Contactless Water Fountain

by MaertenAndreas

For the end of my first year as an MCT student I was tasked to make a project that contained all the skills I had picked up from courses throughout the year.

I was looking for a project that would check all the requirements set by my teachers and at the same time be fun for me to make. When looking for a subject I couldn't help but feel inspired by Covid-19 (This was right before it went to a world wide outbreak.) I chose for a contact less water fountain/dispenser, as it would offer for a way of drinking water without work as well

- 1x Photoresistor
- 1x **HC-SR04** (Ultrasonic Distance Sensor)
- 1x RFID-RC522
- 3x Different colors LEDs (blue, yellow, red)
- 1x LCD1602
- 1x Active Buzzer
- 1x PCF8574
- 1x MCP3008
- 1x Water Pump (A 12v peristaltic pump was used, link to this item)

touching some buttons before water would come out.

This project uses a distance sensor to detect if a cup or glass has been placed under the water output, the fountain will then proceed to output water for 60 seconds (100ml / minute). This is to make it more consistent because detecting if the glass has been pulled away proved to be too difficult/slow of a task which is why a timer was put in place. After your glass has been filled with 100ml of water you can wait for 5 seconds and if the glass is still in front of the distance sensor it will proceed to fill another time (this means there is also a timeout of 5 seconds between filling two different items).

Supplies:

Components

- 1x RaspberryPi (I used the 4th version but older versions might work as well)
- 1x **S8050** transistor or 1x **PN2222** transistor might
- 1x DC Power supply (12v, 600mAh)
- 1x power brick with 3 spots
- 3x breadboards (you could probably use less)
- T-cobbler for RaspberryPi GPIO pins
- T-cobbler cable (for connecting between pi and cobbler)

Materials and tools used

- A drill with the following drill bits:

- 4mm (to pre-drill holes for the screws)
- 15mm (to drill holes for the distance sensor)

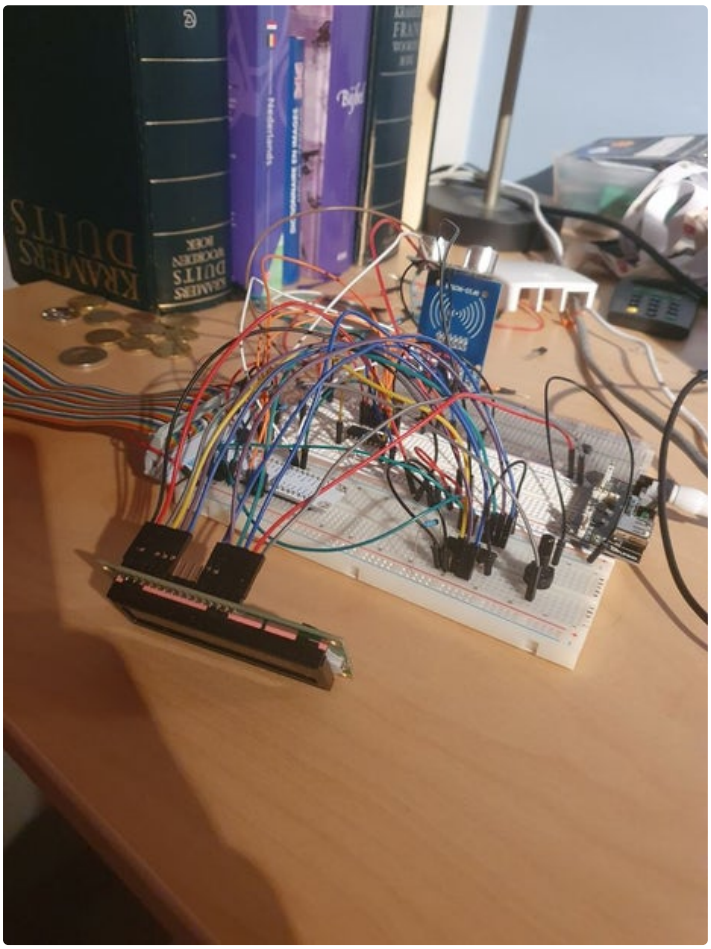
- Any screwdriver
- 30 screws of 45mm long
- 6 screws of 20mm
- 2 hinges for the door
- A plate of MDF of around 130cm by 80cm
- A couple of files

