U SCIENCE TECH
FACULTAT DE CIÈNCIES
I TECNOLOGIA
UVIC-UCC

Treball de Fi de Grau

Building a Smart Mirror

Josep Cumeras i Khan

Grau en Multimèdia

Tutor: Raymond Lagonigro

Vic, juny del 2016

Table of Contents

FINAL YEAR PROJECT ABSTRACT
GRAU EN MULTIMÈDIA

Títle: *Building a Smart Mirror*
Keywords: *smart mirror, Raspberry pi, internet of things, makers*
Author: Josep Cumeras i Khan
Tutor: Raymond Lagonigro (UVic-UCC)
Date: June 2016

This project has been developed within the context of a time where every day we see more and more connected devices. The Internet transformed our lives by connecting us more easily to information and other people in the virtual world. Mobile phones then became smartphones and since then this concept has erupted and morphed into the Internet of Things, things which connect us to everyday objects. There are no end of objects that could be made "smarter", some being more suited to this than others. Mirrors, for example, provide a large surface ideal for displaying information and interacting with. Most people have mirrors at home so the concept of a smart mirror that you can interact with is attractive and has been fantasized in many futuristic movies.

Smart mirrors, such as Magic Mirror and HomeMirror have recently started to be developed by people in the Maker community, with varying degrees of interactivity. However, so far, the features of these mirrors have been limited. This final year project describes how a smart mirror was built from scratch using a Raspberry Pi for the hardware and custom software built on top of Raspbian, a Linux distribution. The goal of the project was to create a Smart Mirror device that people could interact with but also to further develop the technology so that it would let you install and develop your own applications for it.

The Smart Mirror was developed in four months, starting with the software and finally integrating it with the hardware. On the whole results were good because a higher level of interactivity has been achieved by being able to use voice commands, gestures and smartphones. A few problems arose in the construction and software side of the project, such as the glass not being reflective enough and the gesture recognition being unreliable but these drawbacks can be addressed by doing more tests and trials to further develop the Smart Mirror.

List of tables and figures

Acknowledgements

I would like to express my gratitude to Ray, my family and my friends.

1. Introduction

Everyone knows what a mirror is. It is an object found in most people's homes. In mirrors we see our reflections. But what happens when you combine the idea of a mirror with technology? What possibilities are there and how smart could a mirror be? These are some of the questions that inspired my choice of final year project, a project which aimed to develop a smart mirror and a small operating system to power it. The device was to go beyond an ordinary mirror, to have a screen inside that you would be able to interact with by using voice commands, hand gestures and smartphones or other devices.

This final year project was developed during my 4th year in the Multimedia degree at UVic-UCC. Multimedia is a very broad area and I like every aspect of it so it was difficult to choose a specific area and I had many ideas. However, I finally decided to build a smart mirror because it is a great combination of many of the things we have studied: web technologies, electronics, UI design, etc.

The smart mirror is a popular project among DIY enthusiasts and it usually consists of a one-way mirror with a screen attached to it that displays a static web page. However what I wanted to achieve was something you could interact with. My goal was to learn how a Raspberry Pi worked and to understand how to combine the software and the hardware components to create a multimedia project.

I started by obtaining a Raspberry Pi and creating the software. At the same time I began documenting everything and I also searched for a suitable one-way mirror and a computer screen, as well as some sensors to physically interact with the device. I then spent a long time calibrating the sensors to work with the software. Once the software was almost finished I started designing the frame and finally I built the smart mirror and attached all the components.

Developing this project has been a great experience. I have learned a diverse range of skills in different fields, such as DIY, Linux, electronics and web development. To obtain the final result I've had to work with many different technologies. I used Photoshop and Illustrator for the UI designs, web development tools for the software and electronics for the hardware. Not sticking

to just one field has made this project a really fun one and I would recommend it to anyone who is passionate about creating things.

2. Project goals

The main goal of this project was to develop a smart mirror device as well as an operating system to run on similar devices. The device was to look like a regular mirror but would have a screen inside and you would be able to interact with it using voice commands, hand gestures and smartphones.

The operating system would support running apps and would provide a simple API for third-party developers to create their own apps for the Smart Mirror. The main features the Smart Mirror would have would be showing basic weather and time information, being able to add alarms, reminders or notes in a similar way we stick post-it notes on a fridge. We would also be able to play music in some way and see pictures through Instagram, for example. The software needed to be designed to be modular and responsive in order to fit different hardware.

With the project I wanted to learn a lot about the Raspberry Pi as it was the first time I used it. I also hoped to refresh my electronics knowledge as it had been quite some time since I did something with electronics.

Up to now there have been many people who have built Smart Mirrors but in my opinion they lack interactivity. The project aims to change this by letting the user interact using different means. It will be one of the first Smart Mirrors you can interact with and also one of the first to let you install apps.

3. Context

This project was inspired by a “Magic Home Mirror” device that I found while browsing the DIY section in a popular website called Reddit. The “Magic Home Mirror” is a Nexus 7 Android tablet attached to a one-way mirror. The device has a display with a webpage that shows time and weather information and it looks very futuristic. I liked that project a lot and I thought I could improve on it by adding some means of interaction to the device. I also found a similar project that was built using a Raspberry Pi mini computer, but again it was a static panel with no interaction.

This inspired me to begin this project and develop a Smart Mirror with an operating system that would let you install apps that anyone could develop just like on Android or iOS. The project has a very broad scope covering some current popular topics in the IT sector such as the Internet of Things, Maker culture and home automation.

3.1 Internet of Things

The Internet of Things is a concept defined as a network of connected physical objects (Internet of things, 2016). It’s often viewed as the next step for the internet. Recently it has gained a lot of popularity predicting that in the future most everyday objects will be connected to each other and will be able to interact in smart ways. The Smart Mirror will eventually become one of these connected objects in our households and if we think about it being able to communicate with other objects the possibilities become endless.

3.2 Maker culture

The maker culture is a contemporary culture derived from DIY culture and hacker culture (Maker culture, 2016). It focuses in the creation of new devices as well as modifying existing ones. It often supports and embraces open-source hardware and software. This culture has been growing rapidly thanks to tools and technology like the Raspberry Pi, 3D printers and other hardware that have become increasingly affordable and accessible. The Internet also plays a

big part in the community as it enables people to share their ideas, blueprints and code. The Smart Mirror is a good example of a Maker culture project.

3.3 Home automation

Home automation has been around for a long time and it is all about turning the house into an intelligent unit with the goal of increasing comfort and efficiency at home. Some of the typical applications are automatic lights, intelligent thermostats, alarms, window blinds (Home automation, 2013).

In my project I will not be focusing on home automation since I don't have access to any smart home devices. However it would be very easy to write an application to turn on and off the lights using voice commands or gestures on the mirror or even an application to change the temperature of the room, for example. These examples are just the tip of the iceberg as there are new connected devices emerging everyday that could interact with the mirror.

3.4 Benchmarking

There are thousands of people building devices with one-way mirrors. It’s a very popular project within the Maker community and you can find many interesting creations. Five projects relevant to this work are described in this section. Magic Mirror was one of the first known projects in this area, Home Mirror was a development of Magic Mirror and easier to build, Evan Cohen’s Smart Mirror incorporated several extra features to the smart mirror concept, Max Brau’s Smart Mirror, developed at Google included interactivity and PANL is the latest smart mirror with a touch screen device. These projects have acted as benchmarks for the smart mirror I built.

3.4.1 Magic Mirror

Magic Mirror, created by Michael Teeuw, is a mirror device built with a wooden frame, a flat-screen TV, a Raspberry Pi 2 and software running on a web browser (Magic Mirror, 2014). It was initially revealed in 2014 in a blog post by Michael and it quickly gained lots of attention reaching up to more than 2.5 million views. The source code for the software and a step-by-step guide to building the hardware was also posted in Michael’s blog. It was one of the first, if not the first popular project of its kind.

Figure 1. MagicMirror2 (Source: GitHub/MichMich/MagicMirror, 2016)

The open-source software that powers the mirror, MagicMirror2, is still currently in development and it was completely rebuilt in March 2016 using **Electron**, the same tool that I used for my own project. MagicMirror2 supports small widgets called “modules” and anyone can write one. The default installation comes with some basic widgets for time, calendar, weather and news and there are about 20 third-party modules available.

Magic Mirror is a great device that looks very good and has a clean UI. It also has a big community behind it. There’s a specialised MagicMirror forum helping people build their own hardware and write modules, a blog where Michael posts his progress and an active **GitHub** repository (GitHub/MichMich/MagicMirror, 2016) where anyone can contribute to the code. Overall it is a very similar project to my own. Nevertheless, unlike my own device, with the current hardware and software implementation on Magic Mirror you cannot physically interact with the device and so it only acts as a simple information panel.

3.4.2 HomeMirror

HomeMirror (GitHub/HannahMitt/HomeMirror, 2016) is a device created by Hannah Mittelstaedt that was initially revealed in a post on **Reddit** in 2015 gathering millions of views. It was the main inspiration for my project. In this case there's a **Nexus 7** Android tablet behind the mirror so there's no need for a separate computer board and screen.

