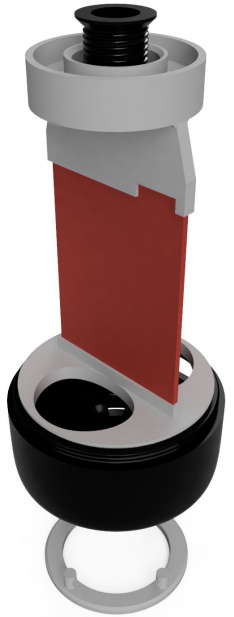
Unimore Smart Bottle

Colombari Mattia, Pirrelli Marco

Water bottle

In the bottle we use, as a microcontroller, an Arduino Nano that communicates with different modules/sensors:

- HC-05 Bluetooth module: Serial communication, we also retrieve the status of the connection (connected or not).
- Battery module: provide battery charge.
- GY-521: Accelerometer connected with I2C.
- FSRs Force Sensing Resistor, there are each in a voltage divider, and we read the analog value.
- LEDs (addressable ws2812b led): connected with a NRZ (Non Return to Zero) protocol.



Bottle states

Internal bottle state:

0. Idle: Starting state where the bottle is moving or not vertical, we get back to this state (form 1 or 2) if the bottle is moving or is not vertical.
1. Waiting: The bottle is steady and vertical, so now we wait before reading.
2. Reading:
    a. We start reading from the weight sensor, every 300 ms, and we fill a buffer of value.
    b. When the buffer is full then we send data to the application (if we have a connection), or print the values on the serial monitor.
3. Resting: after sending the data we wait to restart the cycle, this state doesn’t depend on the position of the bottle.

Bottle flags

We define two variables to verify the state of the bottle: is_still and is_vertical. These boolean variables are updated over the values read on the GY-521

From the sensor, we obtain acceleration data for each axis. Each reading is stored in a vector of 10 elements called *values_x*, then, from that vector, we calculate the means and store it in another vector called *means_X* that we use in the end to study the bottle movement, using the mean helps reduce the noise from the sensor.

- *is_still* variable is obtained by calculating the max distance between two values in the *means_X* vector.
- *is_vertical* variable is obtained by calculating the mean for each axis and comparing it to a set of thresholds.

Each buffer dimension is designed to not exceed the microcontroller memory limit.

Dispenser

The dispenser can be accessed only by our bottle, it reads the NFC tag embedded inside the bottom of the bottle, with a RFID-RC522 sensor, where the bottle ID is stored.

For our implementation we consider a water dispenser not connected to the water system, so with a finite amount of water, we have a potentiometer that simulate the water level detected in the system.

The if the water level is too low the bridge send a distress signal to the server.

Recommendations

When a user refills a bottle at a water cooler, the event is recorded along with the timestamp in the database. Such data is then used to construct recommendations through the python library “Implicit”.

The prediction is done with a collaborative filtering approach: rather than describing items (the dispensers) through a set of attributes and suggesting similar items to the ones a specific user has interacted with, we look for users who behave similarly to the subject and suggest items that they have enjoyed.

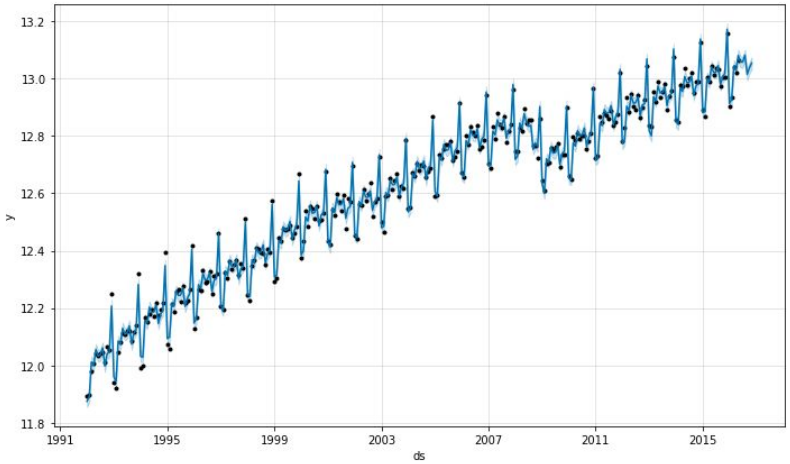
This approach works well in the context of water coolers because the coolers’ only distinguishing feature is their position. Nonetheless users, by interacting with them, will still create patterns.

Engagement

Another key factor in the user's choice is how busy a cooler is.

Cooler usage data is also employed to train an engagement predictor model with the python library "fbprophet".

The model, then, is able to calculate estimates on the amount of people that will interact with a dispenser in a particular timeframe (1 hour).