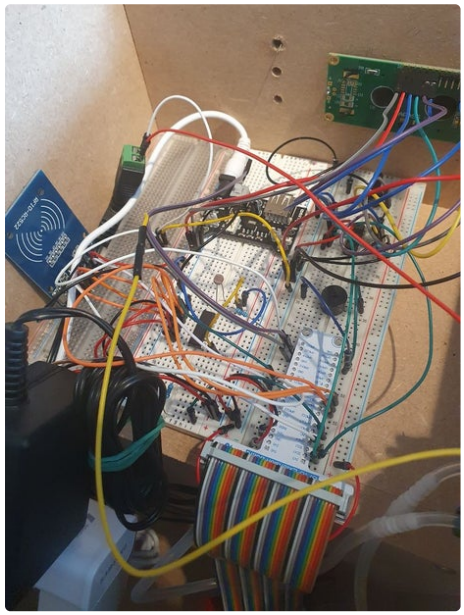
Step 1: Assembling the Circuit

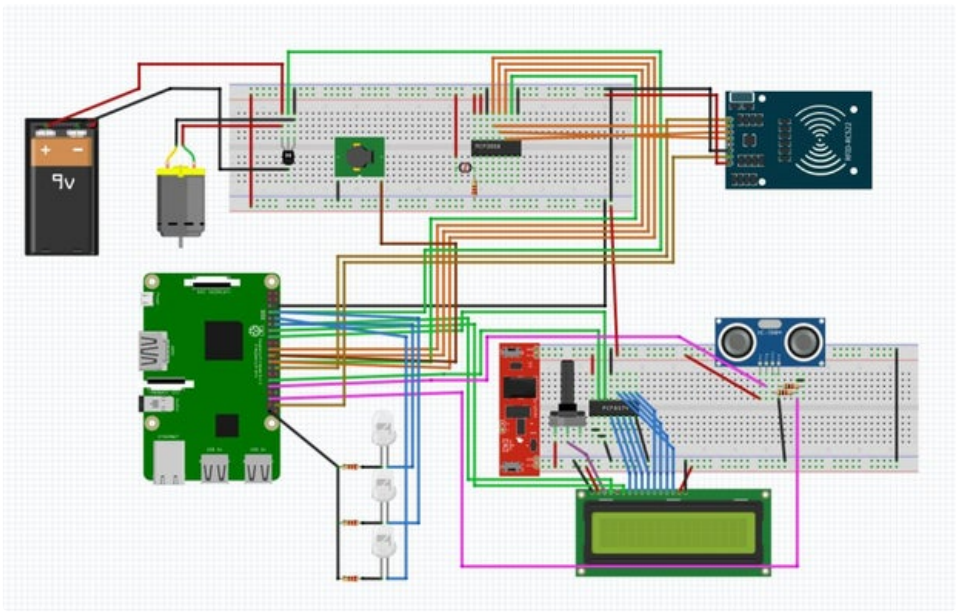
For the circuit we have 2 sensors, a **distance sensor** and a **photoresistor**. The distance sensor is used to detect if a cup has been put in range of the water fountain and optionally I added a photoresistor, this one is used to detect if the casing has been opened by anyone that isn't supposed to open it. On top of that we have an **RFID reader** this can be used to authenticate a mechanic that needs to open the case to refill the water reservoir or for some other mechanical issue.

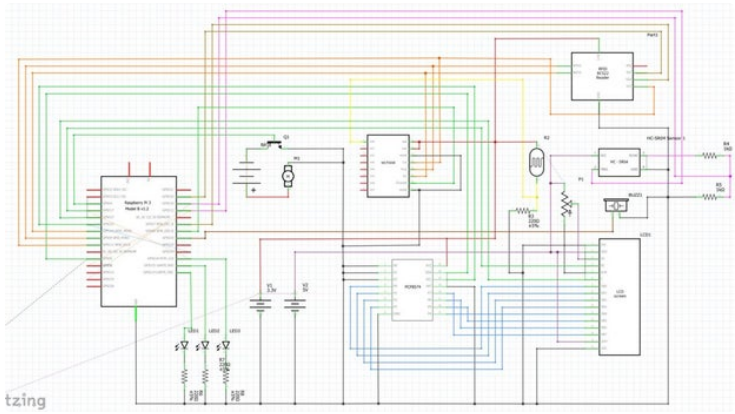
For the active elements we have the **LCD1602, active buzzer** and a **peristaltic pump**, the LCD is used to display status like if the case is open or the pump is running as well as the IP address of the device will be shown, the buzzer is used to make an alarming sound when the case has been opened without someone authorizing it.

