



University of Applied Sciences

HOCHSCHULE
EMDEN-LEER

Technisches Projekt

Entwurf und Bau einer
Bienenstockwaage

11.01.2019



Verfasser: Holger Tammen Stefan Kassen

Studiengang: Maschinenbau und Design

Fachrichtung: Produktionstechnik

Inhaltsverzeichnis

Abbildungsverzeichnis	3
Tabellenverzeichnis.....	4
Einleitung	5
Motivation.....	5
Vorstellung Arduino.....	5
Vorstellung Raspberry Pi.....	5
Zigbee.....	6
HX711.....	7
Auswahl der Komponenten	8
Durchführung.....	10
Arduino Uno.....	10
Teileliste	10
Vorbereitung	10
Verkabelung.....	12
Ersteinrichtung	13
Bibliotheken	14
Kalibrierprogramm	14
Hauptprogramm	18
Raspberry Pi.....	21
Teileliste	21
Ersteinrichtung	21
ThingSpeak.....	25
Programm	29
Zusammenbau	32
Ergebnis	38
Quellen	40

Abbildungsverzeichnis

Abbildung 1: Arduino.....	5
Abbildung 2: Raspberry Pi	5
Abbildung 3: Zigbee.....	6
Abbildung 4: XCTU Konfiguration.....	6
Abbildung 5: HX711 ADC.....	7
Abbildung 6: Blockdiagramm HX711.....	7
Abbildung 7: XBee Shield.....	10
Abbildung 8: Adapter 4 Kabel.....	11
Abbildung 9: Adapter Kabel verlöten	11
Abbildung 10: Adapter Kabel anlöten	12
Abbildung 11: Angeschlossener HX711.....	13
Abbildung 12: Arduino Systemübersicht.....	13
Abbildung 13: Arduino Programmdownload	13
Abbildung 14: Bibliothek einbinden	14
Abbildung 15: Ausgabe Kalibrierung	17
Abbildung 16: SD Karte formattieren.....	21
Abbildung 17: Raspbian Imager	22
Abbildung 18: Raspbian erfolgreich übertragen	22
Abbildung 19: WLAN	22
Abbildung 20: Rechte wechseln	23
Abbildung 21: Crontab öffnen.....	23
Abbildung 22: Neustart einstellen.....	23
Abbildung 23: WLAN Interface öffnen	24
Abbildung 24: Energiesparmodus Code	24
Abbildung 25: ThingSpeak neuen Channel erstellen.....	25
Abbildung 26: Felder definieren.....	25
Abbildung 27: Übersicht über Felder	26
Abbildung 28: Channel ID.....	26
Abbildung 29: API KEY	27
Abbildung 30: MATLAB Visualization	27
Abbildung 31: Thingspeak leere Visualisierung erstellen.....	27
Abbildung 32: Programmcode Schwerpunktdiagramm	28
Abbildung 33: Python öffnen	29
Abbildung 34: Autostart einstellen	31
Abbildung 35: Gewinde schneiden.....	32
Abbildung 36: Distanzstück	32
Abbildung 37: Distanzstück und Schrauben.....	33
Abbildung 38: Wägezelle auf Holzplatte	34
Abbildung 39: Wägezelle mit Distanzstück	34
Abbildung 40: Vier Wägezellen auf Holzplatte.....	34
Abbildung 41: Einstellen Höhe	35
Abbildung 42: Makierung für Bohrung.....	36
Abbildung 43: Verkabeltes Gehäuse	36
Abbildung 44: Stromversorgung löten	37
Abbildung 45: USB Kabel offene Enden	37
Abbildung 46: Isolation der Lötstelle	37
Abbildung 47: Waage ohne Deckel	38

Abbildung 48: Waage mit Deckel	38
Abbildung 49: Gewichtsverlauf	39
Abbildung 50: Schwerpunkt in X und Y	39

Tabellenverzeichnis

Tabelle 1: Teileliste	8
Tabelle 2: Anschluss Wägezelle	12
Tabelle 3: Anschluss Arduino	12

Einleitung

Motivation

Wir haben das Prinzip der Waage nicht neu erfunden, ebenso nicht der Bienenstockwaage. Es existieren auf dem Markt bereits verschiedenste Bienenstockwaagen, welche allerdings als Fertigprodukt recht kostenintensiv sind.

Unsere Motivation bestand nun also darin, mit einfachsten Mitteln eine funktionierende Waage zu bauen, welche sich auch Hobby-Imker leisten können. Die Daten sollten hierbei, ebenso wie bei einer Bienenstockwaage aus dem Laden, an einen Webserver gesendet werden, so dass man von überall aus das Gewicht des eigenen Stocks beobachten kann.

Gegeben war der Anspruch, das System erweiterbar zu gestalten.

Vorstellung Arduino



Abbildung 1: Arduino

Der Arduino Uno ist ein Mikrocontroller, welcher 6 analoge und 14 digitale Ein-/Ausgänge besitzt. Darüber hinaus besitzt der Arduino Uno ein 16 MHz Quartz, eine USB Schnittstelle, einen 6 poligen ICSP Anschluss und einen Reset-Taster.

Die Stromversorgung und die Programmierung erfolgt über die USB Schnittstelle.

Der Arduino Uno hat in unserem Aufbau zwei wesentliche Aufgaben. Zum einen fungiert er als Aufnehmer der Messdaten und zum anderen sendet er die verarbeiteten Messdaten weiter an den Raspberry Pi.

Vorstellung Raspberry Pi



Abbildung 2: Raspberry Pi

Der Raspberry Pi ist ein kostengünstiger Mini-Computer. Das Betriebssystem beruht auf einer Linux Oberfläche. In einzelnen Bestandteilen ähnelt dieses dem Windows-System.

In seiner Grundausstattung verfügt der Minirechner über einen ARM-Prozessor, einen Grafikprozessor und über Arbeitsspeicher. Ebenso können über Schnittstellen wie USB, HDMI, Ethernet, 21 GPIO-Pins etc. andere Geräte mit dem Raspberry Pi verbunden oder Funktionen angesteuert werden. Der Speicherplatz für den Raspberry Pi wird von einer SD-Karte bereitgestellt. Darauf muss zu Beginn auch das Betriebssystem für den Pi kopiert werden, welches von der Herstellerwebsite kostenlos heruntergeladen werden kann. Wir haben uns für das Betriebssystem Jessie entschieden, da es einfach zu bedienen ist und alle benötigten Funktionen bereitstellt. Es stehen unterschiedliche Modelle zur Verfügung, die sich in ihrer Ausstattung und Rechenleistung unterscheiden. In unserem Fall verarbeitet der Raspberry Pi die empfangenen Daten des Arduinos und sendet sie weiter an den Webserver, wo sie grafisch dargestellt werden.

Zigbee



Abbildung 3: Zigbee

Es wird aufgrund der noch folgenden Argumente ein Zigbee verwendet. Zigbee wurde von der Zigbee Alliance entwickelt und dadurch als Standardschnittstelle für die Homeautomation eingeführt. Die Funkmodule mit Zigbee Kommunikation werden XBee genannt und von dem Unternehmen Digi International vertrieben.

Unser Modul ist ein Pro Modul und erreicht auf freiem Feld eine Reichweite von bis zu 1600 Metern und hat beim Übertragen einen Verbrauch von 63mW. Es unterstützt Standard Baudraten von 1200 bps bis 250 kbps. Die Versorgungsspannung liegt zwischen 2,8 und 3,4 Volt. Bei 3,3 Volt ergeben sich fürs Senden eine Stromstärke von 250mA und fürs empfangen 55mA.

Der verwendete XBee besitzt eine aufgedruckte 2,4 GHz Antenne. Mit dem ebenfalls von Digi International angebotenen Programm XCTU lässt sich der XBee leicht konfigurieren und so zum Beispiel auch die Baudrate beliebig einstellen.

Einfachhalber wurde die Baudrate aber auf der Standardeinstellung von 9600 belassen. Das Funkmodul kann also einfach eingesteckt und direkt in Betrieb genommen werden. Es funktioniert in den Standarteinstellungen direkt und kommuniziert auch direkt mit anderen Funkmodulen auf Standarteinstellung.

Benutzt man allerdings mehrere Module mit demselben Kommunikationsstandard, so benötigt das jeweilige Modul eine Konfiguration des Channels und/oder der PAN (Personal Area Network) ID. Dies ist im Programm XCTU leicht zu ändern.

Networking & Security

Modify networking settings







CH Channel	<input type="text" value="C"/>	 
ID PAN ID	<input type="text" value="3332"/>	 
DH Destination Address High	<input type="text" value="0"/>	 

Abbildung 4: XCTU Konfiguration

HX711

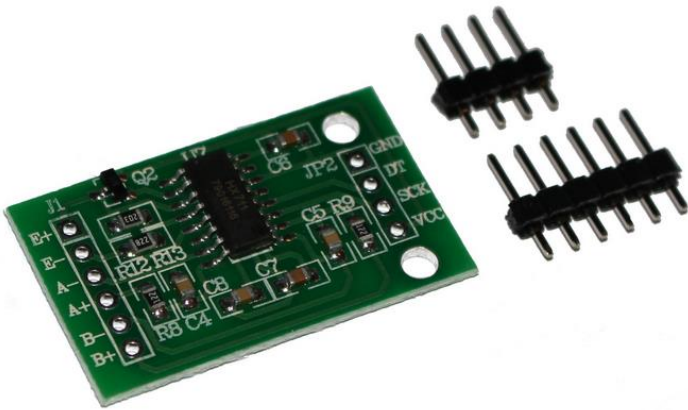


Abbildung 5: HX711 ADC

Bei dem HX711 handelt es sich um einen 24 bit Analog-Digital-Converter (kurz: ADC). Dieser wurde speziell für Wägezellen entwickelt und enthält direkt eine Brückenschaltung. Es sind zumeist zwei Kanäle verfügbar, wobei Kanal A über eine Verstärkung von Standardmäßig 128 oder alternativ 64 angesteuert werden kann und Kanal B über eine Verstärkung von 32. Der Chip muss nicht programmiert werden, sondern kann direkt über die Pins angesteuert werden. Der HX711 besitzt auf der Seite der Wägezellen die Anschlüsse E+ und E- für die Spannungsversorgung, A+ und A- für Daten von Kanal A und B+ und B- für die Daten von Kanal B. Auf der Seite des Arduino besitzt er die Anschlüsse VCC für die Versorgungsspannung von 5 Volt, SCK als Serial Clock für die zeitliche Ansteuerung des HX711, DT als Datenausgang und GND als Erdung.

Wir haben für jede Wägezelle einen eigenen HX711 verwendet, da bei paralleler Nutzung der Kanäle A und B die konstante Spannungsversorgung nicht gewährleistet werden konnte. Infolgedessen kam es zu falschen Messwerten, wodurch die Funktion der Waage nicht gegeben war.

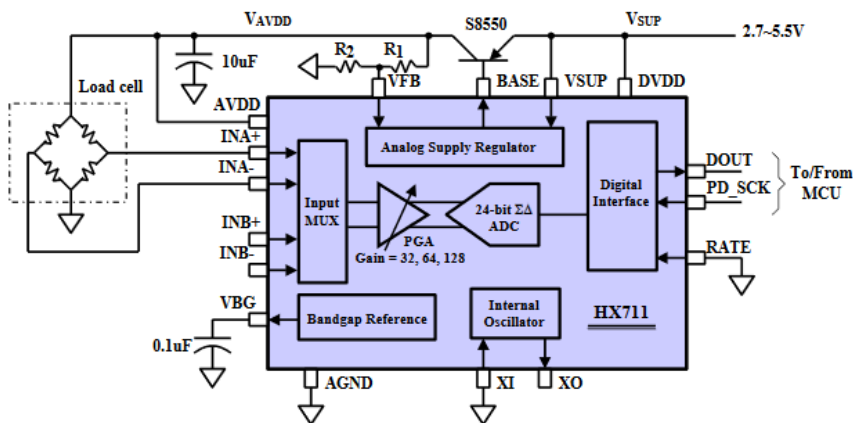


Abbildung 6: Blockdiagramm HX711

Auswahl der Komponenten

Es wurde sich für den Raspberry Pi 3 B+ entschieden, da dieser eine Wlan- & Ethernetschnittstelle beinhaltet. Außerdem besitzt er die meiste Rechenleistung.

Bei der Auswahl des Funksystems für die Verknüpfung zwischen Arduino und Raspberry Pi entschieden wir uns für die Kommunikation via Zigbee. Diese ist zwar etwas teurer als andere Techniken, dafür ist sie aber einfach einsetzbar, hat eine hohe Reichweite und läuft sehr stabil. Außerdem ist es sehr gut verknüpfbar, da diese Technik bei den meisten Hausautomatisierungssystemen zum Einsatz kommt.

