TÉCNICO LISBOA

MEIC-A - Instituto Superior Técnico
Universidade de Lisboa

— Ambient Intelligence —
(Ambientes Inteligentes)

Mixologist
Project report

Students:
Nathan Edely
Florian Mornet

Professors:
Prof. Renato Jorge Caleira Nunes
Prof. Alberto Manuel Ramos da Cunha

Report date: 15/05/2020
`https://fenix.tecnico.ulisboa.pt/disciplinas/AI514/2019-2020/2-semestre`
Repository: `https://github.com/geckoflume/mixologist`

Abstract

This report presents a solution for the creation of an automatic and intelligent bartender. We start by presenting some technologies or solutions that are already developed. This part explains our motivation to use, or avoid some designs, and make a point of view of what is already done, what we can reuse, and what will be our production. After this summary, we present our solution, starting with a general introduction. Then, some details of this project are introduced to explain in concrete terms, how does our project work. Finally, we finish this paper by evaluating our work and all aspects that would be interesting to develop in a future version of this project. Last, a short bibliography is presented.

Keywords: Mixologist, IoT, Intelligent bartender, Orange Pi, Robot, Automation, Cocktail, Google Assistant, Maker, Arduino, Load cells, Peristaltic Pump

domotics / domótica / domotique:
from the Latin domus (‘‘home’’), and robotics
(from science fiction author Isaac Asimov in 1941 from "robot" + "-ics")

Contents

1 Introduction **5**

2 Related work **6**
2.1 Professional robots 6
2.2 Homemade solutions 8

3 Description of the solution **9**

4 Details of the solution **11**
4.1 Electrical layer 11
4.2 Electronics 12
4.2.1 Orange Pi PC 12
4.2.2 Relay board 12
4.2.3 Peristaltic pumps 13
4.2.4 RGB addressable LEDs 13
4.2.5 Load cell 14
4.2.6 Load cells amplifiers 15
4.3 Arduino Uno 16
4.4 Software layer 16
4.4.1 Controlling the RGB strip 16
4.4.2 Calibrating and reading the Load cell values 16
4.4.3 Database 17
4.4.4 Making cocktails 17
4.5 User interaction 18
4.5.1 Web interface 18
4.5.2 Google Smart Home ecosystem 18
4.6 Final design 19

5 Evaluation **21**

6 Conclusion **24**

Glossary **26**

References **28**

List of Figures

1 The Makr Shakr - Photographer: Avocado Studio 6
2 The Barsys 2.0 - Photographer: Barsys [1] 7
3 The Arduino Robotic Bartender - Extracted from DIY Machines YouTube videos [10] 8
4 The Crude Cocktail Machine - Extracted from GreatScott's YouTube video [6] 9
5 Mixologist - Initial design 10
6 Orange Pi PC SBC Board 12
7 The relay board we used to control the pumps 12
8 Peristaltic pump operation - Courtesy of Grayline LLC 13
9 NeoPixels LED strip 14
10 A 5kg load cell 14
11 The 1kg load cell in our prototype 15
12 HX711 module 15
13 Arduino Uno 16
14 Initial database schema 17
15 Google Actions 18
16 Mixologist - Final design 20
17 Base prototype, with removed tubings 21
18 Mixologist dashboard 22
19 Mixologist new recipe form 23
20 HX711 module modification - Photography of gandalf15 Github user [4] 24

1 Introduction

With the quickly expanding market of embedded devices and robotic appliances in everyone's homes, we tend to see more and more industrial/professional-graded solutions entering our lives and thus, automating each and every redundant task and reducing the need for external services.
This fact led us to an interesting observation: there are no affordable device capable of replacing the human in a pretty simple but common and time-consuming task such as brewing cocktails.

This way we had the idea of the *Mixologist*, a cheap (less than 50€) device capable of replacing an amateur bartender in mixing liquids.

In fact, this task is rather simple and for a basic recipe, a robot could be better than a human by providing more accurate liquid measurements and doing this task faster.

2 Related work

2.1 Professional robots

At first, we tried to find similar ideas and the first that came up were professional robots, such as the *Tipsy Robot* in Las Vegas [16] or the *Makr Shakr* [14] as presented by this Bloomberg article [17].

This product uses robotic arms to pour liquids from bar optics and to shake the containers and lets the user interact with it using a mobile application (see Figure 1). However this solution was really impressive, using cutting edge technologies such as Machine Learning, it had two main drawbacks: its robotic arms takes a considerable amount of space and it costs about 99 000€.

Figure 1: The Makr Shakr - Photographer: Avocado Studio

This product was definitely not suitable for in-home use and was mainly designed for professional bars (it is in fact used in Royal Caribbean cruises and bars in Milan and London) where more technical gestures were needed.

Then we found out about a scaled-down device, the *Barsys* [1] (see Figure 2) that provides some common services, removing the constraints above mentioned by removing the arms (thus also removing the ability to shake/mix drinks) and instead has the cocktail glass moving on a treadmill.

This was a huge breakthrough but the price was still quite expensive: about $979.00 as of April 2020.

Figure 2: The Barsys 2.0 - Photographer: Barsys [1]

2.2 Homemade solutions

Seeing that there was no affordable solution available on the market, we then tried to find homemade or DIY solutions, which led us to two interesting projects:

1. The *Arduino Robotic Bartender*, made by Lewis (aka DIY Machines YouTube user) [10]
2. The *Crude Cocktail Machine*, made by Scott (aka GreatScott! YouTube user) [6]

The first one (see Figure 3) seems greatly inspired by the two previous, by using bar optics for measuring the quantity of liquid and activating the liquid flow, and using a treadmill underneath the glass container to move it just under the right bottle.

This solution uses stepper motors to control the belt and has a major drawback: the quantity poured in the glass is measured by time. We found out this might cause problems in case we need a lot of functions (such as serving a web page, reacting to REST requests, pouring the liquid, indicating the progression of the task...) for a small micro-controller, which might add some delay and thus, introducing wrong liquid measurements.

Figure 3: The Arduino Robotic Bartender - Extracted from DIY Machines YouTube videos [10]

The second, however (see Figure 4), uses a different solution which seemed less prone to errors and problems in the future: using motors and peristaltic pumps (cheaper than stepper motors), having the container fixed and the liquid pouring down silicone tubings. The previous problem is sort of solved because it uses a load cell to measure the quantity of liquid being poured, and being able to stop at the right time.

It is also faster than having to move the glass under the different bottles, as we can pour multiple liquids at the same time, and it reduces the risk of breaking and spilling the glass and its content.

This particular project provides a fancy interaction by containing an LCD screen with a rotary encoder for the selection of recipes. However, having the recipes hard-coded is making it quite harder to add or edit them.