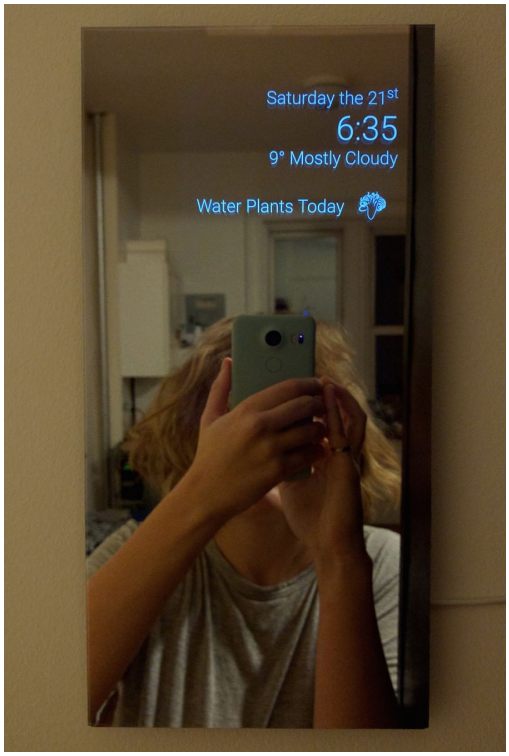


Figure 2. HomeMirror (Source: GitHub/HannahMitt/HomeMirror, 2016)

The HomeMirror's visual interface runs on an Android app and it shows useful information such as weather, time, date and reminders. As for the Magic Mirror, HomeMirror is only intended to be used as an information panel. The software doesn't support custom widgets but it is open-source so anyone can modify it and create a different version. The code is available on GitHub where Hannah also provided a step-by-step guide on how to build the device.

HomeMirror is a device that also looks great and it is also easier to build than other devices as it only requires two main parts, the tablet and the mirror. However it lacks any kind of interaction.

3.4.3 Evan Cohen's Smart Mirror

The Smart Mirror by Evan Cohen (http://smart-mirror.io/) is a device inspired by both MagicMirror and HomeMirror. This project focuses mainly on the software part and offers a wide range of voice commands to interact with the mirror. The voice commands support weather, time and date information, adding reminders, showing maps, showing pictures and even controlling the lights in a house.

Figure 3. Smart Mirror by Evan Cohen (Source: http://smart-mirror.io/)

The hardware requirements for this project is the usual one-way mirror, display and Raspberry Pi setup but with a USB microphone to support voice commands. For the mirror to interact with the house lights Evan uses Phillips Hue light bulbs. The software is open-source, built on Electron and can be found on GitHub. Although it is recommended to run it on a Raspberry Pi it can also run on other systems.

As described, this smart mirror is more advanced and feature-packed than most others, however the software has been approached as a single app that can do many things and this makes it difficult to write your own apps for the system.

3.4.4 Max Braun's Smart Mirror

On January 30th Max Braun, an engineer working at Google, made a blog post on **Medium** (https://medium.com/@maxbraun/my-bathroom-mirror-is-smarter-than-yours-94b21c6671ba#.q4932hjfc) showing his own smart mirror and explaining briefly the parts he used and how he created it. In his post he argues that he built it because there's no one selling this kind of device and that he wanted to feel as if he was in the future. Although there is nothing particularly new or innovative about this device, the news about it was reported by major news outlets around the world because it was built by a Google engineer.

Figure 4.Max Braun's Smart Mirror (Source: Medium, 2016)

Technically, the device is very similar to the others. It uses a one-way mirror, a 15 inch USB powered display panel and it runs on an Amazon Fire TV stick, which is an HDMI Android device.

The software running on the device is an Android app displaying useful information similar to what **Google Now** offers. The author says that you are not meant to interact with the UI, instead it will update with new useful information automatically. However, the device does have voice commands to perform a Google search.

Max Braun's Smart Mirror is another great mirror device. In this case Max introduces a small level of interactivity,compared to the previously mentioned devices, by being able to use voice commands to search Google. However, the device is still basically a static information panel and as of now, the software running on it has not been made open source, so there are not many details available about it.

3.4.5 PANL

PANL (http://getpanl.com/) is one of the first touchscreen smart mirrors. It was created by Ryan Nelwan and it was revealed as a startup in California on April 26th 2016. Ryan initially posted a video of the device on Reddit where it went viral and it was reported by all major technology blogs.

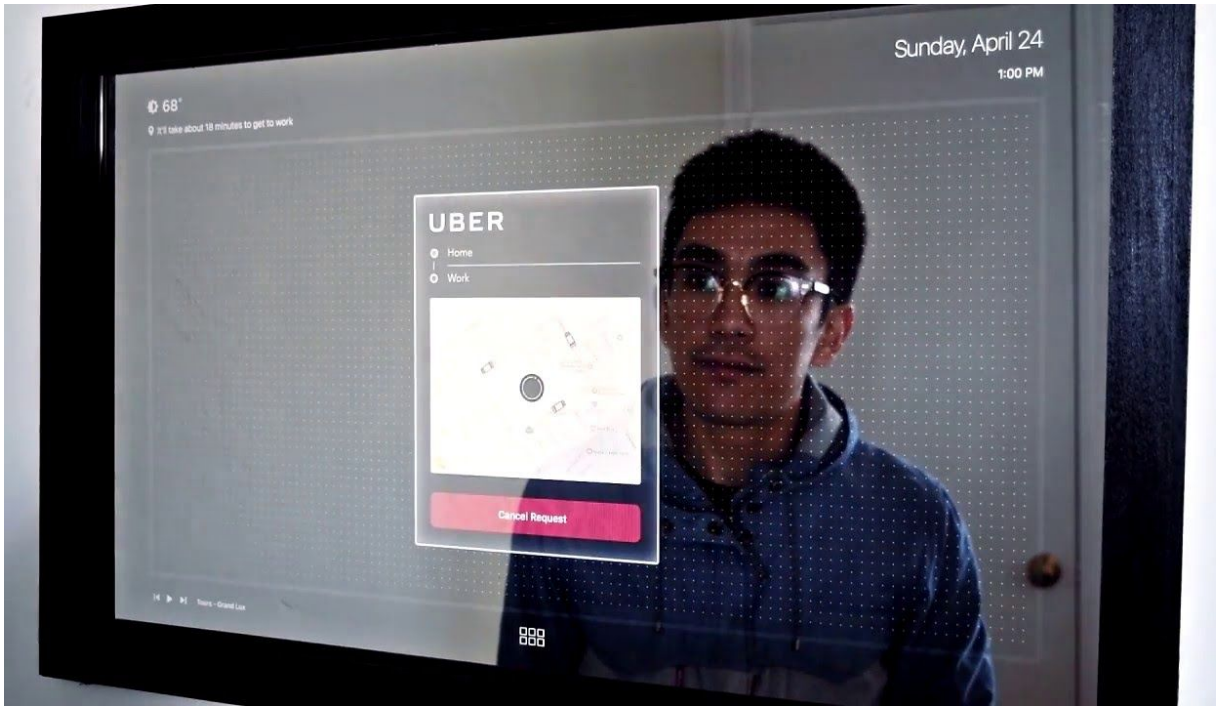


Figure 5. PANL display screen (Source: http://getpanl.com/)

The only information we have about this project is the video Ryan posted on YouTube so we do not know much about it. He has not said anything about the hardware setup or the kind of software he is using but from the video we can see that the device is fully multi-touch and it can display useful information such as weather, time and date. It can play music, youtube videos and it has apps for Photos, DropBox, Messages, AirDrop, Reddit, Nest and Uber. The software also comes with a fully functional on-screen keyboard.

The lack of information provided meant that some people believed the video to be a fake but it was later confirmed by journalists that is indeed real (https://twitter.com/ProductHunt/status/729893082112483328) and that Ryan is not releasing any more information because he plans to sell these devices.

Summing up, these five projects (Magic Mirror, Home Mirror, Evan Cohen's Smart Mirror, Max Brau's Smart Mirror and PANL) illustrate the step-by-step progression in technological advances with smart mirrors right up to the present, which have accompanied the decision-making process during the development of my own project. There has been a clear progression in how smart these mirrors are, with the first smart mirror simply providing information, then easier

programming and construction was created, a small level of interactivity was added and finally a touch screen incorporated. This brings me to the rationale for this project, as although some interactivity has been developed in smart mirrors there is a clear need to further develop this aspect.

