skala.

Technical Project Description

Story

We weigh rebar deliveries on construction sites and match them with the order lists of the construction company. The following tutorial will guide you step-by-step through our solution.

Used Components

Hardware-Components:

- Arduino ESP 8266 Wifi Board
- Load Cell Amplifier HX711
- 4x20 LCD Display HD44780
- Strain gauge (DMS) inside crane scale body
- Crane scale body Hilitand 660Lb (Up to 300 kg structure load)
- Powerbank (3350 mAh)
- Jumper wires
- 2x 10k Ohm resistors;

See Appendix for specific component links

Software dependencies:

- SQL-Database
- Docker
- Node.Js + Node-Package Manager

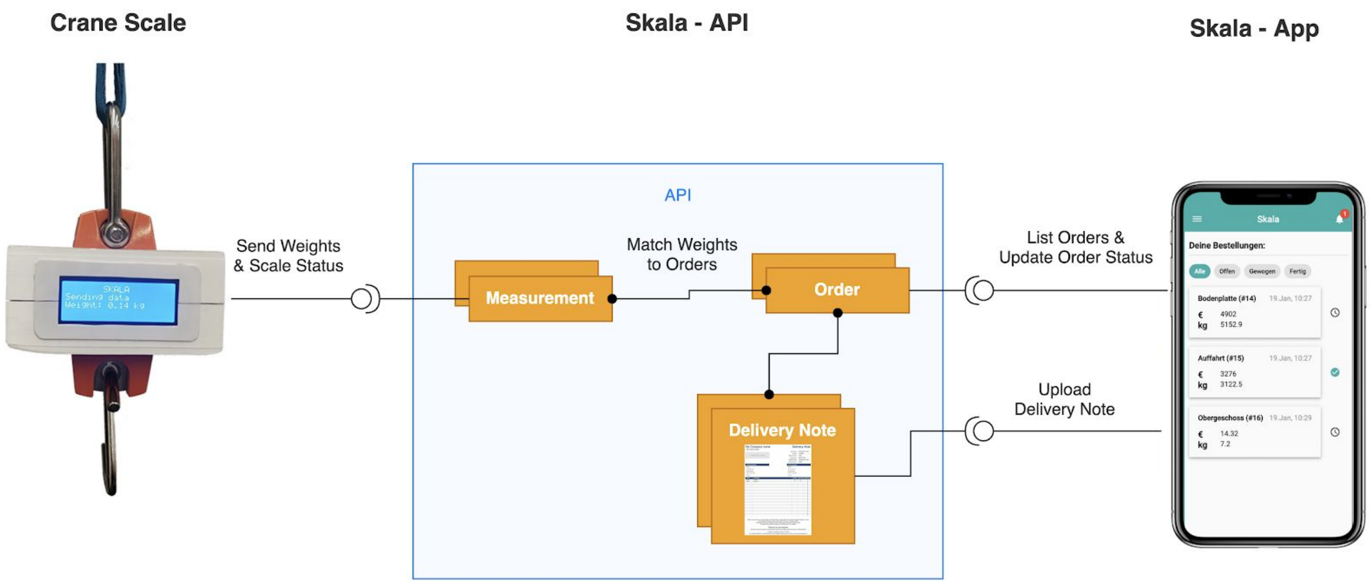
Introduction

Construction site managers get truck deliveries of rebar, which often miss rebar. As the deliveries contain countless (often more than 1000) pieces of rebar they can not count them and have to assume that the deliveries are complete.

Our solution weighs the rebar positions by attaching a scale to the crane that is used for the unloading of the trucks. The measured weights are then matched with the order lists. The resulting status (complete/incomplete) for the orders is displayed in a mobile application on mobile and desktop devices.