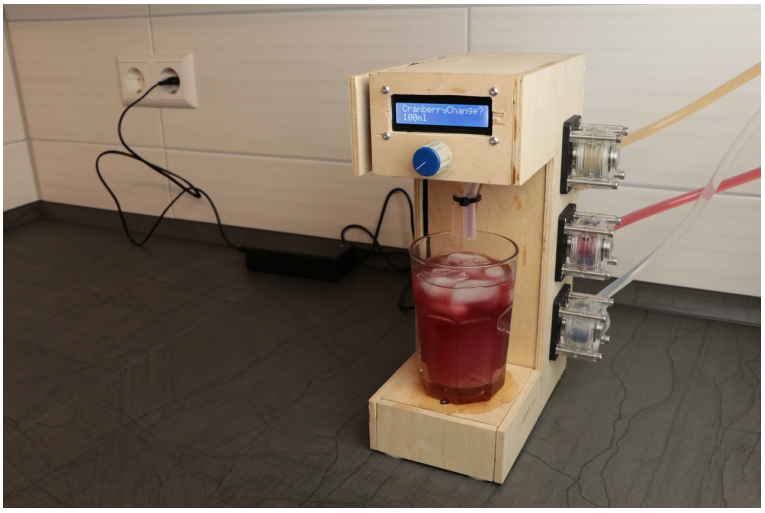

Figure 4: The Crude Cocktail Machine - Extracted from GreatScott's YouTube video [6]

Both devices use micro-controllers such as Arduinos for controlling the motors, the load sensors and the indicators (LCD screen or LEDs)

However, we thought about some more features that would be appreciable, and which would enhance those ideas. Our design is mainly based on the latter one thus making it cheaper, and providing easier maintenance.

3 Description of the solution

During the conception step, we thought about adding features such as:

- a web-server to interact with the bartender, add recipes
- a database to store the recipes and for statistics purposes
- adding load cells to measure the weight of each bottle (indicating if it is possible to make cocktails according to the bottles capacities)
- integration with the Google Assistant ecosystem

Those features would not increase dramatically the cost of the solution, most of them being only software implementation and the other parts being quite cheap (a single-board computer such as the Raspberry Pi costs about 35€).
Thus, we came with the following design (Figure 5):

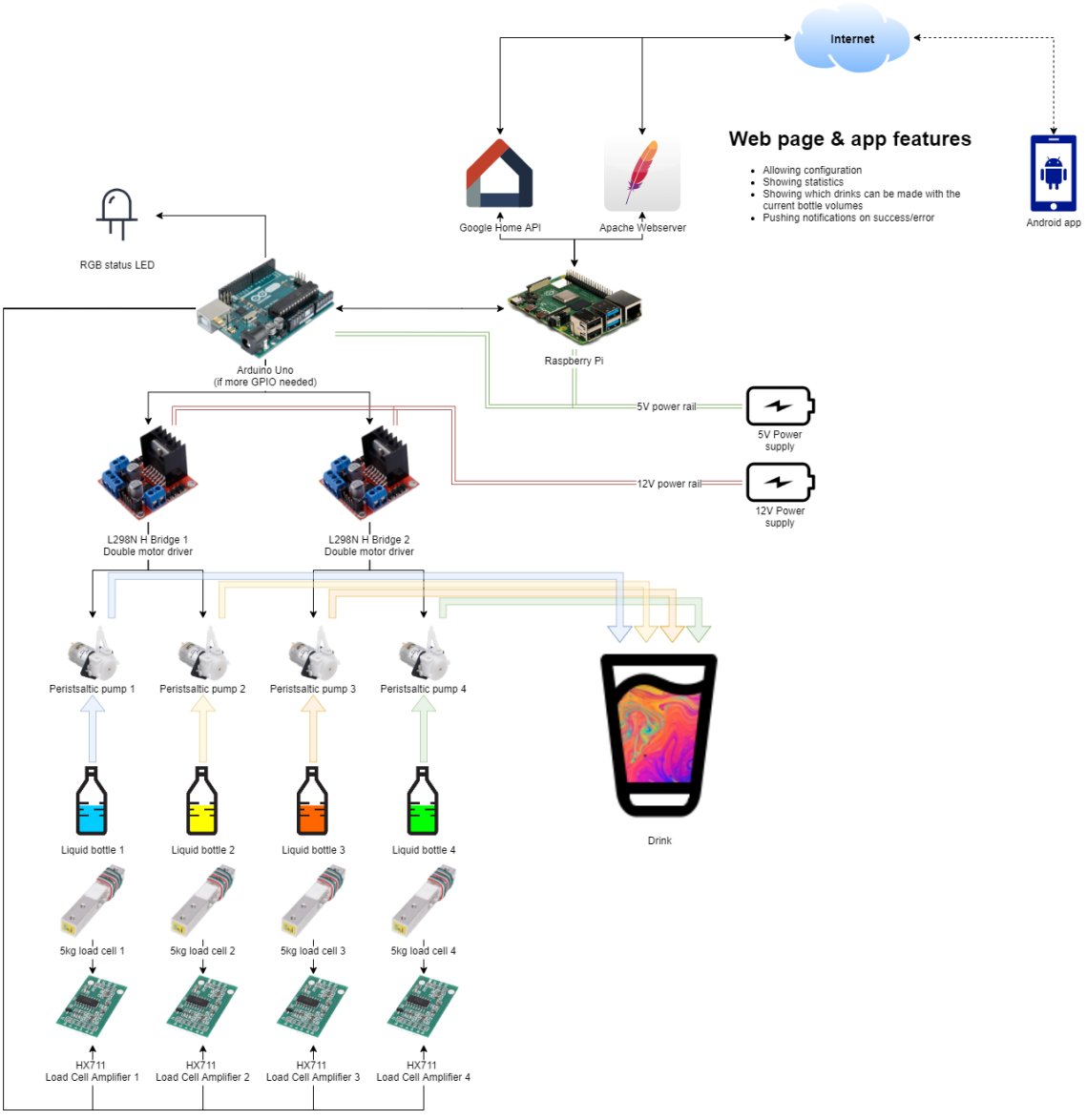

Figure 5: Mixologist - Initial design

This drawing is composed of 3 layers, which will be covered in detail in the next section:

- The electrical layer, that powers the motors, the Raspberry Pi and the Arduino
- The electronics, containing the Raspberry Pi, the Arduino, the motors, the motor drivers, the load cells, the load cell amplifiers and the LED
- The software layer, containing the program we wrote to interface with the hardware

4 Details of the solution

The Covid-19 pandemic situation gave us some more constraints to complete this project. We were not able to order electronic components from China in a reasonable time, Amazon services stopped delivering unnecessary products in France and public post services only worked partial time. Moreover, the lockdown prohibited us to go shopping except for food.