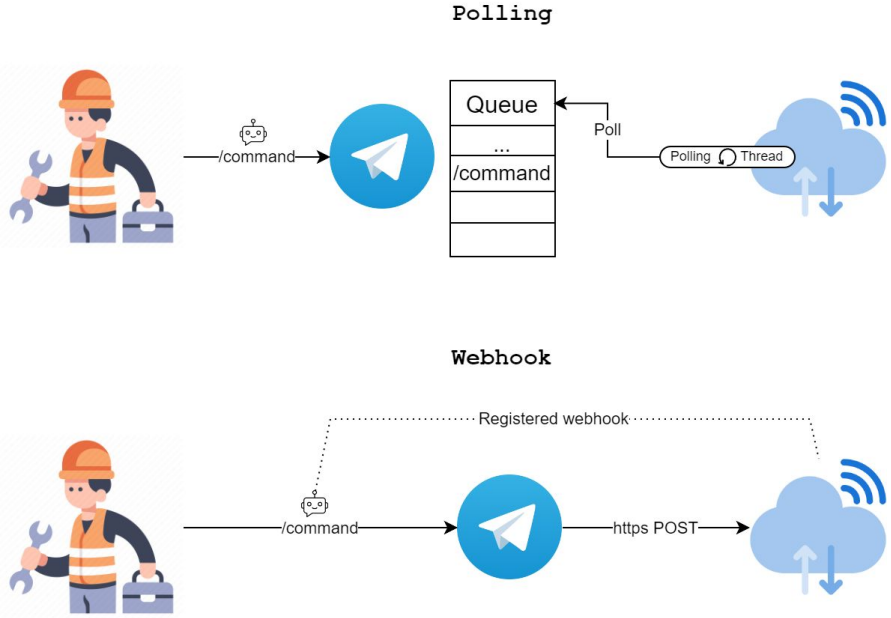
Handling Telegram commands

Telegram allows 2 paradigms for bot programming:

Polling - A dedicated thread continuously loops while polling the Telegram server for updates. Whenever the user sends a command, it is inserted in a queue and will eventually be handled.

Webhook - An https enabled web resource is linked as a webhook for the bot. Whenever a new command is sent, it immediately gets redirected to the webhook to be processed asynchronously.

Since the server is already a looping thread, a webhook paradigm fits the architecture better.



The server

The server is coded with Python using Flask and is hosted remotely on pythonanywhere.com, which includes a MySQL database. The connection to the db is realized through the MySQLdb module.

The default domain at *.pythonanywhere.com* also automatically comes with a https certificate, which is needed to set up a Telegram webhook.

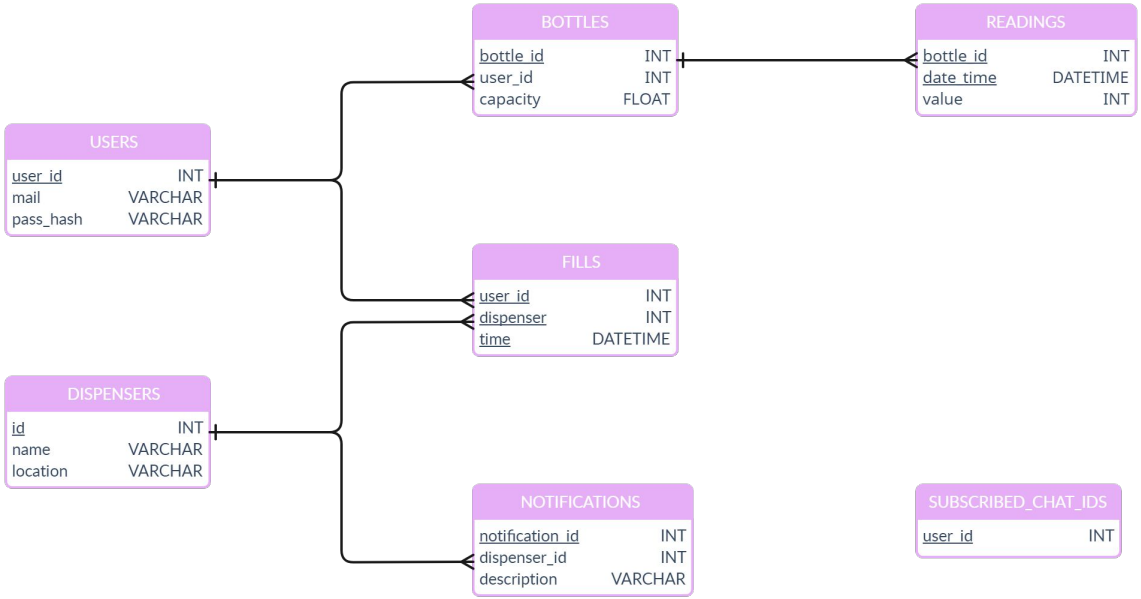
GET

/water_drank/<int:user_id>
/recommendations/<int:user_id>
/engagement/is_busy/<int:dispenser>

POST

/readings
/fills
/notifications
/bottles/register
/telegram

Database MySQL



Thank you for your attention and consideration

Github

Unimore Smart Bottle

Colombari M. and Pirrelli M.

Github