



9/12/2019

Smart Mirror

Formal Report



Rahul Mishra

Rahul Gudwani

Mohammed Dad Khan

Table of Contents

Abstract:	2
Introduction:	2
Components Used:	2
Design:	3
Development:	4
Step-1: Installing Raspbian stretch on Raspberry PI.	4
Step-2: Installation of Magic Mirror modules.....	4
Step-3: Setting up the Face recognition module	5
Configuring the Raspberry Pi:	5
Build OpenCV:	5
Capture & Train the Model:.....	7
Training our model	7
To Test our trained model.....	7
Setting up the Module:.....	8
Step-4: Installing YouTube Screencast	9
Step-5: Installation of Alexa:	9
Product information:	10
Create a Security Profile	11
Enable Security Profile:	13
Authorize Your Alexa Product:	14
Cost Analysis:	15
Conclusion and Future Work:	15
References:	16

Abstract:

The project describes the design, construction, and operation of the smart mirror. Smart Mirror is one of the Raspberry Pi's applications. It is a two-way Mirror Embedded with a monitor. Thus, Providing different information like a clock, news, current weather, weather forecast, and more. Smart Mirror also has Alexa Integrated which enables you to play music, add reminders, get notifications about traffic and much more. It also has facial recognition which enables the user to unlock the mirror and view the home screen.

Introduction:

Smart Mirror is meant as smart home technology to provide convenient access to media and information that people may be interested in daily. The mirror will display the user's customized information so that users can get some basic daily information briefly. This will save users time on their busy morning routine, which will, as a result, relieve stress and help people get out of the door on time in the morning. The mirror will display information that users can access through their phones or other technology in the morning in a way that is readily available as soon as they enter the bathroom. This will allow the user to plan their day while at the same time completing their daily routine in front of the mirror.

Some of the information that will be provided to users includes the time and date, the weather for the day, the daily schedule, the to-do list, the preferred news source for users, traffic delays on the way to work, and more. The mirror will be hands-free, automatically turning on and off when someone enters and exits the room. Since mirrors are commonly used by multiple people, facial recognition will be introduced, and the mirror will show personalized data per person, with a simple guest profile of faces not recognized by the mirror. App communications will be accomplished by voice commands, leaving the device hands free for comfort and ease of use.

One of the key principles behind this project is the platform created for open source development. This will allow developers to expand this project by creating open applications. Users can design widgets and can implement it with the magic mirror module. With the level of flexibility offered by the mirror, users will be able to set up the mirror just as they see fit.

Components Used:

- **Monitor:** The monitor is used for displaying the information. The Thinner the display is, the better it fits in the wooden frame. Thicker displays are wide and are difficult to fit in the frame.
- **Wooden Frame:** This is built based on the size of the two-way mirror. The measurement is done after we have the two-way mirror ready on hand.
- **Two-Way Mirror:** This two-way Mirror is 70% Reflective and 30% Transparent. The reflective helps the user to view themselves and the Transparent side will help user to view information from the monitor.
- **Raspberry Pi 3B+:** This the motherboard of our project and is used for software purposes. Most of the modules are installed with the help of this device.

- **Sound Card:** This is used for providing audio input and output signaling to and from the computer.
- **Speaker:** This is used in audio output from YouTube, Spotify or Alexa.
- **Microphone:** This captures audio by converting sound waves into electrical signals. This is then amplified as an analog signal and converted to a digital signal processed by computer or any other digital device.
- **PI Camera:** The PI camera is used to take high definition photos and videos and is compatible only with Raspberry Pi. The Raspberry Pi is of 8Megapixels and can shoot 1080P video at once.
- **HDMI Cable:** This is used for the connection between Raspberry pi and the monitor.
- **Power Supply:** This is given

Design:

The Designing of this project is done in two parts. The first involves the construction of our wooden frame that describes out an outer layer for our mirror. Both Raspberry Pi and Monitor need to be embedded with the frame and secured to avoid any possible damage to other components that taking our frame around. Once these two components are fixed, the mirror will cover the front layout of the frame and the other information is to be displayed on the monitor. The Characteristics of the mirror is such that the front side of the mirror is reflective and acts as a mirror while the backside is transparent, anything which is displayed on the monitor is seen on the mirror surface. Thus, the mirror needs to be placed on the monitor to view the information. In addition, there is space left out for placing Raspberry Pi, Extension cord for power supply to the monitor and Raspberry Pi. The Raspberry Pi has several USB ports that are used for keyboard and mouse. The Frame is also electrically insulated to prevent damage to the Pi Board and other devices too.

