HCI

ARAGONA DARIO

MAUTONE LUIS

PODO LUCA

PAPERBAR

A NEW WAY TO TYPE

WHAT IS A PAPER TOUCH BAR?

A CHEAP AND EASY TO USE TOUCH BAR FOR ALL PC WITH A WEBCAM

BASIC ASSUMPTION

EVERYONE KNOWS HOW TO USE A KEYBOARD, WE DON'T HAVE TO INTRODUCE RADICAL CHANGES

THE FIRST IDEA

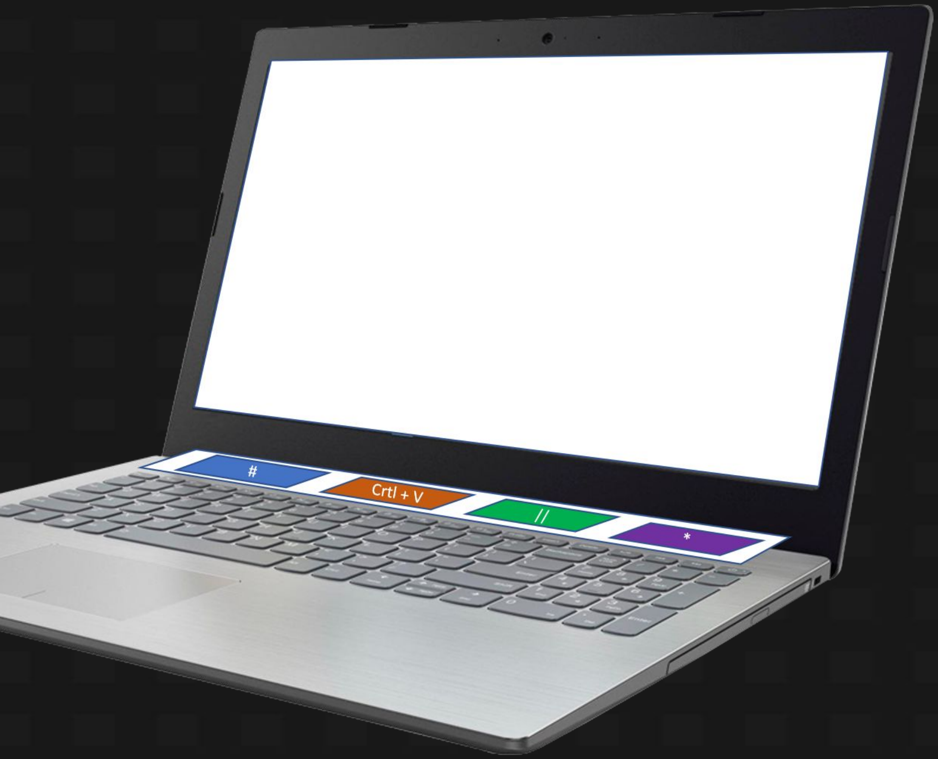
Our first goal was to reproduce a working number pad using only secondary webcam and a piece of paper.

After technical analysis and user interviews we decided to use a different way to realize our prototype, because of the many technology problems that would make the interaction more complex and the problem related to the second webcam.

Problems like: occlusion, illumination variation, position w.r.t to the camera, user movements limitation using the second camera. All these would have required a difficult calibration and limitations in context of use

HCI

THE SECOND IDEA

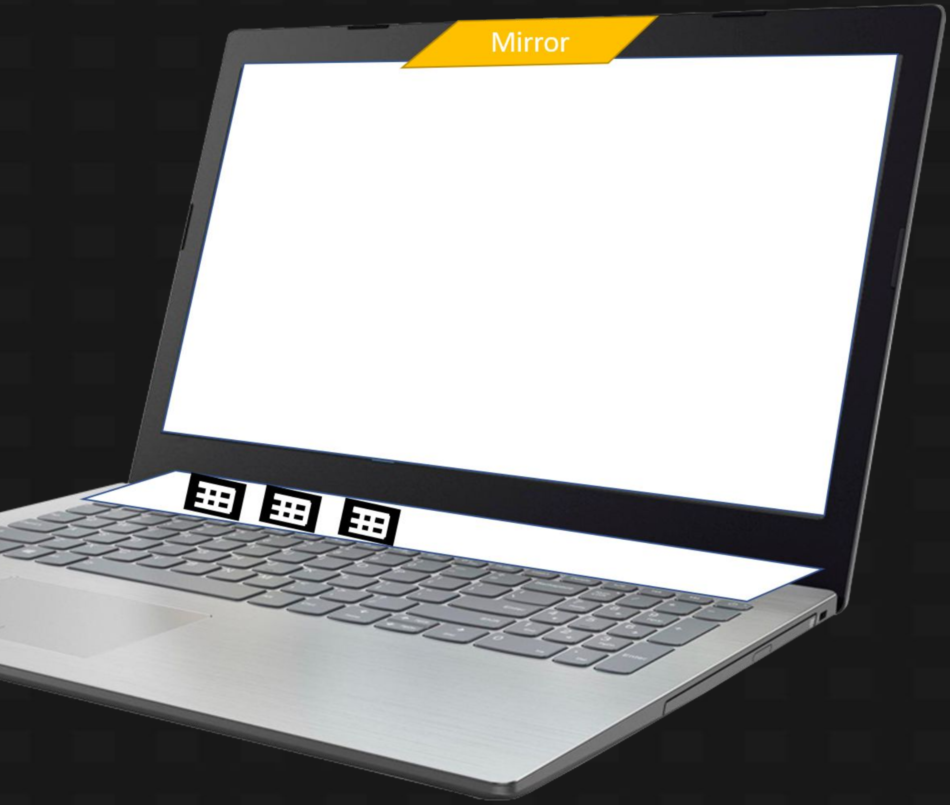


Based on feedback received and the problems found, we moved to the concept of touch bar introduced by Apple in order to solve the problem of occlusion and to avoid the use of the second camera, by exploiting the webcam of the pc with a mirror. This by placing the paper like in the image.

The concept produced was a linear array of colored button. This solution was based on finger tracking and color detection.. This helped us to solve the occlusion problem and position w.r.t the camera, but not the illumination sensibility of colors.

HCI

THE FINAL IDEA



The final solution was based on the study of the previous ones.

The layout was still a linear one, but this time we didn't used the colors and finger detection but we proposed a variation of Aruco Marker used in AR.

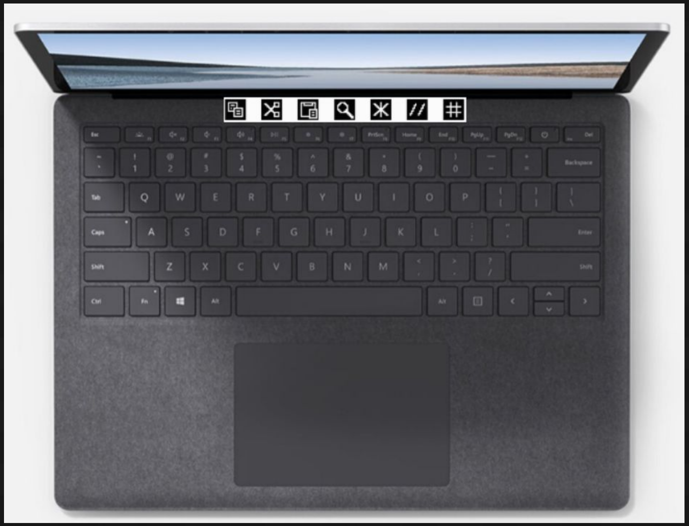
We used this marker, modified for our case, to create simple icons and simulate a keyboard of shortcuts based on the app that the user is using.

This solution is more illumination stable and solve the occlusion problem definitely.

HCI

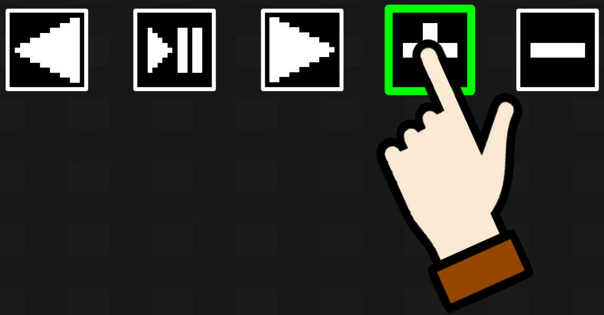
HOW DOES IT WORK? (FOR USER)

1. Buy and plug the mirror



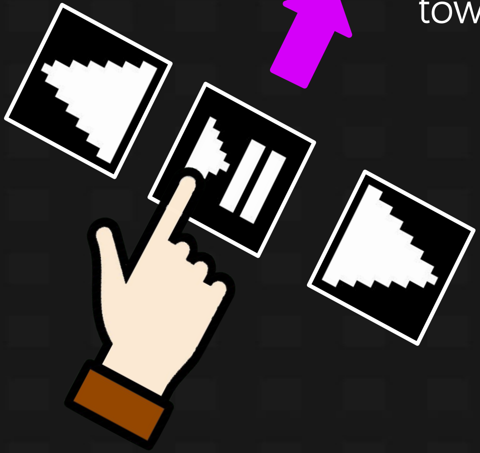
2. Print and place the keyboard

3. Start the app, let it recognize the keyboard and start to type



HOW DOES IT WORK? (TECHNICALLY)

1. The user touches a button on paper correspondent to a visual marker



2. A reflective surface right above the keyboard redirects the image perceived towards the webcam

3. Laptop internal webcam sends image signal to our software

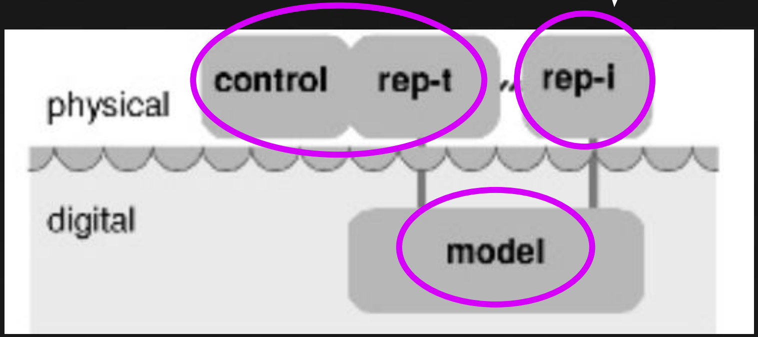
4. The program detects the marker occlusion and identifies it as a "button pressure"

TECHNOLOGY

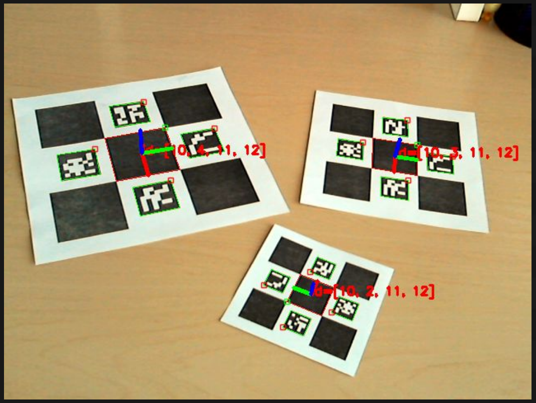
Our keyboard is an "interactive surface" and its interface belongs to the augmented reality field: it uses visual markers on a tangible object.