I have added the breadboard and schematic views of the circuit below.









Step 2: Setting Up Our RaspberryPi

To setup our RaspberryPi, we will download the imaging software from the Raspberry site, with this you can download the version of Raspbian you want and image your SDCARD for you. After this tool has done its work you can open the SDCARD in Windows Explorer, you'll be able to see the boot partition of your RaspberryPi. In here we will find a file called cmdline.txt (don't open this file in notepad, open it in Notepad++ or any other IDE). We will add **ip=169.254.10.1** to the end of this file to make sure we can connect to our device over ethernet (make sure you don't add any ENTERS at the end of your file or you'll have trouble).

Now you can put your SDCARD in your RaspberryPi and boot it up, connect the Pi to your computer and use Putty to connect to your Pi over **SSH.** I use the following command to connect to my Pi instead of using Putty.
"**ssh pi@169.254.10.1**"this might timeout, so be patient and wait for the Pi to boot up. Once prompted for a password we'll fill in the default password of "raspberry". Make sure to change this password after logging in to prevent anyone with ill intent from accessing your Raspberry Pi.

We will now configure our Pi to provide the necessary functionality for our code.
Use "**sudo raspi-config**" to open the configuration menu and in here we will go to **Interfacing Options.**

Under here we will toggle the following options **ON**:
- **SPI**
- **I2C**

Follow this guide to setup a wireless internet connection on your Pi, after you've successfully done this we can get to installing our required packages.

Packages: (run the commands in the order as they are noted here)

The following to get the latest updates for our Pi
- **sudo apt update && apt upgrade -y**

Install our MySQL server and webserver
- **sudo apt install mariadb-server apache2**

I will be using MySQL Workbench to setup the database later in this guide, if you don't use this and prefer **phpmyadmin** you can install this with the following command, you're free to use any other MySQL Client as well as long as you're able to properly import the database.
- **sudo apt install phpmyadmin**

After you've done all of the above we need to create a user for our our database.
Use "**sudo mysql -u root**" to log into your MySQL server, in here we will create a user called db_admin with its respective password, keep this password noted somewhere for later in the instructions.
GRANT ALL PRIVILEGES ON *.* TO "db_admin"@"%" IDENTIFIED BY "yourPasswordHere" WITH GRANT OPTION;

Use the"**\q**"command to exit out of the MySQL terminal.

Python Packages:
We still need to install some python packages before proceeding, run the below command to make sure everything is there for a flawless experience.

sudo pip3 install Flask Flask-Cors Flask-SocketIO gevent gevent-websocket greenlet spi SPI-Py
spidev

As well as the following MySQL connect python package
sudo apt install python3-mysql.connector -y

If all went right you can now visit your Pi on your webbrowser with the following address
http://169.254.10.1/

Step 3: Setting Up the Backend

Here I'm going to explain how you can setup the backend yourself, first download the rar file from below, unrar it to some temporary directory. Connect to your RaspberryPi with FileZilla or WinSCP with the following credentials:

IP: 169.254.10.1
User: pi
Password: **raspberry** (if you changed the password do it here as well)

You can then proceed to transfer the files you unrarred to any directory you want in the home directory of the pi user. For simplicity sake we will assume in this setup that we have uploaded all of our files under the document directory.

Keep your FTP program open for the next step!

Setting Up the Backend


https://www.instructabl...
Download

Now open your command prompt again with your SSH connection because we're going to need to do some changes to the webserver so the frontend can communicate with the backend.
We're going to open the default Apache2 config file and modify it slightly:
sudo nano /etc/apache2/sites-available/000-default.conf

Add the following lines below DocumentRoot in the config file we just opened:
ProxyPass /api/ http://127.0.0.1:5000/api/
ProxyPassReverse /api/ http://127.0.0.1:5000/api/

You can take a look at the image attached for an example.