4. Building a Smart Mirror

4.1 Hardware

For the hardware I used a 24” LG computer monitor, a 50x90x0.5cm one-way mirror a Raspberry Pi 2, two USB microphones and two ultrasonic sensors. Everything was put together in a wooden frame. These are the final sketches for the hardware design:

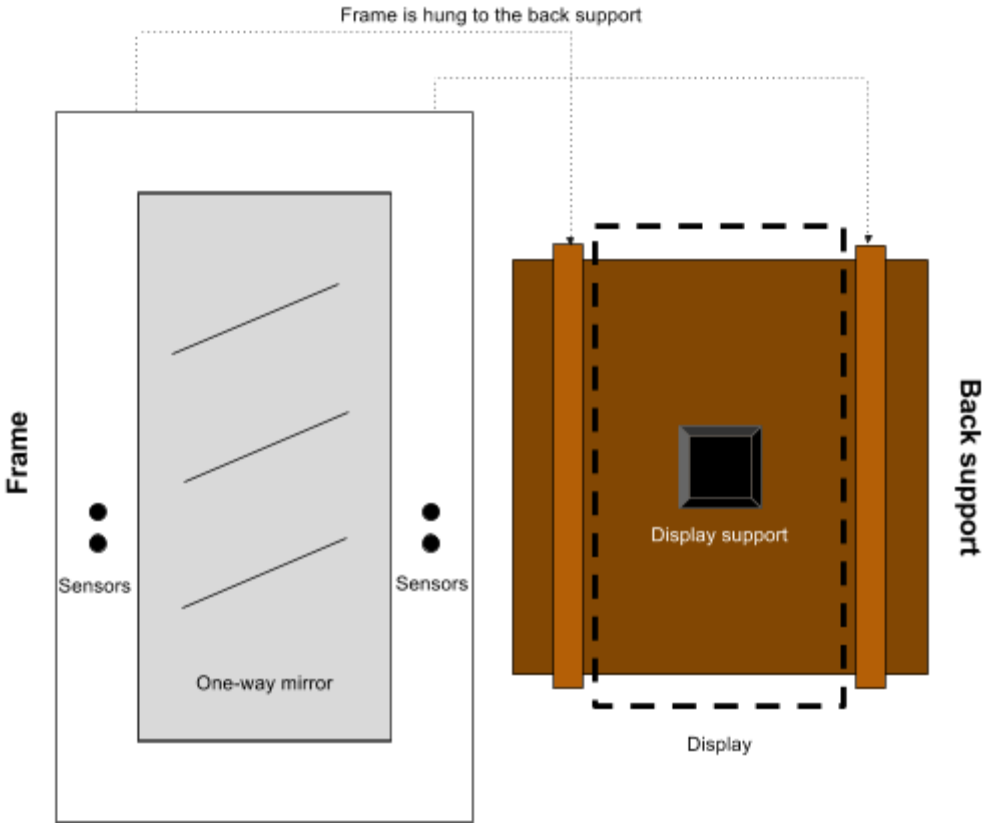


Figure 6. Sketch of the hardware design required for the smart mirror

The device has two wooden parts. The back part holds the display and the Raspberry Pi and is used to support the device so that it can be hung on a wall. The frame is attached to the glass

by two small wooden slats and it has four holes, two on each side, that contain the ultrasound sensors. The frame can be attached and detached from the back part so it's easy to change the glass or even the whole frame. See appendices 1 and 2.

A breakdown of each of the main parts of the smart mirror (the one-way mirror glass, display, Raspberry Pi 2, microphones, ultrasonic sensors and frame) and how they were used is described in the following sections:

4.1.1 One-way mirror

This is probably the most important part of the hardware because it's responsible for creating the futuristic effect and is the biggest part of the smart mirror. Wikipedia provides the following definition:

> A one-way mirror, sometimes called two-way mirror, is a mirror that is partially reflective and partially transparent. When one side of the mirror is brightly lit and the other is dark, it allows viewing from the darkened side but not vice versa (Loy, 1999).

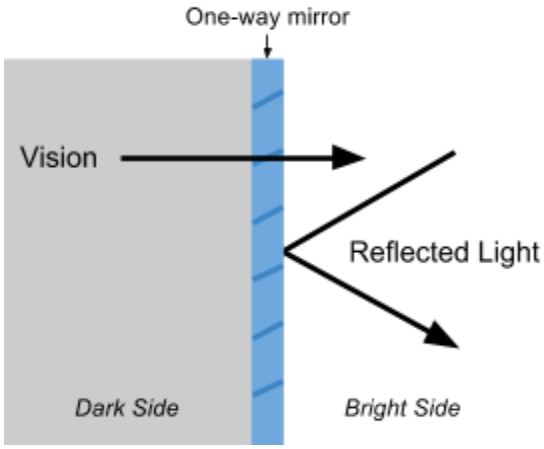


Figure 7. Schematic diagram of light reflection on a one-way mirror

In the case of this project this essentially means that the dark or black parts of the screen will be seen as a reflection and the light parts will be seen normally. So if there is white text over a black background the white text will be seen as an overlay with the user reflected in the background.

This was the most difficult component to find because of these technical requirements, but a one-way mirror was eventually found at a nearby glass store. The one that was bought was unfortunately not very reflective so sometimes you can see the interior of the device. This is not ideal but in the right conditions it works well and it can always be replaced with better quality glass in the future.

4.1.2 Display

For the display a 24 inch LG monitor was bought, which also has built-in speakers and comes with a remote control which is useful to easily turn off the device's screen. The monitor is much smaller than the mirror so a black sticker was used to cover the parts of the glass which are not covered by the display. An HDMI cable was used to connect the display to the Raspberry Pi for video and audio.

4.1.3 Raspberry Pi 2

The Raspberry Pi is a single-board computer developed by the Raspberry Pi foundation in the UK. It has become the most popular computer of it's kind thanks to great support and a big community behind it as well as an inexpensive price.

The Pi does not work out of the box. It lacks a hard drive and it does not come with a preinstalled operating system. To install an OS you need a microSD card prepared with an OS image. And because the software that will be running on the mirror will be coded on the same device at least a screen, a keyboard and a mouse are required.

4.1.4 Microphones

One mode of interaction with the smart mirror is through microphones. Two microphones were used to power the voice recognition capabilities of the device. USB microphones had to be used because the Raspberry Pi does not have a regular microphone input. The first microphone is a cheap simple one connected through a USB sound card to the Pi. The second microphone is actually a PS3 Eye camera that I had at home and that connects directly through the USB. However, only the microphone part of the PS3 Eye is being used. The voice recognition system works by listening for someone to clap with the first microphone and once that happens the second, higher quality microphone is triggered to listen for a voice command.

4.1.5 Ultrasonic sensors

The ultrasonic sensors are the second way to interact with the smart mirror. An ultrasonic sensor has two main parts, a speaker and a microphone. It works by sending an ultrasound with the speaker and returning the time it takes to capture the echo with the microphone. With the time it takes and the speed of sound we can then calculate the distance of an object from the sensor.

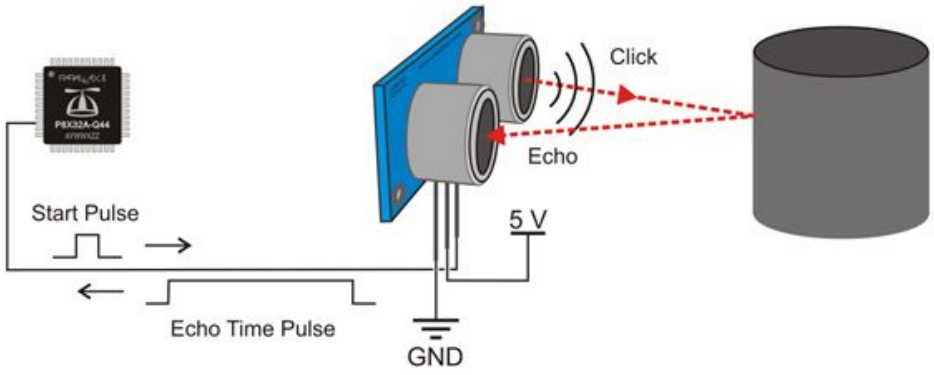


Figure 8. Diagram of an ultrasonic sensor (Source: themakersworkbench.com)

This component needed two resistors to work correctly so I soldered them on a metal plate to and joined the cables to be able to easily attach and detach it. See appendices 2, 3 and 4.
In my device there is one ultrasonic sensor on each side of the frame and they can be used to detect different gestures and navigate through interfaces.

4.1.6 Frame and support

The frame is made of wood and it provides the support for the mirror and all the other components. It frames the glass and provides a way for hanging the mirror on a wall.
It has two parts: the front is painted white and has four holes for the ultrasonic sensors. The back has two wooden bars on the sides that are used to hang the front part. In the center there is a support for the display and at the bottom there is the Raspberry Pi. See appendix 5.

4.2 Software

All the software runs on the Raspberry Pi 2 and there are many operating systems to choose from. I chose to use Raspbian which is the official Linux distribution from the Raspberry Pi Foundation because it has a lot of support and documentation.

To install it, I downloaded Raspbian from the official Raspberry Pi website and I copied it on a microSD card. Then I inserted the card on the Raspberry Pi, I started it and followed the setup instructions which are quite simple. Once Raspbian was installed, the first thing I did was to update the distribution with the latest packages, I configured the basics of the OS as for instance the keyboard layout to match my keyboard and everything was ready to go.

4.2.1 Development Tools

Taking advantage of the fact that I already had an operating system running on the Pi, I gave myself the challenge of writing all the code for the Smart Mirror on the same device. I installed Geany, which is a very lightweight IDE, and I used it to write all the HTML, Javascript, CSS and Python code.

In the end, the entire coding for the software was done on the Raspberry Pi and I only used my Windows laptop to create icons and designs with Illustrator and Photoshop. It turned out to be very convenient to be able to easily test the software directly on the Smart Mirror.