The **paper keyboard** represents and controls the pressure of buttons

Actions executed upon a program, **visual effects**, **sound feedback are the intangible representation of information**



The **software** contains the computational coupling of the markers with individual or combined pressure of keys



Pose estimation is of great importance in augmented reality: this process is based on finding correspondences between points in the real environment and their 2d image projection.

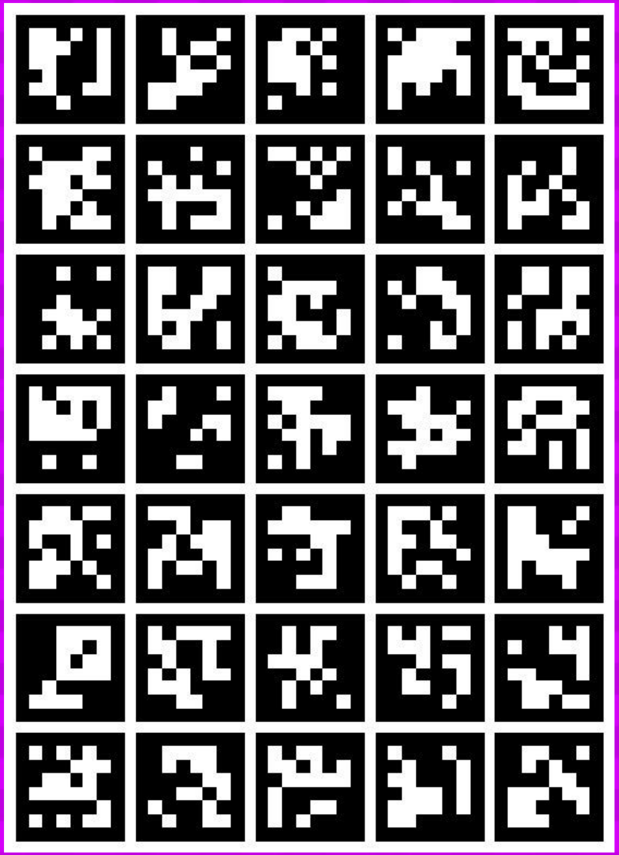
The use of binary square fiducial markers in **OpenCV** is one of the most popular approaches for pose estimation in Computer Vision.

The main benefit is that a single marker provides enough correspondences to obtain the camera pose. Also tag-specific algorithm, such as the inner binary codification, are fast and accurate.

The aruco module in OpenCV is based on the Aruco library, a popular library for detection of square fiducial markers.

An **ArUco marker** is a synthetic square composed by a wide black border that facilitates its fast detection and an inner binary matrix which determines its identifier.

A dictionary of markers is the list of binary codifications of each of its markers. It can be configured manually, so that any codification can be employed.

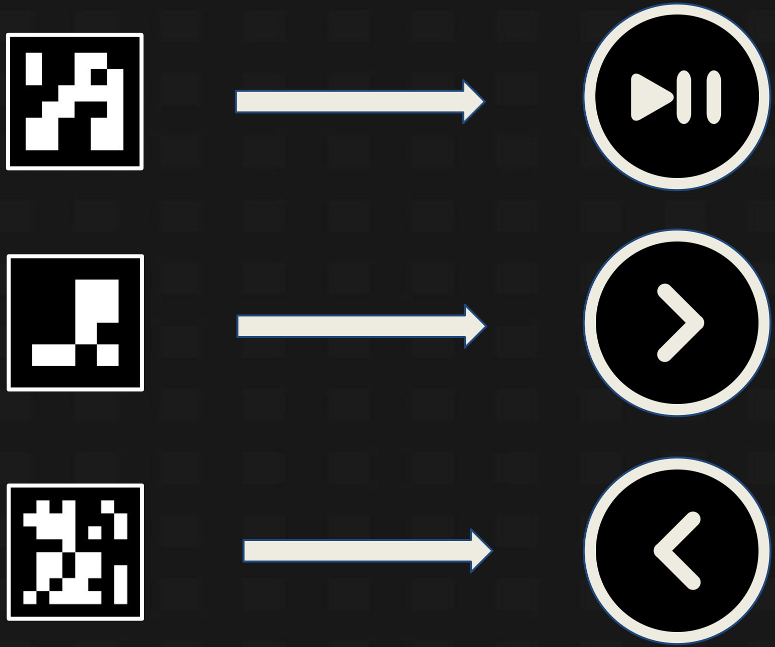


HCI

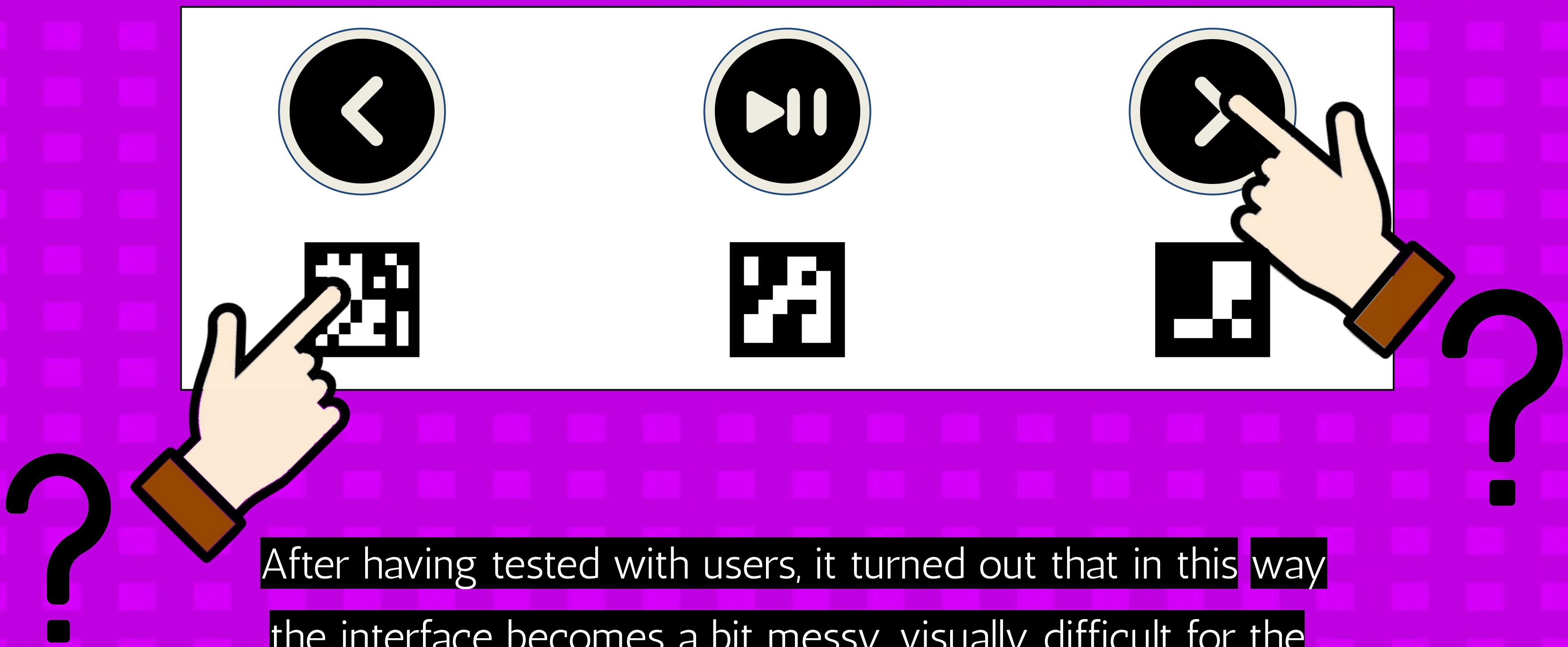
KEYBOARD ICONS - 1

Our first idea was to associate an ArUco marker with each key.

In this way, pressing the ArUco marker encodes the function of the associated button.

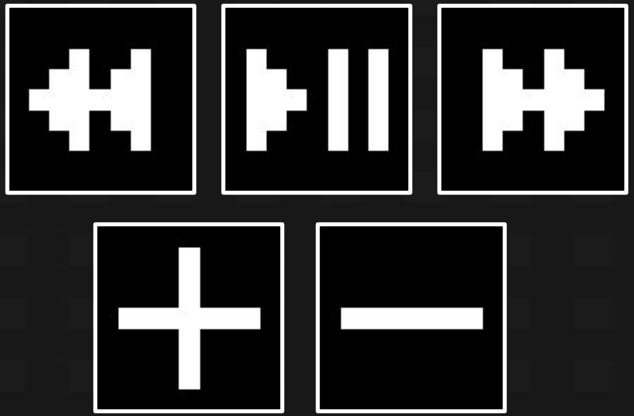


KEYBOARD ICONS - 1



After having tested with users, it turned out that in this way the interface becomes a bit messy, visually difficult for the users to understand where to press.