Our solution consists of the three components seen below:

1) A scale that we attach to the crane`s hook
2) An API that stores the measurements, orders and delivery notes
3) A mobile/web application that displays the orders and allows to update their status

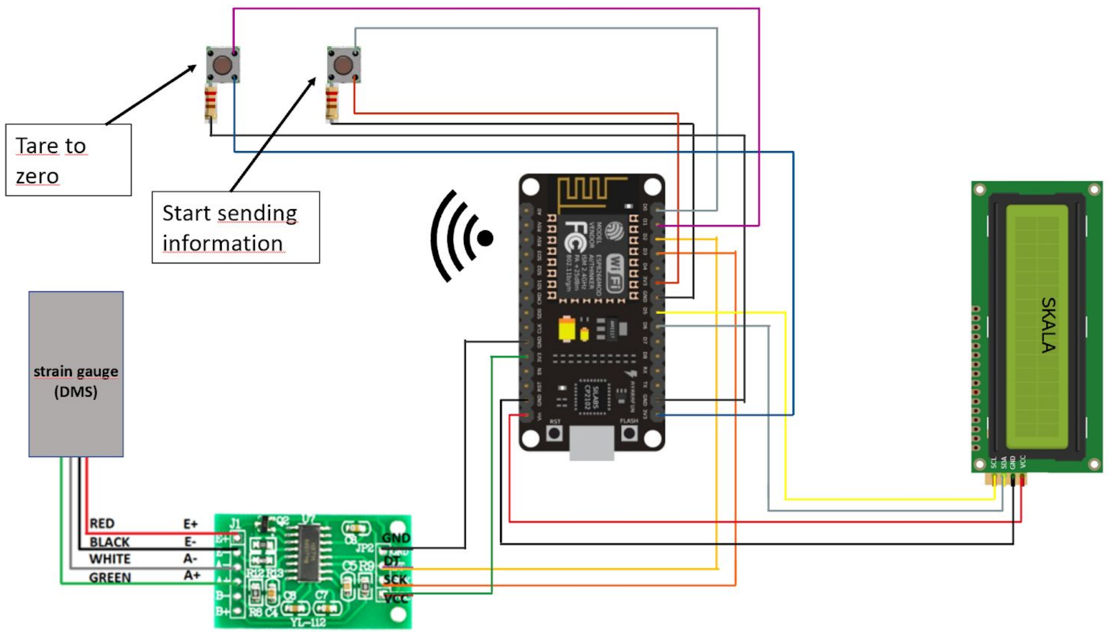


Solution

In the following we will go step-by-step through the implementation of our three components. The source-code is provided for all parts and a list of the exact hardware components is given at the top.