Electron

Electron is a software based on Chromium, the open source version of Google's Chrome, that includes NodeJS and several improvements to make it easy to develop web-based software for desktop computers. The OS was built on top of Electron using web technologies.

NodeJS

NodeJS is a Javascript engine for server side applications. It comes included with Electron and I used it to launch processes to control things that are not available in web APIs such as the ultrasonic sensors for gesture input and microphones for voice recognition. I also use it to access the filesystem and read the app files.

Python

Python is a high-level, general purpose, interpreted programming language. It's very popular in the Raspberry Pi community and it has lots of support and libraries. In my case I used it with the microphone to detect claps and I also used it to control the ultrasonic sensors and detect gestures.

4.2.2 MirrorOS

MirrorOS is the software I created for the Smart Mirror's interface and it runs on top of Raspbian and on top of Electron. In the following figure you can see the layers of the software stack.

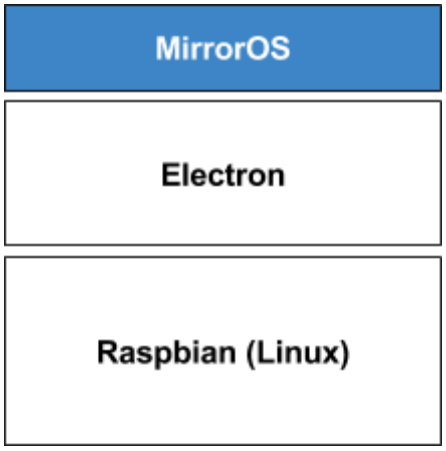


Figure 9. Layers of software stack of MirrorOS

Architecture and features

The OS was designed to be very simple and lightweight as it already runs on top of many layers of software. It's written in HTML, JavaScript and Python and it is basically a framework for web apps that provides APIs for listening for gestures, voice input and smartphone interaction and for displaying messages to the user in a consistent way.

Almost everything in the OS is an app, even the home screen. Each app has to define some keywords to respond to voice input so when a keyword is recognized the OS knows which app to launch. All apps run on a different process so if an app crashes the OS continues to work as usual.

Currently, MirrorOS includes 3 apps by default and the idea is that you will be able to download more from the companion smartphone app. It's also possible to define a different default home screen app.

MirrorOS has three main services:

- The voice input service, used to handle all the voice recognition process using.
- The gesture input service, used to handle gesture recognition using the ultrasonic sensors.
- A socket server which is in charge of communicating with smartphones or other devices.

User Interface

The user interface for the OS is clean and simple. It has an overlaid status bar on the top with the time on the right corner, the IP address of the socket server on the left corner and a status message in the center. The status bar is dynamic and changes depending in the context: it can be hidden in case we want watch something in fullscreen or expanded to show important information.

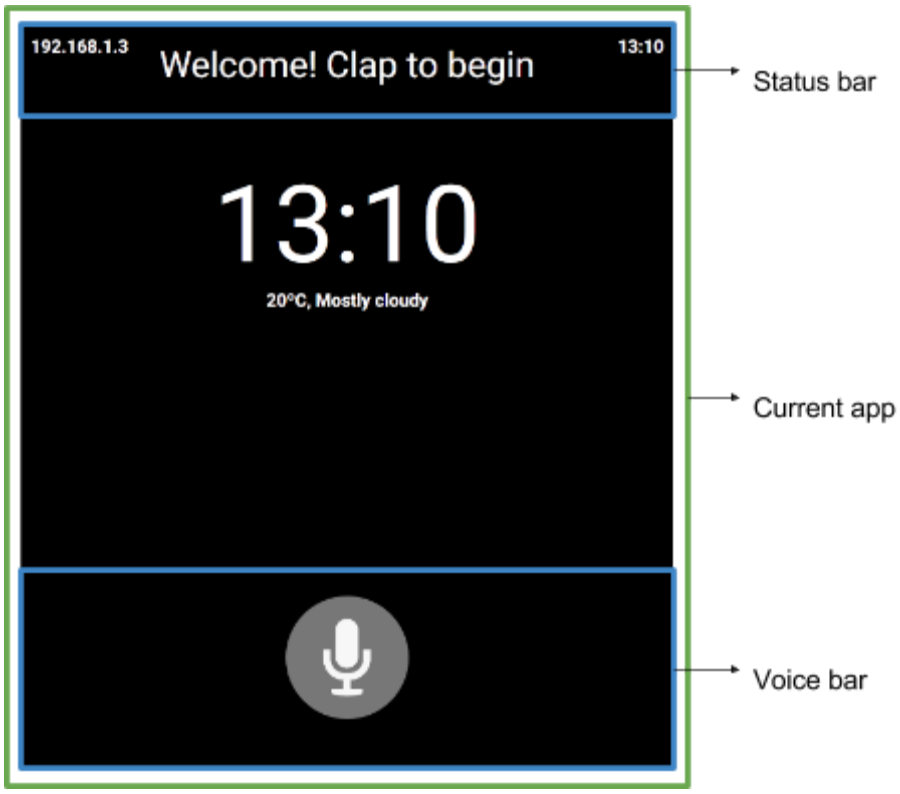


Figure 10. User Interface for MirrorOS

The center of the screen shows the current running app and there currently are no guidelines for UI so developers can show anything they want there. In the example we see the Home app.

Finally, In the bottom part of the screen there's an overlaid microphone icon that pops up when the voice recognition is triggered.

This user interface is completely responsive so it's possible to have different sized mirrors and the OS will adapt to it automatically. The included default apps are also responsive but It's up to app developers to implement this feature.

Voice Input

The voice recognition feature in MirrorOS uses an online API made by Google. The API is not officialy supported and it has a 50 query a-day limit but it is the best one available. To use the API you need to make an HTTP POST request to the API's url with a mono FLAC audio file with a 16000 bit-rate. To integrate the service with the OS I created the following bash script:

```
#1/bin/bash
timeout 3 arecord -D "plughw:0,0" -q -f cd -t wav | avconv -analyzeduration 1 -y -i - -ar 16000 -acodec flac file.flac

wget -q -U "Mozilla/5.0" --post-file file.flac --header "Content-Type: audio/x-flac; rate=16000" -O - "http://www.google.com/speech-api/v2/recognize?lang=en-us&client=chromium&key=AIzaSyCXResRGQcGCQhXChLksKds0OBN7N0_aH8" >out.json
cat out.json

rm file.flac
```

The script records a 3 second sound file using the main microphone, converts it to FLAC, sends it to the API and then prints the result as a JSON string to a file. This was done using a separate

script instead of doing it in JavaScript because access to the Microphone using the Web APIs was not possible in the Raspberry Pi. I also created another script that listens for claps and triggers the other script. This time I wrote it with python because it was a bit more complicated. To call the two scripts from MirrorOS I use the spawn method that NodeJS provides. It works like this:

1. The OS starts.
2. The python script that listens for claps starts.
3. If a clap is heard, the bash voice recognition script is run.
4. The query is processed by the OS and an app is chosen to launch.
5. The app launches and the query is sent to it.

Two months after developing this feature the API stopped responding. As I didn't know how to fix it, I switched to a different API made by IBM which was almost identical in the way it is used. The results, however, were not very good compared to Google's API. It often misunderstood words.

I searched again to find a solution for Google's API but I was not successful. After some testing I discovered that the FLAC file needed to be sent as mono and I was sending it as a stereo file. Changing this fixed it but as the API is not officially supported so things could change without warning and this could happen again. That's why I looked for a better solution and I found an official Web Speech API that works very well and has no usage limits. However, the Web Speech API only works when launching the OS as root to access the microphone so I will not implement it until I can make it work without root.

Gesture Input

The initial idea was to implement this feature using a camera. There were two options a regular USB webcam or the dedicated Pi Cam board which has direct access to the Pi's hardware and provides much higher frame-rate than a USB Cam. However, after doing some tests with OpenCV I found that it was not trivial to detect hands and gestures and it depended a lot on the lighting of the room so I decided to look for alternatives. The first thing that popped up was a board called HOVER specially designed for the Raspberry pi. The board, however, detected gestures from a very small distance so it wasn't ideal for my project.

Finally I decided that I would put two ultrasonic distance sensors on each side of the mirror. This would allow me to detect holding a hand in front of the mirror and hopefully detect left and right swipes. I bought the two sensors and once I got them I wrote a python script (see appendix 8) to detect the different gestures.

The resulting program was quite good although swipe gestures are not very reliable. To integrate the program with the OS I used the same NodeJS spawn child process feature that I used with the voice recognition scripts.

Smartphone Interaction

The smartphone is not a mandatory accessory for the Smart Mirror but it's a very convenient way to interact with the it because it can act like a remote control.

I created a companion app for MirrorOS that works through a web browser so technically it can be used in any device that has a web browser, not only a smartphone. The app connects to the socket server created by the Smart Mirror and it can send and receive messages through it.
To connect to the device the user must enter the IP Address that's shown in the Smart Mirror's status bar. The server and client code is powered by Socket.io, a NodeJS module.

Once connected, the user can see a list of apps to launch and he can interact with the mirror in different ways depending on the app. The app and the API for this are still in early stages of development but the core concept works perfectly.