KEYBOARD ICONS - 2



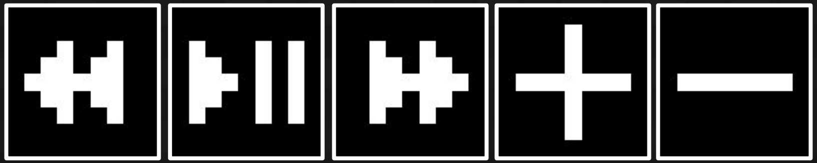
A different approach is to directly give semantic meaning to markers.

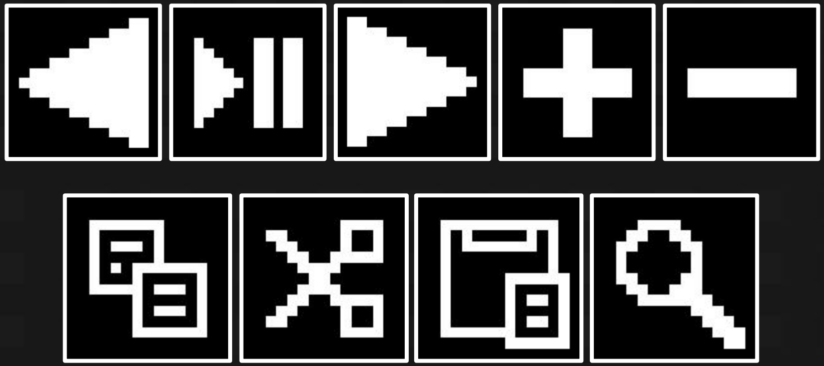
This was possible thanks to ArUco system, which allows to create custom markers given a binary matrix.

HCI

KEYBOARD ICONS - 2

We compared different resolutions in order to find a balance between the size of the markers and their visual understandability.

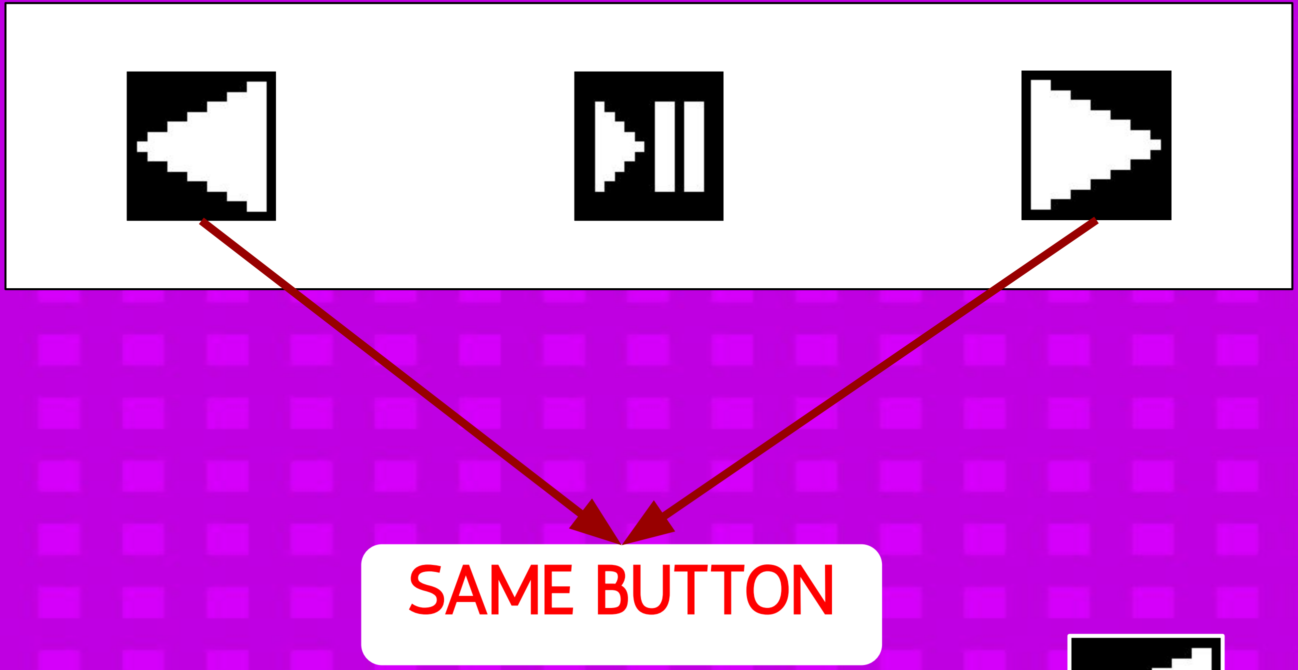

7x7 pixels

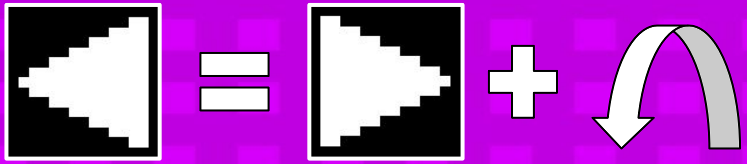

13x13 pixels

Finally, we found out 13x13 pixels resolution is the best option in order to create complex buttons, like the ones used for text editing

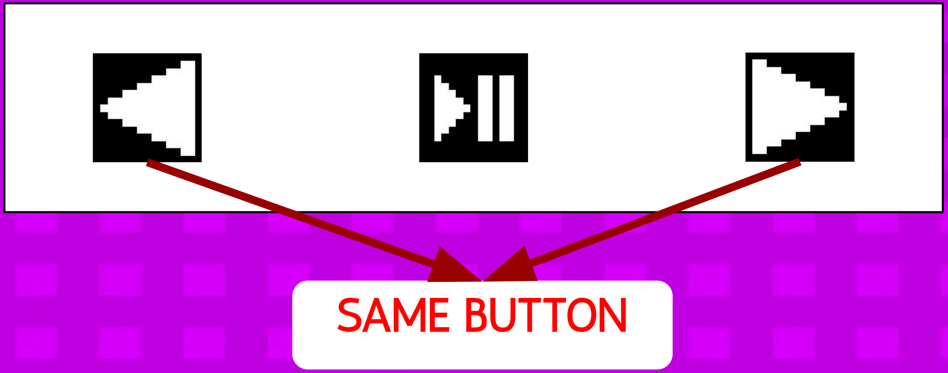
KEYBOARD ICONS - 2

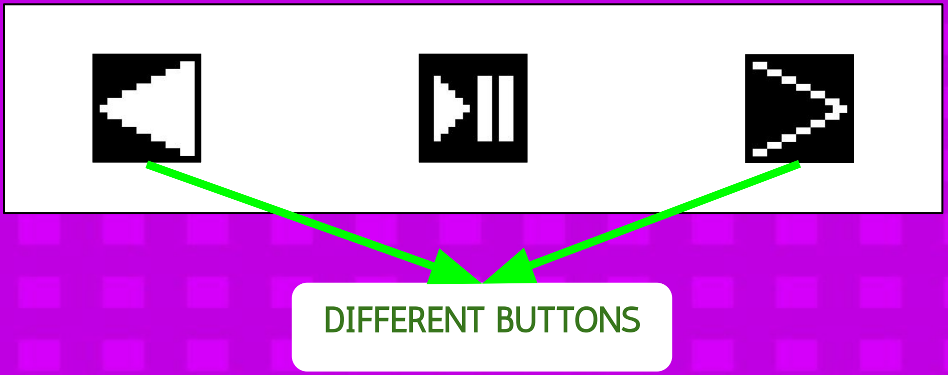
The only problem in this case is the symmetry of markers: two different buttons cannot be represented by symmetrical markers because ArUco system is pose-invariant and will recognize them as the same.





KEYBOARD ICONS - 2

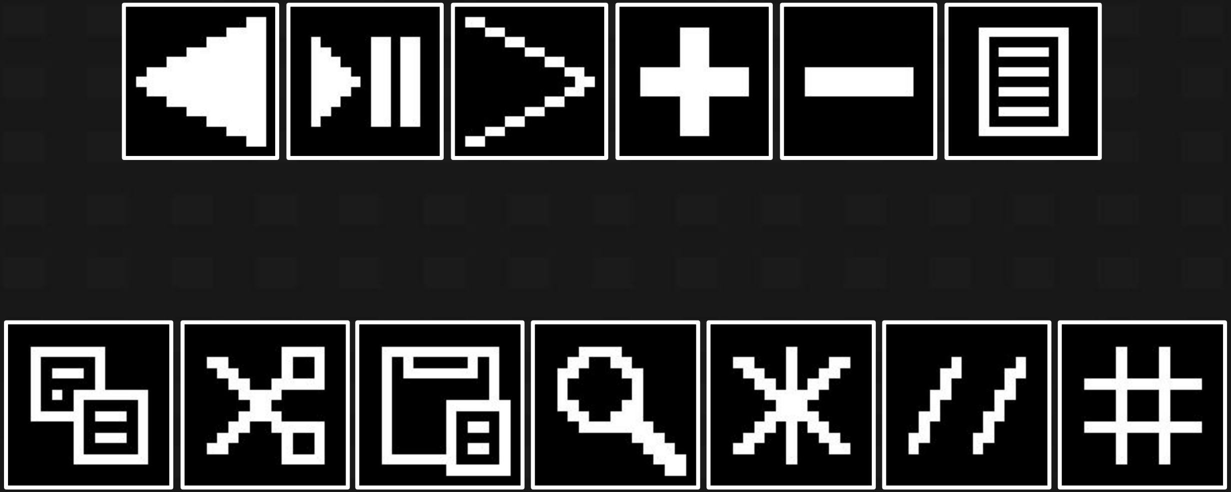




This solution solved symmetrical problems and seems the most simple and understandable for the users

