POLYTECH GRENOBLE

UNIVERSITÉ Grenoble Alpes

RICM4 PROJECT RobAIR Tweet

JEAN Jordan, EZ-ZINE Najwa

2017 - 2018

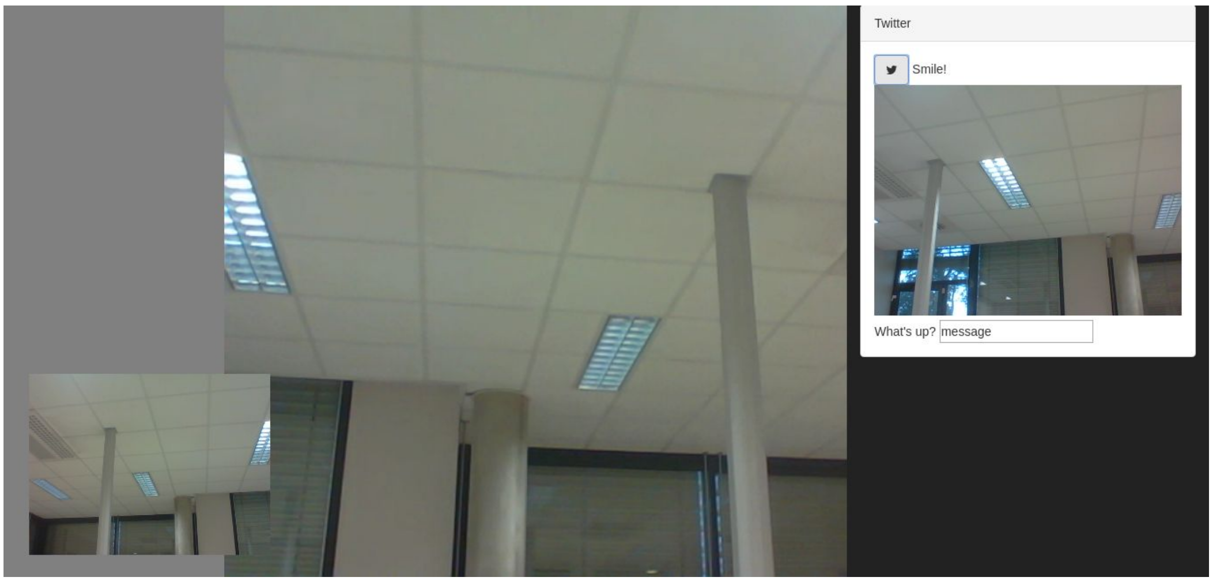
Abstract

RobAIR is an open source and low cost telepresence robot which aims to virtually reproduce a human presence. It can have a lot of uses such as remotely visiting a museum or remotely taking a course. Our project consists of adding a feature to RobAIR which allows it to take pictures using its embedded camera, and then post it on twitter with a text message. We also have to take into account the user experience and the image rights. Pictures have to be posted on social networks after confirmation by the person who appears on it. Many extensions could be considered such as face detection and motion detection. This feature could give an international dimension to the RobAIR project and promote it during public events such as Open Days.

1. Technologies and achievements

● WebRTC

WebRTC[1] is a free open source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps. RobAIR video communication is based on WebRTC. We had to study this technology in order to create a taking picture service. A button on both client and server interfaces calls a function which capture the current picture of the video stream.



RobAIR interface with twitter container

● Twitter API

The twitter API[2] provides services to make authenticated requests to the Twitter platform and makes RobAIR able to post text messages and pictures on twitter. We used the Twit[3] NodeJS library to use this API. First, the content of the tweet, meaning the picture and the text message, are sent to RobAIR NodeJS server using a POST request. Then the server use the Twit library to post the tweet.

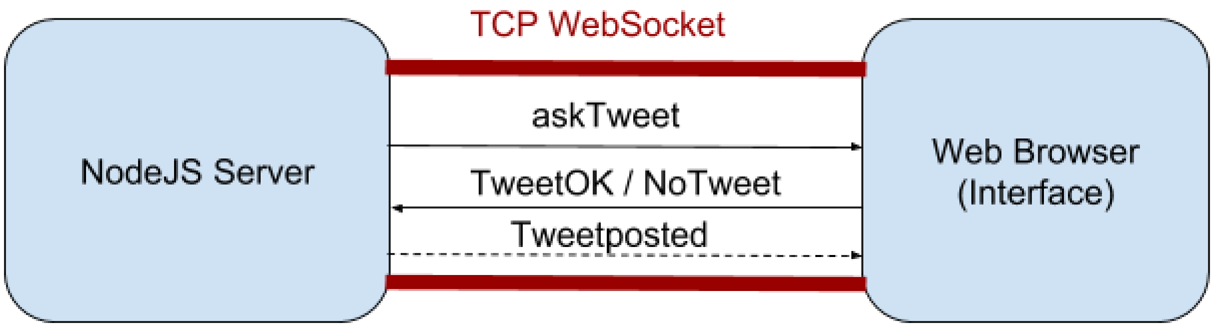
● WebSocket

WebSocket[4] is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. Before posting a picture on twitter

---

1 https://en.wikipedia.org/wiki/WebRTC
2 https://developer.twitter.com/en/docs
3 https://github.com/ttezel/twit
4 https://en.wikipedia.org/wiki/WebSocket

we need to ask permission to the persons who have been photographed. To do so we need to have a connection between the NodeJS server and the web browser on RobAIR to send an alert on RobAIR interface. We used websockets to implement this solution as described on the following figure. When the server receive a picture and a text message from the client or the server, it send a message to the RobAIR browser to ask permission to post. Depend on the user answer, the browser replies OK or not OK for tweeting.