Workflow

MirrorOS boots on top of Raspbian. To achieve this I modified the Raspbian boot sequence so it immediately starts MirrorOS after booting. After the initialization, the voice recognition service, gesture recognition service and a socket server are all started. Then the software looks for all the installed apps in a folder and it starts the default home app.

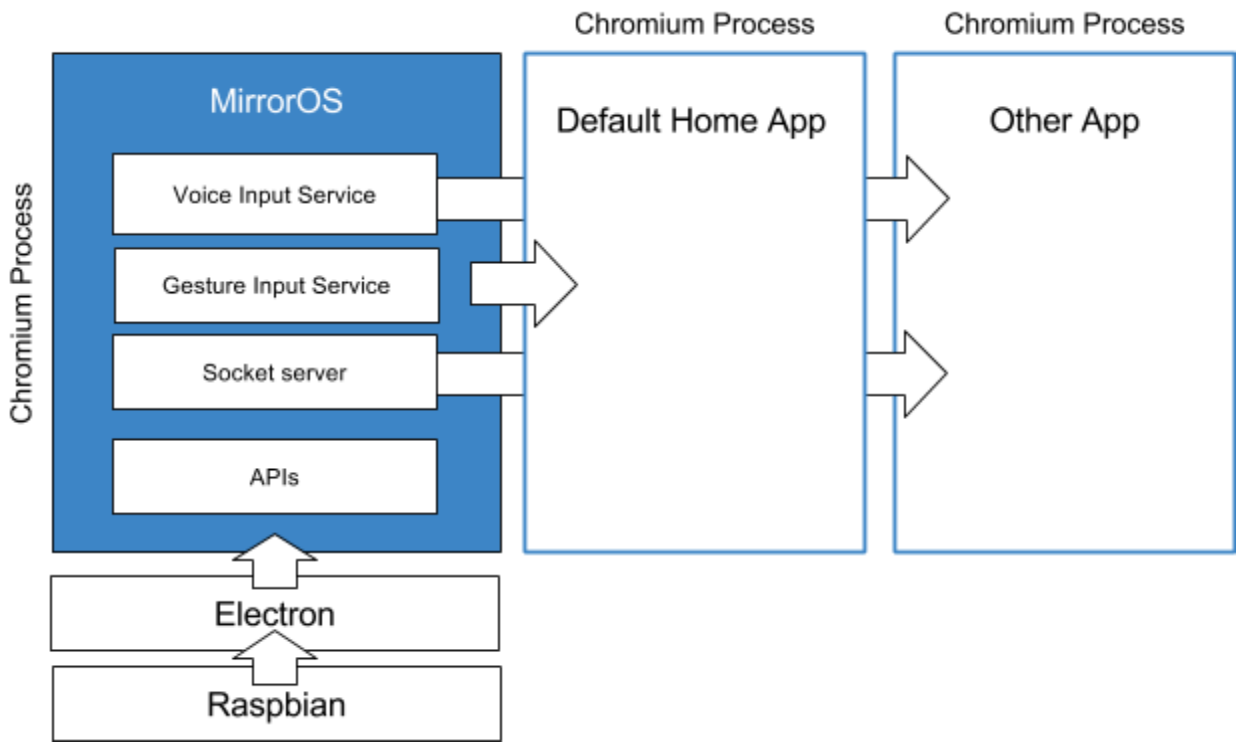


Figure 11. MirrorOS boot sequence and basic operation

Once the home app is open, the OS waits for user input through the voice input service or the socket server. If a user sends a query through one of the possible inputs, the OS processes it and decides which app to open based on the keywords defined by each app.

4.2.3 Developing Apps for MirrorOS

A typical MirrorOS app should have the following file structure:

index.html

Main html file for the app. Contains references to js and css.

js/app.js

Main js loigc file

css/style.css

styles for the app

manifest.json

App description. Defines the name, version, icon and voice input keywords.

icon.png

App icon

API

MirrorOS provides a simple API for developers to perform actions in a consistent way. The API works by making use of the inter process communication (IPC) system provided by Electron. This system enables you to communicate through processes using Javascript.

Developers can call these functions in Javascript to perform different actions consistently:

MOS.showToast(message, duration)

Shows a small message on the bottom part of the screen during the indicated duration.

MOS.showAlert(title, message, alertId)

Shows an alert message with two options that the user can select using gesture input. The alertId parameter is used to obtain the user's input with the *onAlertPositiveOption* and *onAlertNegativeOption* callback methods.

MOS.setMicrophoneEnabled(enable)

Setter to enable or disable the microphone's clap detection. Useful for apps that play audio so they don't trigger the microphone.

MOS.setGestureRecognitionEnabled(enable)

Setter to enable or disable gesture input. Gesture input is disabled by default. Any app that wants to use gesture input must call this function.

MOS.setTitle(title)

Sets the status bar message.

Developers can override these functions in Javascript to obtain information from the different inputs that the mirror has:

function onNewQuery(query)

Is called whenever a user uses the voice input and says one of the app's keywords. The parameter query contains what the user has said.

function onGesture(gesture)

Is called when a user makes a gesture. The parameter gesture contains one of the 4 types of gestures the user can perform. GESTURE_SWIPE_LEFT,GESTURE_SWIPE_RIGHT, GESTURE_HOLD_LEFT,GESTURE_HOLD_RIGHT)

function onAlertPositiveOption(alertId)

Is called when a user responds positively to an alert message. The parameter is used to identify which alert the user is responding to because it contains the id that was originally sent with the *showAlert* method.

function onAlertNegativeOption(alertId)

Is called when a user responds negatively to an alert message. The parameter is used to identify which alert the user is responding to because it contains the id that was originally sent with the *showAlert* method.

4.2.4 Default Apps in MirrorOS

At the moment, I have developed 3 apps for the MirrorOS but the possibilities are endless. These are the apps included so far:

Home

Home acts as a home screen for the Smart Mirror and is the first app to start once MirrorOS boots. It contains time, weather and reminder information. It responds to voice queries to get weather information, set timers and reminders and create notes. The weather information is obtained using the API provided by Forecast.io.

YouTube app

The YouTube app lets you watch YouTube videos on the mirror. You can ask the mirror what video you want to watch or you can send a video from your phone if you have the companion app installed. It responds to the following keywords: “youtube”,”video”.

To be able to play YouTube videos on the Smart Mirror I used a NodeJS module called node-ytdl-core. This module downloads the video on the device and streams it at the same time through an HTML5 video tag.

Image search app

The image search app creates a gallery of pictures from different sources including Bing and Instagram. It responds to the following keywords: “pictures”, “pictures of”, “images”, “photos”. Using the companion app or gestures you can navigate through the gallery.

The app gets the content from the Bing image search API as well as the Instagram API.

Budget

The following table shows an approximation of the total cost of the project and each of the components that were used. The biggest part of the budget was the display, followed by the frame, the mirror and the Raspberry Pi.
This information could be useful for other people that want to build their own Smart Mirror.

Table 1. Budget for the Smart Mirror parts

One-way mirror	47€
LG Monitor TV 24”	179€
TV support	12€
Raspberry Pi 2	40€
Power adapter for Raspberry Pi	5€
MicroSD card (32GB)	9€
HDMI cable	9€
HC-SR04 Ultrasonic Distance Sensors	5€
Jumper wires, resistors, plates	8€
Frame and back support	90€
TOTAL	**404 €**

5. Results

The final results were very satisfying. The device can successfully recognise voice input and hand input and it can connect to other devices such as smartphones and use them as a remote control. It has 3 working apps and a simple developing environment to make it very simple to create further apps for it.

Overall I would say I met most of the goals that were set at the beginning surpassing them in some aspects.

5.1 Hardware

The hardware looks very good but the glass could be more reflective. In some conditions you can see a bit of the interior of the device. See appendix 7.

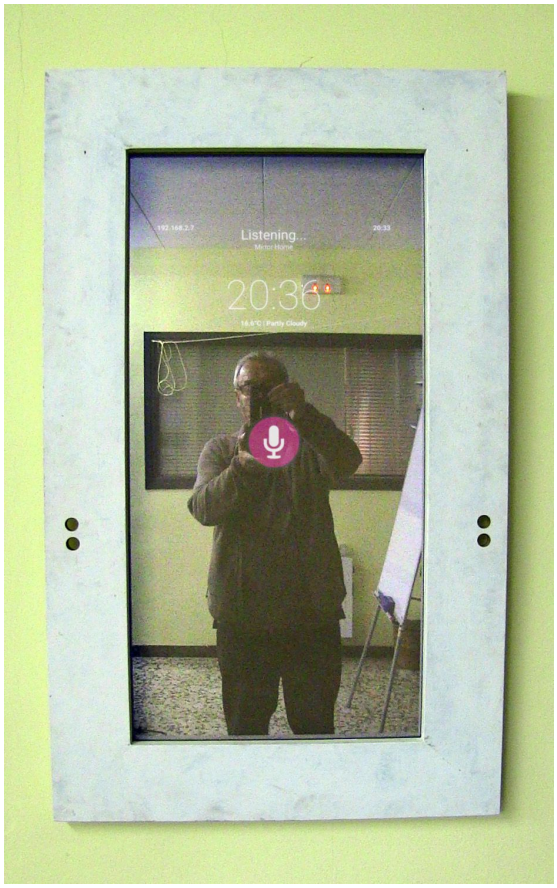


Figure 12. Picture of the near-finalized Smart Mirror hardware.