KEYBOARD ICONS - 2

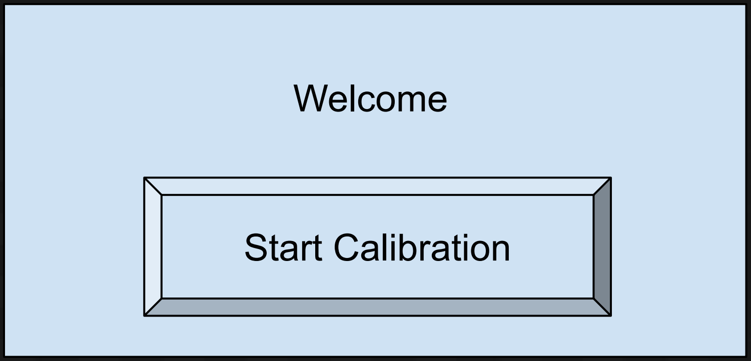
After some tuning, the final icons of our keyboard are the following:



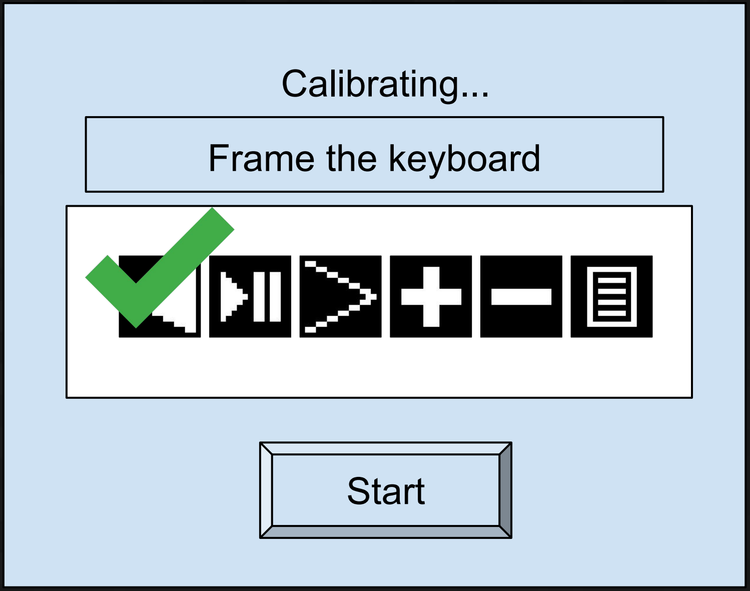
HCI

SOFTWARE INTERFACE

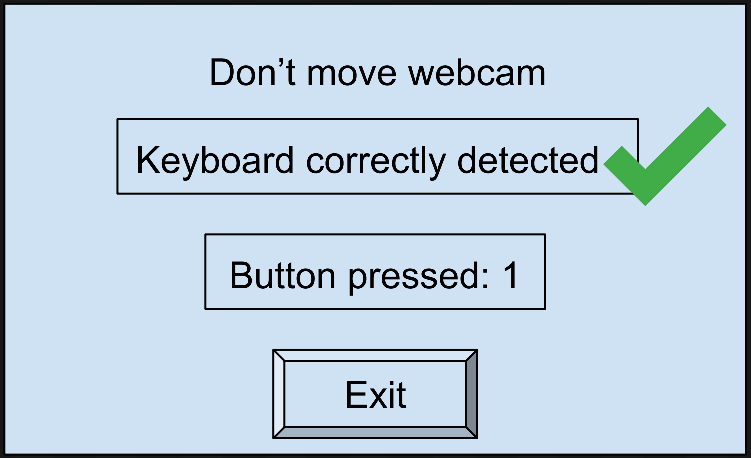
First version of wireframes included:



1. starting window



2. calibration window


Don’t move webcam

Keyboard correctly detected ✓

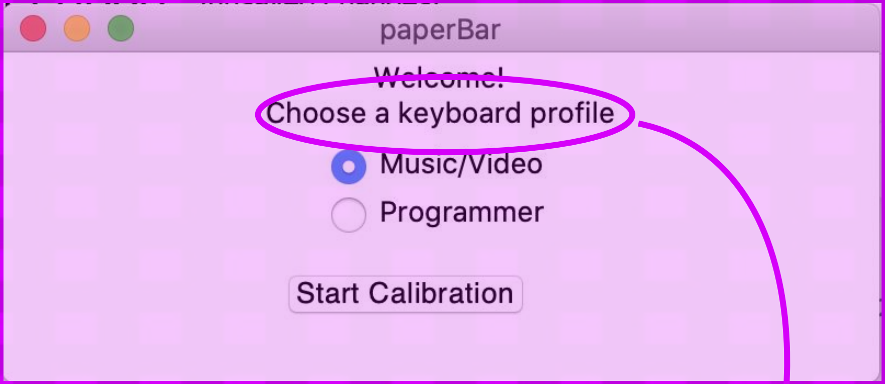
Button pressed: 1

3. keyboard window

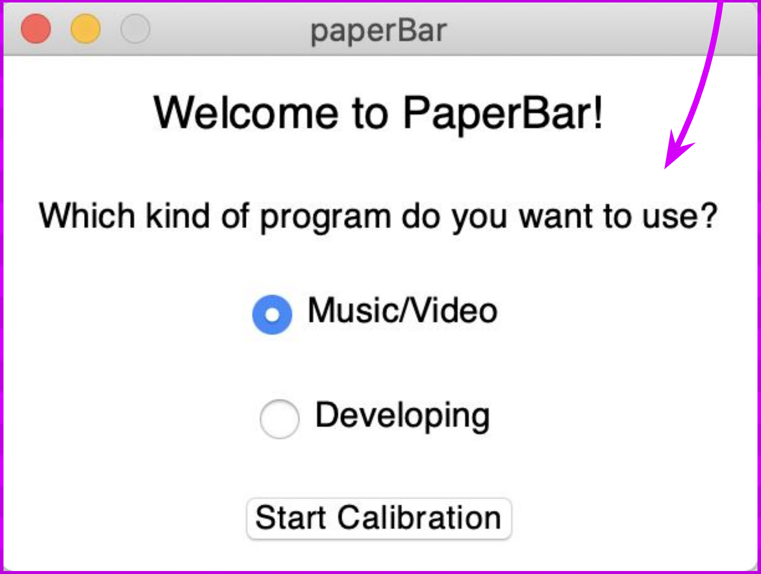
HCI

1.STARTING WINDOW

Changes were made in order to have an ever more usable and clear interface based on user feedback.

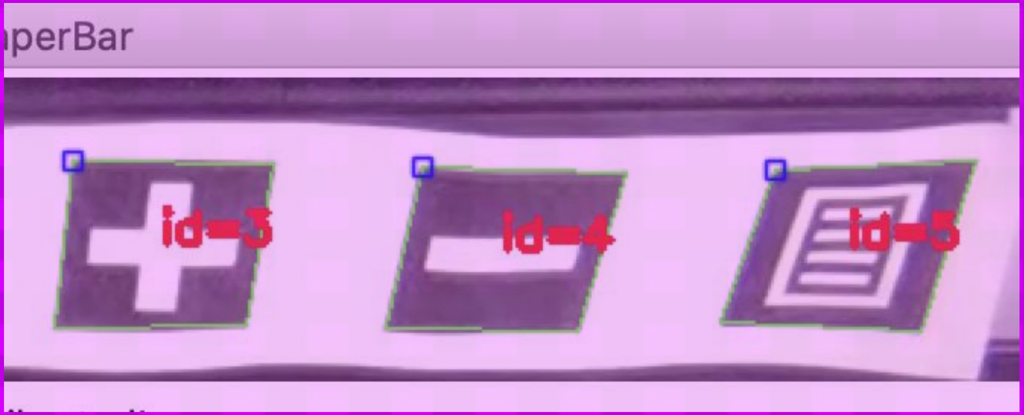


First view includes the selection of the profile linked to a different keyboard aspect

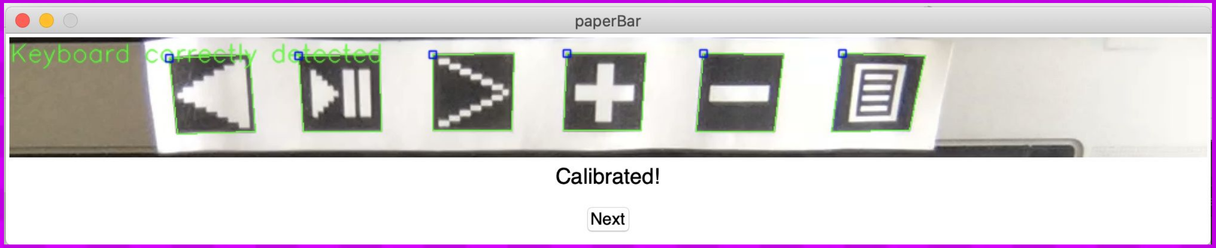


The label accompanying the selection was modified because created some confusion about the meaning of the action (e.g. "what's a keyboard profile?")

2.CALIBRATION WINDOW



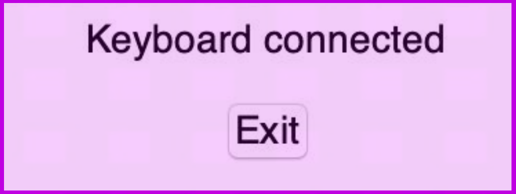
The calibration window shows the image perceived by the camera with bounds around detected markers and a button to finalize the phase



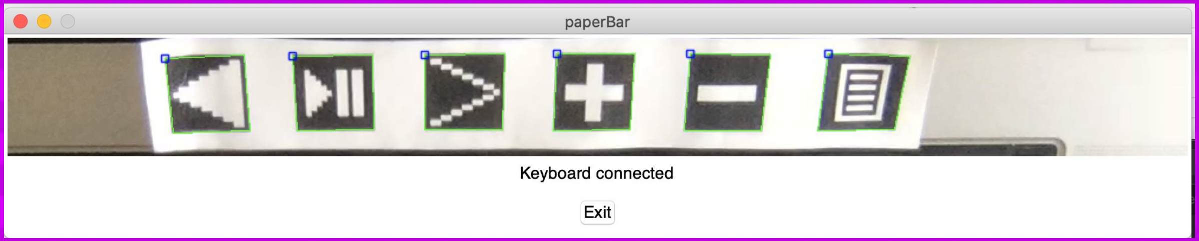
Calibrated!