The Second part involves designing of the Smart Mirror application. Since the objective is to have a mirror that displays certain information, the design must be such that the existing surface of the mirror that covers the monitor is used in the most efficient manner. The information to be displayed must be arranged in such a way that the display area of the monitor is used efficiently and does not stand in the way of reflection. The best way to achieve this is to place the widgets around the corner of the screen leaving the mid part of the display surface empty so that the person can see himself/herself reflected in the mirror clearly.

Development:

Upon Design Completion the development phase begins. The operating system used for our Smart Mirror is Raspbian Stretch with Desktop OS

Step:1: Installing Raspbian stretch on Raspberry PI.

We download the Raspbian software image file and write the same image on the 32GB SD card. This OS is required for the installation of Magic Mirror Software and configuring all the modules. The basic setup is done i.e. setting up the Wi-Fi and updating the software if available.

- First, Expanding the filesystem.
`sudo raspi-config`
- Select advanced options menu item:
- Followed by selecting the Expand file system.
- Once prompted select the 1st option Expand File system and hit Enter on the keyboard,
- You may be prompted to reboot
`sudo reboot`

Step-2: Installation of Magic Mirror modules

1. To set up the magic mirror platform we need to write following command:

```
pm2 startup
```

2. If you get an error that says, “Unable to install dependencies!” then you have to install older version of electrons:

```
sudo npm install -g electron@1.7.6
```

3. Now you are ready to set up the magic mirror and type in the following

```
cd~/MagicMirror
```

```
DISPLAY=:0npm start
```

4. To install the module for the magic mirror or your own custom module follow these steps:
 - Navigate to the modules folder via the following command:

```
cd~/MagicMirror/modules
```

- Now we must make clone the module from Github

```
git clone https://github.com/author/module-name
```

- Add your brand module to your config! this is the file `config/config.js`. After editing the module name, you can edit and add:

```
{
  module: 'module name',
  position: 'position',
  header: 'optional header',
  config: {
    extra option: 'value'
  }
},
```

Step-3: Setting up the Face recognition module

Configuring the Raspberry Pi:

To get the face recognition working, we need to install Open-CV on Raspberry-Pi first. This installation might take somewhere around 2-2.5hrs.

To start with, we need to install and update all the elements before compiling:

```
sudo apt-get update
sudo apt-get upgrade
sudo rpi-update
sudo reboot
sudo apt-get install build-essential git cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev
sudo apt-get install libatlas-base-dev gfortran
cd ~
git clone https://github.com/Itseez/opencv.git
cd opencv
git checkout 3.3.0
cd ~
git clone https://github.com/Itseez/opencv_contrib.git
cd opencv_contrib
git checkout 3.3.0
```

Build OpenCV:

Building this will take a lot of time. First, get the PIP Cross version:

```
wget https://bootstrap.pypa.io/get-pip.py
```

For Building OpenCV, we have Python 3.0 is used here.

```
sudo apt-get install python3-dev
sudo python3 get-pip.py
```

Before we start the compile, the process increases the Swap Size. This enables openCV to open up all the four cores of the Raspberry PI without compile hanging due to memory issues.

Open /etc/dphys-swapfile and then edit the CONF_SWAPSIZE variable:

```
# set size to absolute value, leaving empty (default) then uses computed value
# you most likely don't want this, unless you have a special disk situation
# CONF_SWAPSIZE=100
CONF_SWAPSIZE=1024
```

To activate the new swap space, restart the swap service:

```
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
```

Note: It is highly recommended that you change this setting back to the default when you are done compiling and testing the install.