We then had to do with the materials and components we already owned, which led us to redesign our idea in some ways.

For example, we had a Raspberry Pi 1 Model B and an Orange Pi PC laying around, which we tested to find which one was more suitable.
However the software support for the Raspberry Pi was way better due to its community, we were not able to run most of the software because of its outdated hardware. We chose to go with the Orange Pi, which was much more powerful.

Same goes for the L298N motor drivers, which we did not receive in time, and that we replaced with a 4 relay board (removing the possibility to control the motors speeds).

Finally, we also did not receive the single RGB LED we ordered, which we replaced with an addressable RGB LED strip we already had.

4.1 Electrical layer

Here we have two power supplies, delivering the 12V to power the motors and the motor drivers, and the 5V to power the Arduino Uno and the Orange Pi PC.

The 12V power supply is a spare power supply from an old project, which provides 10A.
The 5V power supply is a robust 2A power supply, which should be sufficient according to our needs.

4.2 Electronics

4.2.1 Orange Pi PC

These single-board computers made by Chinese manufacturer Xunlong are pretty popular among hobbyists because of their aggressive price-performance ratio.
They are made to look like Raspberry Pis, for a cheaper price tag. For example, this board has an *Allwinner H3* SoC, with 1GiB DDR3L RAM, HDMI output, 4 USB ports, Fast Ethernet and the operating system running on a MicroSD card.

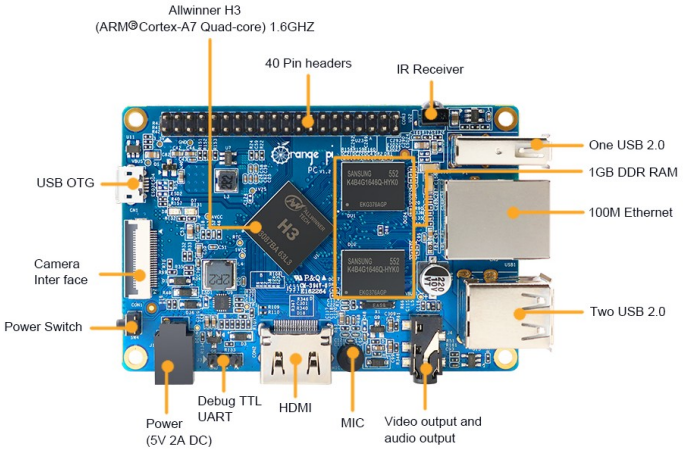

Figure 6: Orange Pi PC SBC Board

The problem with these cheap ARM-based boards is their software support from their manufacturers, which can be fortunately solved with the help of a community operating system based on Debian/Ubuntu, called *Armbian*.

4.2.2 Relay board

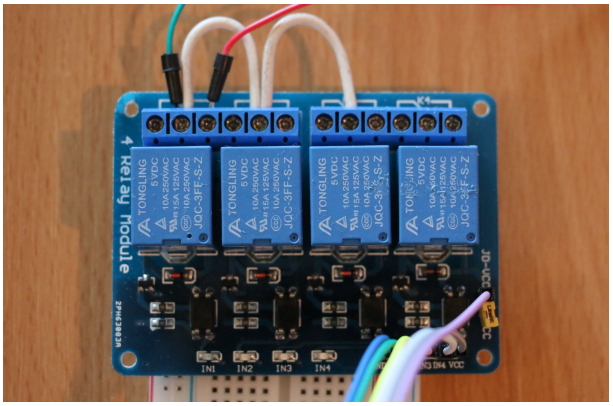

Figure 7: The relay board we used to control the pumps

As explained in the beginning of this section, we decided to replace our motor drivers with a 4-relay module board.

This module simply consists of 4 optocouplers, driven by the pins on the bottom of the picture (see Figure 7), and 4 5V-driven relays, which can control devices rated up to DC 30V/10A or AC 250V/10A.

Our SBC having a working under 3.3V logic, we needed 2 power supplies: one for driving the optocouplers and one to trigger the relays.

4.2.3 Peristaltic pumps

These pumps, also called roller pumps, are 12V DC motors which are connected to a rotor with 3 rollers. Each rotor has a silicone tubing in its enclosure, which is compressed when the motor is powered on, by creating pressure and thus letting the liquid flow (see Figure 8 to see the working principle) [2].

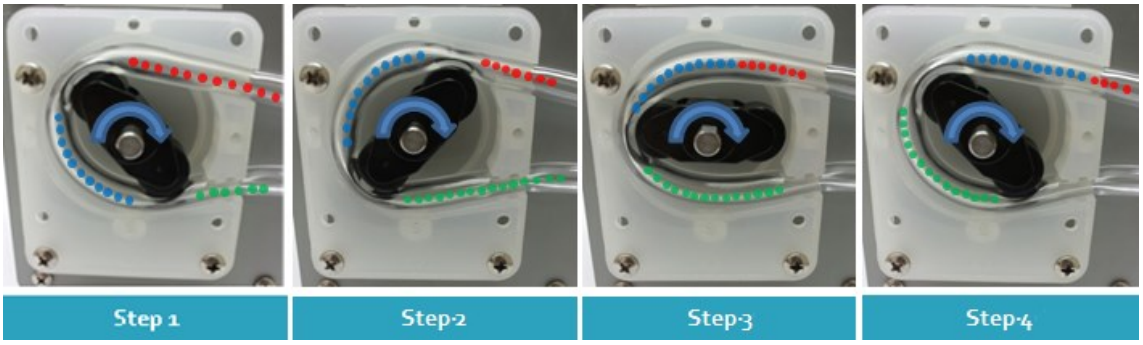

Figure 8: Peristaltic pump operation - Courtesy of Grayline LLC

Unlike standard pumps, peristaltic pumps can be used for beverage dispensing, since they are food-graded. Indeed, their unique design allows liquid flowing without exposing the fluids to contamination from exposed pump components.

They allow precise amounts of liquids to be poured, thanks to their constant motor rotation speed. The ones we had in our possession were rated 3W, which makes the 12V 10A power supply we had not scaled for our usage:

$$P = U * I$$
$$I = \frac{P}{U}$$
$$I = \frac{3 * 4}{12}$$
$$I = 1\text{A}$$

4.2.4 RGB addressable LEDs

Replacing the single RGB LED we had in our initial device, we were able to drive a LED strip using the SPI protocol.
This LED strip is quite common and is also known as WS2812 or *NeoPixels* [18].

