



SmartPost: Smart Postal Package Locker



by Kirsten Vermeulen

To end my first year of New Media and Communication Technology (NMCT), I had to make a project in which I integrated all the courses of the past year.

I came up with the idea to make a smart locker. I can be used to automate a collection point for packages.

To make my idea reality, I used a raspberry pi. I programmed the code in python and I used a Flask webserver to host a webapplication that collects data and controls the locker.

In this instructable you can find all steps I went through to get my project working.





Step 1: Analysing My Idea

Before I started working on my project, I wanted to investigate if people liked my idea.

I started asking my closest family what they thought of my project and what feature they would implement to make it even better and usefull.

I also asked my fellow students and friends if they

thought this could be innovative project.

Most of the people where enthusiastic about the idea and wanted to see it become reality.

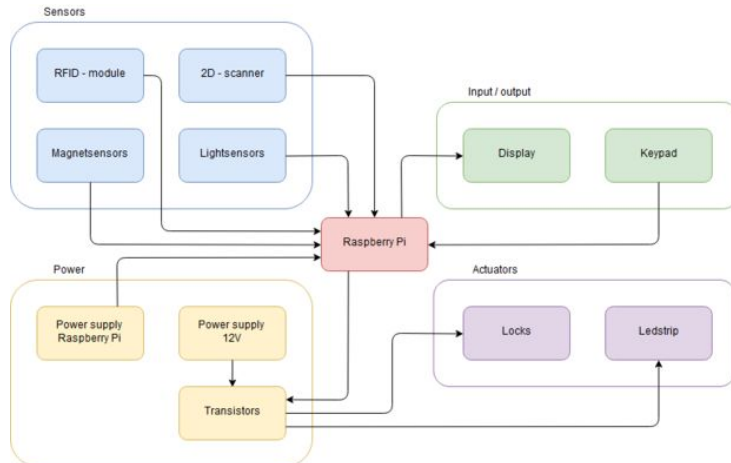
For me, this was the sign to go for it and start creating the project.



Step 2: Gathering Materials

The first step was thinking of all the materials and components I would need to build my smart locker.

To do this, I made myself a block diagram to sketch my needs and started searching in my electronics kit. I could use some of the component from my kit and I had to order some of them online. The list of parts I used can be found in the attachments.



<https://www.instructabl...>

Download

Step 3: Creating a Suitable Database

After doing some research and buying the right materials, it was time to create a database.

First, I made an entity relationship diagram using MySQL Workbench (as in the picture above). Then I forward

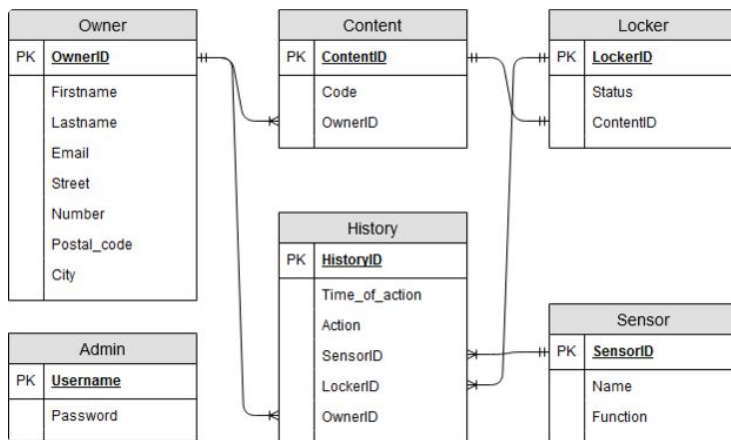
engineered this ERD and made a database. Here I added some random data to test it again and again till I didn't found errors anymore.


My database contains 6 tables. The main tabel is the history table. This is the tabel where I store all the data that I receive from my sensor.

The intention of the tabel 'owner' is to keep some info stored about the owner of which the locker contains a package. If the owner doesn't come to collect the package within 14 days then could the owner of the locker contact this person with this info.