5.2 Software

OS and Home app

The operating system’s UI simply consists of the top status bar and the bottom microphone icon.

The simple Home app works as intended, showing time and weather information and being able to set reminders, timers and notes.

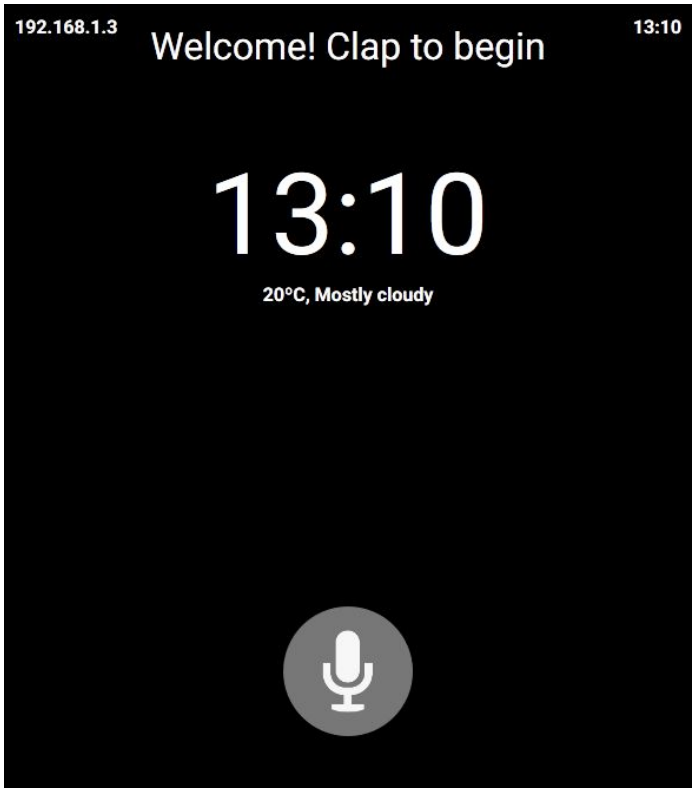


Figure 13. Home App on the Smart Mirror

YouTube App

The YouTube app shows a loading indicator while downloading the initial part of the video. Then it plays the video including sound, as expected. In this app you can perform a holding gesture on one of the sensors to pause the video and resume it.

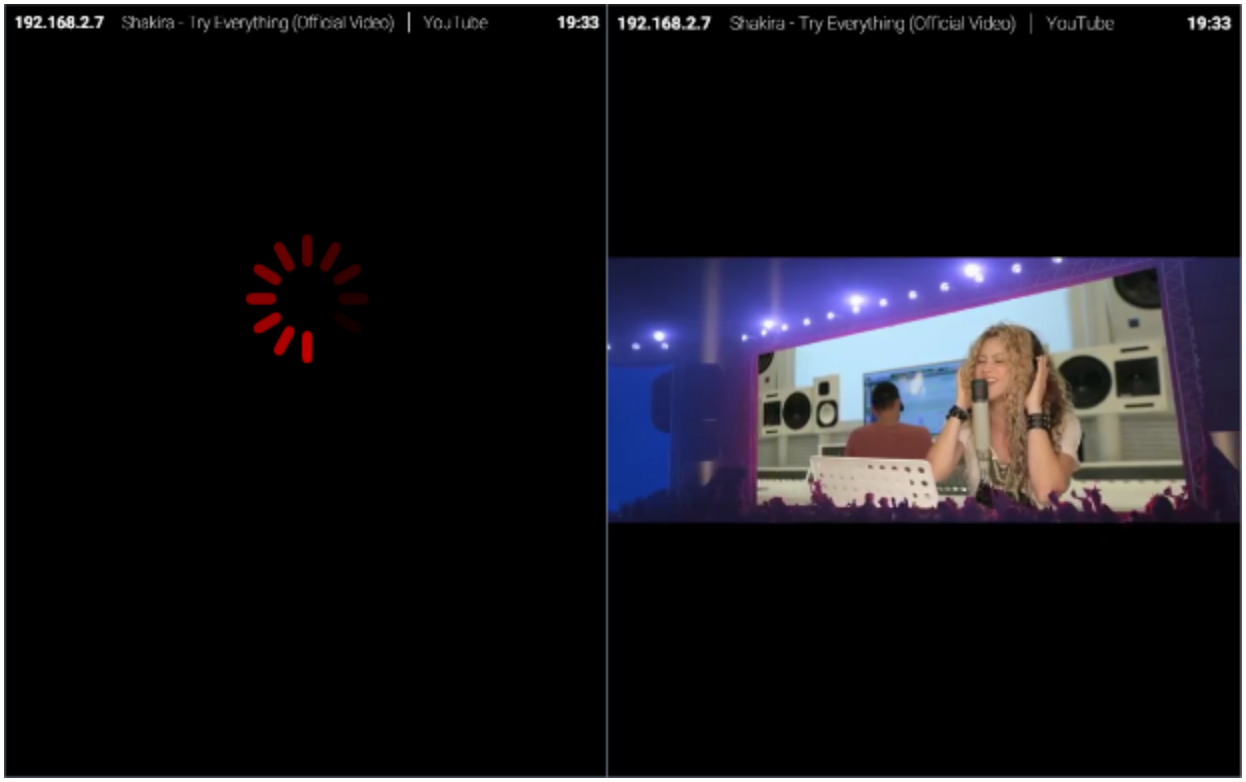


Figure 14. YouTube App playing a video in response to the query “Shakira video” on the Smart Mirror

Image Search App

The image search app has a nice grid user interface where it places images gathered from different sources. In this app you can use swipe gestures and holding gesture to navigate through the gallery of pictures.

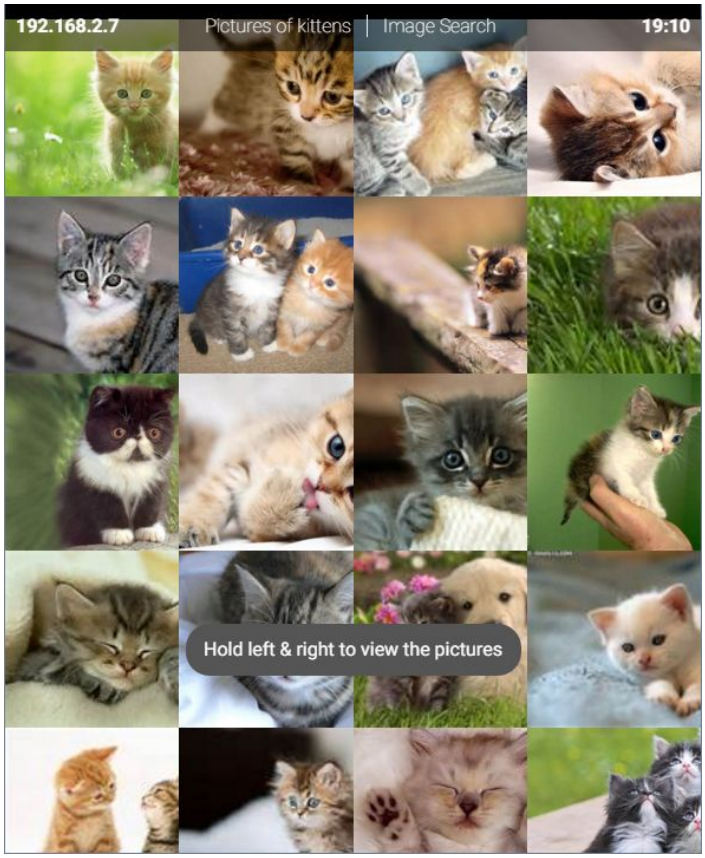


Figure 15. Image search App showing pictures in response to the query “Pictures of kittens”

Companion App

The companion app’s user interface was not finished on time but the functionality is working as initially intended. You are able to connect to the Smart Mirror by entering it’s IP address and you can send any query to it.

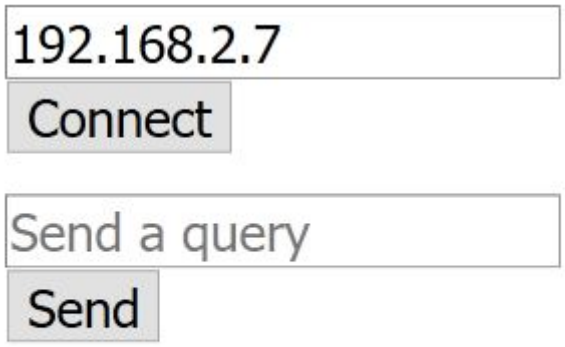


Figure 16. Companion App running on a smartphone

In the future I would like to extend the functionality of the app by adding features such as a store to find and install Smart Mirror apps, pairing with the mirror using NFC or a QR code and finally, being able to configure some of the Smart Mirror's settings.