Figure 9: NeoPixels LED strip

They are powered in 5V and draw about 20mA per LED compound according to their datasheet (so 60mA in total for Red, Green, Blue), times 18 modules we used to fit our project, which leads us to 1080mA.
The fact that we had more liberty on this side allowed us to produce an indicator based on the number and the colours of the LEDs to indicate the status of the device.

4.2.5 Load cell

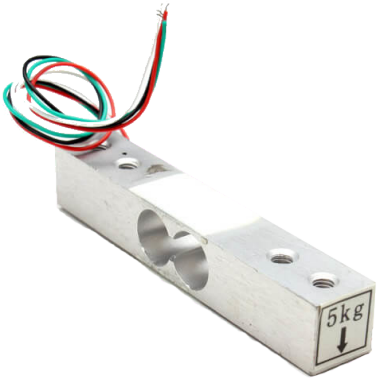

Figure 10: A 5kg load cell

Load cells are transducers that convert a force such as tension, compression or pressure into an electrical signal. Here, our strain gauge load cells are composed of Wheatstone bridges, just as in bathroom scales to check our weight.

We have one measuring up to 1kg under the support for the cup (see Figure 11), and 4 of them measuring up to 5kg under each bottle support.
It is noticeable that each one needs to be calibrated in order to be accurate.

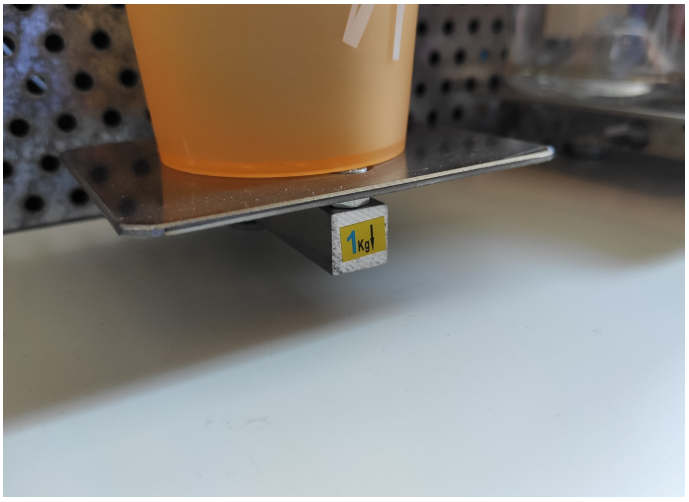

Figure 11: The 1kg load cell in our prototype

4.2.6 Load cells amplifiers

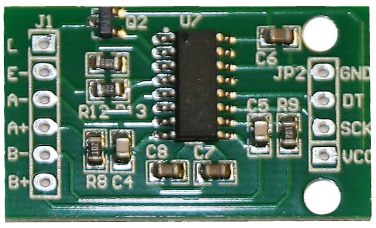

Figure 12: HX711 module

Due to the nature of our strain gauge load cells, we needed a component to interface between them and our computer.
The HX711 are cheap 24-bit ADC, sold on easy to connect breakout boards [13].
However they support 2 channels (A and B in the datasheet), the channel A would allow a bigger precision with a gain of 128 (by default, or 64), whereas the channel B was fixed to a gain of 32. Thereby, we chose to use multiple ADCs instead of losing precision.

Here, we quickly found out that their support was rather limited on Raspberry Pis and not existing on Orange Pis. The solution we found to tackle this was to use an *Arduino Uno* micro-controller to take the readings, removing the need for real-time access to GPIO that those SBCs were not able to provide, and sending them to the Orange Pi using a USB-Serial connection.
With more time, we could have tried to write our own driver based on the datasheet and their particular protocol.

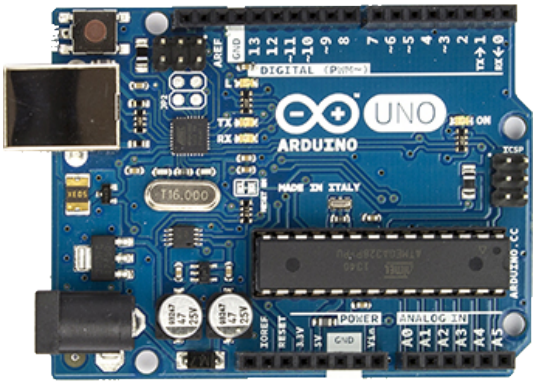

Figure 13: Arduino Uno

4.3 Arduino Uno

This micro-controller is the most common open-source design of the Arduino company.

It is designed around the well known ATmega328P processor, has relatively good specs for its aggressive price tag. This board was only used to answer to the Orange Pi requests.

Connected to the 5 HX711 modules through digital GPIO ports, we were able to read their values and then computing the weight of each load cell after calibration.

4.4 Software layer

Since it is common to find code snippets, library and projects about Raspberry Pis (and other SBCs) in the Python language, therefore we decided to use it.

Our software stack is based on *Python 3* with the *Flask* framework [12] to handle requests.

4.4.1 Controlling the RGB strip

We were able to use *joosteto*'s Python library made for the Raspberry Pi [8] that we adapted to fit our needs (by removing the *Numpy* calls originally made for patterns generation, fixing what was not made for the *H3* platform [3]).

The use of multiple LEDs allowed fancier usages such as showing a progression in the cocktail being made or to signal different errors or warnings.

4.4.2 Calibrating and reading the Load cell values

For this part, we used *olkal*'s C library [11] that allows using multiple HX711 at a time. We then wrote a simple program to read the values on the Arduino, which

transmits the data to its serial port.

This data is read by the Orange Pi thanks to *Chris Liechti*'s *pySerial* Python library [9].

4.4.3 Database

In order to allow the user to add custom recipes and to keep persistent data, we added a local MariaDB DBMS, which is a replacement for the well-known MySQL.

We will see in the next sections that the user can add precise recipes, allowed by the preliminary schema in Figure 14.