Für die Verbindung der Zigbee werden für den Arduino ein Shield und für den Raspberry Pi ein USB Explorer benötigt. Dies hat den Vorteil der einfachen Plug and Play Verbindung ohne weitere Kabel.

Tabelle 1: Teileliste

<u>Bezeichnung</u>	<u>Stk.</u>	<u>Link</u>	<u>Einzel Preis [€]</u>	<u>Gesamt Preis [€]</u>
Raspberry Pi 3 B+	1	https://www.exp-tech.de/plattformen/raspberry-pi/8551/raspberry-pi-3-modell-b	38,21	38,21
Arduino Uno	1	https://www.exp-tech.de/plattformen/arduino/4380/arduino-uno-r3	21,66	21,66
Speicherkarte 32GB	1	https://www.exp-tech.de/zubehoer/speicherkarten/8689/kingston-32gb-canvas-select-sdcs-klasse-10-microsd-speicherkarte	13,26	13,26
USB-Adapter	1	https://www.exp-tech.de/zubehoer/netzteile/7906/hnp12-usbl6-5v-2-4a-usb-steckernetzteil-12w?c=1106	7,25	7,25
USB-Kabel Micro	1	https://www.exp-tech.de/zubehoer/kabel/usb/8247/micro-usb-kabel-2m?c=1186	2,90	2,90
USB-Kabel Mini	1	https://www.exp-tech.de/zubehoer/kabel/usb/7877/mini-usb-kabel-usb-a-auf-mini-usb-b-1.8m?c=1186	1,85	1,85
USB-Kabel B-Stecker	1	https://www.exp-tech.de/zubehoer/kabel/usb/5285/usb-kabel-2.0-a-stecker-auf-b-stecker-1.8m?c=1186	1,79	1,79
Zigbee	2	https://www.exp-tech.de/module/wireless/xbeezigbee/5068/xbee-pro-10mw-pcb-antenna-series-1-xbp24-api-001j	43,14	86,28
Xbee Shield	1	https://www.exp-tech.de/module/schnittstellen/6212/sparkfun-xbee-shield	14,95	14,95
Xbee Explorer	1	https://www.exp-tech.de/module/wireless/xbeezigbee/5774/sparkfun-xbee-explorer-usb-wrl-11812	24,95	24,95
Jumperkabel m/m	1	https://www.exp-tech.de/zubehoer/kabel/jumper-wires/7798/premium-male/male-jumper-wires-40-x-6-150mm?c=1179	4,25	4,25
Jumperkabel f/m	1	https://www.exp-tech.de/zubehoer/kabel/jumper-wires/8035/premium-female/male-extension-jumper-wires-40-x-12-300-mm?c=1179	7,66	7,66
Steckverbinder	1	https://www.exp-tech.de/zubehoer/steckverbinder/5421/arduino-stackable-header-kit-r3	1,50	1,50

Powerbank	1	https://www.reichelt.de/powerbank-li-po-20800-mah-usb-ans-pb-20800-p195298.html?&trstct=pol_3	27,60	27,60
Gehäuse Raspberry Pi	1	https://www.reichelt.de/gehaeuse-fuer-raspberry-pi-3-schwarz-grau-rasp-3-case-bg-p167287.html?ARTICLE=167287&&trstct=pol_14;SID=94W9i6dKwQATUAAD1boIU6a2ae08c7232c4e1306113ed33e7a9b5	7,60	7,60
Kunststoffgehäuse transparent	1	https://www.reichelt.de/kunststoffgehaeuse-240-x-120-x-60-mm-ip65-rnd-455-00138-p193352.html?&trstct=pol_1	11,35	11,35
Kunststoffgehäuse grau	1	https://www.reichelt.de/kunststoffgehaeuse-240-x-120-x-60-mm-ip65-rnd-455-00127-p193341.html?&trstct=pol_2	11,30	11,30
Kabelverschraubung M12	5	https://www.reichelt.de/metrische-kabelverschraubung-3-0-6-5-mm-m12-grau-mbf-12-p44825.html?&trstct=pos_0	0,30	1,50
Anschlusskabel 2,1mm	1	https://www.conrad.de/de/niedervolt-anschlusskabel-niedervolt-stecker-kabel-offenes-ende-55-mm-21-mm-21-mm-bkl-electronic-2-m-1-st-734183.html	3,36	3,36
Kabel & Buchse Rundstecker	1	https://www.conrad.de/de/tru-components-1229336-rundstecker-konfektioniert-buchse-gerade-stecker-einbau-gesamtpolzahl-2-1-st-1571074.html	11,24	11,24
USB-Kabel offenes Ende	1	https://www.conrad.de/de/inline-usb-20-kabel-a-an-offenes-ende-schwarz-2m-bulk-800404946.html	2,68	2,68
Blechdeckel	1	https://www.holtermann-shop.de/Beutenzubehoer/Blechdeckel--halbkonisch-/Blechdeckel-halbkonisch-Segeberger-Beuten.html	16,50	16,50
Wägezellen	4	https://www.tinkerforge.com/de/shop/accessories/sensors/load-cell-50kg-czl601.html	11,99	47,96
HX711	4	https://www.roboter-bausatz.de/204/hx711-24-bit-gewichtssensor	1,76	7,04
Siebdruckplatte 50cmx50cm	2	Baumarkt	-	14,99
Distanzstücke Aluminium/Holz/ etc.	4	-	-	-
Gesamtsumme				389,63

Schrauben etc. werden nicht aufgeführt, da sie je nach Anforderungen ausgewählt werden müssen und als vorausgesetzt gelten.

Durchführung

Arduino Uno

Teilleiste

Benötigt wurden für den Aufbau des Arduinos:

- Arduino Uno
- USB-Kabel B-Stecker
- Wägezellen (4Stk.)
- HX711 (4Stk.)
- Zigbee
- XBee Shield
- Jumperkabel (diverse)
- Steckverbinder

Darüber hinaus werden zusätzlich noch folgende Hilfsmittel benötigt

- Lötkolben
- Lötzinn
- Schrumpfschlauch
- Laptop/PC

Vorbereitung

Da das XBee Shield ohne Steckverbinder geliefert wird, müssen im Vorfeld die Steckverbinder mit dem XBee Shield verlötet werden. Hierbei muss darauf geachtet werden die Steckverbinder an die richtige Position zu bringen. Der kürzere muss an den analogen Pins A0 bis A5 und der längere an den digitalen Pins 8 bis SCL angelötet werden. Die beiden mittleren Steckverbinder sind dementsprechend an den Pins RX bis 7 und den Pins der Spannungsversorgung zu verlöten.

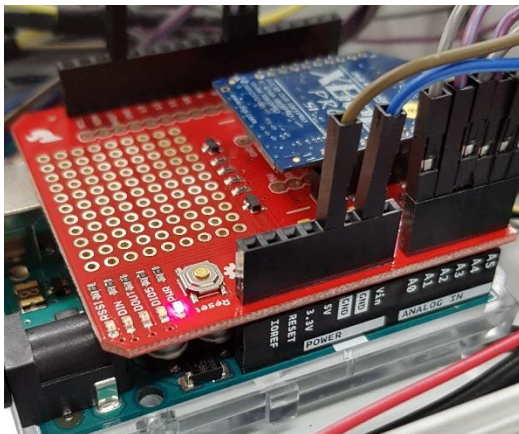


Abbildung 7: XBee Shield

Ebenso müssen die mitgelieferten Steckverbinder des HX711 verlötet werden.

Da bei der Wägezelle keine direkte Anschlussmöglichkeit an den HX711 Analog-Digital-Wandler besteht, muss diese durch erneute Lötarbeit geschaffen werden. Hierfür benötigt man die weiblichen Seiten der Jumperkabel, optimalerweise in den Farben der Kabel (rot,schwarz,grün,weiß) und für die Abschirmung die männliche Seite in Gelb. Die Jumperkabel können an beliebiger Stelle durchgeschnitten werden und müssen anschließend abisoliert werden. Optimalerweise werden die Kabel so kurz wie möglich, aber so lang wie nötig zugeschnitten. Diese verlötet man entsprechend mit den Kabeln der Wägezelle.

Ebenso ist es günstig sich „Adapter“ zusammenzusetzen, um ein Breadboard, also eine Steckplatine, zu vermeiden. Solch einen „Adapter“ benötigt man, da man vier Wägezellen hat und damit auch vier HX711, die alle eine 5 Volt Versorgung und eine Erdung benötigen, aber man nur einen 5 Volt Anschluss am Arduino Uno zur Verfügung hat und nur drei Erdungen (GND). Für diesen Adapter benötigt man je vier Mal die weibliche Seite und einmal die männliche Seite der Jumperkabel. Einen „Adapter“ benötigt man für die vier Abschirmungen der Wägezellen (idealerweise hier auch wieder mit gelben Kabeln), einen für die Spannungsversorgung der HX711 und einen für die Erdung der HX711. Man benötigt also drei Adapter.

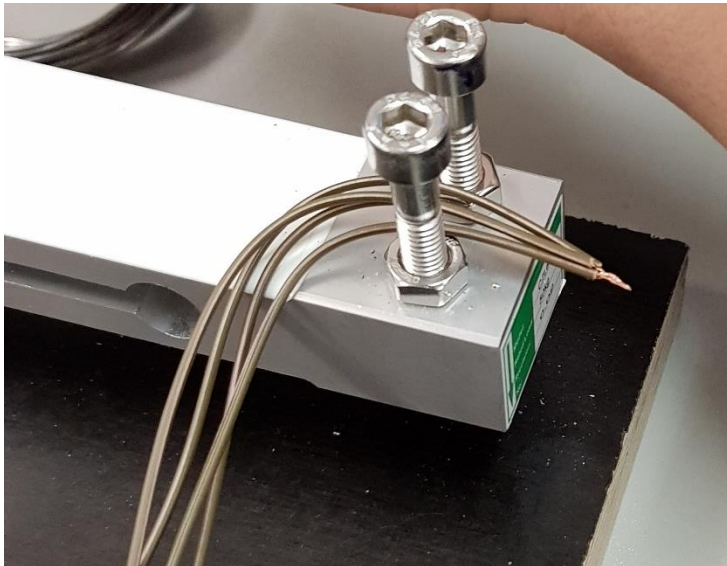


Abbildung 8: Adapter 4 Kabel

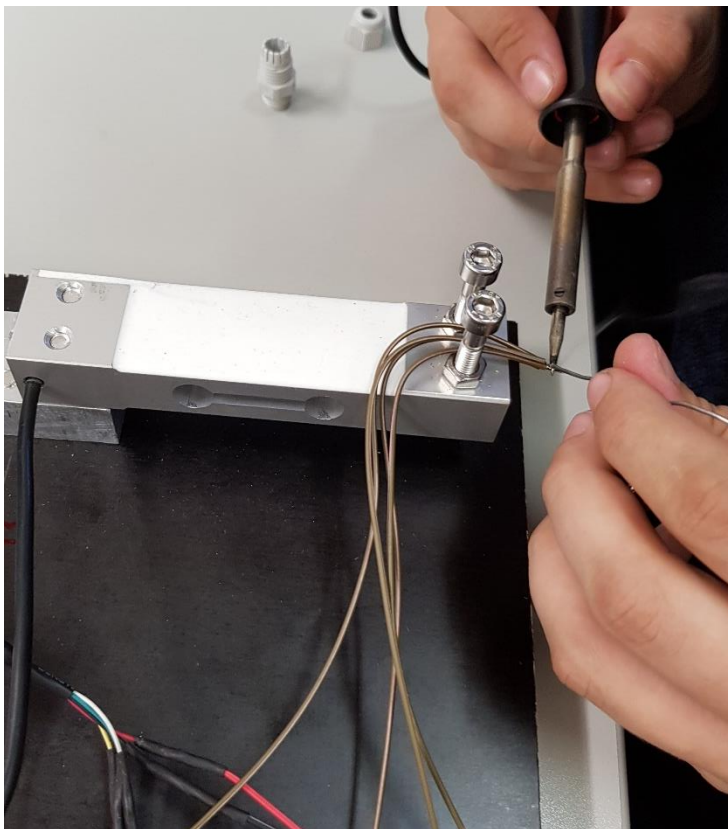


Abbildung 9: Adapter Kabel verlöten

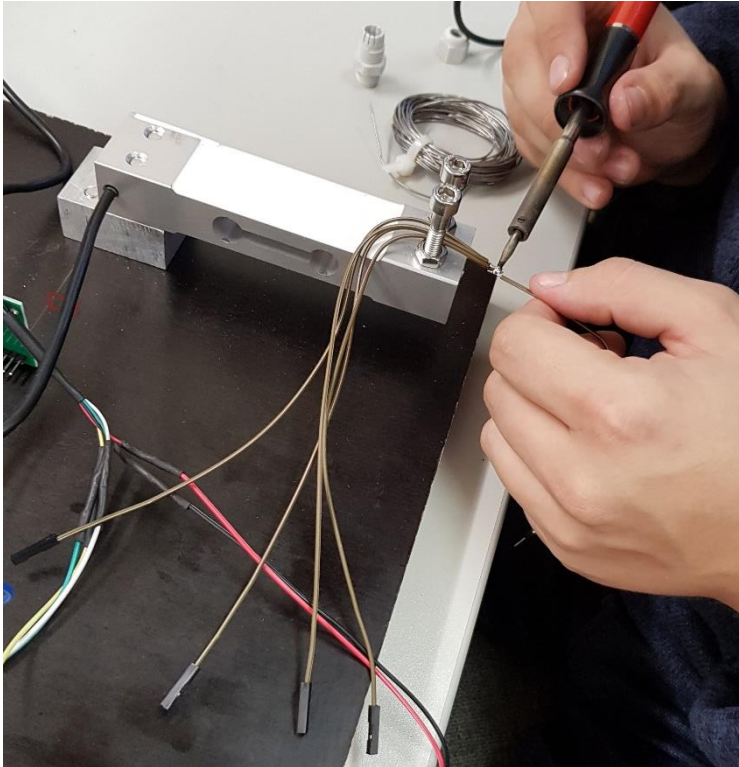


Abbildung 10: Adapter Kabel anlöten

Verkabelung

Für die Funktion der Bienenstockwaage ist die richtige Verkabelung grundlegend. Hierbei muss darauf geachtet werden, dass jedes Kabel richtig angeschlossen wird.