The last step is to build OpenCV:

```
pip install numpy
cd ~/opencv
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
      -D CMAKE_INSTALL_PREFIX=/usr/local \
      -D INSTALL_C_EXAMPLES=OFF \
      -D INSTALL_PYTHON_EXAMPLES=ON \
      -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
      -D BUILD_EXAMPLES=ON ..
make -j4
sudo make install
sudo ldconfig
```

As mentioned earlier, installation may take around 2 - 2.5hrs. Detailed information for installing and building OpenCV is provided on the link below:

<https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/>

Step:3 Installing the modules and Dependencies for MMM-Facial Recognition:

1. To make the Face Recognition working we have to get the module using the following command line:

```
cd ~/MagicMirror/modules/

git clone https://github.com/normyx/MMM-Facial-Recognition-OCV3.git
```

2. In the MMM-Facial-Recognition-OCV3 modules directory, enter in terminal:

```
cd ~/MagicMirror/modules/MMM-Facial-Recognition-OCV3
```

3. Install the need dependencies by running the command :

```
npm install
```

4. If you will run the different tools with Python 2.7, install the cross-compatible features with :

```
pip install future
```

Capture & Train the Model:

For capturing the Image to create the dataset of the person.

Run python tools.capture.py.

1. Run `python tools.capture.py`.
2. Decide whether you want to capture images from your picam or convert existing .jpg images.
3. Enter the name of the person you are about to capture. Images will be stored in a folder named after the captured person in `training_data/`.
4. Follow on-screen instructions.

Training our model

1. Make sure you have captured all your images.
2. Run `python tools.train.py`. The script will automatically scan the directory `training_data/` for your images.
3. Wait. You will end up with a `training.xml` file in the current directory.
4. Copy down the `['name1', 'name2', 'name3']` part because you will later need it for setting up your mirror's face recognition and to test your face recognition model.

To Test our trained model

1. Make sure your `training.xml` from running `train.py` is in this directory
2. Specify your user labels in the environment with

```
export FACE_USERS=name1,name2,name3
```

3. Run `python tools.facerecognition.py` to test your trained model.

Setting up the Module:

To set up the module in MagicMirror², add the following script in the `config.js` file in the `config/ MagicMirror2` directory (Modify the script regarding the Algorithm, file location, ...).

```
{
  module: 'MMM-Facial-Recognition-OCV3',
  config: {
    // Threshold for the confidence of a recognized face before it's
    // considered a
    // positive match. Confidence values below this threshold will be
    // considered
    // a positive match because the lower the confidence value, or distance,
    // more confident the algorithm is that the face was correctly detected.
    threshold: 80,
    // force the use of a usb webcam on raspberry pi (on other platforms
    // this is always true automatically)
    useUSBCam: false,
    // Path to your training xml
    trainingFile: '/home/pi/MagicMirror/modules/MMM-Facial-Recognition-
    OCV3/training.xml',
    // recognition intervall in seconds (smaller number = faster but CPU
    // intens!)
    interval: 2,
    // Logout delay after last recognition so that a user does not get
    // instantly logged out if he turns away from the mirror for a few seconds
    logoutDelay: 15,
    // Array with usernames (copy and paste from training script)
    users: [],
    //Module set used for strangers and if no user is detected
    defaultClass: "default",
    //Set of modules which should be shown for every user
    everyoneClass: "everyone",
    // Boolean to toggle welcomeMessage
    welcomeMessage: true
  }
}
```

In order for this module to do anything useful, you have to assign custom classes to your modules. The class-default (if you don't change it) is shown if no user is detected or a stranger. The class everyone (if you don't change it) is shown for all users. To specify modules for a certain user, use their name as class name.

```
{
  module: 'example_module',
  position: 'top_left',
  //Set your classes here seperated by a space.
  //Shown for all users
  classes: 'default everyone'
},
{
  module: 'example_module2',
  position: 'top_left',
  //Only shown for name1
  classes: 'name1'
}
```

Detailed Information for Facial Recognition installation and procedure is given in the below link.

<https://github.com/normyx/MMM-Facial-Recognition-OCV3/blob/master/README.md>

Step-4: Installing YouTube Screencast

To install the YouTube Screencast, follow the steps as mentioned below:

1. Navigate to the modules directory via the follow command:

```
cd MagicMirror/modules
```

2. Clone the module from github:

```
git clone https://github.com/kevinatown/MMM-Screencast.git
```

3. Navigate to the MMM-Screencast directory:

```
cd MMM-Screencast
```

4. Install the dependencies:

```
npm install
```

5. Add the following configuration to the modules array in the config/config.js file:

```
{
  module: 'MMM-Screencast',
  position: 'center', // This position is for a hidden and not
                      // the screencast window
  config: {
    position: center,
    height: 300,
    width: 500,
  }
}
```

Step-5: Installation of Alexa:

After you've registered for an Amazon developer account, you'll need to create a new Alexa product and security profile.