Marker ids were removed in order to show only useful informations to the user

3.KEYBOARD WINDOW



Keyboard connected



Keyboard connected

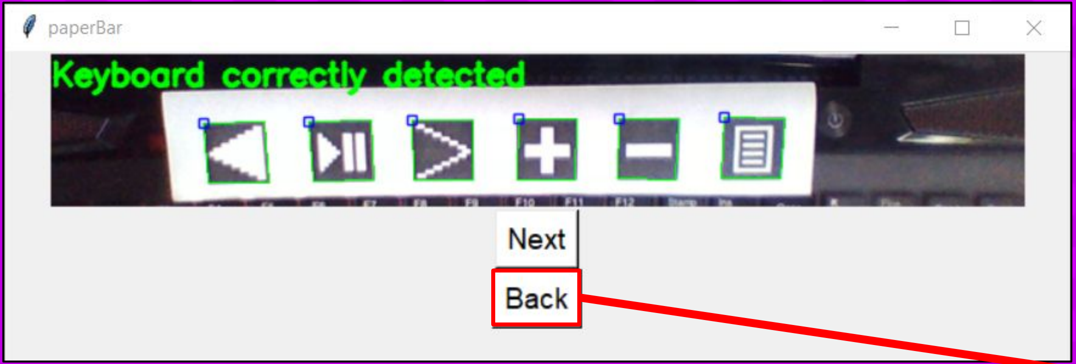
To have a clear view of the system status along time, text feedback alone was not enough!

Also the status window now includes constant camera feed to monitor marker correct detection

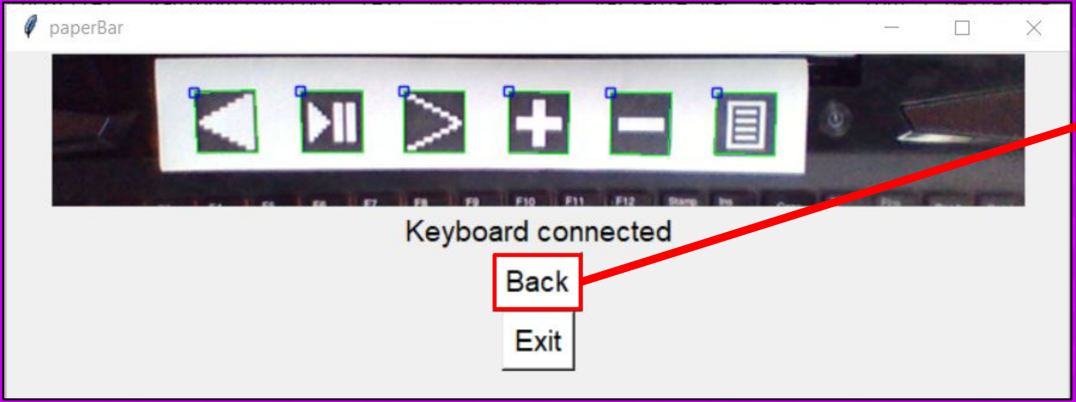
BACK BUTTON

Once a keyboard is connected, some users may want to go back to main menu and change the keyboard profile in use.

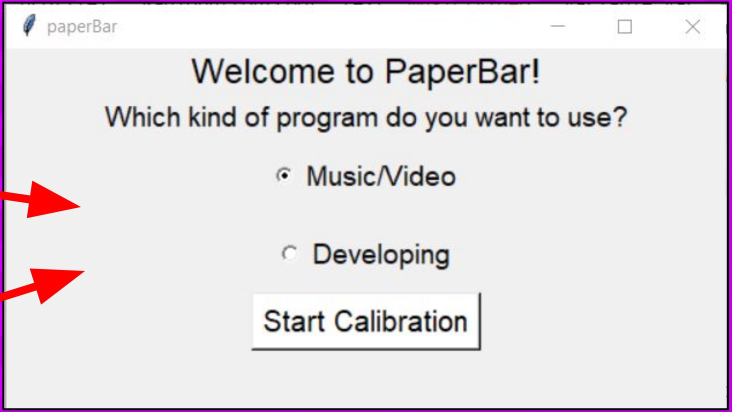
For this reason we added a "back" button both in the calibration and type windows.


paperBar

Keyboard correctly detected


paperBar

Keyboard connected


paperBar

Welcome to PaperBar!

Which kind of program do you want to use?

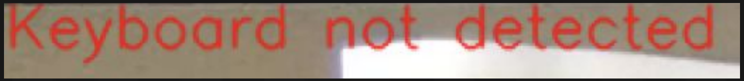
[✓ Music/Video]

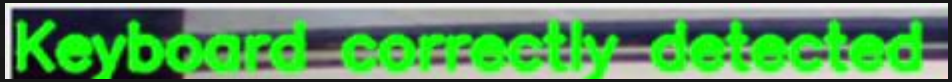
Developing

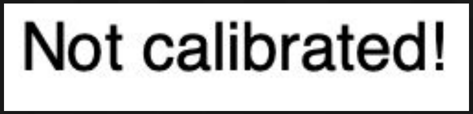
SOFTWARE FEEDBACKS

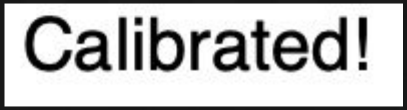
Software implements **visual feedbacks** such as:

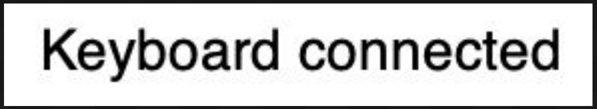
Labels, describing the current situation


Keyboard not detected

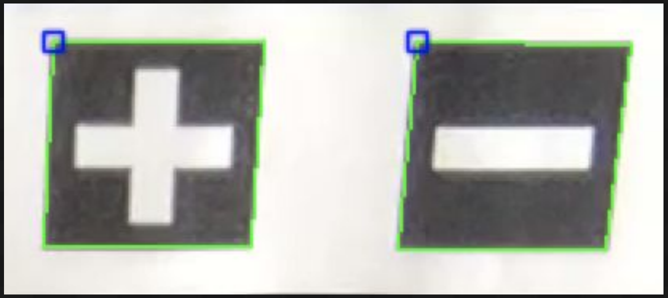

Keyboard correctly detected


Not calibrated!


Calibrated!


Keyboard connected

Marker bounds, visualizing a correct detection



SOFTWARE FEEDBACKS

User testing made clear the necessity of **sound feedbacks**, in order to improve the User Experience.

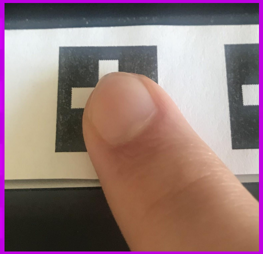
Sound feedbacks for

- correct/incorrect keyboard detection
- button pressure

It's important to let the user know immediately what are the consequences of his actions on the system

Not calibrated!

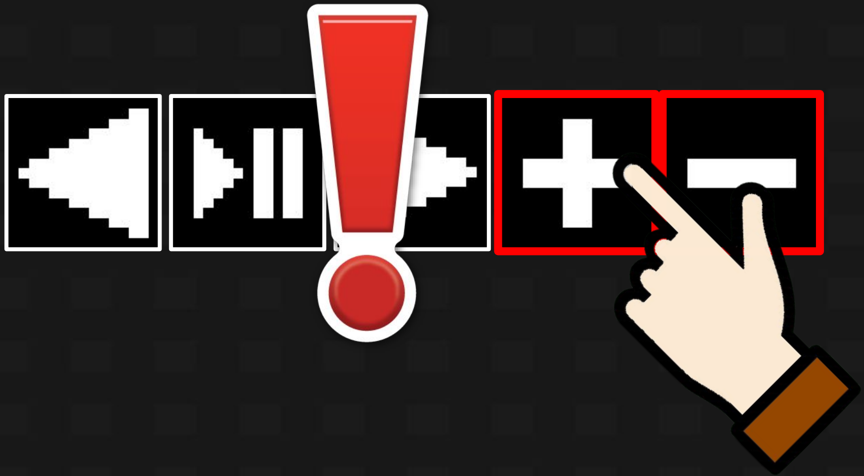
Calibrated!