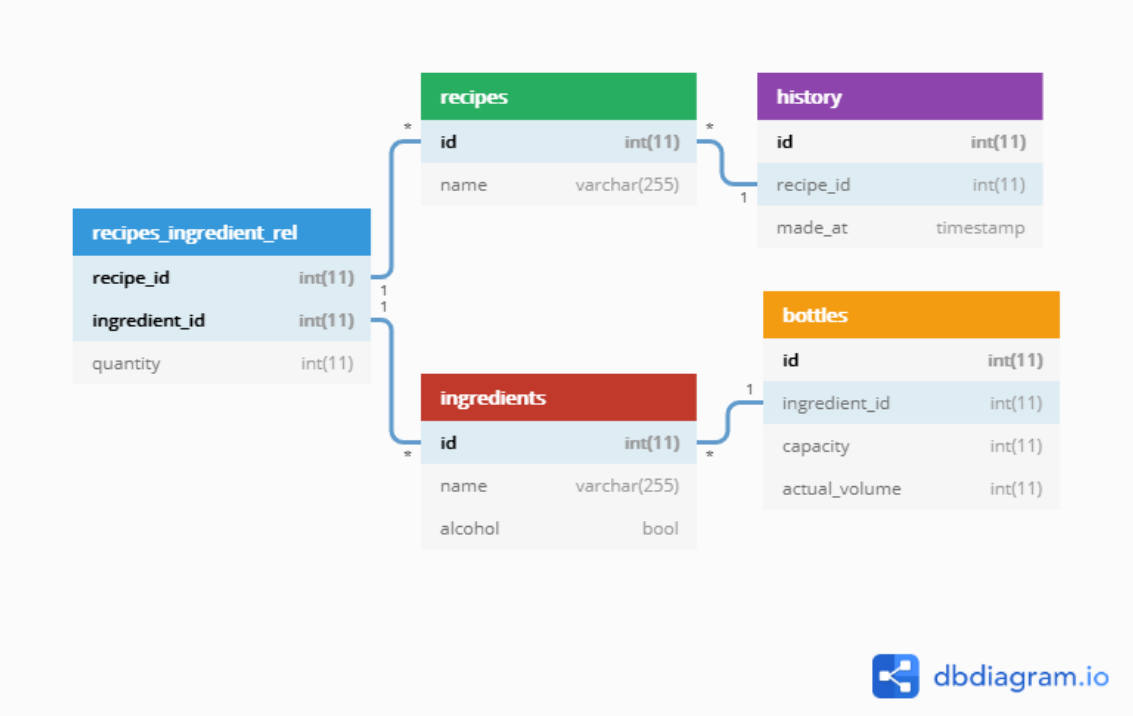

Figure 14: Initial database schema

Later, we could think of an integration with online services such as the IBA which maintains a list of cocktails and their official recipes, to build a pre-populated database.

4.4.4 Making cocktails

Based on the related works and our design, it seemed that the most straightforward solution was to use the 1kg load cell under the glass to detect when to start or stop the pumps.

Thus, we just enable the corresponding digital output of our Orange Pi that is connected to the relay board to power the motor. Then, the load cell measures the increasing weight and after calibration (and adding the density parameter) we were able to compute how much weight was needed for a corresponding volume of liquid. Then, we just disable the digital output of the Orange Pi when that value is reached to stop the pouring.

Here, we used a simplification about the benefit we mentioned earlier. Despite we could be able to enable multiple pumps at the same time, the fact that each liquid should have a precise amount poured down is measured by the weight would not allow us to know when a specific liquid reached its recipe volume.

4.5 User interaction

4.5.1 Web interface

The web interface was developed to allow configuration as well as control of the device. It is made simple and user-friendly thanks to the *Bootstrap* framework [15] which helped to focus on more complicated parts of the project.

A capture of this interface can be seen in Figure 18 and Figure 19.

4.5.2 Google Smart Home ecosystem

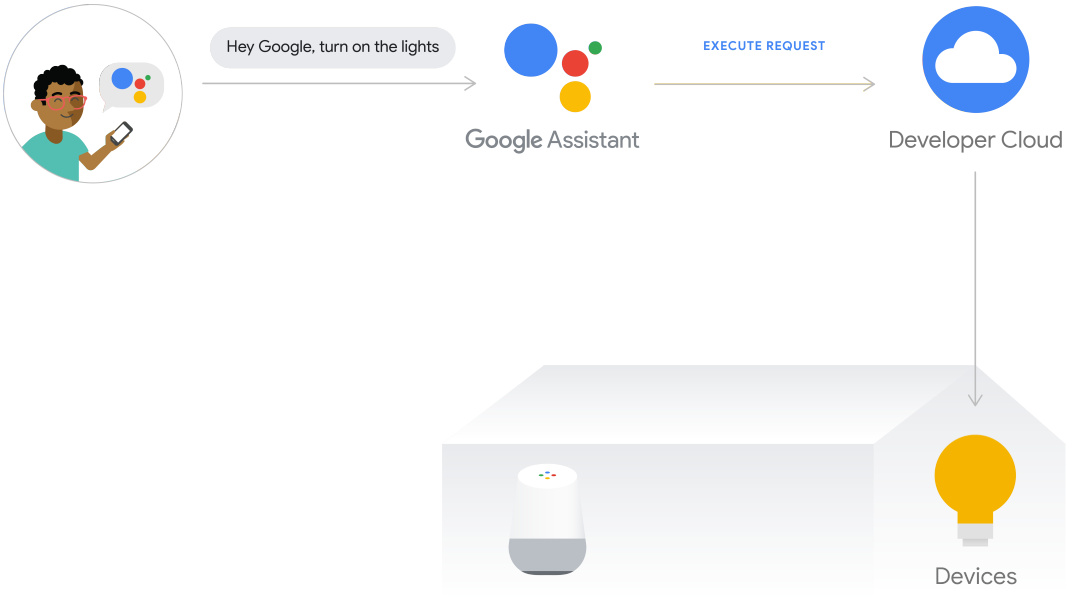

Figure 15: Google Actions

Google Assistant Google Assistant is an artificial intelligence-powered virtual assistant developed by Google that is primarily available on mobile and smart home

devices that can engage in two-way conversations. It allows interaction with devices such as lamps, smart sockets, cameras or thermostats in multiple languages.

Our primary goal was to allow an interaction with this service, to enable users to voice-command their cocktail machine.

Google Actions Google Smart Home Actions is a powerful API to interact and connect devices with the Google Assistant. Assistant handles how users trigger the Action (in multiple languages), the remaining part is to respond to the requests sent by the API.
However we have followed the documentation and samples [5], we were not able to reach our server from the Google Console (the platform to configure all Google APIs). The reason behind this is that our Internet service provider wasn't able to provide us with a public IP address on a 4G/LTE modem, and thus our web server was not accessible from the outside.
With more time and maybe more tests on different connections, we would have been able to tackle this issue.

IFTTT As a fallback, the easier solution we found was to use a free bridge Android application, named IFTTT (If This Then That) that binds with the Google Assistant and allows custom actions, such as sending requests to a server.

Moreover, this tool allows more complex, bidirectional actions by using HTTP Webhooks [7], which allows server responses using POST requests to the configured URL:

```
https://maker.ifttt.com/trigger/<event>/with/key/<api-key>
```

We used this feature to send a notification to the user when there is a new status for the machine: a cocktail was made, a problem occurred (eg: low level in one of the bottles).

