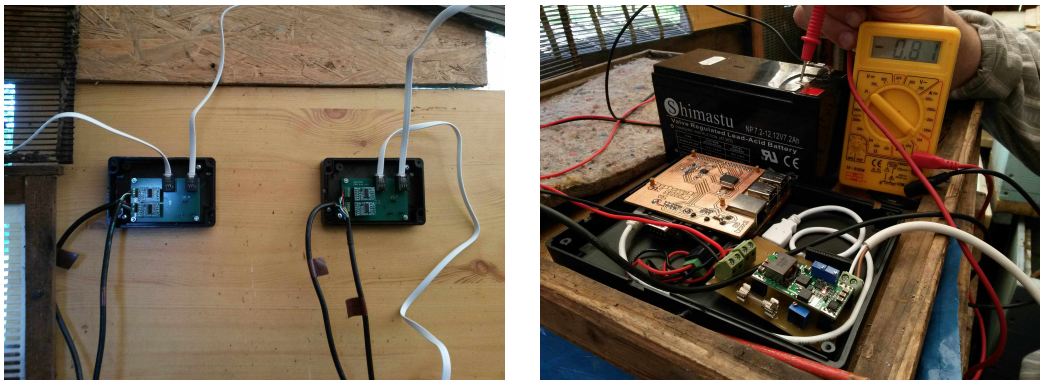

Figure 6.1: Bus units monted on the wall with connected load cells and telephone cables (left). Testing of connected devices, powered only from the solar panel (right).

may have. Selecting cabled system for interconnection of the bus-units makes sense in case of specialized deployment (tens of beehives in one chain), where replacing batteries every year or two may make it unnecessary overhead.

Source code as well as designed boards along with schema are published as an open-source. Provided design files may not guarantee that the system will be replicated by somebody else, but makes a space for collaboration with general public beekeepers.

Future work on the system includes:

- Redefinition of necessary functionality
- Getting in touch with more beekeepers to collect special requirements (Reference beekeeper has beehives only on garden close to WiFi source)
- Considering making the system low-power with wireless transmission based on SIGFOX/LoRaWAN technology
- Simplifying the system to one PCB per beehive with minimal wiring
- Creating bus-units for various other sensors, e.g. temperature, humidity

Outcomes of this thesis are:

- Hardware: Design and implementation of bus-units interfacing CAN bus with weight converters
- Hardware: Manufactured bus-units by a professional PCB manufacturing company
- Hardware: Design and implementation of CAN shield for Raspberry Pi 3 – milled on CNC
- Hardware: Design and implementation of Power management board (includes under-voltage protection, battery charger, fusing, voltage regulation) – milled on CNC
- Software: Port of Ivory Tower Framework to STM32F0
- Software: Firmware for STM32F042 MCU – acquiring data from HX711 converters and sending them to CAN bus
- Software: Web interface with charts for monitoring
- Software: Program running on Raspberry Pi listening on CAN bus, repacking received CAN frames to HTTP requests

Achieved error $40.54g$ on reference scale with dead-weight shows satisfactory results as defined in goals. Price of the whole set of devices necessary for measuring is 4097 czk with one bus-unit, not including the price of the load cells. Additional bus-unit costs 327 czk each.

Bibliography

[1] ALYA, s.r.o.: Pricelist of beehive weights. http://www.alya.sk/alya/userfiles/documents/vcely/Cennik_vcelarskych_vah_2-2015.pdf. 2015, [cite. 2016-05-12].

[2] ALYA, s.r.o.: VILKO - multifunkčné vážiace zariadenie (MVZ) pre včelárov. http://www.alya.sk/vcelarska-vaha-vilko. 2015, [cite. 2016-05-12].

[3] Arnia: Arnia – remote hive monitoring, System Overview. http://www.arnia.co.uk/how-it-works/. 2011, [cite. 2016-05-13].

[4] AVIA Semiconductor: 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales. https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf. 2009, [cite. 2016-05-10].

[5] Beewise: BeeWISE, product information. http://www.beewise.eu/bee-hive/gms-sms-scale.php. 2005, [cite. 2016-04-13].