Step 4: Setting Up the Frontend

Before transferring our files we'll have to do something before we can start transfering our frontend files.
Open your command prompt with the SSH connection you made previously and use the below command to switch to the root user of our RaspberryPi: "**sudo su -**"

After this we can change the password of our root user with the following command: "**passwd**"
This will ask you to input a new password, after you've done this you can switch back to your FTP program and login with your root credentials:

IP: **169.254.10.1**
User: **root**
Password:

Download the rar file from below and unrar it in a temporary folder, you can the move these files to your RaspberryPi to the following directory **/var/www/html/**, after you've done that you can visit the frontend on **http://169.254.10.1**, you can't interact yet because the backend is not running yet, I'll show you later on in this guide how to do this.

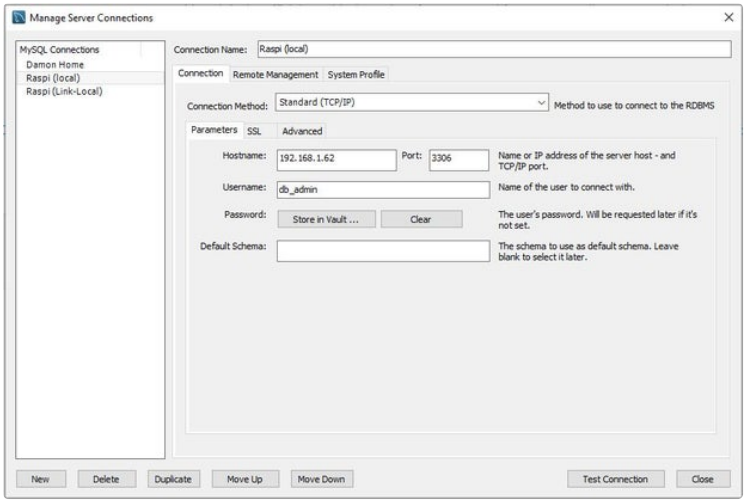

https://www.instructabl...
Download

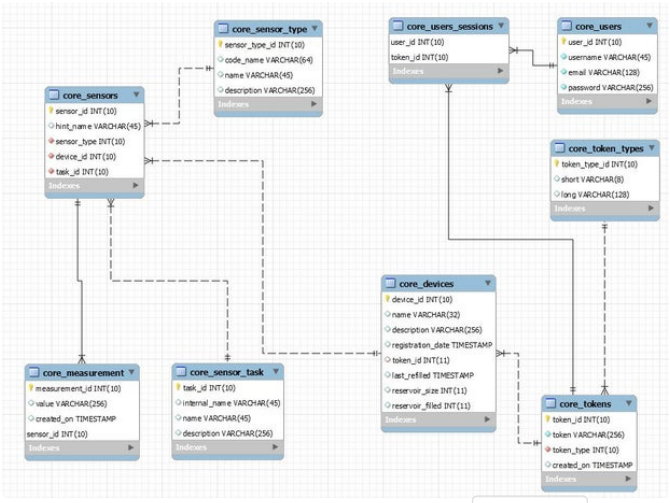
Step 5: Importing the Database for Our Project

Open your favorite MySQL server management program and connect to your Raspberry Pi with the credentials we created in **Step 2.**

Download the database dump from below and import it as you normally would, MySQL workbench you would go to **File > Open SQL Script** and select the database dump you downloaded.
Then press **CTRL + SHIFT + ENTER** and the SQL script should be ran and the structure for the database should be created.

I added the credentials I used for my RaspberryPi as an example below as well as several pictures of the Database structure, you can take a look at it and try and get a general idea of how everything works.





Download

https://www.instructabl...

Step 6: Starting Up Our Project

Before we can startup our project we need to change the database credentials in the config.py file, if you followed the instructions exactly as this guide said then you can find these under **/home/pi/Documents/Backend/src/config.py** in here you need to change the credentials of the db_config variable to match those we created earlier for our database. I've added an example of what you'll see in this file below.

After that you we will add a **.service** file this file will make sure our project starts when the RaspberryPi starts, make sure you change the directory appropriately of where you installed the backend files. Use the following command to create the service file: **sudo nano /etc/systemd/system/dispenser.service** This will create a service file and copy paste the below code into this file.

add an example file below.