Hardware

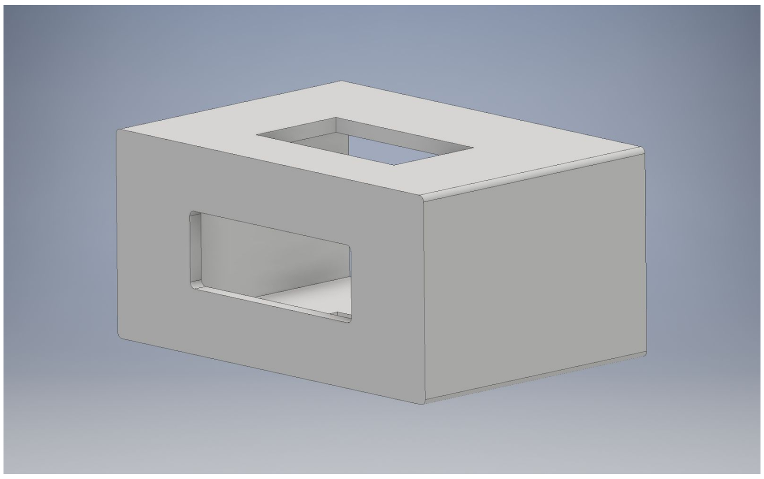
Schematics

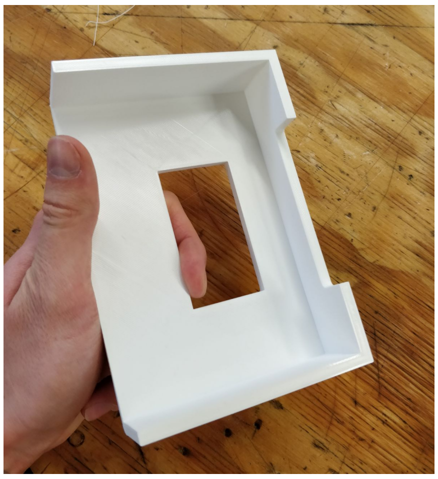


1. Explanation

   To measure the weight with our smart scale, we bought a crane scale body with a possible structure load of up to 300 kg. We connected the strain gauge (Dehnungsmessstreifen/DMS), which changes its electrical resistance while being deformed. We use this change in the measured voltage, to compute the weight.

   To convert the resulting weak analog signal from the DMS to a stronger digital signal we use the HX711 load cell amplifier. The load cell amplifier as well as two buttons are connected to the GPIO Pins 0-3, 3V Output and to GND (see schematics for details). For displaying the measured weights and the status of the scale we connected the HD44780 display to the GPIO Pins 5-6, VCC and Ground. The measured voltage of the DMS is then used to show the weight on the display and the Arduino Wifi Board ESP8266 additionally sends those measurements via Wifi to our Skala-API.


Inventor file


printed case part

2. Case

   All components and electronic parts were now integrated into a specially modelled shell using Autodesk Inventor, which was produced in three parts. The model is available as well in the repository at https://github.com/tpfeifle/skala/blob/master/scale/Case.stp . We 3D printed the model with a Ultimaker 2+ , which took around 24 hours to print . The cover was then glued and provided with a frame as well as holes for control buttons.

3. Code

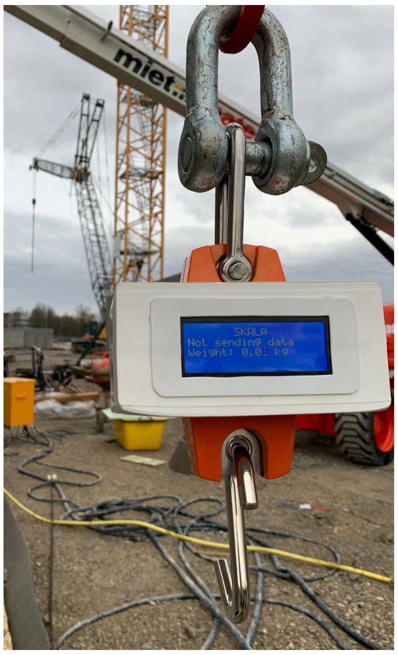
   ESP8266 runs code written in Arduino language. In order to upload the code to the board, one should download Arduino IDE. In the preferences menu of the Arduino IDE, "*http://arduino.esp8266.com/stable/package_esp8266com_index.json*" should be added to Additional Boards Manager URLs. After that, one can download ESP8266 package in the Boards Manager. Finally, ESP8266 board will appear in Tools/Board menu, so that one can choose it and upload the code to it.

   Libraries needed to be installed to run the code:

   1. HX711 - for working with weight sensor
   2. ESP8266WiFi and ESP8266HTTPClient - for connecting to WiFi and sending HTTP requests to the server.
   3. LiquidCrystal_I2C and Wire - for working with LCD display

You can find the code in the repository at https://github.com/tpfeifle/skala/blob/master/scale/arduino_script.ino. To adjust this script to a different scale one has to change the following:

1. Scale is calibrated using `calibration_factor` which is specific for each weight measure sensor (-12500 in our case). This variable is located at line #35 and has to be set up for each sensor individually.
2. A couple of thresholds are set.
    a. The Microcontroller sends the weight from scale only when the weight on the scale is changed by more than a specific value. In our case we choose `0.01` in order to ignore very small changes.
    b. The Microcontroller sends the weight from scale if it has been unchanged for a specific amount of seconds (5 seconds in our case). This has been done to prevent incorrect values to be sent. For example, during acceleration or during attachment of the weight to the scale, the weight on it might increase/decrease. This threshold of 5 seconds prevents this incorrect weight from being sent.