1. Login to Amazon Developer Portal
2. Navigate to the Alexa > Alexa Voice Service console.
3. Choose Create Product, and follow then instructions below to 1). register a product, 2). create a security profile, and 3). authorize your security profile to use Login With Amazon (LWA).



Product information:

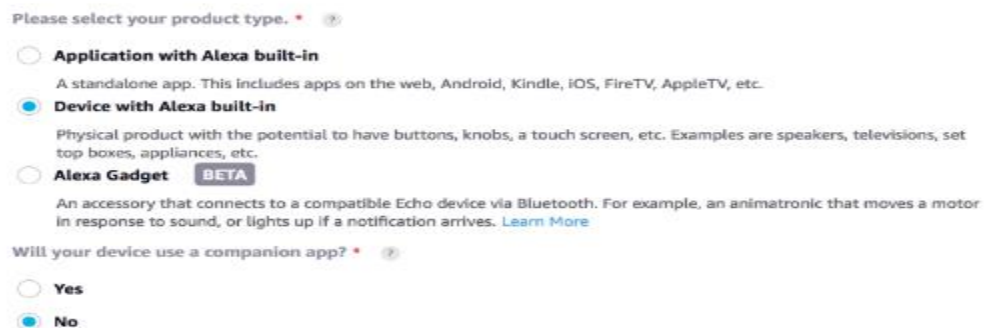
Now you will provide information to register your product

1. Product Name - This is what gets displayed to users when they register an instance of the product with Amazon.
2. Product ID - A simple identifier for your product



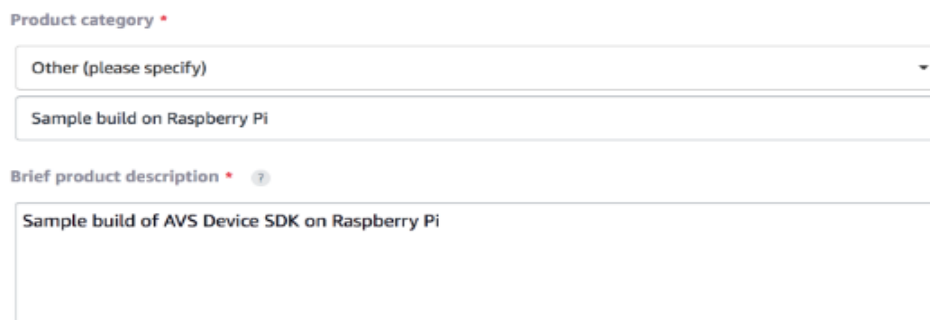
The screenshot shows a form titled "Product information" with the subtitle "Tell us about your Product". It contains two input fields: "Product name" with the value "Sample product for AVS" and "Product ID" with the value "sample_product_avs".

3. Choose the product type Device with Alexa built-in. When asked if your product will use a companion app, select No.
4. For Product Category, select Other, and enter a description of the product type, such as Raspberry Pi.
Note: For a commercial product, you should select the best option available.



The screenshot shows a form titled "Please select your product type." with three radio button options: "Application with Alexa built-in", "Device with Alexa built-in" (which is selected), and "Alexa Gadget" (marked as BETA). Below the "Device with Alexa built-in" option is a description: "Physical product with the potential to have buttons, knobs, a touch screen, etc. Examples are speakers, televisions, set top boxes, appliances, etc." Below the "Alexa Gadget" option is a description: "An accessory that connects to a compatible Echo device via Bluetooth. For example, an animatronic that moves a motor in response to sound, or lights up if a notification arrives. Learn More". Below these options is a question "Will your device use a companion app?" with two radio button options: "Yes" and "No" (which is selected).

5. Enter a Brief product description.



The screenshot shows a form titled "Product category" with a dropdown menu set to "Other (please specify)". Below the dropdown is a text input field with the value "Sample build on Raspberry Pi". Below this is a section titled "Brief product description" with a text input field containing the value "Sample build of AVS Device SDK on Raspberry Pi".

- When asked how users will interact with your product, select the option(s) that best describe your product's functionality.

Example:

How will end users interact with your product? *

☒ **Touch-initiated**
A user's primary way to interact with Alexa is by tapping or holding a button.

☒ **Hands-free**
Hands-free products allow users to interact with Alexa by using their voice at a close distance.