After you've done that and exited out of the text editor we can enable the service with the following commands:

- **sudo systemctl daemon-reload**
- **sudo systemctl enable dispenser**
- **sudo systemctl start dispenser**

```
db_config = {
    'user': 'db_admin',
    'password': 'K*G7hBoIBz$&Kc869&Ft1M',
    'host': '127.0.0.1',
    'port': '3306',
    'database': 'water_dispenser'
}

rest = {
    'host': 'http://localhost',
    'token': 'test_token'
}
```

[Unit]
Description=Water Dispenser
After=mysql.service

[Service]
Type=simple
Restart=always
RestartSec=1
User=pi
ExecStart=/usr/bin/python3 /home/pi/Documents/Backend/index.py

[Install]
WantedBy=multi-user.target

Modify the line where it says /home/pi/Documents/Backend/index.py to where you installed your backend files, if you don't do this correctly the project won't be started correctly! I will

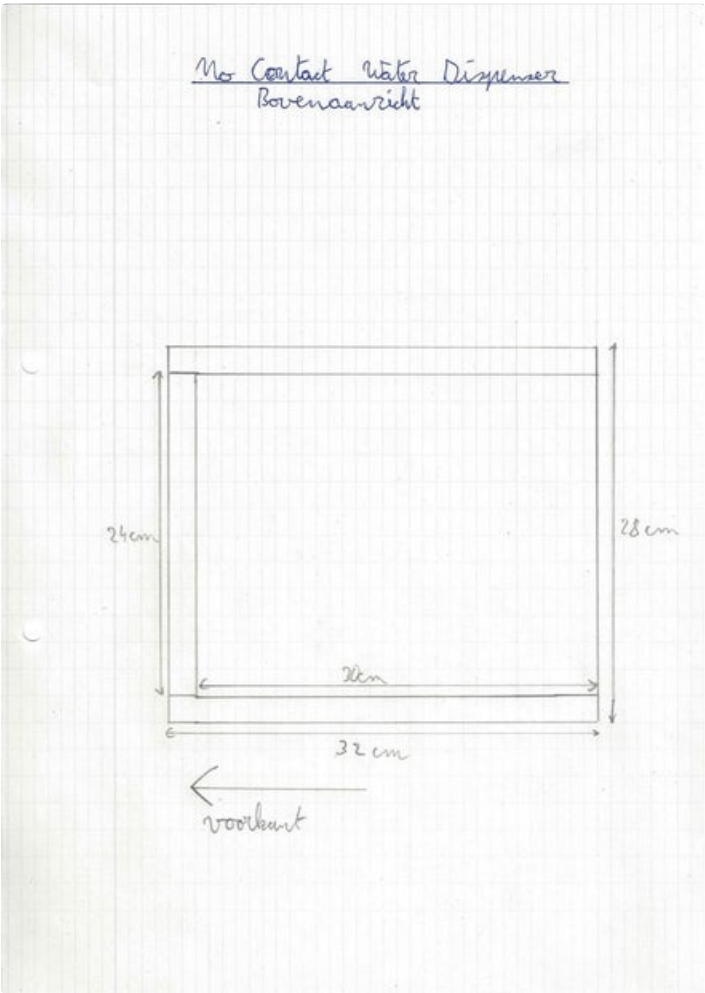
And as an extra we can run:
sudo systemctl status dispenser
This will show some information around our service, if it's active or not, ...