Our scale attached to a crane

Skala-API

This is the central piece of the Skala platform (see Components Diagram) and exposes REST-endpoints for the CRUD(**C**reate, **R**ead, **U**pdate, **D**elete)-operations on the orders, delivery notes and measurements. It also maps the stream of measured weights to their respective orders.

Dependency-Setup

We use the asynchronous event driven JavaScript runtime Node.js. This makes it easy to build scalable network applications. Specifically we use the framework Nest.js which provides many features to help structure the application and to provide HTTP utilities and middlewares. To run the API first install the following dependencies:

1. Install nodejs (https://nodejs.org/en/download/)

   e.g. `$ brew install node`

2. Install Nestjs

   ```
   $ npm install -g @nestjs/cli.
   ```

For data storage we use a simple SQL database. In this demo setup we use a Docker-instance:

1. Install docker desktop (https://docs.docker.com/install/)
2. Run the mysql docker instance on port 3306, create a database “techchallenge” and set your root-password

   ```
   $ docker run -p 3306:3306 --name some-mysql -e
   MYSQL_ROOT_PASSWORD=CHOOSE_YOUR_PASSWORD -e
   MYSQL_DATABASE=techchallenge -d mysql:5.7
   ```

On top of the SQL database we use the object relation mapper (ORM) *TypeORM*. It stores each entity in a file, specifying e.g. relations simply by annotations. It is automatically installed with the Node-Package-Manger (NPM) once you build the application

Skala-API-Setup

After installing the dependencies and setting up the database we can now build the Skala-API.

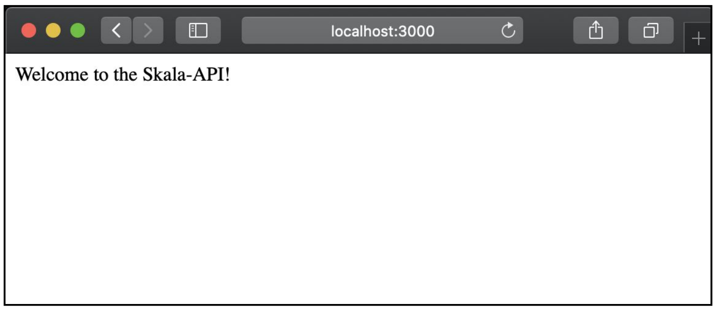
The source-code is available on Github (https://github.com/tpfeifle/skala/backend).

Follow these steps to get it running on your machine:

```
1. $ git clone https://github.com/tpfeifle/skala
2. $ cd skala/backend
3. $ export DB_TECHCHALLENGE_PASSWORD="CHOOSE_YOUR_PASSWORD"
4. $ npm install
5. $ nest start
```