6. Conclusions

The main strengths of this project are that this is a new kind of smart device that people don't see every day and it looks very spectacular. The platform has a very simple API that makes it very easy for developers to make apps. The voice recognition is very accurate thanks to Google's services. The smartphone integration works very well and it is something that hasn't been done with smart mirrors before.

Of course there are also weaknesses: the app ecosystem is currently very small, the glass could be more reflective but it can be easily changed, the swipe gestures are sometimes unreliable and finally I would have liked to have the hardware and software more decoupled because currently the sensors and microphones are tied to the software and it can be difficult to make the OS work with different hardware. However, this can also be solved given enough time by making the software more modular.

There are many future possibilities for this project and hopefully it will be continued. For the software, It would be interesting to create an installer for it or even bundle it as a Linux distribution to be able to install it very easily on any Raspberry Pi device. It would also be good to make some changes to make it truly multiplatform. The companion app needs a new UI, maybe an app repository and also the ability to easily change settings for the mirror. A community around the OS and the hardware should be created so people can help each other build and evolve these devices and create apps for them. Once polished, the software could be made open-source. Finally, for the hardware part, the glass panel could be replaced for a more reflective one and there's a new, recently released Raspberry Pi 3 that could bring improvements to the overall performance of the device.

7. Final Thoughts

This has been a very satisfying project to work on. It has been enriching and most importantly, I had fun doing everything and for me this is very important because it motivated me to continue developing it and to keep adding features.

Developing this project I have learned many fascinating things. I feel like this list would be endless but I will try to explain some of the most important ones. Electron is a very interesting project that is growing very fast and I think it will soon become the main tool for developing desktop apps. Raspberry Pi is also incredible and I can't wait to see how the project evolves and what new things the Raspberry Pi foundation will bring us.

Also, after three years of barely touching any electronics it has been thrilling to get the ultrasonic sensors to work and communicate with a web application. I was also very impressed with how web sockets work and how simple it is to implement them thanks to Socket.io. Finally I've learned lots of basic things about how linux works but I still feel I have much to learn about it and that's a good thing.

As you can see the list of positive results is very long and I think I even left out lots of things but in general my conclusion is that I really enjoyed this project and I hope that I can continue building similar things and I'm very excited about what the future of the Maker community will bring us.

References

Internet of Things Global Standards Initiative. *ITU*. Retrieved 26 April 2016. http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx

Maker Culture (chapter in Innovating Pedagogy 2013) (PDF). The Open University. Retrieved 20 April 2016. http://www.open.ac.uk/people/my-profile

How Can I Get Started with Home Automation? (2013) Retrieved 20 April 2016. http://lifehacker.com/how-can-i-get-started-with-home-automation-510246491

Magic Mirror (2014) Retrieved 20 April 2016. https://www.raspberrypi.org/blog/magic-mirror/

GitHub/MichMich/MagicMirror (2016) Retrieved 20 April 2016. https://github.com/MichMich/MagicMirror

GitHub/HannahMitt/HomeMirror (2016) Retrieved 20 April 2016. https://github.com/HannahMitt/HomeMirror

Smart Mirror Retrieved 20 April 2016. http://smart-mirror.io/

Medium (2016) Retrieved 24 February 2016. (https://medium.com/@maxbraun/my-bathroom-mirror-is-smarter-than-yours-94b21c6671ba#.q4932hjfc

PANL Retrieved 20 April 2016. http://getpanl.com/

Twitter Product Hunt Retrieved 20 April 2016 https://twitter.com/ProductHunt/status/729893082112483328

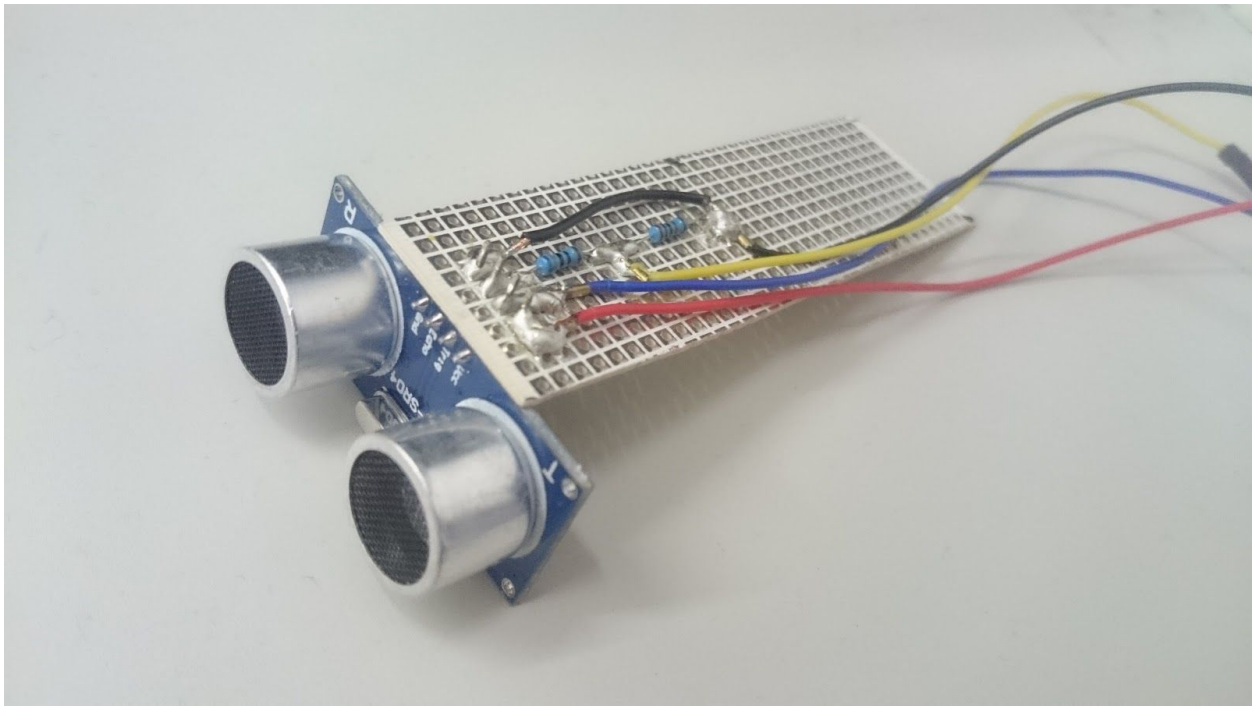
Loy, J (1999) Two-Way Mirrors. Archived from the original on March 13, 2005. Retrieved 8th May 2016

Appendices

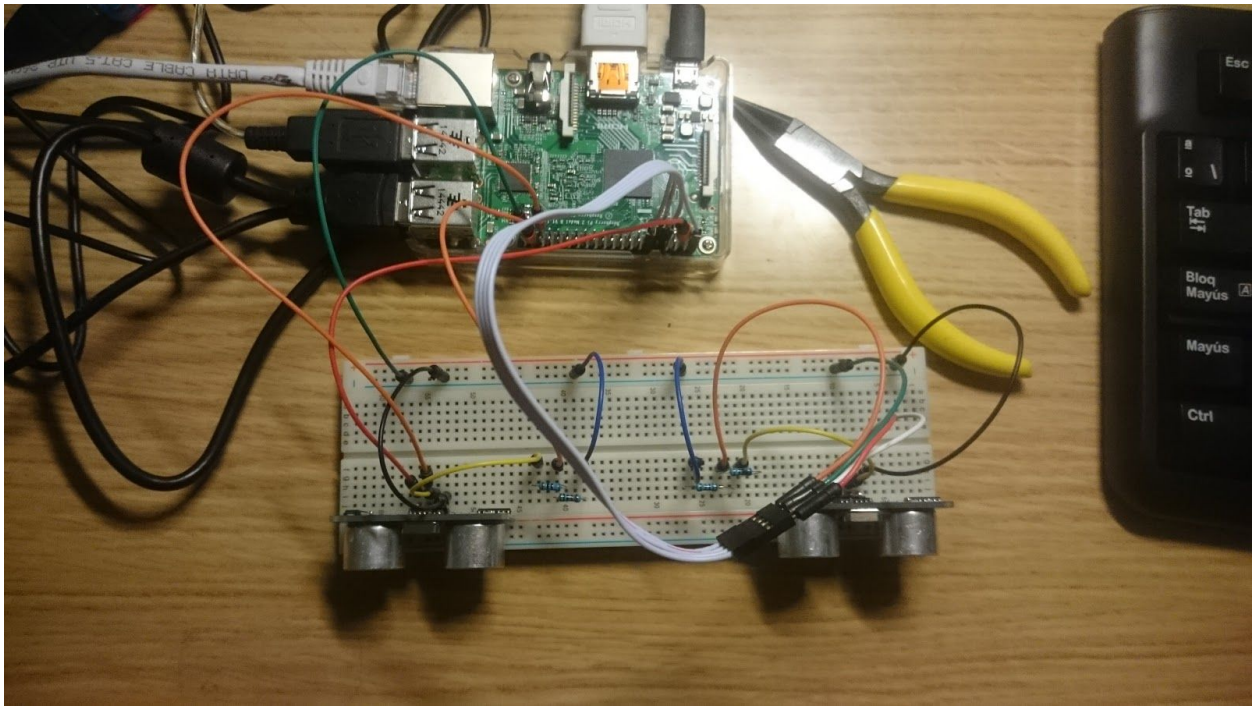
Appendix 1. Wooden back support

Appendix 2. Wooden front frame.