Der Zigbee ist in die entsprechenden Pins des Shields zu stecken und diese anschließend auf die PINS des Arduinos.

Das USB-Kabel mit dem B-Stecker ist zum einen mit der B Seite in den Arduino zu stecken und mit dem USB Teil in den Laptop oder PC. Hierüber wird zunächst auch die Stromversorgung gewährleistet.

Jede Wägezelle wird an einen eigenen HX711 angeschlossen. Grundsätzlich liegt den Wägezellen ein Zettel bei, auf dem definiert ist, welches Kabel, was bedeutet. Hier gilt:

Tabelle 2: Anschluss Wägezelle

Wägezelle	HX711
Rot	E+
Schwarz	E-
Weiß	A-
Grün	A+

Die Abschirmung (Gelb) ist mithilfe des Adapters an GND (Erdung) anzuschließen

Tabelle 3: Anschluss Arduino

HX711	Arduino
GND	GND (Adapter)
DT	A1 / A3 / A5 / 7 (für 4 Wägezellen nach unserm Programmcode)
SCK	A0 / A2 / A4 / 6 (für 4 Wägezellen nach unserm Programmcode))
VCC	5V (Adapter)

Für die erste Wägezelle ist entsprechend A1 für DT und A0 für SCK anzuschließen. Am Ende sollten dann alle Adapter, die Pins A0 bis A5 und 6 und 7 angeschlossen sein, dazu noch zwei Mal die Pins GND, einmal 5V und bei den HX711 sollten nur noch die Anschlüsse B- und B+ frei sein, sofern sie vorhanden sind.

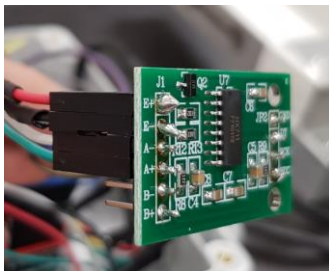


Abbildung 11: Angeschlossener HX711

Ersteinrichtung

Die Software des Arduino ist auf einem externen Zusatzgerät, zum Beispiel einem Laptop herunterzuladen. Sowohl Software, als auch Tutorials und alles weitere ist direkt auf der Homepage von Arduino verfügbar. Unter dem Link <https://www.arduino.cc/en/main/software> stehen sämtliche Versionen zur Verfügung.

ARDUINO 1.8.8
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
[Get](#)

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

Abbildung 12: Arduino Systemübersicht

Hier muss für das entsprechende Betriebssystem die richtige Version ausgewählt werden.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **29,073,354** TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

\$3 **\$5** **\$10** **\$25** **\$50** **OTHER**

[JUST DOWNLOAD](#) [CONTRIBUTE & DOWNLOAD](#)

Abbildung 13: Arduino Programmdownload

Es gibt die Möglichkeit zu spenden oder nur zu downloaden.

Nachdem das Programm installiert worden ist, kann es geöffnet und mit dem Programmieren gestartet werden.

Bibliotheken

Das Programm arbeitet, wie alle Programmier-Programme mit Bibliotheken, die zunächst eingebunden werden müssen.

Für den HX711 muss die Bibliothek zunächst als „zip“ heruntergeladen werden:

<https://github.com/bogde/HX711>

Die im Programm fehlende Bibliothek wird nach folgedem Schema eingebunden:

Sketch>Bibliotheken einbinden>.ZIP-Bibliothek hinzufügen.

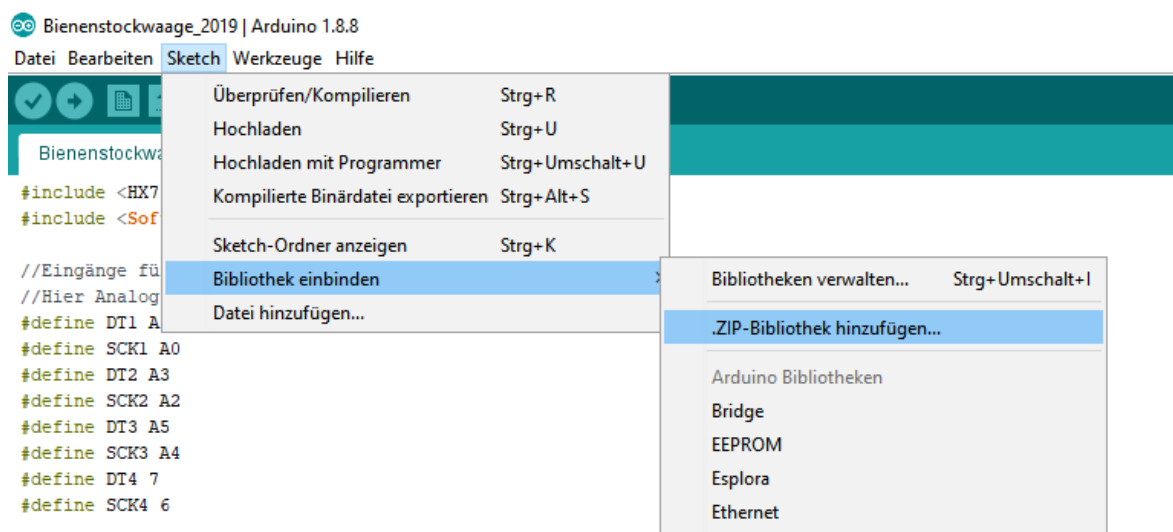


Abbildung 14: Bibliothek einbinden

Kalibrierprogramm

Um die Eigenschaften der Wägezellen zu ermitteln müssen diese zunächst kalibriert werden.

Dies läuft nach einem einfachen Schema ab.

Als erstes muss eingetragen werden, welche Eingänge beim Arduino mit der zu kalibrierenden Wägezelle verbunden sind.

Als zweites kann das Programm auf den Arduino geladen werden und gestartet werden.

Im seriellen Monitor erscheint nun die Vorgehensweise:

Im dritten Schritt muss, sofern die Waage unbelastet ist, die „1“ eingegeben werden und mit Enter bestätigt werden.

Im vierten Schritt ist die Waage mit dem Kalibriergewicht von 1000 Gramm zu belasten und anschließend ist die „2“ einzugeben und mit Enter zu bestätigen.

Nun ermittelt das Programm selbstständig den Skalierungsfaktor und das Taragewicht und gibt dieses im seriellen Monitor aus. Diese Werte sind zu notieren und im späteren Hauptprogramm einzutragen.

Das folgende Programm kann so übernommen werden und muss lediglich in den Eingängen, wie beschrieben, abgeändert werden.

Bei uns war die Wägezelle im aktuellen Beispiel mit den Datenpins des HX711 an die Pins 6 und 7 angeschlossen. Auf dem HX711 sind die Pins DT (Data) und SCK (Serial Clock) für die Datenbereitstellung zuständig.

Zusätzliche Kommentare und Hinweise beginnen im Programmcode mit einem „//“ und sind ausgegraut.

Programmcode für das Kalibrieren

```
#include <HX711.h>
//Hier müssen die jeweiligen Eingänge für jeden HX711 eingetragen werden
//Bei uns vier Durchgänge mit A0-A5 und Digital 6&7
#define HX711_DT      7
#define HX711_SCK    6
HX711 scale(HX711_DT, HX711_SCK);    //HX711 den Eingängen zuordnen

//Variablen deklarieren
char c;
long Gewicht = 999999;
long LetztesGewicht = 0;
const long Max_Abweichung = 500;
long Taragewicht = 1;
float Skalierung = 1.0;

// hier kann das Kalibriergewicht in Gramm eingetragen werden
long KalibrierGewicht = 1000;    // 1000 = 1kG Referenzgewicht

void setup() {
    //Start des Seriellen Monitors mit der Baudrate von 9600
    Serial.begin(9600);
    //Serial.println := Ausgabe auf Seriellen Monitor mit Zeilenumbruch
    //Serial.print := Ausgabe auf Seriellen Monitor ohne Zeilenumbruch
    //Bei Eingabe mit "" erfolgt direkte Ausgabe des Textes, ohne erfolgt
    Ausgabe des Wertes
    Serial.println("Waage Kalibrierung");
    Serial.print("DT auf ");
    Serial.println(HX711_DT);
    Serial.print("SCK auf ");
    Serial.println(HX711_SCK);
    Serial.println();

    //Zeitverzögerung von 5 Sekunden
    delay(5000);

    Serial.println("Zur Kalibrierung der Stockwaage bitte den Anweisungen
    folgen!");
    Serial.println();
    Serial.println("Waage ohne Gewicht - Kalibrierung mit '1' und 'Enter'
    starten!");
    Serial.println();
    while (c != '1') {                //Warte auf Eingabe einer "1"
        c = Serial.read();
    };
    c = 'x';
    Serial.println();
    Serial.print("Kalibriere ... ");
    scale.set_scale();                //Standard Skalierung setzen
    Serial.print(" ... ");
    delay(100);                        //Warte 0,1 Sekunden
    scale.read_average(20);           //Feststellen der Aktuellen Werte
    Serial.println(" ...");
}
```



```

    Taragewicht = scale.read_average(20); //Taragewicht setzen, ohne
Gewicht
    Serial.print("Waage mit genau ");
    Serial.print(KalibrierGewicht);
    Serial.println(" Gramm beschweren - Kalibrierung mit '2' und 'Enter'
starten!");
    Serial.println();
    while (c != '2') { //Warte auf Eingabe einer "2"
        c = Serial.read();
    };
    Serial.println();
    Serial.print("Kalibriere ... ");
    scale.set_offset(Taragewicht); //Einfügen des Taragewichtes als Offset
    Serial.print(" ... ");
    delay(100);
    scale.get_units(20); //Auslesen des Durchschnitts von 20
gemessenen Werten
    Serial.println(" ...");
    Skalierung = ((scale.get_units(20)) / KalibrierGewicht);
//Zwischenspeichern des tausendstel des Durchschnittswertes
    scale.set_scale(Skalierung); //Skalierung für die Wägezelle abspeichern
//Ausgabe der Ermittelten werte (Aktuelles Gewicht, Taragewicht und
Skalierung)
    Serial.print("Pruefe Gewicht: ");
    Serial.println(scale.get_units(20), 1);
    Serial.println();
    Serial.print("Taragewicht: ");
    Serial.println(Taragewicht);
    Serial.println();
    Serial.print("Skalierung: ");
    Serial.println(Skalierung);
    Serial.println();
}

void loop() {
    for (byte j = 0 ; j < 3; j++) { // Anzahl der Wiederholungen, wenn
Abweichung zum letzten Gewicht zu hoch
        Serial.println("Messung wird vorgenommen ...");
        Gewicht = (long) scale.get_units(10);
        if ( ((Gewicht - LetztesGewicht) < Max_Abweichung) and ((LetztesGewicht
- Gewicht) < Max_Abweichung) ) break; // Abweichung für Fehlererkennung
        Serial.println("Warte auf stabilen Messwert ...");
        if (j < 3) delay(3000); // Wartezeit zwischen Wiederholungen
    }
    LetztesGewicht = Gewicht; //Zwischenspeichern des Gewichtes
    Serial.print("Gewicht: "); //Ausgabe des Gewichtes
    Serial.print(Gewicht);
    Serial.println(" g");
    delay(5000);
    Serial.print("Skalierung: "); //Ausgabe von Skalierung und Taragewicht
    Serial.println(Skalierung);
    Serial.print("Taragewicht: ");
    Serial.println(Taragewicht);
    delay(5000);
}

```

```
Waage Kalibrierung
DT auf 7
SCK auf 6

Zur Kalibrierung der Stockwaage bitte den Anweisungen folgen!

Waage ohne Gewicht - Kalibrierung mit '1' und 'Enter' starten!

Kalibriere ... ..
Waage mit genau 1000 Gramm beschweren - Kalibrierung mit '2' und 'Enter' starten!

Kalibriere ... ..
Pruefe Gewicht: 999.8

Taragewicht: 35269

Skalierung: 84.36

Messung wird vorgenommen ...
Warte auf stabilen Messwert ...
Messung wird vorgenommen ...
Warte auf stabilen Messwert ...
Messung wird vorgenommen ...
Warte auf stabilen Messwert ...
Gewicht: 999 g
Skalierung: 84.36
Taragewicht: 35269
Messung wird vorgenommen ...
Gewicht: 1000 g
Skalierung: 84.36
Taragewicht: 35269
Messung wird vorgenommen ...
Gewicht: 1000 g
Skalierung: 84.36
Taragewicht: 35269
Messung wird vorgenommen ...
Gewicht: 1000 g
Skalierung: 84.36
Taragewicht: 35269
Messung wird vorgenommen ...
Gewicht: 1000 g
```

Abbildung 15: Ausgabe Kalibrierung

Hauptprogramm

Zunächst werden die Bibliotheken mit `#include` eingebunden.