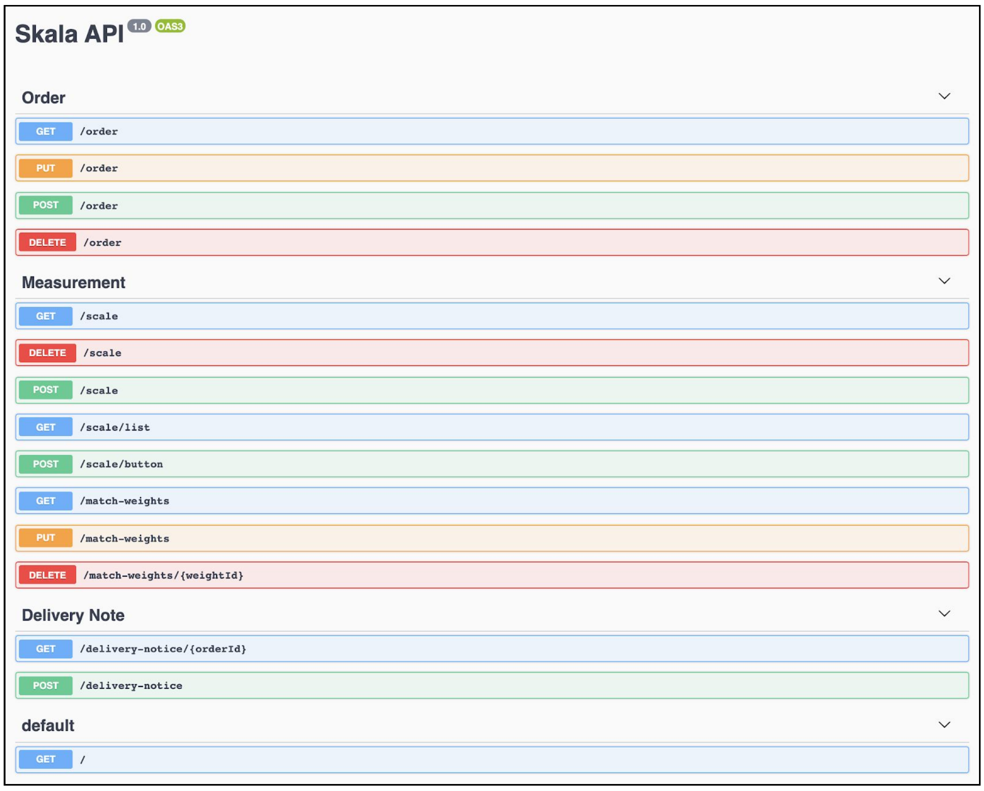
After those steps the REST-API will be available on your machine on port 3000

Check this by opening http://localhost:3000 in your browser. You should be greeted with the following screen:



Welcome to the Skala-API!

To see which endpoints are now available open http://localhost:3000/api and you will get the following list. This is the documentation of the API and how to request/change the stored data:


Skala API 1.0 OAS3

Order

GET /order
PUT /order
POST /order
DELETE /order

Measurement

GET /scale
DELETE /scale
POST /scale
GET /scale/list
POST /scale/button
GET /match-weights
PUT /match-weights
DELETE /match-weights/{weightId}

Delivery Note

GET /delivery-notice/{orderId}
POST /delivery-notice

default

GET /

Now you have setup all the backend that you require for this project. Next we will show how to compile and serve the application.

Mobile-/Web-Application

Our application is built with the framework Angular, which brings many useful features for interactive applications as client side routing, dependency injection and data bindings.

This part requires that you have already installed the API as described above.

The source-code is available in the same Github repository in the folder app (https://github.com/tpfeifle/skala/app). Just navigate to the folder app

Steps:

1. `$ cd skala/app`
2. Install Angular

   ```
   $ npm install -g @angular/cli
   ```

3. Install the dependencies

   ```
   $ npm install
   ```

4. Build the application and serve it locally on port 4200

   ```
   $ ng serve --host=0.0.0.0
   ```

After running those steps the application will be available on your machine on port 4200

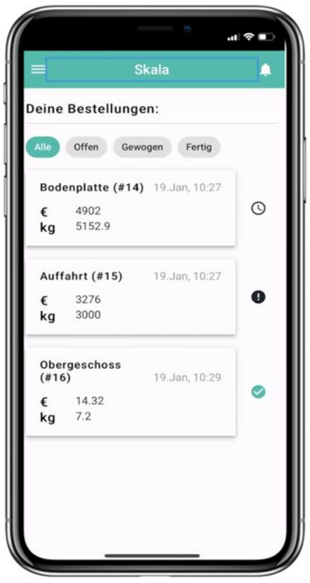
Check this by opening http://localhost:4200 in your browser. You should be greeted with a screen similar to the screen 1) below.

The orders can be created with the REST-API to match the content seen above

While you can use the app on a desktop device, you can also open this page on mobile devices and will see the mobile optimized views there.