☐ **Far-field**
Far-field products allow users to interact with Alexa by using their voice from a long distance.

- When asked if you will distribute this project, select No. Note: For a commercial product, you should select Yes.
- When asked if this product is for children, select No.

Do you intend to distribute this product commercially? *

☐ Yes

☒ No

Is this a children's product or is it otherwise directed to children younger than 13 years old? * [Learn More](#)

☐ Yes

☒ No

NEXT

xi. Click **Next**.

Create a Security Profile

On this page, we'll create a *new* Login with Amazon (LWA) security profile. This associates user data and security credentials with one or more products.

- Select Create a New Profile.

Select a Security Profile

A security profile associates user data and security credentials with one or more related products.

Security Profile *

or **CREATE NEW PROFILE**

2. Enter a name and description for your security profile, then click Next. For example:

- Security Profile Name: AVS Test Product
- Security Profile Description: This is for a reference build of the AVS Device SDK on Raspberry Pi. Note: These are suggested values. You provide *custom information* for Security Profile Name and Security Profile Description.

Create a new Security Profile

A security profile associates user data and security credentials with one or more related products.

Security Profile Name • ⓘ

AVS Test Product

Security Profile Description • ⓘ

This is for a reference build of the AVS Device SDK on Raspberry Pi

NEXT

3. Now you'll generate a Client ID. Navigate to Platform Information > Other devices and platforms. Enter a Client ID name, then select Generate ID.

Platform information

Specify the settings for the websites or mobile apps that will use Login with Amazon with the selected Security Profile.

Web Android / Kindle iOS **Other devices and platforms**

Code-based linking authorization on a device or other platform requires Client IDs to be used when making requests to get a device access tokens. Different devices can use their own Client ID. [Learn More](#)

Client ID name

Identifies the device or platform which you are using for authorization. For example, prototype TV set top box. This value is not customer-facing.

avs_test_product

GENERATE ID

4. Now, select Download. This will save a config.json file to your computer that includes the clientId and productId associated with your security profile. This information will be used by the sample app as part of the setup process.

Platform information

Specify the settings for the websites or mobile apps that will use Login with Amazon with the selected Security Profile.

Web Android / Kindle iOS **Other devices and platforms**

Code-based linking authorization on a device or other platform requires Client IDs to be used when making requests to get a device access tokens. Different devices can use their own Client ID. [Learn More](#)

Client ID name

test product

Client ID

amzn1.application-oa2-client.1d5e1f1bc92047178e2b9a31b13a9769 COPY

DOWNLOAD

+ ADD ANOTHER

☒ I agree to the Amazon Developer Services Agreement, including the Alexa Voice Service Program Requirements.

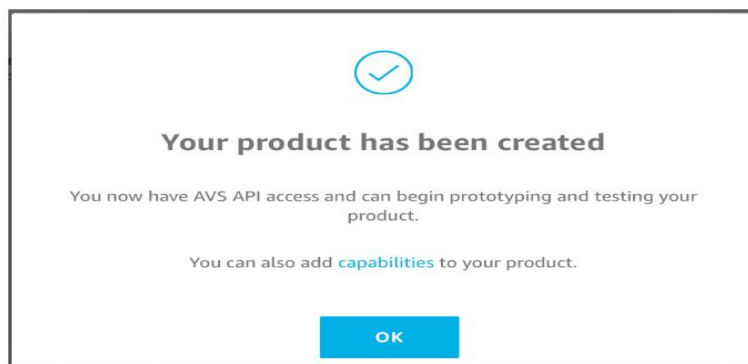
FINISH

5. Review and agree to the [AVS Developer Services Agreement](#), including the [Alexa Voice Services Program Requirements](#), and then select Finish.

☒ I agree to the [Amazon Developer Services Agreement](#), including the [Alexa Voice Service Program Requirements](#).

FINISH

6. You should see pop up confirmation that "Your Product has been created". Select OK.



You are now ready to generate self-signed certificates.

Enable Security Profile:

1. Open a web browser and <https://developer.amazon.com/lwa/sp/overview.html>.
2. Near the top of the page, select the security profile you created earlier from the drop-down menu and click Confirm.
3. Enter a privacy policy URL beginning with http:// or https://. For this example, you can enter a fake URL such as <http://example.com>.
4. Optional You may upload an image. The image will be shown on the LWA consent page to give your users context.
5. Click Save.