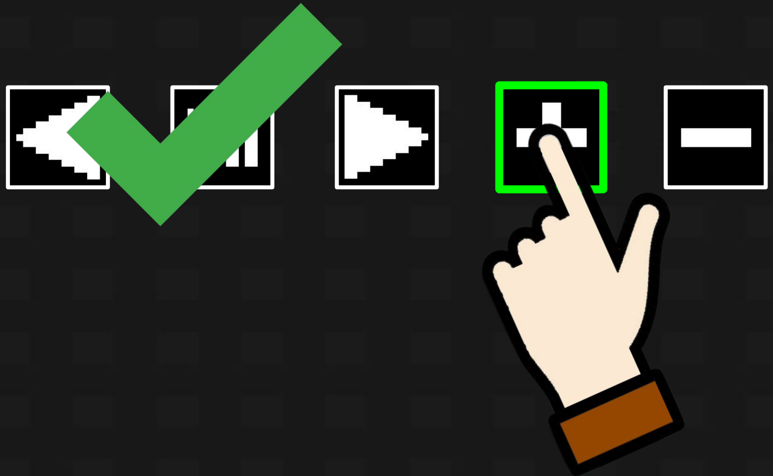


BUTTONS SIZE

After some tests, we noticed that buttons size represent an important aspect to define. In fact we have to find a trade-off between:

- **Bigger** markers means better recognition by the system, but buttons are closer to each other and this could lead to multiple pressing at once



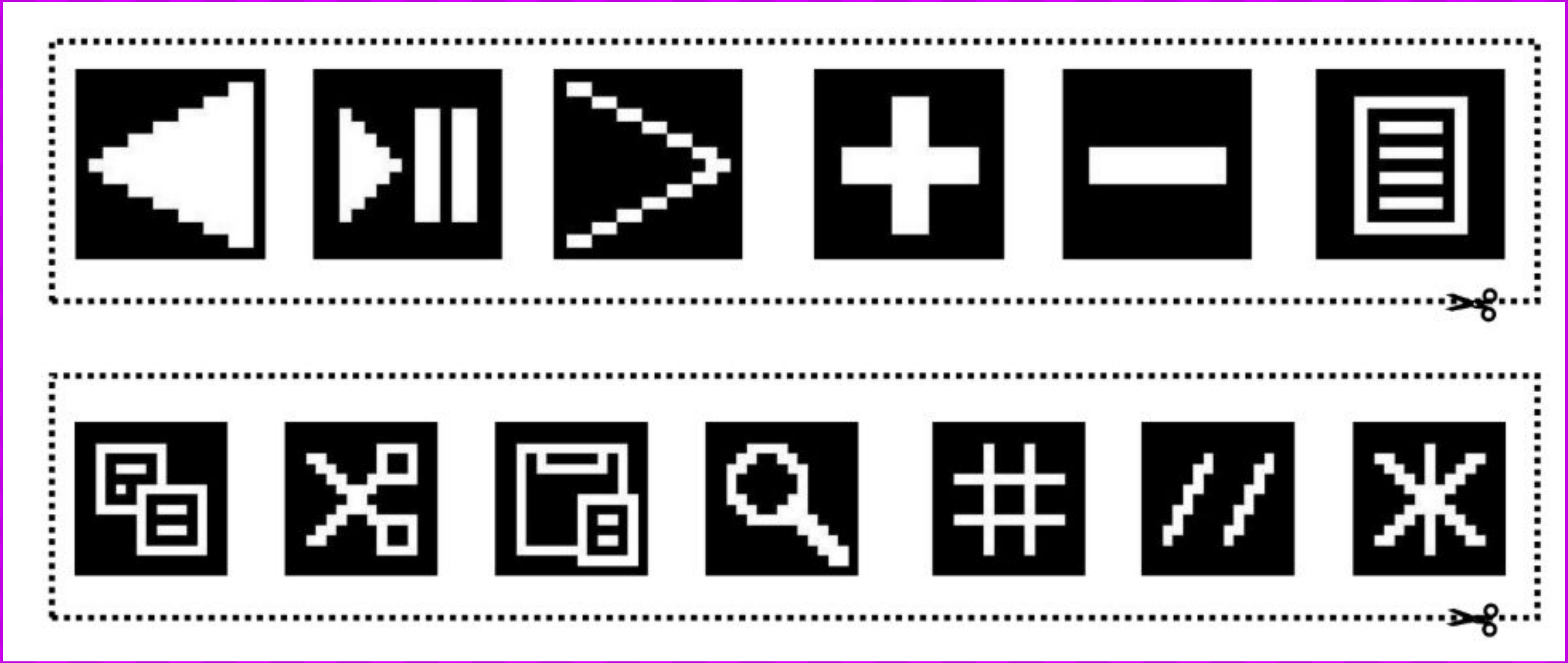


- **Smaller** markers means worse recognition by the system, but the issue of multiple button pressing is reduced.

HCI

BUTTONS SIZE

Finally we find a button size that works well in both recognition and pressing, and we produced standard-size keyboards for the user to print.



HCI

PRESSING A BUTTON

The study of button pressure was a key aspect during the development of the system:

- avoid false pressing due to illumination changes
  that make markers disappear for few milliseconds

- avoid double pressing when two buttons are missing

- select the best pressure time threshold to make the system reacts efficiently to the user

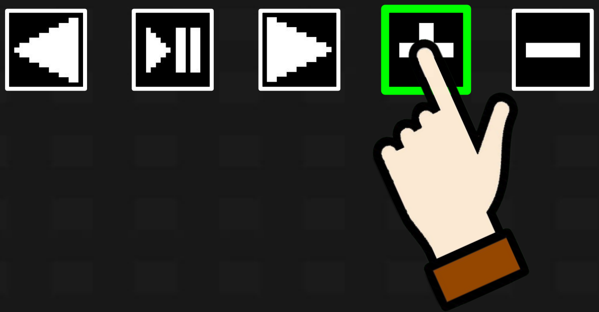
HCI

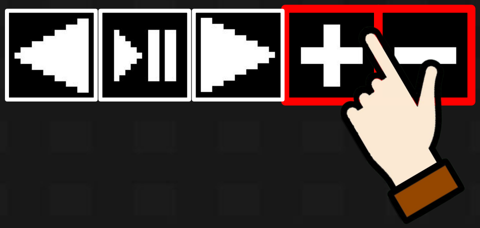
HOW IT WORKS 1/3

1. The system detects markers and if the keyboard turns out to be fully identified, the system highlights it to the user and it starts capture pressing

keyboard connected

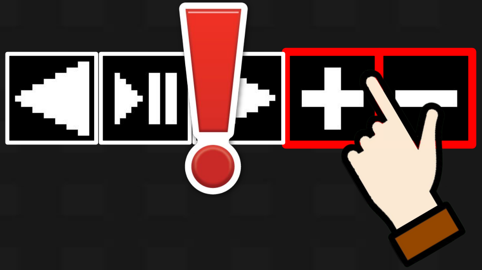
2. When a button is pressed, the system checks to see if it's missing just one or more than one



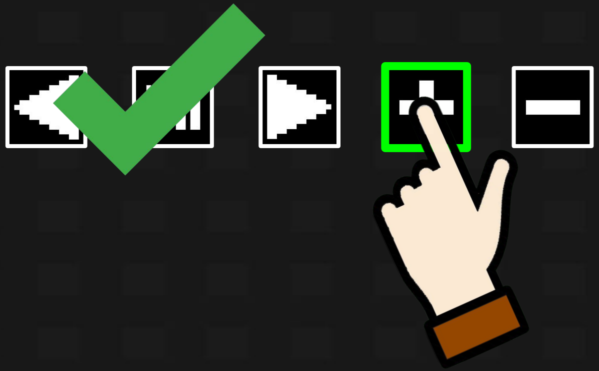


HOW IT WORKS 2/3

3. If is more than one, it is considered as occlusion



4. Otherwise the system takes the id of the missing one and press the button on the keyboard



HOW IT WORKS 3/3

5. A sound feedback is released

USER NOTES

At the beginning, when system did not detect a marker, it was considered as a button pression.
Users warned us that using different webcams in different light conditions make the system consider a button as pressed though it was not.
This is due to **short accidental detection loss** of the marker, depending on **light conditions**.

This creates a lot of problems:

- **sound feedback** are randomly reproduced
- **buttons** are randomly pressed, making the overall system not usable

For these reasons, we added a **threshold** on the **pression time** of the button, saying that if a marker is missing for enough time, it will be considered as pressed, otherwise not.
We tested three different thresholds:

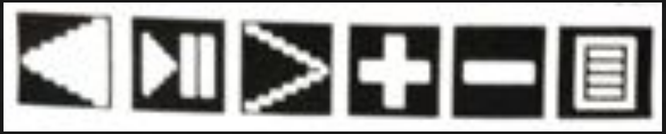
- **100ms**: still too short, creates the same problems.
- **300ms**: perfect trade-off between response time and prevent illumination problems.
- **450ms**: prevent illumination problems but response time was too slow

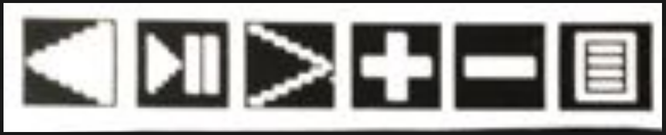
HCI

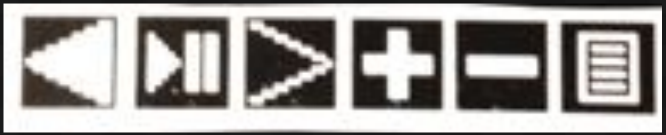
KEYBOARD MATERIAL

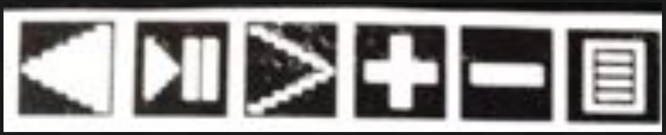
We tried to figure out how to create the best printed keyboard to be easily detected by the camera and that could work in different situations.

We tried the following settings:


Printer: indifferent; Paper:Normal Quality ink: Normal


Printer: Inkjet; Paper: Matt Photo; Quality Ink: High


Printer: Laser; Paper: Glossy Photo: Quality Ink: High


Printer: Laser; Paper: Matt Photo: Quality Ink: High

HCI

KEYBOARD MATERIAL - RESULT

Based on the previous study and test, the best configuration turned out to be:

Printer: Inkjet

Paper: Matt Photo;

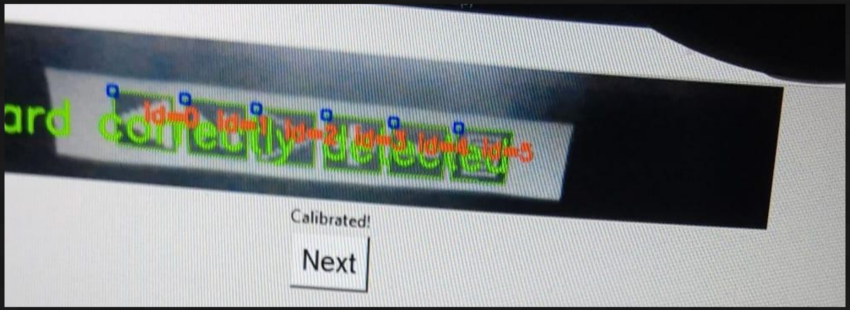
Quality: Ink: High

This study helped us to improve the recognition of the keyboard in different situations, with simple and very common devices to use.