I also keep the username and password of all the aministrators stored in the database. So they can login from the webapplication to get a overview of all the data collected by the locker.

You can find a MySQL dumpfile in the attachments




<https://www.instructabl...>

Download

Step 4: Design a Responsive Website

Now I had a database, I could start creating a responsive webapplication.

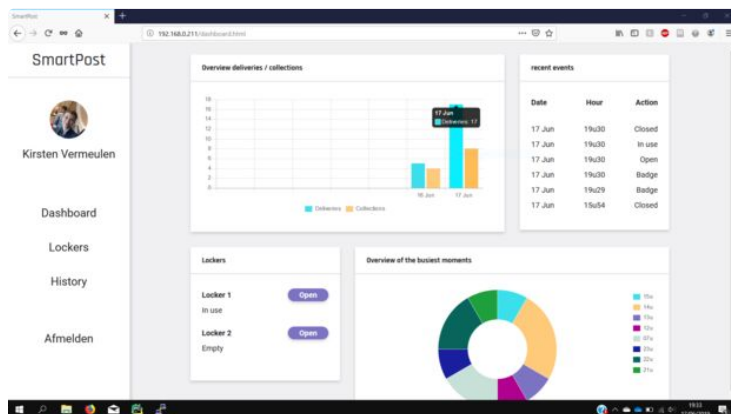
Before I started programming the whole thing, I made a user experience design and a user interface design for the mobile aswell as the web version of my webapplication using Adobe XD.

administrators of the locker and an overview of all collected data.

I attached the designs for the website to this step.

With this concrete plan, it was very easy to recreate it using HTML and CSS to become a responsive webapplication.

My webapplication contains of 2 parts. The first part is meant for general users. It is a small explanation of my project. The second part consists of a login for the




<https://www.instructabl...>

Download

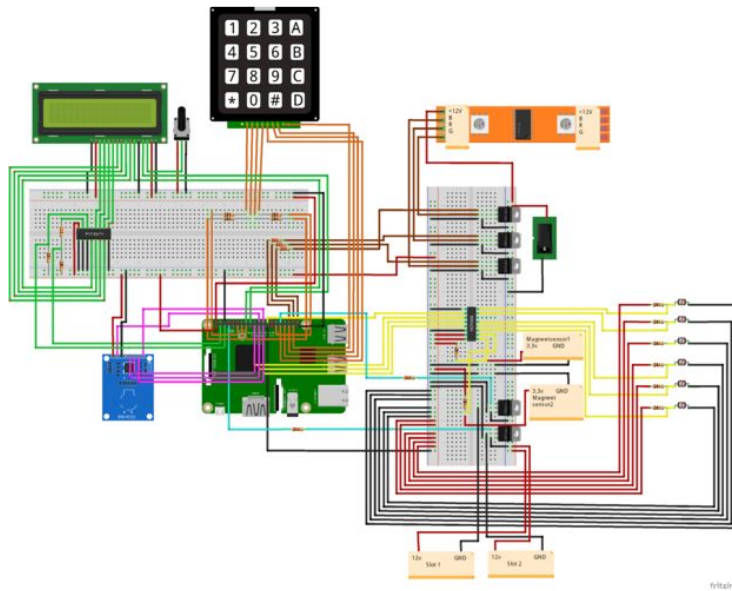
Step 5: Making the Circuit

When I had all the components, I could start making the circuit.

First, I made a fritzing scheme to visualize everything and then I started recreating it.

When all the wires were in place, I turned on the power to see if everything was alright. In my chase, it was not... The wires I used to run 12V through were too thin and they burned. So I replaced them with thicker wires.

I attached the wiringschemes on this step.



<https://www.instructabl...>

Download



<https://www.instructabl...>

Download

Step 6: Making the Circuit Come to Life

Now I have the circuit, we can finally start coding. First, I wrote some testcode to make sure all of my components worked individually.

When I could control almost all of the components separately, I started to put it all together in a Flask backend for my webapplication.

You can find the code in [this github repository](#)