[6] Blackiston, H.: *Beekeeping for Dummies*. For Dummies. Wiley. second edition. 2009. ISBN 0470430656,9780470430651,9780470496299. Retrieved from: http://gen.lib.rus.ec/book/index.php?md5=ED57670EBF7D2ADC5B61FD381EE16DEE

[7] Cetinkunt, S.: *Mechatronics With Experiments, 2nd Edition*. Wiley. 2015. iSBN 9781118802465.

[8] Dewesoft: Strain measurement. http://www.dewesoft.com/pro/course/strain-measurement-1. 2016 [cite. 2016-01-27].

[9] Elliott, T.; Pike, L.; Winwood, S.; et al.: Guilt Free Ivory. In *Haskell Symposium*. ACM. 2015. available at http://www.cs.indiana.edu/~lepike/pub_pages/ivory.html.

[10] Fortich, P.: Flexible Extremely Low Power Battery Protector. http://www.schematics.com/project/flexible-extremely-low-power-battery-protector-16385/. Jan 2017, [cite. 2017-04-10].

[11] Integrated, M.: Microprocessor Voltage Monitors with Programmable Voltage Detection. https://datasheets.maximintegrated.com/en/ds/MAX8211-MAX8212.pdf. 2009.

[12] Kalfus, R.; Hégr, T.: *Ultra Narrow Band Radio Technology in High-Density Built-Up Areas*. Cham: Springer International Publishing. 2016. ISBN 978-3-319-46254-7. pp. 663–676. doi:10.1007/978-3-319-46254-7_54.
Retrieved from: http://dx.doi.org/10.1007/978-3-319-46254-7_54

[13] Sauter, M.: *From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*. John Wiley & Sons. first edition. 2011. ISBN 0470667117,9780470667118,9780470978245,9780470978238,9780470978221.
Retrieved from:
http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118861957.html

[14] SimpleCell: Technolgia sigfox.
http://www.simplecell.sk/pages/technologia_sigfox/. 2017, [cite. 2017-05-10].

[15] Steve Corrigan: Application Report: Introduction to the Controller Area Network (CAN). http://www.ti.com/lit/an/sloa101a/sloa101a.pdf. 2002, [cite. 2017-04-10].

[16] Steve Corrigan: Controller Area Network Physical Layer Requirements. http://www.ti.com/lit/an/slla270/slla270.pdf. 2008, [cite. 2017-04-10].

[17] STMicroelectronics: RM0091 Reference Manual.
www.st.com/resource/en/reference_manual/dm00031936.pdf. 2017.

[18] TechTarget: PHP (Hypertext Preprocessor).
http://searchenterpriselinux.techtarget.com/definition/PHP. Nov 2006, [cite. 2017-04-10].

[19] Tektronix: Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements.
http://www.nortelcoelectronics.se/document-file5116?lcid=1053. 2013, [cite. 2017-05-10].

[20] Texas Instruments: *LM1117 800-mA Low-Dropout Linear Regulator*. 2 2000. rev. Jan 2016.

[21] User:saper_2: CAN bus on raspberry pi with MCP2515.
https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=141052. Mar 2016, [cite. 2017.1.2017].

[22] Zemic: Loadcells.
https://www.zemiceurope.com/categories/load-cells.html. 2017, [cite. 2017-04-10].

Appendices

Appendix A

Contents of the DVD

- `/raspberry_pi` – partition table, root and boot image
- `/boards` – repository with boards and schematics
- `/hexamon-firmware` – firmware for the MCU
- `/hexamond` – source files of the program forwarding CAN frames to the web server, installation not needed as is integrated in Raspberry Pi image
- `/src` – source files of the thesis in LaTeX
- `/models` – 3D printed spacers for the boxes
- `projekt.pdf` – electronical version of the thesis