4.6 Final design

According to the previous changes brought to our project, we came to a new, final design that we will then evaluate (see Figure 16).

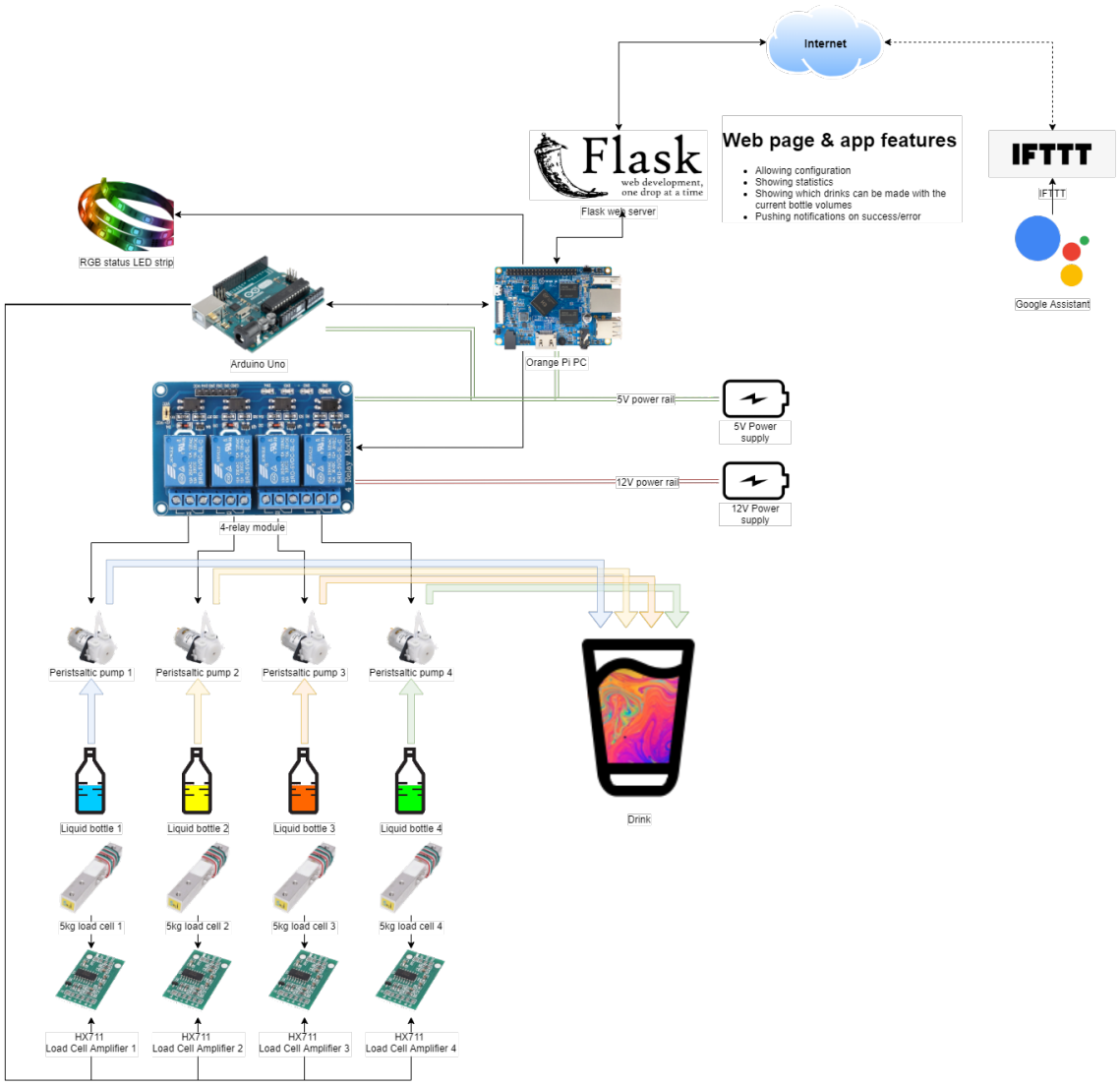

Figure 16: Mixologist - Final design

5 Evaluation

First, we built a working (hardware) prototype, as in Figure 17. That was used as a base for developing the software we were going to use and was easier for debugging than blind testing.

Figure 17: Base prototype, with removed tubings

Our approach was based on the "trial and error" method. We learned for previous designs (from other devices or our own) what worked and what did not. Incrementally, we added features on top of more basic functional code.

After rigorous testing of each part (load cell weight computations, pumps control and timings, LEDs control, proper network configuration and serial data exchange between the Arduino and the Orange Pi) we were able to gather everything and produce the first tests scripts that were able to pour precise amount of liquids.

Then, we tried to build a more user-friendly interface to launch the scripts.
We achieved to produce a web dashboard that allows cocktail making, live status, recipe management and configuration (bottles contents, taring...).

Finally, we added the IFTTT layer mentioned earlier, to allow voice commands and feedback through an Android phone. Since everything was already in place, it was straightforward. We just had to add methods to allow REST requests, using the Flask framework routing system. It works as expected, except that this adds a