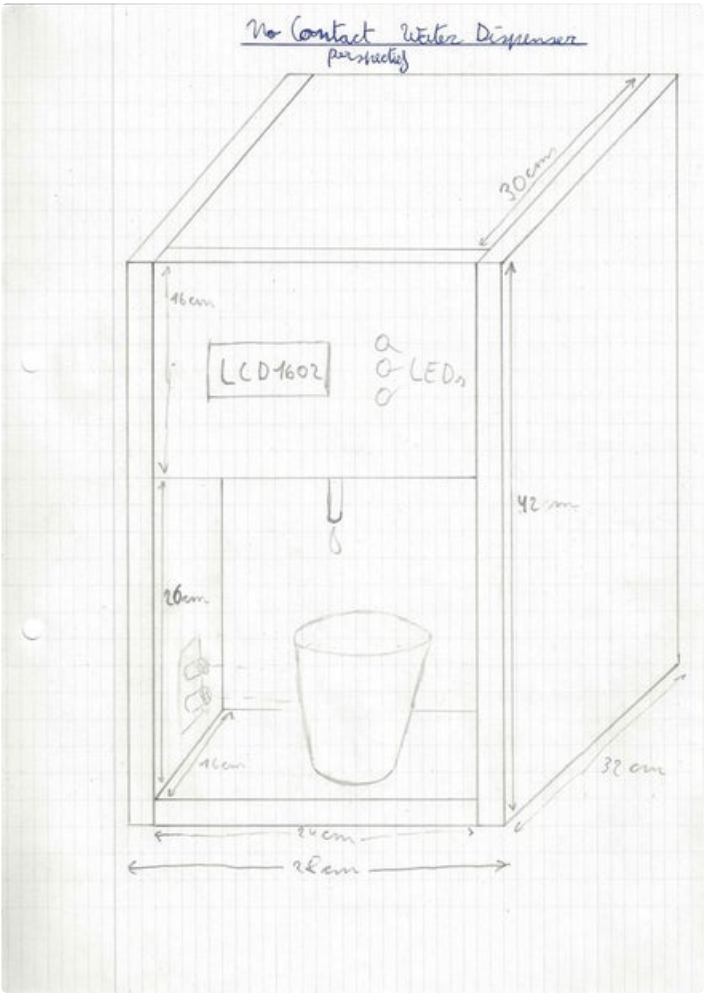
```
dev@damon-s-pie:~ $ cat /etc/systemd/system/dispenser.service
[Unit]
Description=Water Dispenser
After=mysql.service

[Service]
Type=simple
Restart=always
RestartSec=1
User=dev
ExecStart=/usr/bin/python3 /home/dev/project1/Code/Backend/index.py

[Install]
WantedBy=multi-user.target
```