Appendix B

Bill of material

Cost of the material needed for weighting one beehive decreases with the increasing length of chain of beehives connected to one master node on the CAN bus. Currency in following tables is Czech Crown (CZK). Following tables list necessary components needed to recreate whole system:

count	name	unit price
1	Raspberry Pi 3	1359 czk
1	8GB SD Card (Industrial Temp. Range)	229 czk
1	Plastic box KP22	79 czk
1	Plastic case for Raspberry Pi 3	200 czk
1	RJ12 connector WEBP 6-6	10 czk
1	FR4 16x10 cm copper clad, single side	29 czk
1	20x2 2.54mm pin header female connector	15 czk
1	MCP2515 CAN controller	50 czk
1	SN65HVD230 or SN65HVD233 CAN transceiver	61 czk
1	MicroUSB cable 20cm	119 czk
1	Solar panel 20W 18V	543 czk
1	3A fuse	10 czk
1	12V Battery	459 czk
_	Resistors & capacitors	~100 czk
_	Row of 2.54mm female header	~15 czk

Table B.1: BOM of the single Master unit (Raspberry Pi with CAN shield) along with accessories

Price of one bus-unit is 327 czk (approx. 12,40 EUR), master unit with accessories costs 3278 czk (123,72 EUR) and power management board costs 492 czk (18,50 EUR)

count	name	unit price
1	Plastic box KM78	79 czk
2	RJ12 connector WEBP 6-6	10 czk
1	STM32F042F6P6	48 czk
1	1206 SMD LED	2 czk
2	HX711 weight converters	30 czk
2	4 pin 2.54mm screw terminal block	15 czk
1	SN65HVD230 or SN65HVD233 CAN transceiver	61 czk
1	FR4 16x10 cm copper clad, single side	29 czk

Table B.2: BOM of the single Bus unit

count	name	unit price
1	MAX8212ESA	119 czk
1	IRFR5305 P-MOSFET	23 czk
1	5A solar charger	309 czk
3	2 pin 5.08mm screw terminal block	4 czk
1	FR4 16x10 cm copper clad, single side	29 czk

Table B.3: BOM of the single Power management board along with accessories

Appendix C

Replication manual

This chapter includes step-by-step list of actions needed to replicate the device set.

C.1 Bus unit, CAN shield and power management board

1. Manufacture bus unit board – two beehives need one unit, master unit and power management board are needed only once per chain (schematics are available in folder `/boards`)
2. Gather all components as shown in BOM with resistors and capacitors as shown in schemas: (`/boards/busunit/busunit.sch`, `/boards/master_can_shield/master_pi.sch`, `/boards/power_board/power_board.sch`)
3. Solder components on all boards

C.2 Compilation of firmware and flashing to the MCU

For the flashing, ST-Link v2 is required.

1. Compile firmware, as described on https://github.com/lucansky/hexamon-firmware.
2. Serial number of the unit has to be modified for each bus-unit in file `src/Hexamon/Platforms.hs` on line containing „hexamonConfigUID“.

C.3 Web interface

1. Clone repository https://github.com/lucansky/hexamon-web
2. Rename `.env.example` file to `.env` and modify credentials
3. Install `composer`[1]
4. Issue `composer install` command in project directory to install necessary packages
5. Issue `php artisan migrate` command to create SQL structures in database
6. Issue `php artisan serve` to start web-server. URL of the web is `http://<IP>:8000`
7. On URL `http://<IP>:8000/sensors` add sensor IDs to the database

[1] https://getcomposer.org

C.4 Raspberry Pi

1. Write `master/image.bz2` to at least 8GB SD card with commands:

   ```
   # modify mmcblk0 name of the system device appropriately
   cat raspberry_pi/partitions.txt | sudo sfdisk -f /dev/mmcblk0
   bzcat raspberry_pi/boot.bz2 | sudo dd of=/dev/mmcblk0p1
   bzcat raspberry_pi/root.bz2 | sudo dd of=/dev/mmcblk0p2
   ```

   After successful SD card flashing, attach CAN shield, insert SD card and boot the Raspberry Pi with the Ethernet connected to a local network.

   Find the IP address of the device on LAN and connect to it with SSH client with username `pi` and password `changeme`. Issue following commands:

   ```
   sudo bash
   vim /etc/wpa_supplicant/wpa_supplicant.conf # edit WPA2 credentials
   rm /var/lib/tor/hidden_service/hostname
   rm /var/lib/tor/hidden_service/private_key
   reboot

   cd hexamond
   vim src/Hexamon.hs # modify URL of the web server according to your setup
   stack exec -- hexamond
   ```

   Program shall now start transmitting received CAN frames to the server.