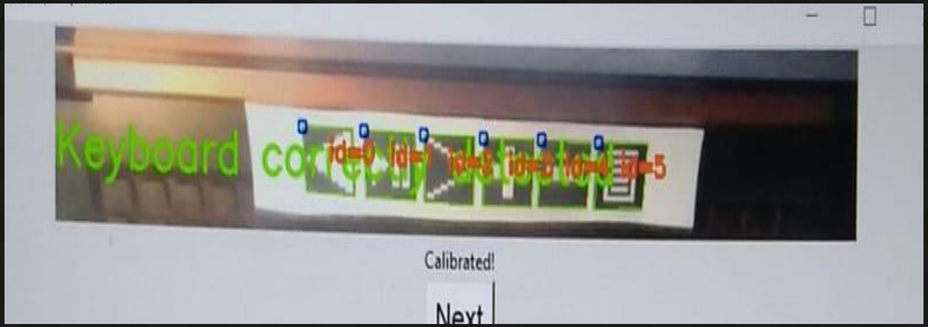
HCI

KEYBOARD MATERIAL - TEST

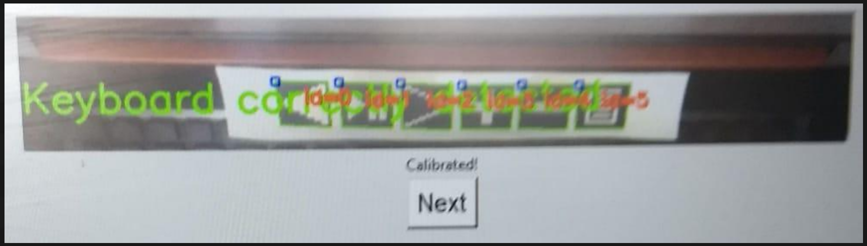
This is how the prototype revealed the keyboard printed with the suggested configuration:



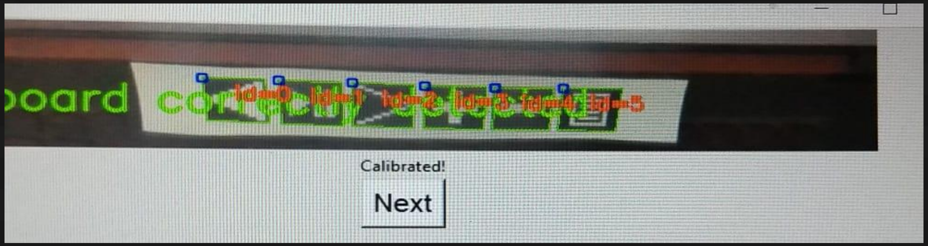
Completely Dark - Correctly detected



Warm Light - Correctly detected



Natural Light #1 - Correctly detected



Natural Light #2 - Correctly detected

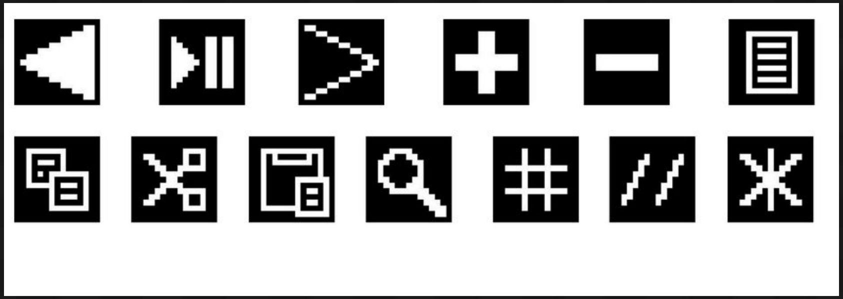
USER NOTES

The settings founded out to get the best result of from the system have been validated by the user. The users approved the simplicity and the speed reproducibility of the keyboard. The settings have been considered not difficult to set.

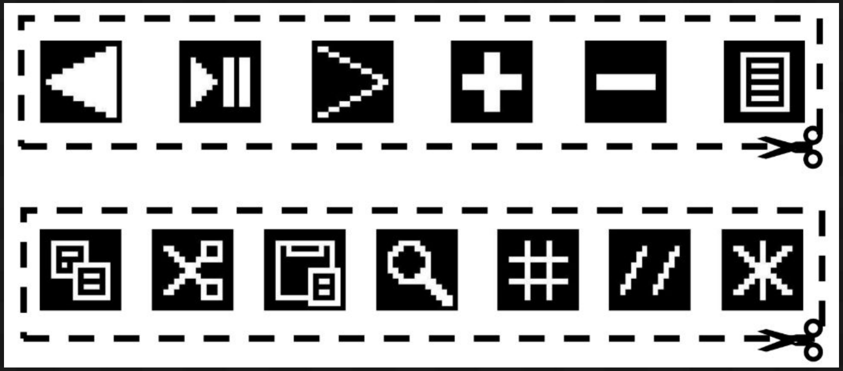
One problem highlighted, was that printing just one keyboard was uncomfortable, because if they wanted to print another keyboard they would need to use a new paper, or to adapt the old one into the printer.

USER NOTES

To help the user easily use the keyboards, we created a simple template. This template contains all the keyboards compatible with the app, like below



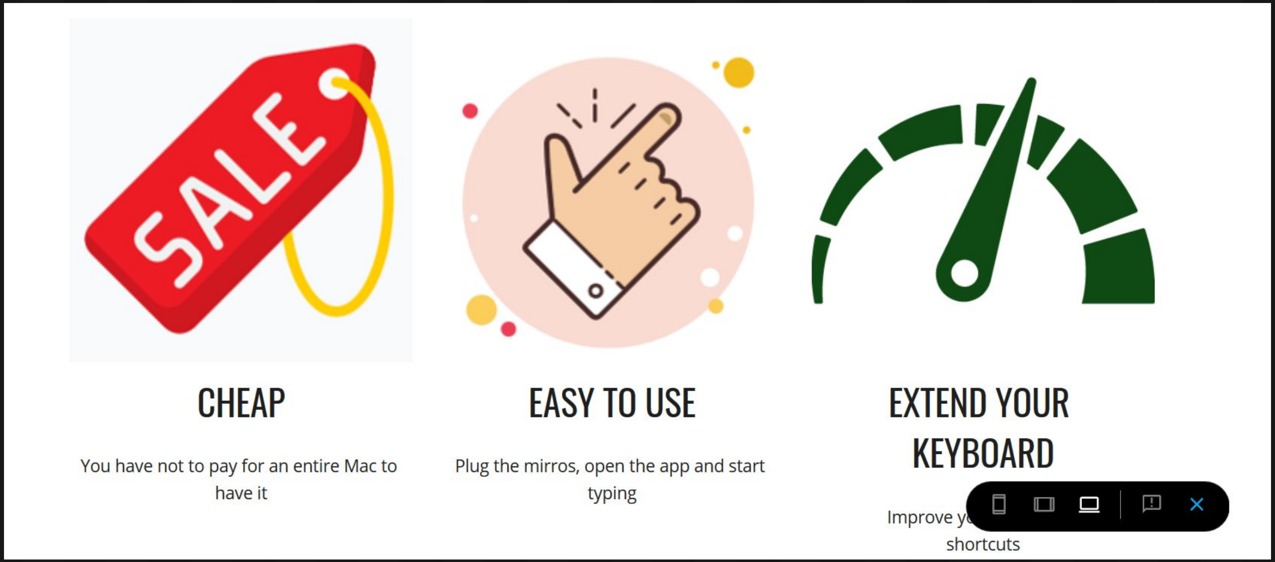
The template turned out to be not easy understandable, because the previous schema create confusion on what to cut. So we redesigned it in this way.



WEBSITE

To improve the user experience we decided to create a simple website where the user can find all the informations and instructions related to the system proposed. The website provides:

1. value proposition of the system



CHEAP

You have not to pay for an entire Mac to have it

EASY TO USE

Plug the mirros, open the app and start typing

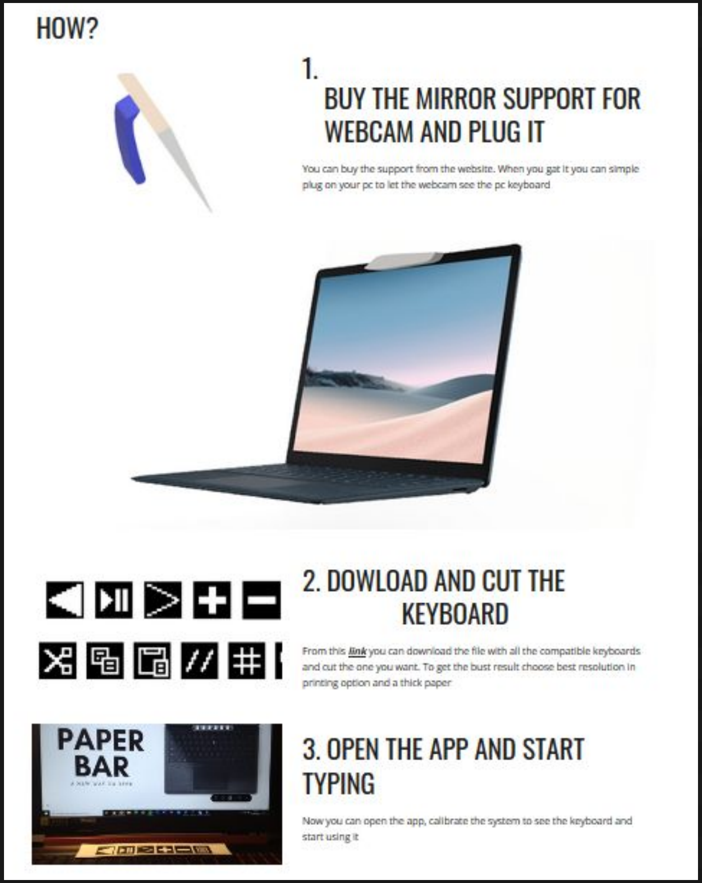
EXTEND YOUR KEYBOARD

Improve y shortcuts

HCI

WEBSITE

2. quick guide on how to install the system



HOW?