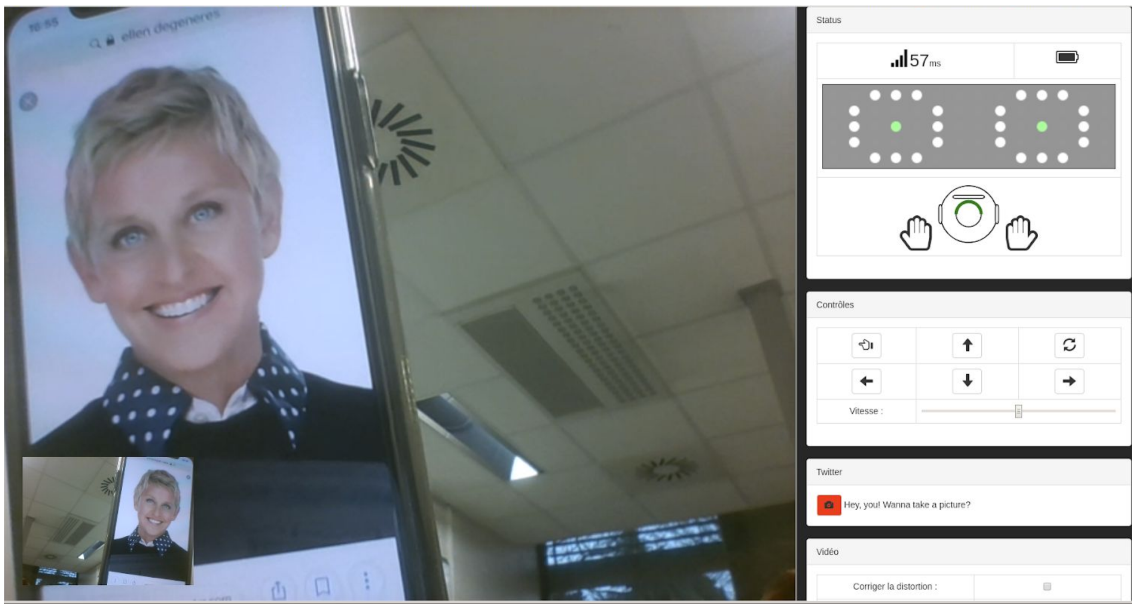


- ## Face tracking

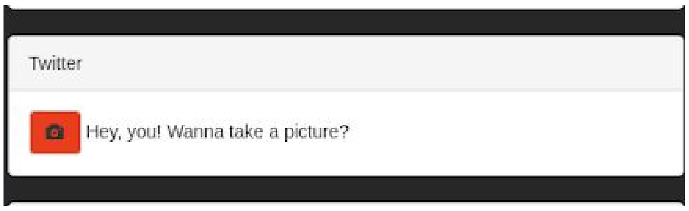
To put in place a face tracking system we used the tracking.js library. This library offers countless features, and easy to use tools, all that without having your processor go into overdrive. Basically, the library was chosen for the improvements it can offer and its fitting to RobAIR requirements.

How did we put it in place?

To make it simple the library, and more precisely the face detection feature, uses an object called a tracker. You simply tell the tracker to track a face and collect the events. When a face is detected, we set the text, message wanted in the dedicated area as shown below. (“Hey you! Wanna take a picture?”)



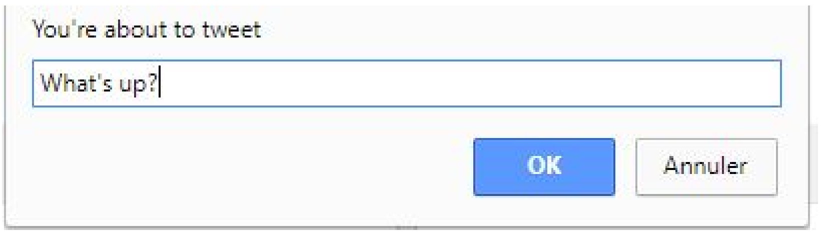
Hey, you! Wanna take a picture?



- ## Bootstrap

Bootstrap one of the most popular project on GitHub, is a framework developed by Twitter. Based on HTML, CSS, SASS and JavaScript, it allows you to create design for websites but also web applications.

We wanted to establish a more formal dialog with the user when tweeting. Our first solution used the JavaScript window prompt method. When tweeting, a message was displayed, allowing you to tweet, to customize your tweet post message and confirm it.