Once we created the product and security profile, we need to start the configuration by following the steps below.

1. Update the Raspberry-Pi first
 - a. `sudo apt-get update`

2. Download all the installation and configuration script

- a. `wget https://raw.githubusercontent.com/alexa/avs-device-sdk/master/tools/Install/setup.sh`
- b. `wget https://raw.githubusercontent.com/alexa/avs-device-sdk/master/tools/Install/genConfig.sh`
- c. `wget https://raw.githubusercontent.com/alexa/avs-device-sdk/master/tools/Install/pi.sh`

3. Run the setup.h script

Before installation, we need to move the config.json file from /Downloads to your Alexa directory and then enter below command in the terminal.

```
sudo bash setup.sh config.json -s 998987
```

Note: 998987 is the Device Serial Number(DSN). If you don't supply any DSN number, then it is by default set to 123456.

Select "AGREE" when prompts to accept the license agreement.

Authorize Your Alexa Product:

All the credentials of Alexa's AVS Device SDK have been installed on Raspberry-Pi. But to get it running you need a refresh token to enable your client to maintain a connection to the Alexa voice service in the cloud.

1. To authorize, enter below command in the terminal:

```
sudo bash startsample.sh
```

2. Wait to display a message like this:

```
#####  
# > > > > NOT YET AUTHORIZED < < < < < #  
#####  
#To authorize, browse to: 'https://amazon.com/us/code' and enter the  
code: {XXXX} #
```

3. Use any browser and navigate to that URL and enter the code shown. Select "Allow"
4. Wait (for around 30 seconds) for CBLAuthDelegate to successfully get an access and refresh token from Amazon.
5. Once authorized, you will see a message like this.

```
#####  
# Alexa is currently idle! #  
#####
```

6. You are now ready to use your own Alexa.
7. If the sample app isn't already running, you'll need to initialize the sample app by navigating to your Alexa directory and enter:

```
sudo bash startsample.sh
```

8. When you say ""Alexa..."", you should see a bunch of messages scroll in your terminal window. One of those will show the status changing to Listening, indicating the wake word has been recognized. Then say "Tell me a joke." If Alexa responds with Thinking..., then Speaking, you have a working prototype... and probably, a very bad joke.

Cost Analysis:

The table below indicates all the cost of the materials used in the project.

Component	Cost(\$)	Quantity
Wooden Frame	\$30	1
Glass Mirror	\$150	1
Raspberry PI 3B+	\$35	1
Monitor	\$50	1
Pi-camera	\$33	1
MicroSD card	\$16	1
Sound Card	\$35	1
Speaker	\$25	1
AUX cord	\$12	1
Microphone	\$24	1
HDMI Cable	\$20	1
DC Fan for Raspberry PI	\$7	1
Extension Board	\$16	1

Conclusion and Future Work:

This project contributes to upgrading technology in smart homes. Although the smart mirror offers the minimum functions but could offer more with improvement and further development. Now it displays some features on the mirror but in the future, additional features will be added to make more user-friendly and customizable. Users will be able to control the mirror on his/her terms like using a smartphone application, hand gestures. This project was done to provide an overview of the available solutions and details information to build your own smart mirror. This project can be a great opportunity for future integration and development than other smart home projects.

References:

- <http://www.eecs.ucf.edu/seniordesign/sp2016fa2016/g24/docs/SD1FinalDoc.pdf>
- <https://www.twowaymirrors.com/diy-smart-mirror/>
- <http://proj354.com/itcs/10to19/2018/05/common/05%20Team%20Reflect%20-%20Final%20Report.pdf>
- [https://www.researchgate.net/publication/322812753 DIY Smart Mirror](https://www.researchgate.net/publication/322812753_DIY_Smart_Mirror)
- <https://www.theverge.com/circuitbreaker/2017/8/17/16158104/smart-mirror-diy-raspberry-pi-commute-weather-time-gadget>
- <https://www.postscapes.com/diy-smart-mirrors/>
- <https://www.howtogeek.com/414647/how-to-build-a-smart-mirror/>
- <https://github.com/kevinatown/MMM-Screencast>
- <https://github.com/alexa/avs-device-sdk/wiki/Raspberry-Pi-Quick-Start-Guide-with-Script>
- <https://github.com/alexa/avs-device-sdk/wiki/Create-Security-Profile#enable-your-security-profile>