1. BUY THE MIRROR SUPPORT FOR WEBCAM AND PLUG IT

You can buy the support from the website. When you gat it you can simple plug on your pc to let the webcam see the pc keyboard

2. DOWLOAD AND CUT THE KEYBOARD

From this ***link*** you can download the file with all the compatible keyboards and cut the one you want. To get the bust result choose best resolution in printing option and a thick paper

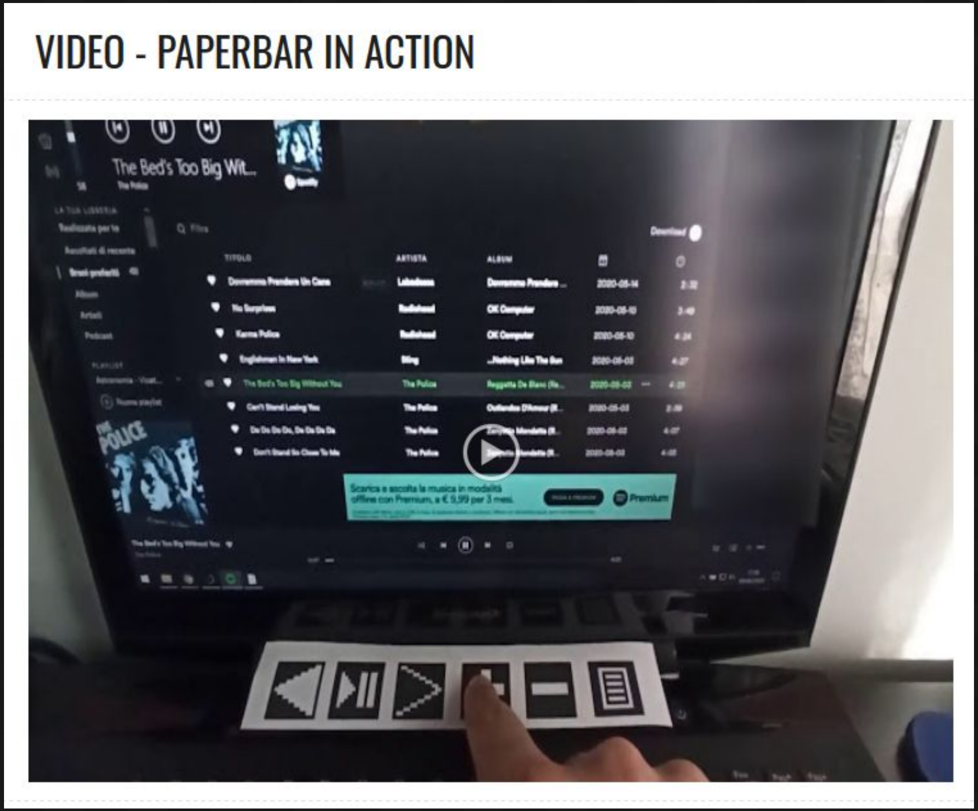
3. OPEN THE APP AND START TYPING

Now you can open the app, calibrate the system to see the keyboard and start using it

HCI

WEBSITE

3. demonstrative video from the very early step to the last one

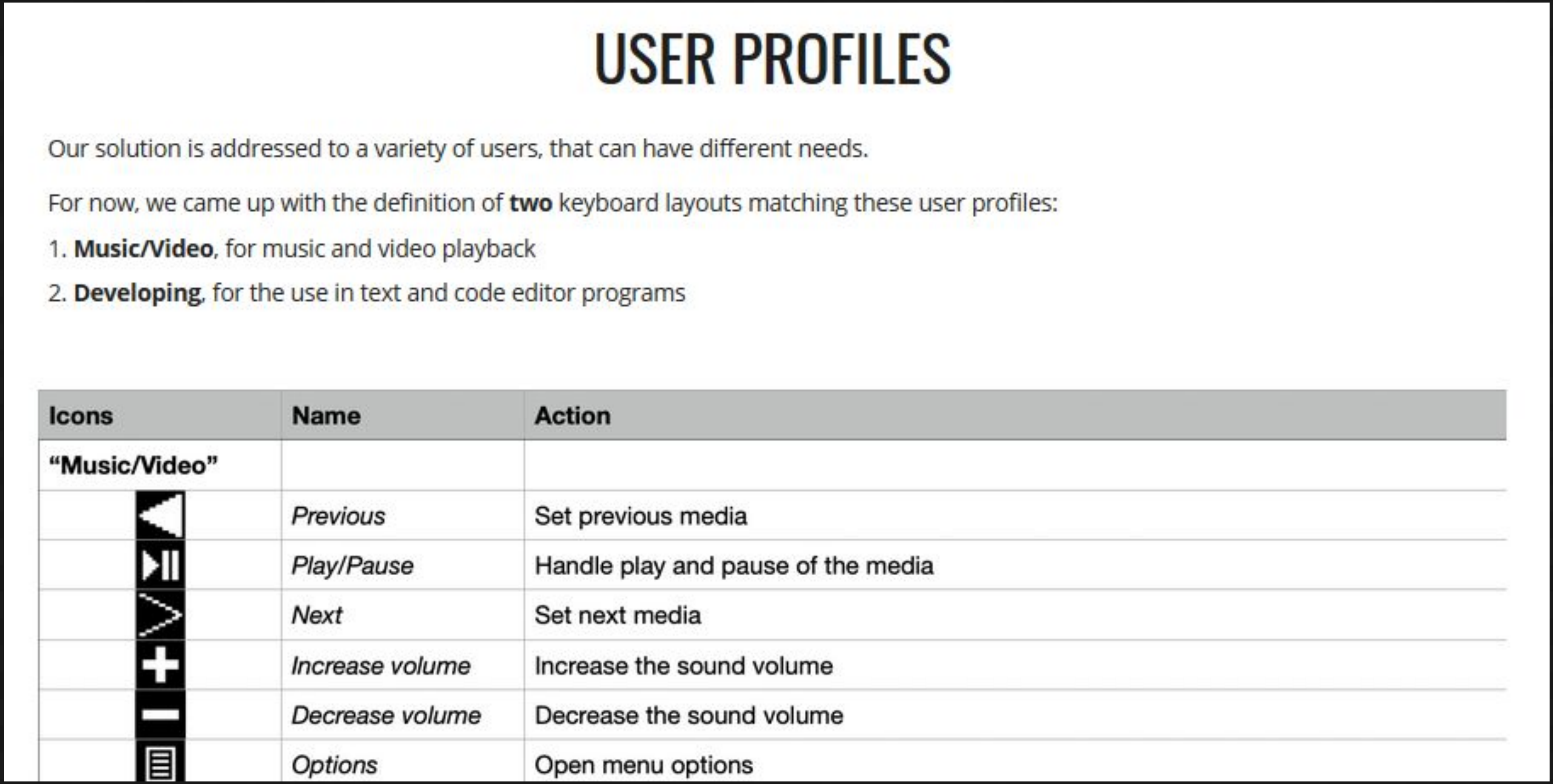


This video was introduced on user feedback. The simple guide could not be understandable at all, so they suggested us to add a video tutorial to be more esplicative and easy to understand

HCI

WEBSITE

4. quick section where the icons are explained



USER PROFILES

Our solution is addressed to a variety of users, that can have different needs.

For now, we came up with the definition of **two** keyboard layouts matching these user profiles:

1. **Music/Video**, for music and video playback
2. **Developing**, for the use in text and code editor programs

Icons	Name	Action
"Music/Video"		
◀	*Previous*	Set previous media
⏯	*Play/Pause*	Handle play and pause of the media
▷	*Next*	Set next media
➕	*Increase volume*	Increase the sound volume
➖	*Decrease volume*	Decrease the sound volume
☰	*Options*	Open menu options

SYSTEM TESTING

The system is composed by our software and three "hardware" elements:

- webcam (if not integrated)
- support for reflective surface
- printed keyboard

We do not developed a final version of the support yet: for our tests we used "home-made" mirrors placed in front of the webcam.

Tests are exploited using both keyboard profiles:

- Music/Video profile tested on Spotify, music/video players...
- Development profile tested on code editors and text editors (block notes...)

HCI

SYSTEM TESTING

LIMITATIONS

- Users must have a printer.
- Light conditions affect markers recognition: for example a sunbeam hitting a portion of the keyboard makes markers unrecognizable.
- Keyboard movements: when tapping the paper keyboard, it may move and cause disconnections.
- Space in front of the monitor, between printed keyboard and webcam, must not be occluded because that space is needed for the webcam to detect markers.

FUTURE DEVELOPMENT

- Keyboard can be constrained to its position in order to avoid improper movements and calibration errors after button pressure.
- Give user the possibility to create a custom keyboard based on his needs: this means let the user decide which shortcuts encode to buttons.
- Solve the problem given by critical light condition, like sunbeams and shadows.

HCI

THANK YOU!

HCI

USEFUL LINKS

- to final prototype: PaperBar.zip
- to the video tutorial: Tutorial.mp4
- to the keyboard paper: Buttons.pdf
- to the website: https://sites.google.com/view/paperbar/home-page
- to the survey (market investigation): https://docs.google.com/forms/d/1GLmstcINZaqYpjxbjnzIP--QFU__9Jm0LVwt5WJdhlc/edit?usp=sharing
- to code: Code

HCI

RESOURCES

- Aruco Markers, Rafael Muñoz and Sergio Garrido, (PDF) Automatic generation and detection of highly reliable fiducial markers under occlusion

- OpenCV library, opencv-python · PyPI

- Playsound library, playsound · PyPI

- Pynput library, pynput · PyPI

- Tkinter library, tkinter — Python interface to Tcl/Tk — Python 3.8.3 documentation