Im nächsten Schritt wird definiert welche Kabel vom HX711 zu welchen Anschlüssen auf dem Arduino gehören. Diese müssen als nächstes dem jeweiligen HX711 zugeordnet werden. Da wir vier Wägezellen (englisch: scale) haben, müssen auch vier Zuordnungen vorgenommen werden.

Im Anschluss werden die Werte aus der Kalibrierung für die jeweilige Wägezelle eingetragen und die Anschlüsse des XBee und die benötigten Variablen definiert.

Hauptprogramm Abschnitt 1 (Definitionen)

```
#include <HX711.h>
#include <SoftwareSerial.h>

//Eingänge für die Verbindung mit den jeweiligen HX711 definieren
//Hier Analog 0-5 & Digital 6-7
#define DT1 A1
#define SCK1 A0
#define DT2 A3
#define SCK2 A2
#define DT3 A5
#define SCK3 A4
#define DT4 7
#define SCK4 6

//Eingänge den jeweiligen HX711 zuordnen
HX711 scale1(DT1, SCK1);
HX711 scale2(DT2, SCK2);
HX711 scale3(DT3, SCK3);
HX711 scale4(DT4, SCK4);

//Skalierungs- & Tarawerte aus Kalibrierprogramm eintragen
#define Skalierung1 86.5
#define Taragewicht1 36715
#define Skalierung2 87.59
#define Taragewicht2 -48381
#define Skalierung3 84.48
#define Taragewicht3 24110
#define Skalierung4 84.36
#define Taragewicht4 35269

//Serielle Schnittstelle für den Zigbee zuordnen. Bei Shieldverwendung
Digital 2&3
SoftwareSerial XBee(2, 3); // RX, TX

//Variablen deklarieren
long Gewicht1 = 999999;
long Gewicht2 = 999999;
long Gewicht3 = 999999;
long Gewicht4 = 999999;
long LetztesGewicht = 0;
float Gesamtgewicht;
float Schwerpunkt_X;
float Schwerpunkt_Y;
int KurzerAbstand = 400;
int LangerAbstand = 450;
//Maximale Abweichung zwischen zwei aufeinanderfolgende Messwerte
const long Max_Abweichung = 1500;
```

In „void setup()“ werden nun die Schnittstellen geöffnet und die Skalierung und Tarierung den Wägezellen zugeordnet.

Hauptprogramm Abschnitt 2 (Setup)

```
void setup() {
//Serielle & Zigbee Schnittstelle öffnen
Serial.begin(9600);
XBee.begin(9600);
Serial.println("Wiegevorgang");
Serial.println();
delay(1000);

//Die jeweiligen Taragewichte und Skalierungsfaktoren zuordnen
scale1.set_offset(Taragewicht1);
scale1.set_scale(Skalierung1);
scale2.set_offset(Taragewicht2);
scale2.set_scale(Skalierung2);
scale3.set_offset(Taragewicht3);
scale3.set_scale(Skalierung3);
scale4.set_offset(Taragewicht4);
scale4.set_scale(Skalierung4);
}
```

Im Folgenden wird die Hauptschleife „void loop()“ beschrieben, welche im laufenden Betrieb immer wieder ausgeführt wird. Hier wird zunächst für jede Wägezelle das Gewicht aufgenommen und gespeichert. Anschließend wird dieses umgerechnet in ein Gesamtgewicht und in die Schwerpunkte in X und Y Richtung. Zudem wird noch eine Abfrage in Form einer for-Schleife durchgeführt, in der ein Abgleich mit der vorherigen Messung stattfindet, um so Ausreißer zu eliminieren. Abschließend werden die Messwerte über den XBee an den Raspberry Pi gesendet und im seriellen Monitor ausgegeben.

Hauptprogramm Abschnitt 3 (Loop)

```
void loop() {
delay(100);
for(byte j=0 ;j < 3; j++) // Anzahl der Wiederholungen, wenn Abweichung
zum letzten Gewicht zu hoch
{
Serial.println("Messung wird vorgenommen ...");
//Werte der einzelnen Wägezellen auslesen
Gewicht1 = (long) scale1.get_units(10);
Gewicht2 = (long) scale2.get_units(10);
Gewicht3 = (long) scale3.get_units(10);
Gewicht4 = (long) scale4.get_units(10);

Gesamtgewicht= Gewicht1 + Gewicht2 + Gewicht3 + Gewicht4;

//Vergleich des aktuellen Messwertes mit dem vorherigen
if ( ((Gesamtgewicht-LetztesGewicht) < Max_Abweichung) and
((LetztesGewicht-Gesamtgewicht) < Max_Abweichung) ) break; // Abweichung
für Fehlererkennung
Serial.println("Warte auf stabilen Messwert ...");
if (j < 3) delay(3000); // Wartezeit zwischen Wiederholungen
}
```

```

//Berechnung des Schwerpunkts
Schwerpunkt_X=(Gewicht1+Gewicht3-Gewicht2-Gewicht4)*
(KurzerAbstand/2)/Gesamtgewicht;
Schwerpunkt_Y=(Gewicht1+Gewicht2-Gewicht3-Gewicht4)*
(LangerAbstand/2)/Gesamtgewicht;

LetztesGewicht = Gesamtgewicht;
XBee.print(Gesamtgewicht/1000); //Gesamtgewicht per Zigbee an den
Raspberry Pi schicken
XBee.print(';'); //Trennpunkt fuer Splittfunktion im
Raspberry Pi
XBee.print(Schwerpunkt_X); //Schwerpunkte übertragen
XBee.print(';');
XBee.print(Schwerpunkt_Y);
XBee.println(';');

//Gewichte im seriellen Monitor ausgeben, zur Kontrolle wenn PC an Arduino
angeschlossen
Serial.print("Gesamtgewicht: ");
Serial.print(Gesamtgewicht);
Serial.println(" kg");
Serial.print("Gewicht1: ");
Serial.print(Gewicht1);
Serial.println(" g");
Serial.print("Gewicht2: ");
Serial.print(Gewicht2);
Serial.println(" g");
Serial.print("Gewicht3: ");
Serial.print(Gewicht3);
Serial.println(" g");
Serial.print("Gewicht4: ");
Serial.print(Gewicht4);
Serial.println(" g");

delay(600000); //Wartezeit, bis nächster Messwert erfasst werden soll.
Angabe in Millisekunden. Alle 10 min Messwert
}

```

Der gesamte Hauptprogrammcode kann von der Definition bis zur Loop so übernommen werden, wobei natürlich die Texte zwischen den Abschnitten weggelassen werden müssen.

Raspberry Pi

Teilleiste

Benötigt wurden für den Aufbau des Raspberry Pi:

- Raspberry Pi 3 Model B+
- Gehäuse
- Netzteil (Am besten das originale oder mind. 5V & 2500 mA)
- Micro SD 32GB
- Zigbee
- Xbee Explorer
- Mini USB Kabel

Darüber hinaus werden für die Einrichtung folgende Hilfsmittel benötigt:

- Laptop/PC
- PC-Monitor (am besten mit HDMI-Anschluss)
- HDMI-Kabel
- Maus & Tastatur
- Kartenlesegerät

Ersteinrichtung

Der erste Schritt ist das Beschreiben der SD Karte mit dem Betriebssystem. Es kann zwischen NOOBS und Raspbian gewählt werden (siehe <https://www.raspberrypi.org/downloads/>). In unserem Fall wurde Raspbian, aufgrund der kompletten Funktionsbandbreite, gewählt. Zum Beschreiben der SD Karte werden die Programme SDFormatter und Win32 Disk Imager benötigt. Diese sind herunterzuladen und zu installieren. Als erstes ist SDFormatter zu öffnen und das Laufwerk der SD Karte auszuwählen. Es ist die Option „FORMAT SIZE ADJUSTMENT ON“ und im Anschluss auf Format zu klicken.

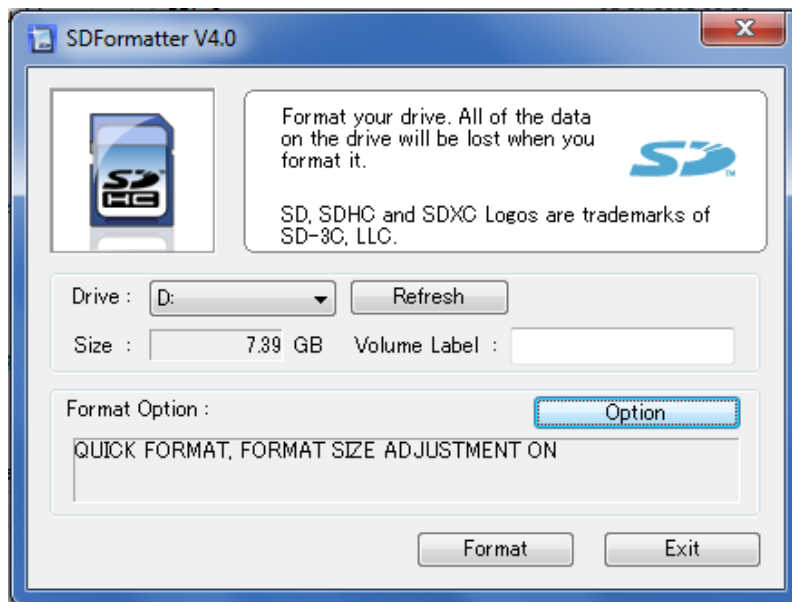


Abbildung 16: SD Karte formattieren

Nun ist die Karte entsprechend formatiert und das Raspbian Image kann übertragen werden mit WIN32 Disk Imager. Dort ist das zuvor heruntergeladene Image & das Laufwerk auszuwählen und auf Write zu drücken.