However, with such a method we encountered several setbacks.

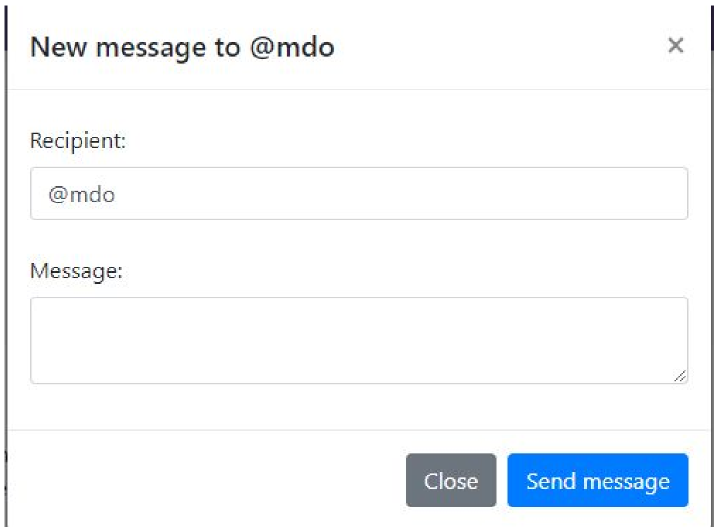
As a major setback, it was impossible for the user to check the picture before posting, as the prompt function stopped the background execution and asked to be treated first. Only after posting, meaning after treating the prompt, could the user see its picture displayed in the twitter container.

As a minor setback, the prompt wasn't really user-friendly, the interface wasn't very pleasant, several weaknesses that eventually led to a poor tweeting experience.

We came up with another solution : deleting the prompt method and adding a textarea to type the message. The user would also be able to see the post result as a text displayed overlapping the picture. At this point we had no problems with undisplayed image but the overall tweeting experience was still poor.

That's when we discovered bootstrap and more precisely its JavaScript modal[5] plugin that allowed adding dialogs and custom content. In order to implement it, we adapted an already existing code (see varying modal content section of 5th note). We added a canvas to draw the image taken and 2 buttons to tweet or cancel. Bootstrap provide us with several style of buttons, we took advantage of the blue color of the info class button that's similar to the twitter color. But also of the red color of the danger class button to cancel the tweet.

[5] https://getbootstrap.com/docs/4.0/components/modal/


New message to @mdo

Recipient:

@mdo

Message:

- ## Timer

We wanted to have a countdown displayed when clicking on the button to take a picture. It would give 5 seconds to the user to prepare its picture. To implement this, we started from code[6] found on stackoverflow, by Tom Koker. We then customized it to trigger the tweeting function and dynamically set the text.

2. Project management

- ## Methods

To manage this project we used one of the most famous agile methodology : SCRUM. Guided by this method, we choose to deliver our product in an incremental and iterative way. However due to the size of our team, we had to adapt the method, implying both members somehow played more or less every role needed. Stand-ups and all sorts of meeting advised by this method were also adapted to the size of our team, meaning they weren't as formal.

Each sprint lasted one week and each time we started by asking each other what we were going to do and who was responsible for what. Of course, we also evaluated the difficulty of each goal set. During the process, we took time to have a sort of stand up while sitting to check if we were going to make it and if everything was going on well. After reviewing, we took a break and took a deeper look at what had been made and how it could've been improved.

Throughout the process, we were able to adapt and put in place our tutors requests without overwhelming ourselves with unbearable loads of work.

[6] https://stackoverflow.com/questions/31106189/create-a-simple-10-second-countdown/31106229

- Setbacks
  - Installation of RobAIR : Despite having a detailed guide in the README of the project, we encountered several problems when installing RobAIR, including missing some modules, version problems, etc.
  - Little to no experience with languages required : The general architecture of the project was somehow complex but the major setback was not knowing most of the languages or technologies used. We didn't know anything about the JavaScript language, implying it was very hard to, at first, understand how the part we had to work on worked. During our training, we happened to work with HTML and CSS. But as they weren't part the main purpose of the project, our knowledge in this domain remained little.

3. Conclusion and ideas for the future

This project was the opportunity to take part in a famous project and as an introduction to the robotic world, it allowed us to get a broader view of Computer Sciences. We end this project with brand new and sharpened skills as we have learned a new language : JavaScript but we also gained tremendous experience with web technologies. Finally, we learned how to quickly adapt to an unfamiliar environment. We also were lucky enough to be able to work at the Fablab in direct contact with the main contributors to this project and got a deeper understanding of the initial project.

The work we provided makes it very easy to now take pictures. New features using it could be implemented.

The same goes for social networks. We mainly worked on twitter but other social networks could be used as a further improvement. Unfortunately, posting on Instagram was one of our goals but for economical reason they disabled this feature.

As previously stated, tracking.js offers countless features such as games, the possibility to draw on the canvas and these could be developed in the project.

Finally, tracking.js has a pretty low accuracy, when trying to detect faces.