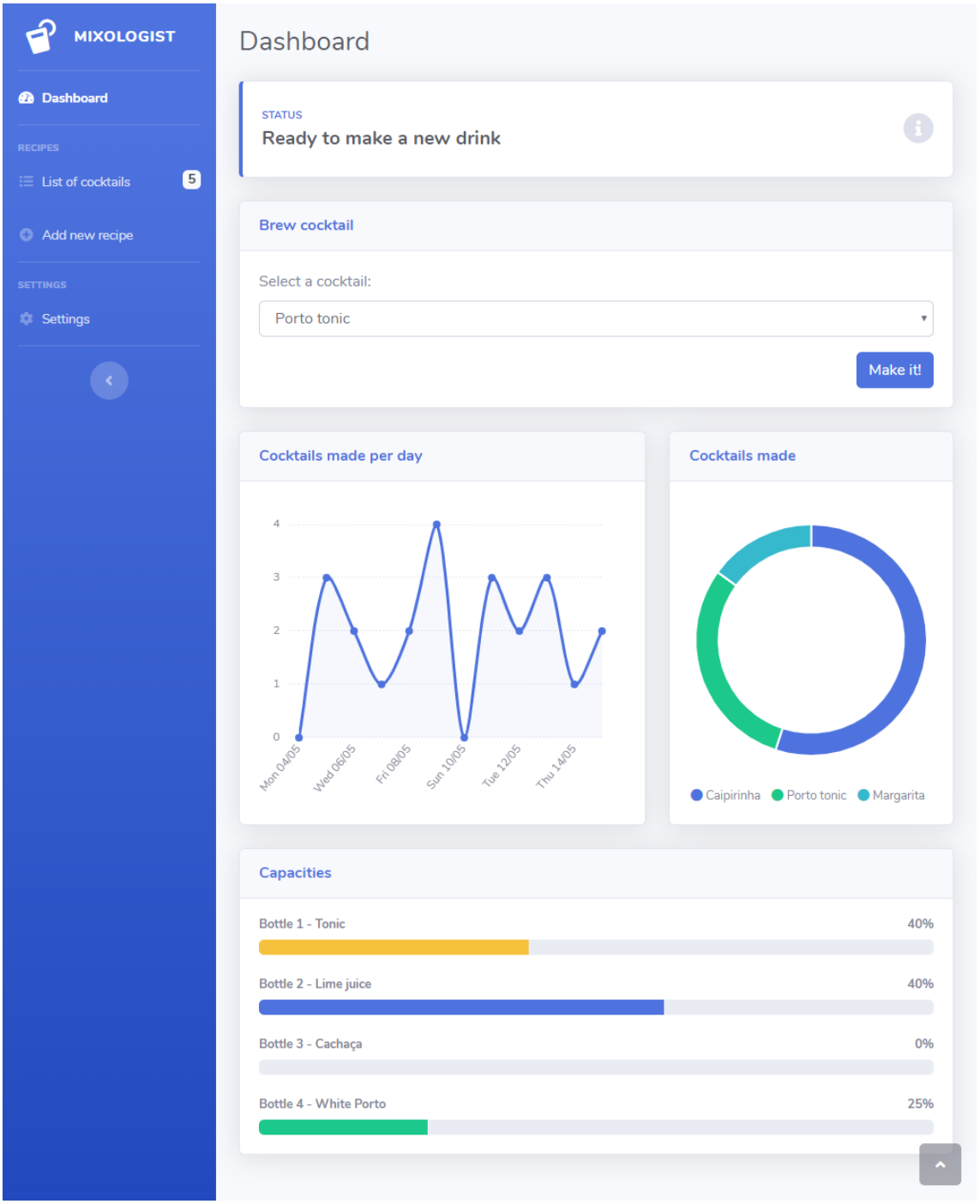

Figure 18: Mixologist dashboard

delay of some seconds to our system because the recognized triggered first goes to the IFTTT servers for analysis.

At the time of writing this report, we did not complete the code of our project yet, but most separate features are working as expected:

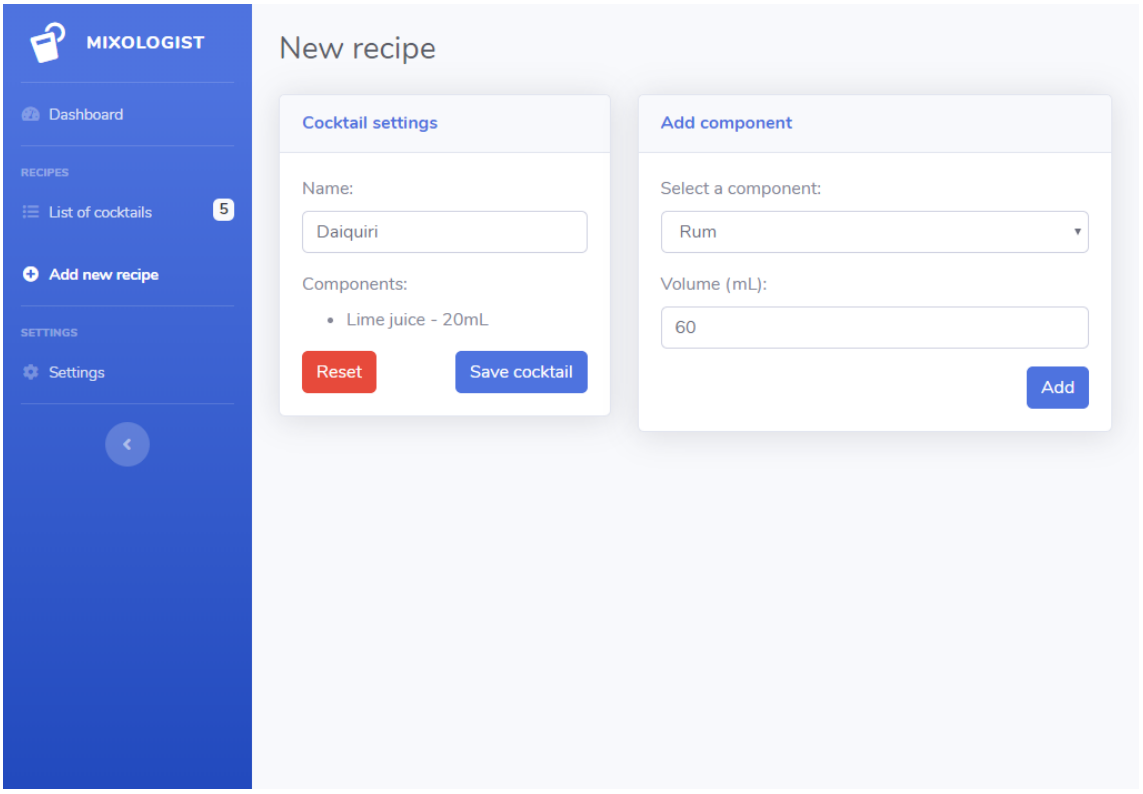

Figure 19: Mixologist new recipe form

- Cup filling up to the specified weight/liquid quantity
- Beverage pouring only if a cup is present
- Cocktail making only if the quantities of the required ingredients are available
- Computation of the number of cocktails that it is possible to make with the current bottles
- Alerts on empty bottles / full glass
- Storage, management of recipes in a database
- Voice control and interaction
- Notifications for status updates

The source code in the Github repository will be updated by the date of the project presentation.

6 Conclusion

This project happening in special circumstances gave us quite a lot of difficulties to overcome.
This pushed us to work harder on searching for solutions instead of going for the simplicity of just buying and changing a component.

We are quite satisfied with our project, however, a lot of features could still be implemented and a lot of things could be improved, to name but a few:

- testing if the B channel of HX711 would really loose in precision
- enhance the HX711 boards precision and data gathering speed by connecting the *RATE* pin to the *VCC* to get 80 samples per second instead of the default 10 samples per second.