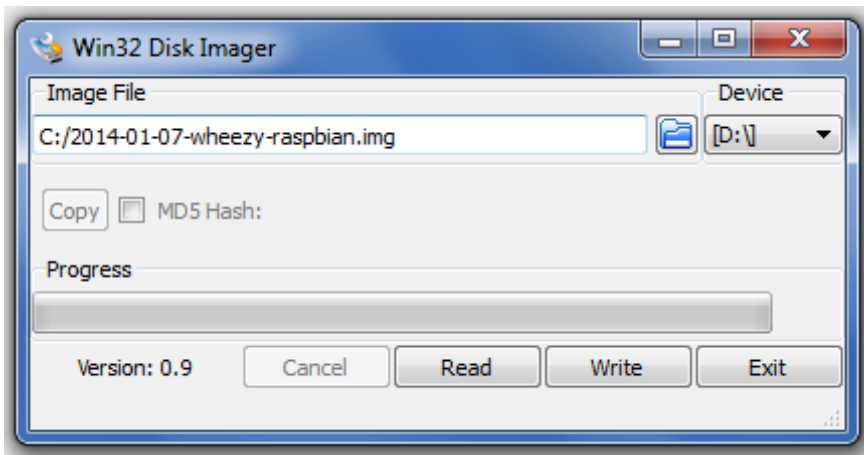


Abbildung 17: Raspbian Imager

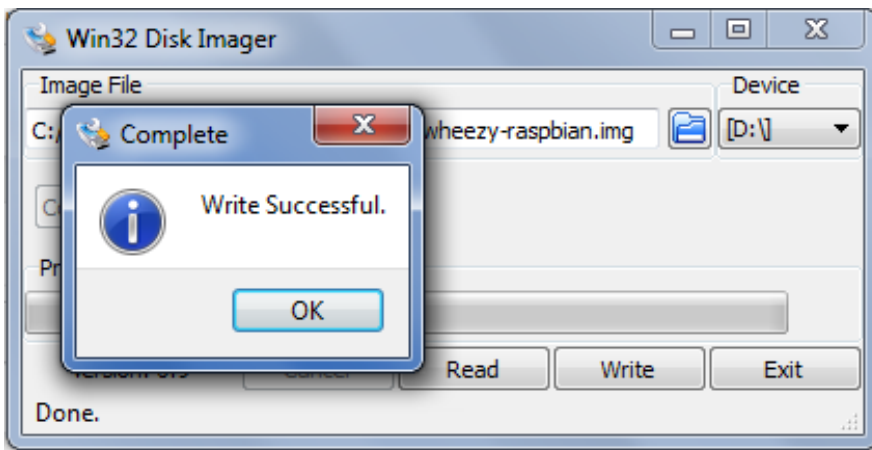


Abbildung 18: Raspbian erfolgreich übertragen

Nach Beenden dieses Vorgangs kann die SD Karte entnommen und in den Raspberry Pi eingesetzt werden. Nach dem Einsetzen kann der Pi durch Anlegen der Stromversorgung gestartet werden. Im Vorfeld ist der Monitor, die Tastatur & Maus und der Xbee Explorer mit Zigbee anzuschließen. Im Anschluss muss eine Internetverbindung, entweder per WLAN oder Ethernet, hergestellt werden. Eine Verbindung per WLAN kann mittels des Buttons in der oberen Taskleiste hergestellt werden.



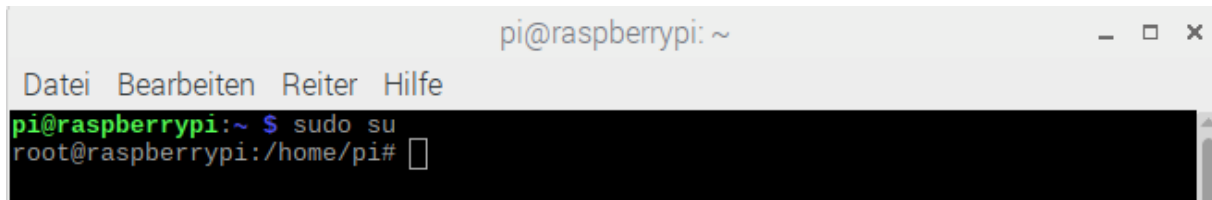
Abbildung 19: WLAN

Bei jedem Start des Raspberry Pi ist ein Update zu empfehlen. Dieses erfolgt über das Eingeben der folgenden Befehle im Terminal:

- sudo apt-get update
- sudo apt-get upgrade

Danach muss der Pi neugestartet werden, z.B. über das Terminal mit „sudo reboot“.

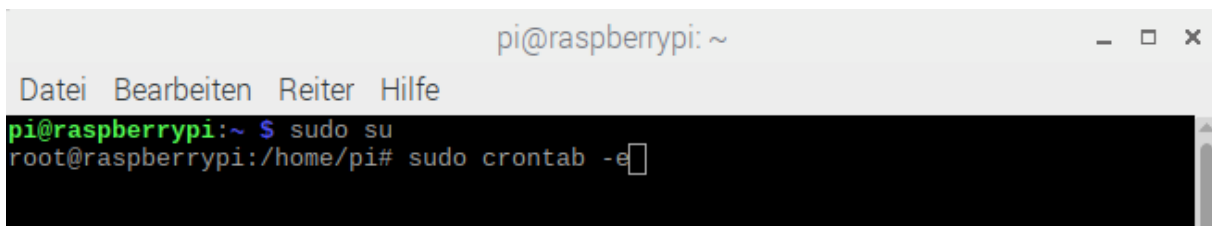
Damit der Raspberry Pi Zuverlässig auf lange Zeit läuft, ist ein regelmäßiger Neustart zu empfehlen. Hierfür ist im Terminal in den root-Modus mit „sudo su“ für die benötigten Rechte zu wechseln.



```
pi@raspberrypi: ~  
Datei Bearbeiten Reiter Hilfe  
pi@raspberrypi:~ $ sudo su  
root@raspberrypi:/home/pi#
```

Abbildung 20: Rechte wechseln

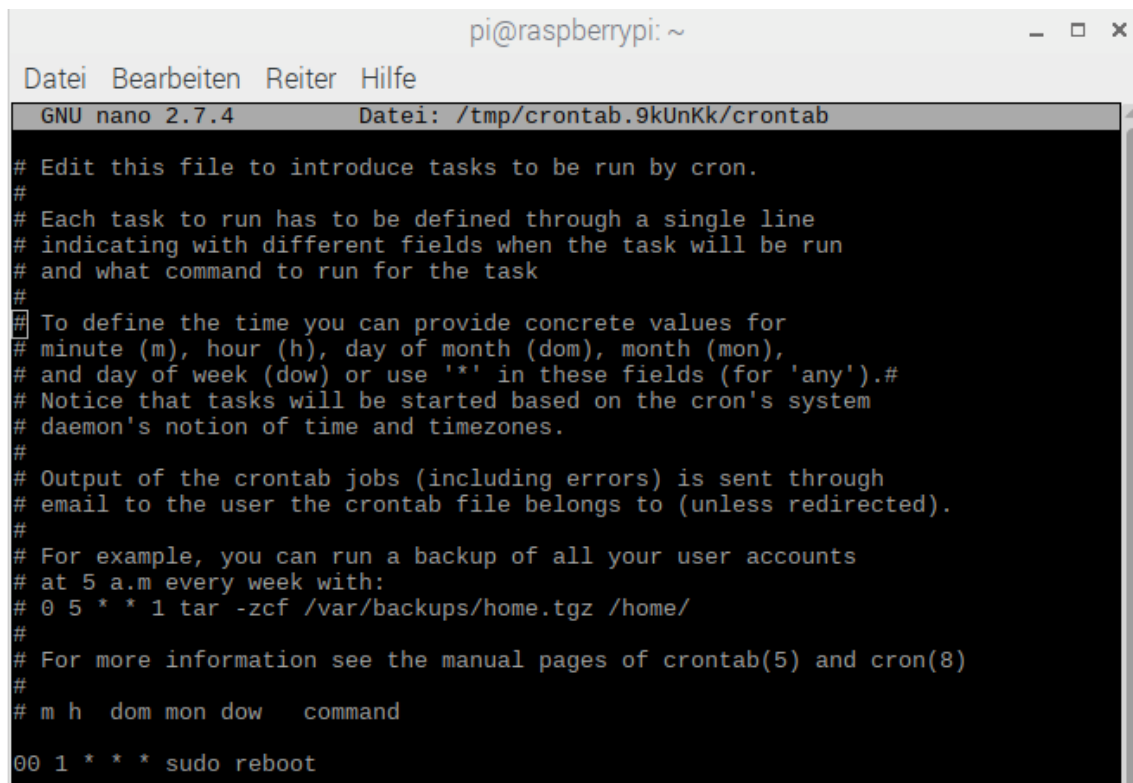
In der neuen Zeile ist der Befehl „sudo crontab -e“ einzugeben.



```
pi@raspberrypi: ~  
Datei Bearbeiten Reiter Hilfe  
pi@raspberrypi:~ $ sudo su  
root@raspberrypi:/home/pi# sudo crontab -e
```

Abbildung 21: Crontab öffnen

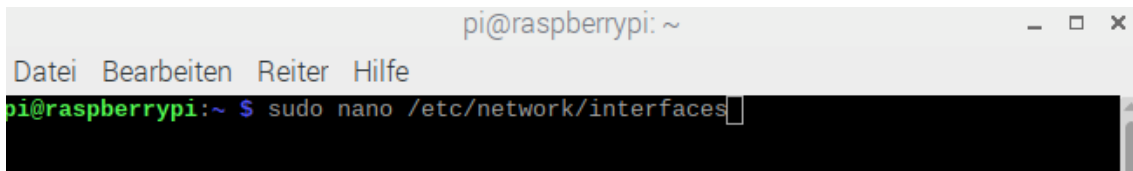
In dem neu erschienen Fenster kann die Zeit für den Neustart eingegeben werden. In unserem Fall wurde die Zeit 1 Uhr morgens gewählt. Der dafür benötigte Code ist „00 1 * * * sudo reboot“. Nach der Eingabe muss mit STRG+O & ENTER gespeichert und mit STRG+X das Fenster geschlossen werden.



```
pi@raspberrypi: ~  
Datei Bearbeiten Reiter Hilfe  
GNU nano 2.7.4 Datei: /tmp/crontab.9kUnKk/crontab  
# Edit this file to introduce tasks to be run by cron.  
#  
# Each task to run has to be defined through a single line  
# indicating with different fields when the task will be run  
# and what command to run for the task  
#  
# To define the time you can provide concrete values for  
# minute (m), hour (h), day of month (dom), month (mon),  
# and day of week (dow) or use '*' in these fields (for 'any').#  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow command  
00 1 * * * sudo reboot
```

Abbildung 22: Neustart einstellen

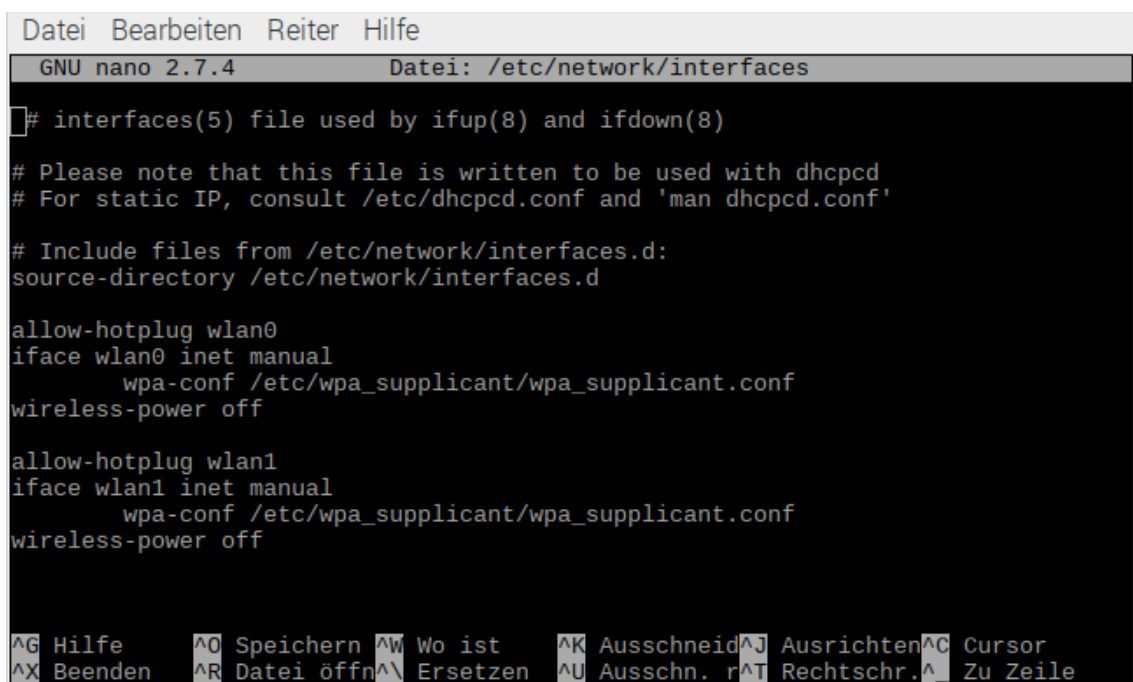
Wenn als Netzwerkschnittstelle Wlan gewählt wurde, ist nun noch der Stromsparmodus für die Wlan-Schnittstelle auszuschalten. Dies sorgt dafür, dass der Raspberry Pi ununterbrochen mit dem Internet verbunden ist. Dafür muss im Terminal „sudo nano /etc/network/interfaces“ für die Netzwerkeinstellungen eingegeben werden.



```
pi@raspberrypi: ~  
Datei Bearbeiten Reiter Hilfe  
pi@raspberrypi:~ $ sudo nano /etc/network/interfaces
```

Abbildung 23: WLAN Interface öffnen

In dem öffnenden Fenster ist der folgende in Abbildung 24 dargestellte Programmcode einzugeben, um den Stromsparmodus auszuschalten. Im Anschluss muss wieder mit STRG+O & STRG+X gespeichert und beendet werden.



```
Datei Bearbeiten Reiter Hilfe  
GNU nano 2.7.4 Datei: /etc/network/interfaces  
# interfaces(5) file used by ifup(8) and ifdown(8)  
  
# Please note that this file is written to be used with dhcpd  
# For static IP, consult /etc/dhcpd.conf and 'man dhcpd.conf'  
  
# Include files from /etc/network/interfaces.d:  
source-directory /etc/network/interfaces.d  
  
allow-hotplug wlan0  
iface wlan0 inet manual  
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf  
wireless-power off  
  
allow-hotplug wlan1  
iface wlan1 inet manual  
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf  
wireless-power off  
  
^G Hilfe      ^O Speichern ^W Wo ist      ^K Ausschneid ^J Ausrichten ^C Cursor  
^X Beenden    ^R Datei öffn ^\ Ersetzen   ^U Ausschn. r ^T Rechtschr. ^_ Zu Zeile
```

Abbildung 24: Energiesparmodus Code

Nun muss der Raspberry Pi noch einmal neugestartet werden. Danach kann mit dem eigentlichen Programm begonnen werden.