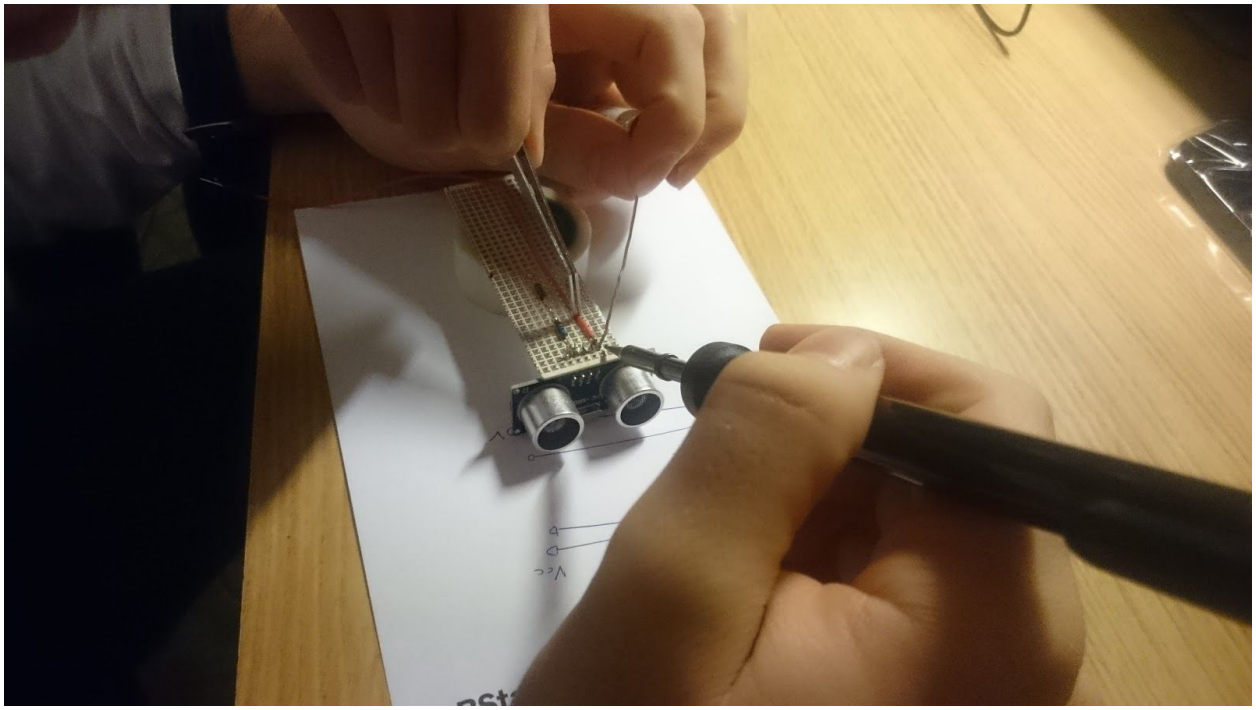
Appendix 3. Ultrasonic sensor with the soldered plate



Appendix 4. Testing the ultrasonic sensors on a breadboard.

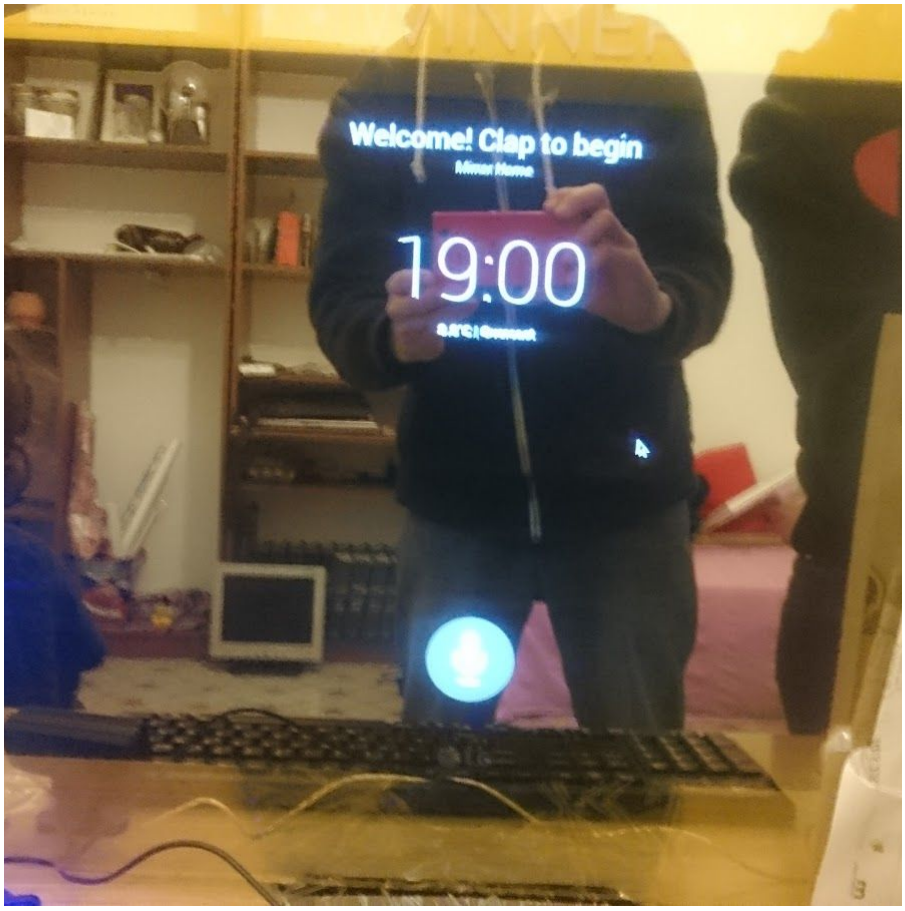


Appendix 5. Soldering the resistors for the sensors



Appendix 6. Fully assembled device (only the glass panel missing)

Appendix 7. Fully assembled and working device.



Appendix 8. Python script

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

TRIG = 7
ECHO = 12
TRIG2 = 35
ECHO2 = 38

MIN_DISTANCE = 15
TARGET_HOLD_COUNT = 8

GPIO.setup(TRIG,GPIO.OUT)
GPIO.output(TRIG,0)

GPIO.setup(ECHO,GPIO.IN)

GPIO.setup(TRIG2,GPIO.OUT)
GPIO.output(TRIG2,0)

GPIO.setup(ECHO2,GPIO.IN)

time.sleep(0.1)

prevSensorL = False
prevSensorR = False
holdCountL = 0
holdCountR = 0
lastInteractionTime = 0

print ("Starting gesture recognition")

try:
        # here you put your main loop or block of code
        while True:
        #print "Starting measurement"
        GPIO.output(TRIG,1)
        time.sleep(0.00001)
        GPIO.output(TRIG,0)

        while GPIO.input(ECHO) == 0:
                pass
        start = time.time()

        while GPIO.input(ECHO) == 1:
                pass
        stop = time.time()

        GPIO.output(TRIG2,1)
        time.sleep(0.00001)
        GPIO.output(TRIG2,0)

        while GPIO.input(ECHO2) == 0:
```

```
                pass
        start2 = time.time()

        while GPIO.input(ECHO2) == 1:
                pass
        stop2 = time.time()

        interaction = False

        distance = (stop - start) * 17000
        distance2 = (stop2 - start2) * 17000

        sensorR = distance < MIN_DISTANCE
        sensorL = distance2 < MIN_DISTANCE

        #if prevSensorL and sensorR:
                #print "Swipe right"
        #if prevSensorR and sensorL:
                #print "Swipe left"

        if prevSensorL and sensorL:
                holdCountL = holdCountL + 1
                if holdCountL < TARGET_HOLD_COUNT:
                print ("0")

        if prevSensorR and sensorR:
                holdCountR = holdCountR + 1
                if holdCountR < TARGET_HOLD_COUNT:
                print ("1")

        if holdCountL >= TARGET_HOLD_COUNT:
                holdCountL = 0
                print ("10")

        if holdCountR >= TARGET_HOLD_COUNT:
                holdCountR = 0
                print ("11")

        interaction = sensorL or sensorR

        prevSensorL = sensorL
        prevSensorR = sensorR

        if interaction:
                lastInteractionTime = time.time()
        elif time.time() - lastInteractionTime > 1:
                #print "Resetting"
                prevSensorL = False
                prevSensorR = False
                holdCountL = 0
                holdCountR = 0
                lastInteractionTime = time.time()

        time.sleep(0.1)
except KeyboardInterrupt:
        # here you put any code you want to run before the program
        # exits when you press CTRL+C
```

```
        print ("exiting")
except:
        # this catches ALL other exceptions including errors.
        # You won't get any error messages for debugging
        # so only use it once your code is working
        print ("Other error or exception occurred!")
finally:
        GPIO.cleanup() # this ensures a clean exit
```