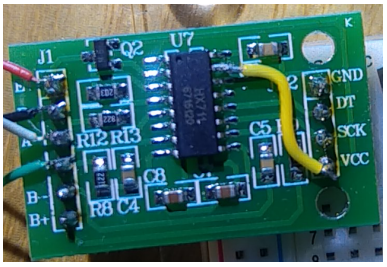

Figure 20: HX711 module modification - Photography of gandalf15 Github user [4]

- replacing the 3W DC motor with more powerful ones and larger tubings for faster cocktails making (however, it would be interesting to calculate the limit, to avoid splashing and vibrations)
- replacing the 5V power supply with a buck/DC-DC converter to use more efficiently our 12V power supply
- update our algorithm so that we would be able to pour multiple ingredients at a time
- develop an Android application to interact with the machine, instead of just a web page
- implement a simple authentication protocol, even if the device is in the LAN, a malicious attacker could trigger the actions by finding the web server endpoints and mess with the owner
- reduce the power consumption to a minimal by disabling uused features on the Orange Pi PC (HDMI output, Gigabit Ethernet, GPU...)
- implement automatic restocks, using Amazon APIs for example when the bottles are almost empty (or at least writing a shopping list)

- enhance the web interface to filter cocktails by ingredients, or to make only "virgin"/alcohol-less cocktails
- automate the brewing of drinks according to calendar events (meetings, parties...)
- add a dispenser for solid ingredients (mint, lime, citrus, ice...)

Thanks to the democratization of cheap, open-source electronic and embedded solutions like the Raspberry Pis, the Arduinos and the growing community or makers (online or physical, in Fab labs and workshops), more and more people will be eventually designing their own systems and machines, and make simple tasks of their daily routine automated.

As we showed in this report, despite some concepts are hard to understand and to put into practice, complete resources are available online and are just waiting for people's creative ideas to be put together.

Glossary

API Application Programming Interface, computing interface which defines interactions between multiple software intermediaries. 19, 24

Arduino Open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits. 2–4, 8, 9, 11, 15, 16, 21, 25

bar optic Trademarked pub measure created by Gaskell and Chambers, that operates as a device which is mounted in the neck of an inverted spirit bottle and dispenses a measure of alcohol when the lever is pushed or pulled. 6, 8

C Compiled, low-level, general-purpose, procedural programming language developed at Bell Labs by Dennis Ritchie between 1972 and 1973. 16

database Organized collection of data stored and accessed electronically from a computer system. 3, 4, 9, 17, 23

DBMS DataBase Management System (DBMS), a software system that enables users to define, create, maintain and control access to the database. 17

GPIO General Purpose Input/Output, a type of pin found on an integrated circuit that is customizable and can be controlled by software. 15, 16

IBA International Bartenders Association. 17

load cell Transducers that converts a force such as tension, compression or pressure into an electrical signal. 2–4, 8, 9, 11, 14–17, 21

Orange Pi Open-source single-board computers developed by Xunlong. 2–4, 11, 12, 15–18, 21

peristaltic From the peristalsis, a radially symmetrical contraction and relaxation of muscles that propagates in a wave down a tube. 2–4, 8, 13

Python Interpreted, high-level, general-purpose programming language created by Guido van Rossum in 1991. 16, 17

Raspberry Pi Single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and developing countries. 9, 11, 12, 15, 16, 25

REST REpresentational State Transfer, software architectural style that defines a set of constraints to be used for creating Web services. 8, 21

SBC Single-board computer, a complete computer built on a single circuit board, with microprocessor(s), memory, input/output and other features required of a functional computer. 4, 12, 13, 15, 16

SPI Serial Peripheral Interface, a synchronous serial communication interface specification used for short-distance communication. 13

web server Server software that can satisfy client requests on the World Wide Web. 19, 24

References

[1] Barsys. *Barsys | Bring The Bar Home – Barsys LLC.* URL: https://thebarsys.com/ (visited on 05/15/2020).

[2] Wikipedia contributors. *Peristaltic pump - Wikipedia.* URL: https://en.wikipedia.org/wiki/Peristaltic_pump (visited on 05/15/2020).

[3] DoubleHP. *Getting OrangePi Zero work with WS2811.* URL: http://www.orangepi.org/orangepibbsen//forum.php?mod=viewthread&tid=3318&page=1&extra=#pid21903 (visited on 05/15/2020).

[4] gandalf15. *gandalf15/HX711: Read HX711 ADC for Weigh Scales on Rasperry PIs.* URL: https://github.com/gandalf15/HX711 (visited on 05/15/2020).

[5] Google. *Actions on Google.* URL: https://console.actions.google.com/ (visited on 05/15/2020).

[6] GreatScott! *Make your own crude Cocktail Machine - YouTube.* URL: https://www.youtube.com/watch?v=Z7GkGeZrb2Y (visited on 05/15/2020).

[7] IFTTT. *Webhooks works better with IFTTT.* URL: https://ifttt.com/maker_webhooks (visited on 05/15/2020).

[8] joosteto. *joosteto/ws2812-spi: python routines to program the WS2812 RGB LED chips on the raspberry, using the hardware SPI MOSI.* URL: https://github.com/joosteto/ws2812-spi (visited on 05/15/2020).

[9] Chris Liecht. *pySerial 3.0 documentation.* URL: https://pythonhosted.org/pyserial/ (visited on 05/15/2020).

[10] DIY Machines. *Robotic Bartender Arduino Project - YouTube.* URL: https://www.youtube.com/playlist?list=PLS1TjqbD3o7ctNpTd9CtWJFY8zvc2RQTd (visited on 05/15/2020).

[11] olkal. *olkal/HX711_ADC: Arduino library for the HX711 24-bit ADC for weight scales.* URL: https://github.com/olkal/HX711_ADC (visited on 05/15/2020).

[12] Pallets. *Flask Documentation (1.1.x).* URL: https://flask.palletsprojects.com/en/1.1.x/ (visited on 05/15/2020).

[13] AVIA SEMICONDUCTOR. *HX711 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales datasheet.* URL: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf (visited on 05/15/2020).

[14] Makr Shakr. *Makr Shakr – Your drink with a splash of robotics.* URL: https://www.makrshakr.com/ (visited on 05/15/2020).

[15] Bootstrap team. *Bootstrap · The most popular HTML, CSS, and JS library in the world.* URL: https://getbootstrap.com/ (visited on 05/15/2020).

[16] TripAdvisor. *Tipsy Robot (Las Vegas) - Tripadvisor.* URL: https://www.tripadvisor.fr/Attraction_Review-g45963-d12831317-Reviews-Tipsy_Robot-Las_Vegas_Nevada.html (visited on 05/15/2020).

[17] Siobhan Wagner. *The $110,000 Robot Bartender Mixing Great Cocktails - Bloomberg.* URL: https://www.bloomberg.com/news/articles/2019-

08-14/the-110-000-robot-bartender-mixing-great-cocktails (visited on 05/15/2020).

[18] Worldsemi. *WS2812 Intelligent control LED integrated light source datasheet.* URL: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf (visited on 05/15/2020).