ThingSpeak

ThingSpeak ist eine kostenlos zur Verfügung stehende Internetdatenbank. Sie speichert die empfangenen Daten und stellt diese grafisch dar. Wir haben diesen Anbieter gewählt, da er leicht verständlich aufgebaut und zu bedienen ist. Außerdem werden zahlreiche Anpassungsmöglichkeiten und die Möglichkeiten zur Programmierung eigener Funktionen oder Grafiken gegeben. Weiter stehen für ThingSpeak mehrere Smartphoneapps zur Verfügung.

Als erstes muss dort (<https://thingspeak.com/>) jetzt ein Benutzerkonto erstellt werden. Im Anschluss ist ein neuer Channel zu erstellen.

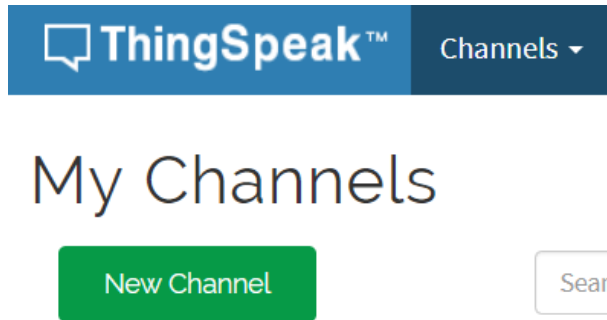


Abbildung 25: ThingSpeak neuen Channel erstellen

Jetzt kann dem Channel und den benötigten Feldern ein Name gegeben werden. Diese Angaben können jedoch später auch noch verändert werden. Im Anschluss muss der Channel noch gespeichert werden.

New Channel

Name	<input type="text" value="Bienenstockwaage"/>
Description	<input type="text"/>
Field 1	<input type="text" value="Gesamtgewicht"/> <input checked="" type="checkbox"/>
Field 2	<input type="text" value="Schwerpunkt X"/> <input checked="" type="checkbox"/>
Field 3	<input type="text" value="Schwerpunkt Y"/> <input checked="" type="checkbox"/>

Abbildung 26: Felder definieren

Über die Optionen der einzelnen Felder können diese im Anschluss angepasst werden in Hinsicht auf Beschriftung, Anzahl Datenpunkte, Achseinteilung etc.

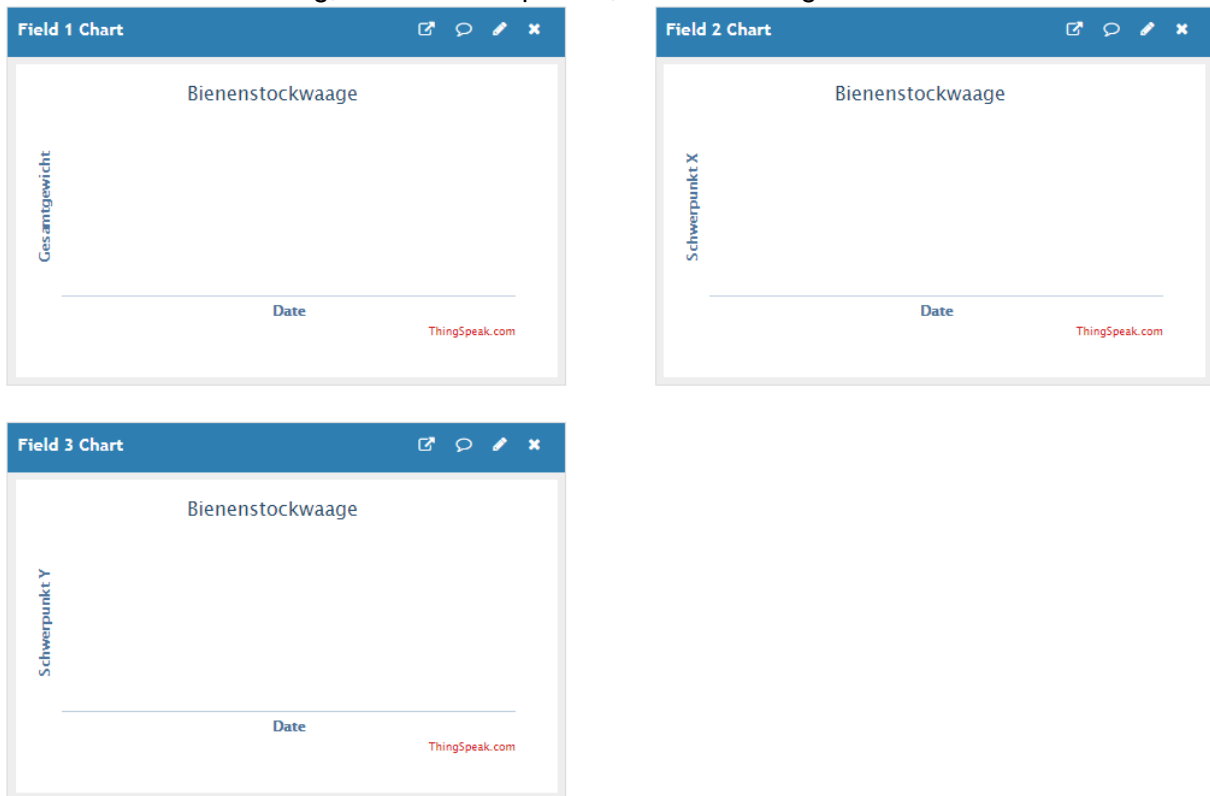


Abbildung 27: Übersicht über Felder

Der nächste Schritt ist das Notieren der „Channel ID“ und der „API Keys“. Diese werden dafür benötigt, dass das später erklärte Programm eine Verknüpfung zum eigenen Channel herstellen kann.

Private View Public View **Channel Settings** Sharing API Keys

Channel Settings

Percentage complete 30%

Channel ID XXXXXXXXXX

Name

Abbildung 28: Channel ID

Write API Key

Key

[Generate New Write API Key](#)

Read API Keys

Key

Abbildung 29: API KEY

In unserem Fall wurde noch ein eigenes Field programmiert, um den Schwerpunkt anzuzeigen. Der Programmcode im Arduino und Raspberry Pi ist dementsprechend aufgebaut, dass ein Wert für die Schwerpunktlage in x-Richtung und einer in y-Richtung übergeben wird. In ThingSpeak muss dafür eine MATLAB „Visualization“ erstellt werden.



Abbildung 30: MATLAB Visualization

Im Anschluss war ein leeres Template ohne Startercode auszuwählen und „Create“ auszuwählen.

Templates:

- Custom (no starter code)
- Create a filled area 2-D plot
- Create a 2-D line plot
- Create 2-D line plots with y-axes on both left and right side
- Create a correlated data plot
- Create a discrete sequence data plot

Examples: Sample code to visualize data

- Use a histogram to understand variation in data
- Visualize directional data with compass plot
- Use area plot to compare traffic data sets
- Compare temperature data from three different days
- Plot temperature and wind speed on two different y-axes
- Visualize correlation between temperature and humidity

[Create](#)

Abbildung 31: Thingspeak leere Visualisierung erstellen

Jetzt muss nur noch der folgende Code aus Abbildung 32 eingetragen werden. In unserem Fall wurde der Schwerpunkt in x-Richtung an das Field 2 und der Schwerpunkt in y-Richtung an das Field 3 übergeben. Sollte dieses auch so gewählt worden sein, kann der Programmcode ohne Änderungen übernommen werden.

Name

MATLAB Code

```
1 % Liest Schwerpunkt in X und Y-Richtung vom ThingSpeak Channel und Visualisiert die
2 % Beziehung dazwischen mit Hilfe eines SCATTER plots.
3
4 % Kanal 123456 enthält Daten der Bienenwaage, augenommen in der Hochschule Emden.
5 % Feld 2 Enthält den Schwerpunkt in X-Richtung in Millimetern, Feld 3 in Y-Richtung.
6
7 % Kanal ID zum lesen der Daten
8 LeseKanalID = 123456;
9 % Feld in dem der Schwerpunkt in X steht
10 SchwerpunktX = 2;
11 % Feld in dem der Schwerpunkt in Y steht
12 SchwerpunktY = 3;
13
14 % Kanal Lese API Key
15 readAPIKey = 'ABC123ABC123ABCD';
16
17 % Liest Schwerpunkt in X und Y
18
19 data = thingSpeakRead(LeseKanalID,'Fields',[SchwerpunktX SchwerpunktY], ...
20                       'NumPoints',1, ...
21                       'ReadKey',readAPIKey);
22 % Lese Schwerpunktsdaten
23 SchwerpunktX_Daten = data(:,1)/10;
24
25 SchwerpunktY_Daten = data(:,2)/10;
26
27 % Darstellung der Daten
28 scatter(SchwerpunktX_Daten*(-1),SchwerpunktY_Daten*(-1));
29 axis([-25 25 -25 25])
30 ylabel('Schwerpunkt in Y [cm]');
31 xlabel('Schwerpunkt in X [cm]');
```

Save and Run

Save

Abbildung 32: Programmcode Schwerpunktdiagramm

Danach ist nur noch auf „Save and Run“ zu klicken und das Diagramm ist erstellt.

Im Anschluss muss die ThingSpeak Datenbank im Raspberry Pi heruntergeladen werden. Dafür ist Terminal aufzurufen und „sudo pip install thingspeak“ einzugeben.

Programm

Nachdem alle Vorkehrungen getroffen wurden, kann mit dem Schreiben des eigentlichen Programms begonnen werden. Dazu muss das Programm Python 3 geöffnet werden.