The mobile application has the following screens and features:

1) Overview of orders and their status
2) List of unmatched weights:
    a) Assign unmatched weights to incomplete order
    b) Call supplier
    c) Delete unmatched weight
3) Detail view of order
    a) Upload delivery note
    b) Change status of order



Skala

Deine Bestellungen:

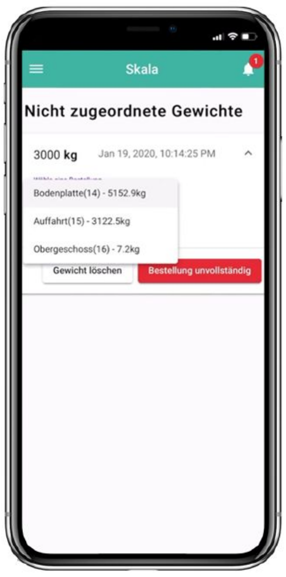
Alle | Offen | Gewogen | Fertig

Bodenplatte (#14) 19.Jan, 10:27
€ 4902
kg 5152.9

Auffahrt (#15) 19.Jan, 10:27
€ 3276
kg 3000

Obergeschoss (#16) 19.Jan, 10:29
€ 14.32
kg 7.2

Screen 1)



Skala

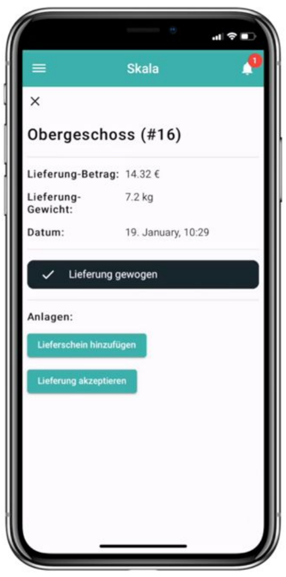
Nicht zugeordnete Gewichte

3000 kg Jan 19, 2020, 10:14:25 PM

Bodenplatte(14) - 5152.9kg
Auffahrt(15) - 3122.5kg
Obergeschoss(16) - 7.2kg

Gewicht löschen | Bestellung unvollständig

Screen 2)



Skala

Obergeschoss (#16)

Lieferung-Betrag: 14.32 €
Lieferung-Gewicht: 7.2 kg
Datum: 19. January, 10:29

Lieferung gewogen

Anlagen:

Lieferschein hinzufügen
Lieferung akzeptieren

Screen 3)

For a production version you can compile the source code, package it and upload it to the cloud (to serve it as a Progressive-Web-Application), or upload it to the App-Store of your choice.

Next Steps

- Integration with ERP-Systems
    - GAEB DA XML - Standard
    - iTWO - RIB
    - Procore API

Appendix

Hardware-Components:

- Arduino ESP 8266 Wifi Board
  - https://www.amazon.de/gp/product/B06Y1LZLLY/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1
- Load Cell Amplifier HX711
  - https://www.amazon.de/gp/product/B07KPJCDCJ/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
- 4x20 LCD Display HD44780
  - https://www.amazon.de/gp/product/B07DDGNPX1/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1
- Strain gauge (DMS) inside crane scale body
- Crane scale body Hilitand 660Lb (Up to 300 kg structure load)
  - https://www.amazon.de/gp/product/B07KW6STF3/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1
- Powerbank (3350 mAh)
  - https://www.amazon.de/gp/product/B005QI1A8C/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1
- Jumper wires + Resistors;
  - https://www.amazon.de/gp/product/B01EV70C78/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1
  - https://www.amazon.de/gp/product/B07CC4V9ZY/ref=ppx_yo_dt_b_asin_title_o07_s00?ie=UTF8&psc=1
  - https://www.amazon.de/ELECTRONIQUE-Package-Komponente-Kurse-Arduino-Sarter-A01/dp/B00MHUCYZA/ref=sr_1_4?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&keywords=arduino+10K+widerstand&qid=1579616358&s=ce-de&sr=1-4

Code

https://github.com/tpfeifle/skala