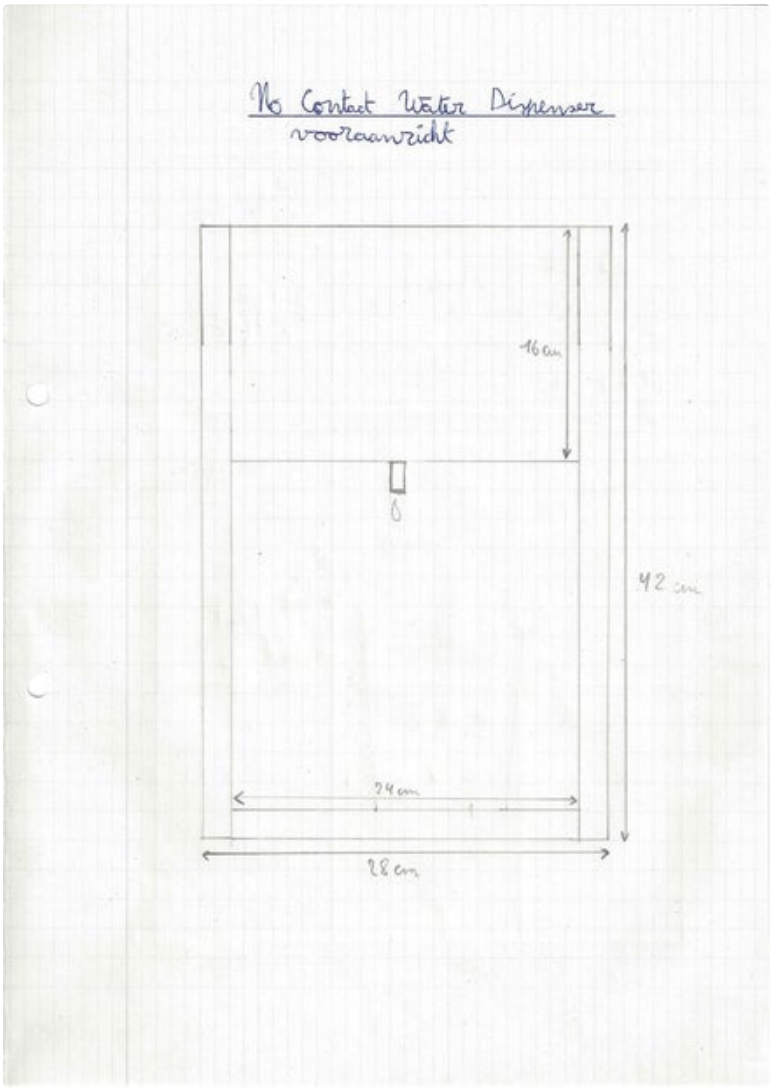
Step 7: The Case

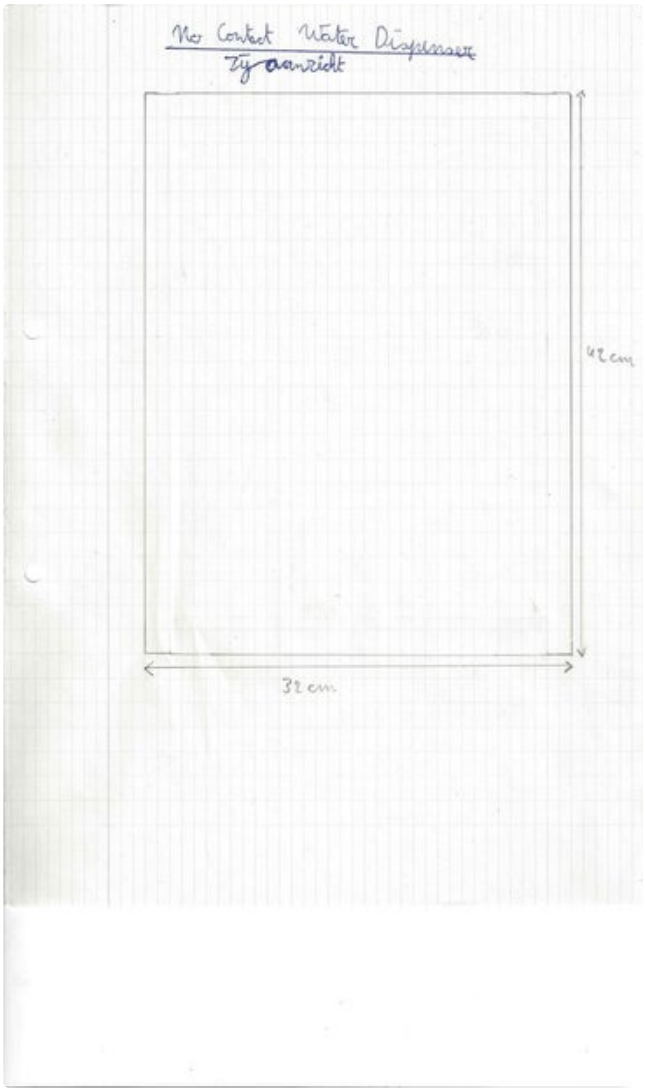
Congrats we're almost there, I will add some pictures that will accurately show the dimensions I used for my project, I used MDF plates of 18mm thick, you can optionally use a different thickness. My casing can be used as a guideline to design yours or you can recreate what I made. (If you use a different thickness of MDF my drawings will no longer allow you to make my design, make sure to adapt it!)

The panels I made:
- 2 panels of 32cm by 42cm (side panels)
- 1 panel of 24cm by 32cm (bottom plate)
- 2 panels of 16cm by 24cm (front plate where LCD stays and neighbouring plate)
- 1 panel of 28cm by 24cm (middle plate seen from the front)
- 1 panel of 30cm by 24cm (top plate)









Step 8: Admire the Final Product

You've reached the end and by now hopefully managed to make the entire thing a reality. If you're just a passerby reading through, also welcome, I thank you for reading until the last step!

I spent a lot blood, sweat and tears into this project so I would appreciate it if you left a comment, any criticism on improving it is welcome!

Step 9: The Problems

I would put the project in its current state as a working prototype that can see a lot more improvements.

The code base of the backend is structured in such a way that a master slave relation can be perfectly made where one fountain would act as the main frontend and all other fountains would push data and changes over the REST api of the master. There are also remnants of an API token system in the code as this was intended to be implemented but cut later on due to time constraints.

I have uploaded my code to my Gitlab server and there you can take a look at the code as whole: https://git.damon.sh/Yimura/project-1