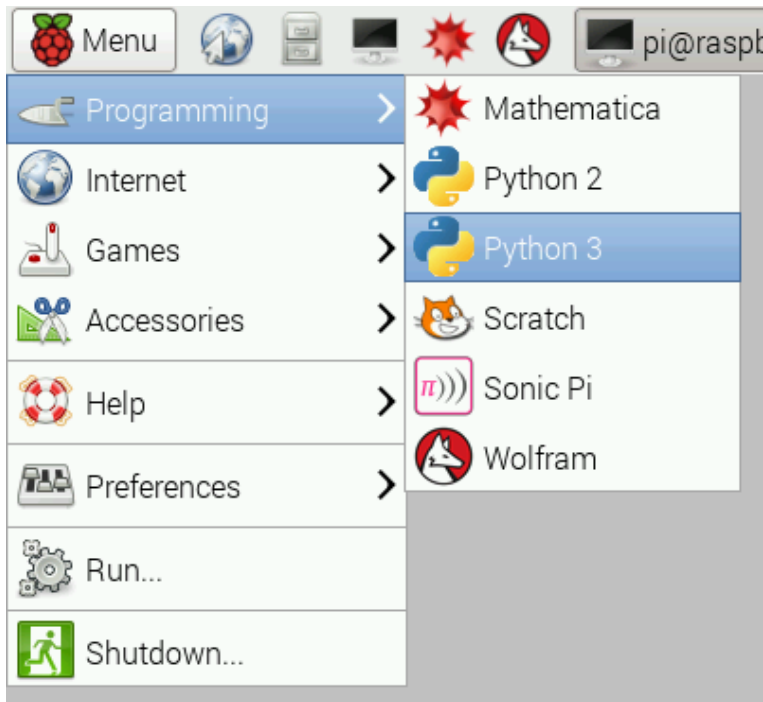


Abbildung 33: Python öffnen

Hier kann nun der folgende Programmcode für das Empfangen der Daten vom Arduino und der Weitergabe an ThingSpeak geschrieben werden. Die grün geschriebenen Texte hinter den Rauten stellen dabei Kommentare dar. Der Code kann vorerst identisch übernommen werden. Im Anschluss müssen folgende Punkte angepasst werden. Die persönliche Channel ID und der write bzw. read API-Key muss eingetragen werden an die davor vorgesehenen Stellen. Ggfs. muss die Baudrate angepasst werden, je nachdem wie der Zigbee konfiguriert wurde. In unserem Fall werden vom Arduino drei Werte (Gesamtgewicht, Schwerpunkt in X- und Schwerpunkt in Y-Richtung). Diese Werte können je nach Anforderung angepasst oder ergänzt werden nach dem vorliegenden Muster. Kongruent dazu ist die Zeile „response = channel.update({'field1': Gesamtgewicht, 'field2': SchwerpunktX, 'field3': SchwerpunktY})“ anzupassen mit den definierten Begriffen. Der übrige Programmcode kann unverändert bleiben.

```
#!/usr/bin/env python
#Datenbanken importieren
import thingspeak
import time
import serial

#Hier Channel-ID vom Thingspeak Channel eintragen
channel_id = 123456
#Hier Write-Key vom Thingspeak Channel eintragen
write_key = 'ABCDEFGHJIJ123456'
#Hier Read-ID vom Thingspeak Channel eintragen
read_key = ' ABCDEFGHJIJ123456'

#Serielle Schnittstelle mit 9600 Baudrate fuer Zigbee oeffnen
ser = serial.Serial('/dev/ttyUSB0', 9600)
ser.isOpen()
#Wartezeit zum oeffnen der Seriellen Schnittstelle
```



```

time.sleep(5)

print("Uebertragung laeuft")
#Schreiben des Mess- bzw. Uebertragungsprogramms fuer spaeteren Aufruf
def measure(channel):
    try:
        while True:
            #Daten vom Arduino per Zigbee empfangen
            gewicht=ser.readline()
            try:
                #Empfangene Daten in string umwandeln
                gewicht=gewicht.decode()
            except:
                print("Decodierungsfehler")

            #Letzten 3 Zeichen vom string entfernen (Zeilenumbruchszeichen
etc.)
            gewicht=gewicht[:len(gewicht)-3]

            try:
                #string splitten bei Semikolon
                werte=gewicht.split(';')
                #Geschriebene Arrays zuordnen
                Gesamtgewicht=werte[0]
                SchwerpunktX=werte[1]
                SchwerpunktY=werte[2]

            except:
                #Wenn splitten nicht moeglich
                print("Typconvertierungsfehler")

            #Erhaltene Daten in ein Thingspeak Feld eintragen
            response = channel.update({'field1': Gesamtgewicht, 'field2':
SchwerpunktX, 'field3': SchwerpunktY})

        except:
            print("Verbindung fehlgeschlagen")
            ser.close() # Serielle Schnittstelle schliessen

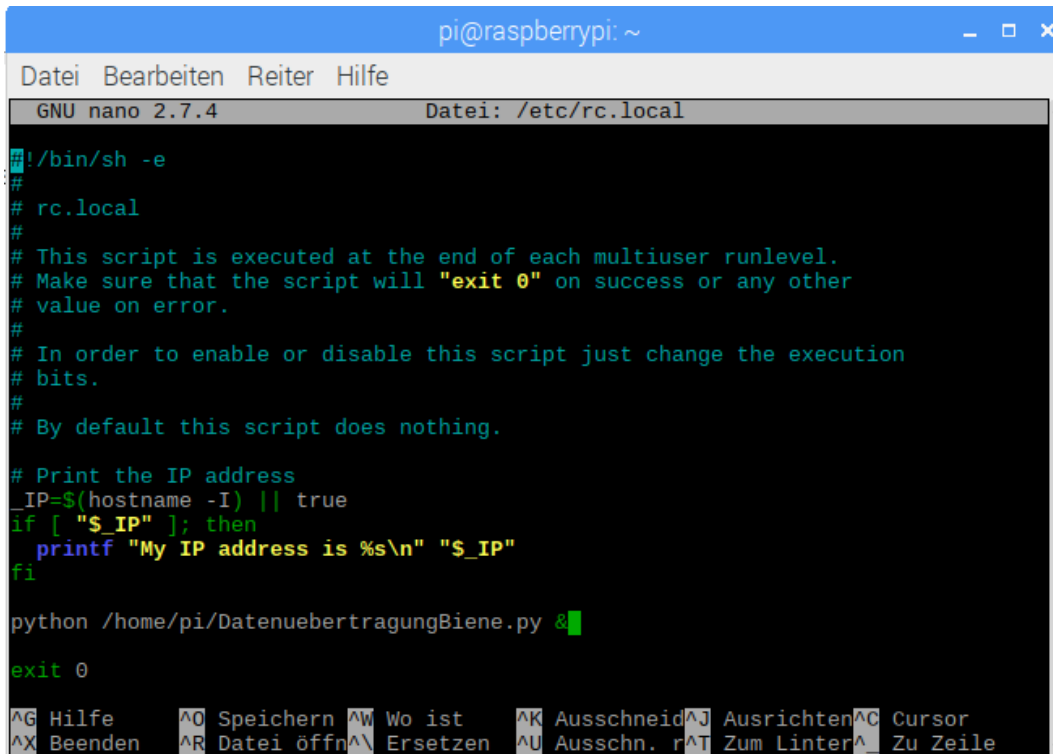
#Hauptprogramm starten
if __name__ == "__main__":
    #Verbindung mit Thingspeak Channel herstellen
    channel = thingspeak.Channel(id=channel_id, write_key=write_key,
api_key=read_key)
    while True:
        measure(channel) #Mess- bzw. Uebertragungsprogramm
        aufrufen
        time.sleep(15) #Uebertragung alle 15sec durchfuehren,
        solange neue Messwerte ankommen

```

Nach Schreiben und Anpassen des Programmcodes muss die Datei gespeichert werden unter dem Verzeichnis „/home/pi“. Wir haben als Benennung „DateneubertragungBiene.py“ gewählt. Danach kann die Datei geschlossen werden.

Weiter vorne im Bericht wurde erklärt wie der Pi täglich neu rebootet. Damit das Programm dabei immer wieder neu aufgerufen wird, muss dieses jetzt noch mit dem Autostart verknüpft werden.

Dafür muss als erstes die Autostartdatei im Terminal mit „sudo nano /etc/rc.local“ geöffnet werden. Dort muss der folgende Text aus Abbildung 34 eingetragen werden. Im Anschluss ist wieder mit STRG-O zu speichern und mit STRG-X das Fenster zu schließen.



```
pi@raspberrypi: ~
Datei Bearbeiten Reiter Hilfe
GNU nano 2.7.4 Datei: /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
#
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
python /home/pi/DatenuebertragungBiene.py &
exit 0
^G Hilfe      ^O Speichern  ^W Wo ist     ^K Ausschneid^J Ausrichten^C Cursor
^X Beenden    ^R Datei öffn^E Ersetzen   ^U Ausschn. r^T Zum Linter^_ Zu Zeile
```

Abbildung 34: Autostart einstellen

Falls ein anderer Dateiname oder Speicherort gewählt wurde, ist dies entsprechend anzupassen. Zuletzt sollte der Raspberry Pi noch einmal neugestartet werden. Ab jetzt kann der Pi auch ohne Monitor, Maus & Tastatur betrieben werden. Auch wenn er vom Strom getrennt und neu verbunden wird, startet er selbstständig und ruft das Programm zur Datenübertragung selbstständig auf.

Zusammenbau

Der erste Schritt ist das Fertigen der Distanzstücke zwischen Wägezelle und der unteren Holzplatte. In unserem Fall wurde der Werkstoff Aluminium gewählt. Es kann aber ebenso Holz oder ähnliches verwendet werden. Diese sind dann zuzuschneiden und mit Bohrungen entsprechend der Wägezellen & Holzplatte zu versehen. In die Bohrungen für die Verbindung mit der Holzplatte wurden Gewinde eingebracht. Die Größe der Distanzstücke ist hierbei beliebig. Es sollte allerdings darauf geachtet werden, dass die Wägezelle hierdurch nicht gestört wird und der Abstand der Gewindebohrung der Wägezelle eingehalten wird.



Abbildung 35: Gewinde schneiden



Abbildung 36: Distanzstück



Abbildung 37: Distanzstück und Schrauben

Ebenfalls gebohrt werden muss jetzt die untere Siebdruckplatte, passend zu den Gewindebohrungen der Distanzstücke. Außerdem musste eine Öffnung gesägt werden, um das Gehäuse der Powerbank im späteren Betrieb von unten entnehmen zu können.



Abbildung 38: Ausgesägte Öffnung Holzplatte

Danach können die Distanzstücke mit den einzelnen Wägezellen und der Holzplatte verschraubt werden.

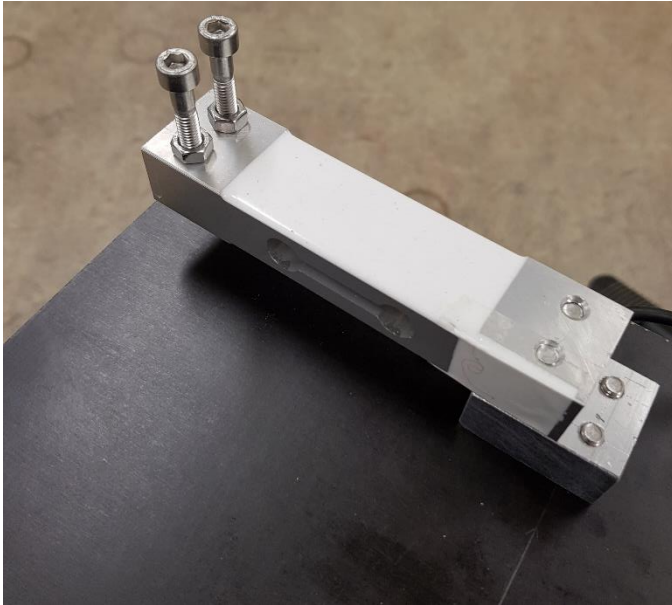


Abbildung 38: Wägezelle auf Holzplatte



Abbildung 39: Wägezelle mit Distanzstück

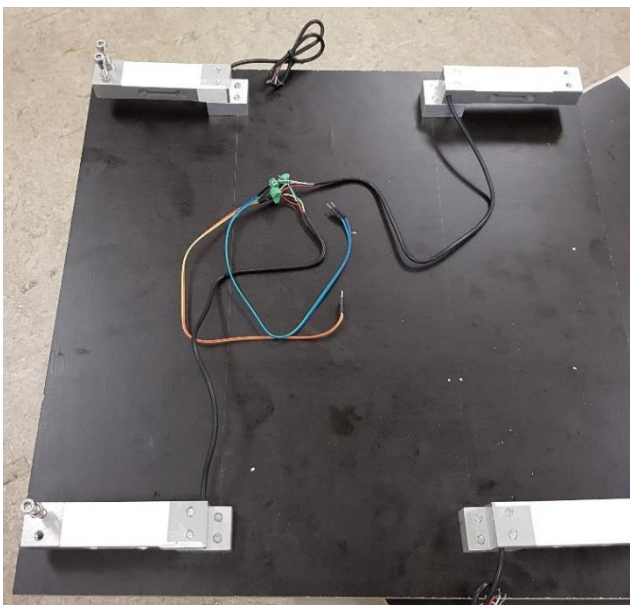


Abbildung 40: Vier Wägezellen auf Holzplatte

Zur Vermeidung innerer Spannungen, welche die Messungen verfälschen würden, wurde die obere Siebdruckplatte nicht verschraubt, sondern nur lose aufgelegt. Dafür waren in die Wägezellen Zylinderkopfschrauben mit Innensechskant einzuschrauben mit dem benötigten Abstand. Diese wurden in der passenden Höhe mithilfe einer Mutter gekontert. Verwendet wurden Edelstahlschrauben- & muttern für die Witterungsbeständigkeit. Nachdem bei der ersten Schraube eine Höhe festgelegt wurde, war das gleiche Höhenniveau mithilfe einer Wasserwaage auf die anderen Schrauben zu übertragen.



Abbildung 41: Einstellen Höhe

Im Anschluss mussten die Löcher in der oberen Platte gebohrt werden. Diese wurden vom Durchmesser größer als die Schrauben gebohrt, damit keine inneren Spannungen entstehen können. Ebenso wurde diese als Sackloch ausgeführt, um die Platte auflegen zu können. Zum exakten Anzeichnen der Bohrungen wurde die Konstruktion im Gesamten gedreht.

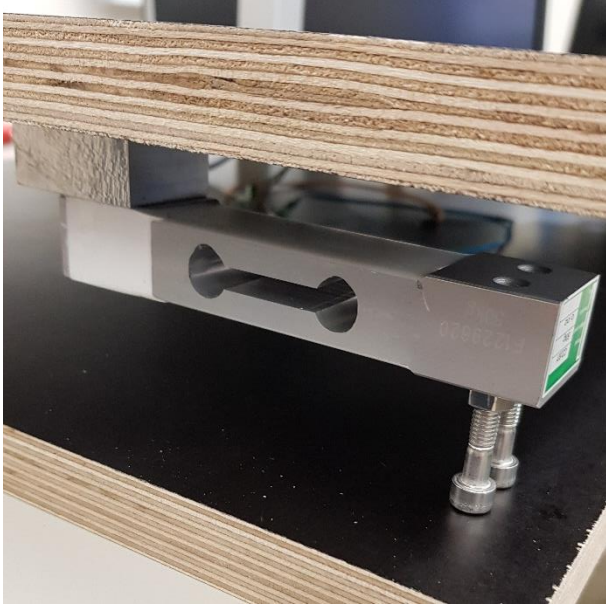


Abbildung 42: Makierung für Bohrung

Jetzt konnte das Gehäuse für den Arduino mit den Kabelverschraubungen versehen werden und auf die untere Siebdruckplatte geschraubt werden. Anschließend war der Arduino mit den Wägezellen zu verbinden. Dabei ist ratsam die einzelnen Kabel miteinander zu verlöten und anschließend mit einem Schrumpfschlauch zu versehen.

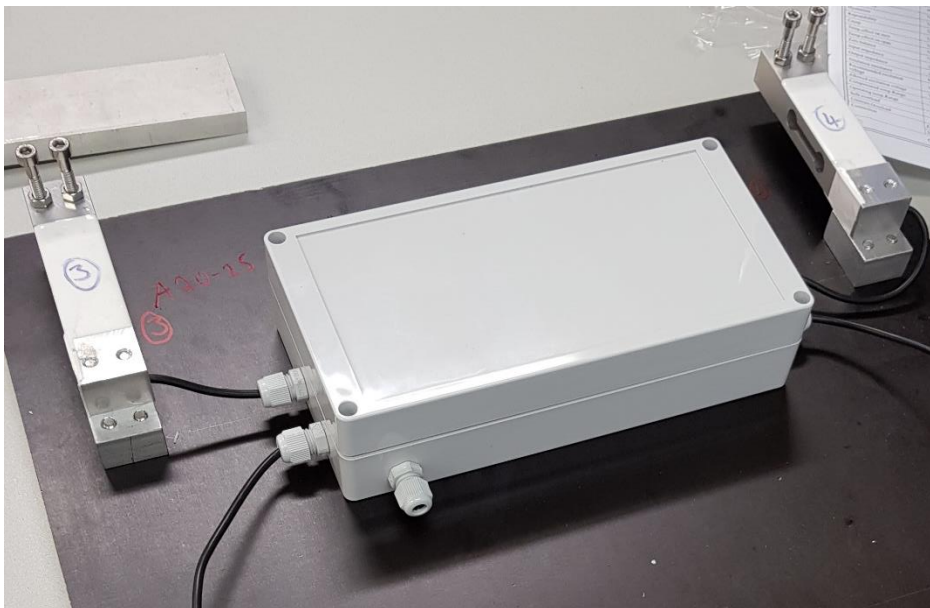


Abbildung 43: Verkabeltes Gehäuse

Zur Stromversorgung wurde eine Powerbank gewählt, welche ebenfalls in einem Gehäuse Platz findet. Um diese mit dem Arduino verbinden zu können, wurde ein wasserfestes Kabel und eine wasserfeste Buchse gewählt. Das Kabel wurde an dem offenen Ende mit einem Anschlusskabel 2,1 mm verlötet und durch die zugehörige Kabeldurchführung geführt. Das Anschlusskabel konnte dann mit dem Arduino verbunden werden.



Abbildung 44: Stromversorgung löten

Die Buchse wurde in das Gehäuse für die Powerbank eingeschraubt. Auf der Innenseite war dann das USB-Kabel mit den benötigten Adern an den jeweiligen Pins der Buchse zu verlöten. Das USB-Kabel konnte dann mit der Powerbank verbunden werden.

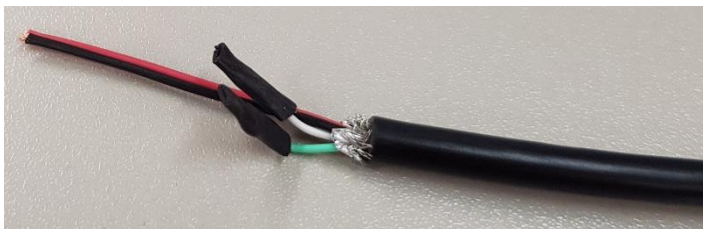


Abbildung 45: USB Kabel offene Enden



Abbildung 46: Isolation der Lötstelle

Das Gehäuse mit der Powerbank musste nicht verschraubt werden, es soll lediglich auf der unteren Holzplatte aufliegen.

Ergebnis



Abbildung 47: Waage ohne Deckel

Die Waage ist nun über die Powerbank strombetrieben und voll funktionsfähig.

Der Deckel schützt die Waage zudem vor äußeren Witterungen, wie zum Beispiel Regen. Sollte es dennoch dazu kommen, dass Feuchtigkeit zwischen die Holzplatten gelangt, was selten zu verhindern ist, so sind alle Elektronikkomponenten dagegen durch die Kunststoffgehäuse geschützt. Die Stromverbindung ist flexibel abschraubbar und nach IP68 geschützt.

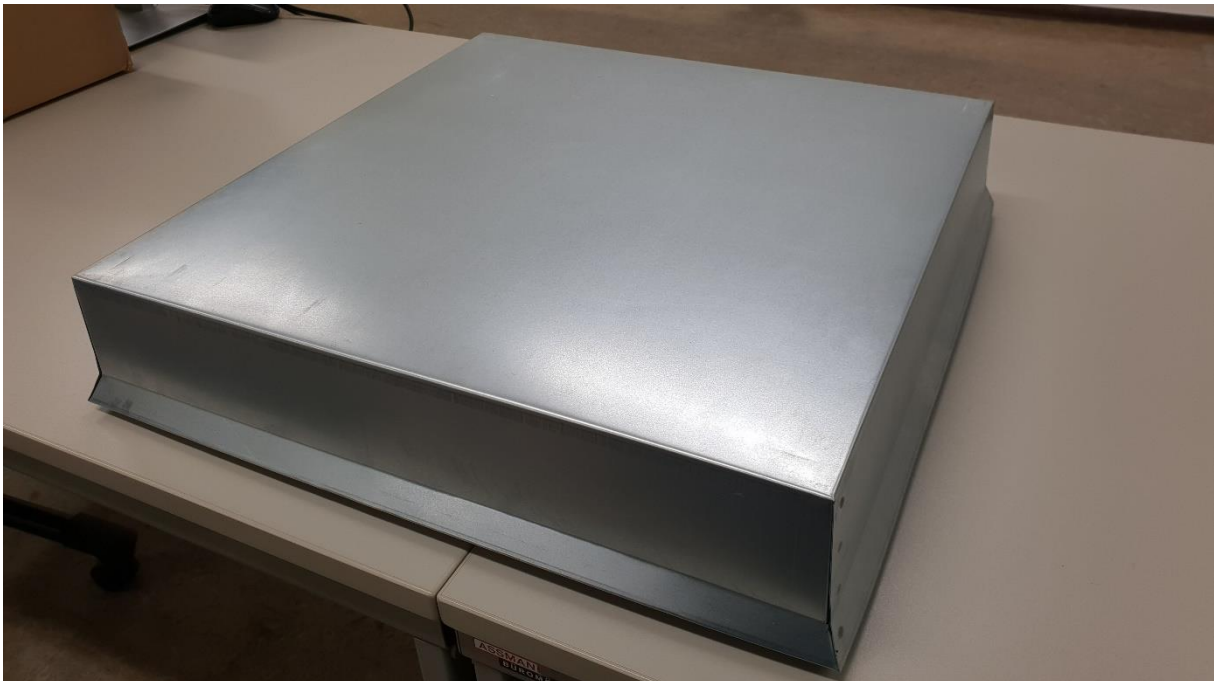


Abbildung 48: Waage mit Deckel

Der Arduino liest alle 10 Minuten die Werte der Wägezellen ein, rechnet sie in ein Gesamtgewicht und einen Schwerpunkt in X und in Y um und sendet diese über den XBee an den Raspberry Pi.

Dieser sendet, wenn er einen Wert erhält, diese nun per WLAN an die Webapp ThingSpeak.com. Über die Seite ThingSpeak.com kann man von überall aus auf seinen Channel zugreifen und die Messwerte auswerten.

Das Gesamtgewicht wird über den Messzeitraum dargestellt, wobei der Channel beliebig wieder auf null gesetzt werden kann. Der Schwerpunkt wird in einem Scatter angezeigt, wobei nur der aktuelle Schwerpunkt angezeigt wird.

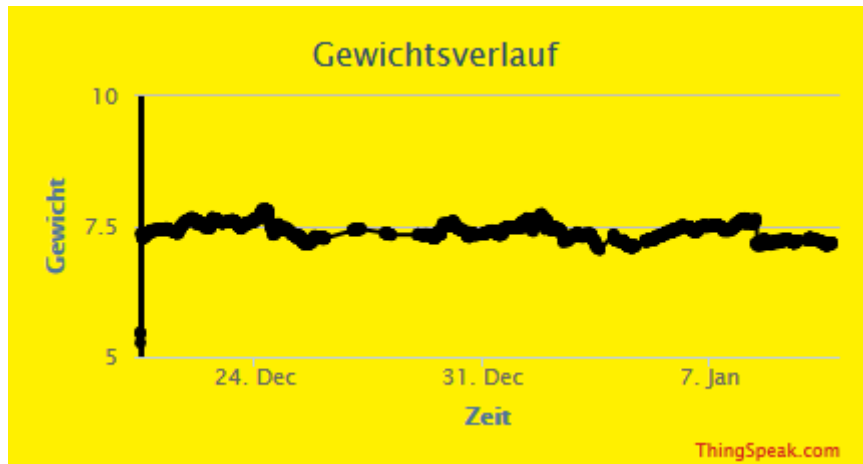


Abbildung 49: Gewichtsverlauf

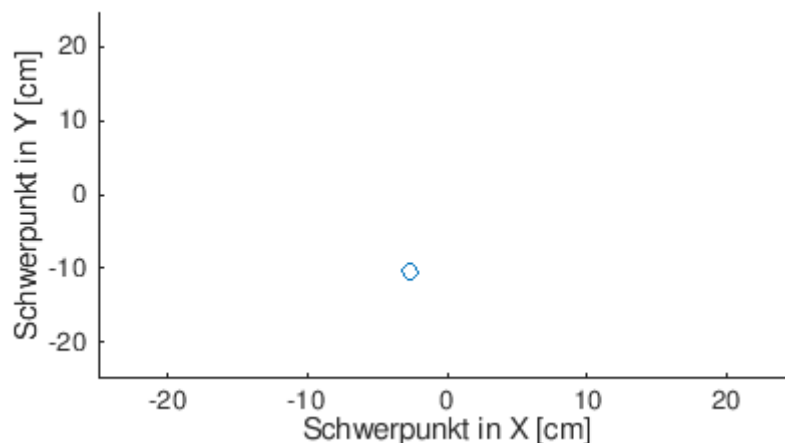


Abbildung 50: Schwerpunkt in X und Y

Hiermit kann nun eine Analyse der Beute durchgeführt werden und es kann anhand der Gewichtsanalyse ein präziser Eingriff erfolgen. So kann ein günstiger Zeitpunkt zur Entnahme des Honigs gefunden werden, sowie frühzeitig festgestellt werden, wann sich das Bienenvolk ggf. teilt oder ähnliches.

Quellen

<https://www.watterott.com/de/Arduino-Uno>

<https://www.elektronik-kompodium.de/sites/com/1904221.htm>

<https://tutorials-raspberrypi.de/>

<https://www.raspberrypi.org/downloads/raspbian/>

https://www.sdcard.org/downloads/formatter_4/

https://www.chip.de/downloads/Win32-Disk-Imager_46121030.html

<https://thingspeak.com/>

<https://www.exp-tech.de/>

<https://www.arduino.cc/>

<https://github.com/bogde/HX711>

<https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>

https://www.mouser.com/ds/2/813/hx711_english-1022875.pdf

<https://www.roboter-bausatz.de/204/hx711-24-bit-gewichtssensor>