Progetto di Internet of Things

Gabriele Dominici:
mail di ateneo: dominici.gabriele@spes.uniud.it
numero di matricola: 143859

Aprile 2021

Indice

Capitolo 1

Introduzione

Il progetto d'esame in questione verte sulla realizzazione di un Magic Mirror, supportato su una RaspberryPi 3B+, con l'implementazione dell'assistente vocale Alexa, di un algoritmo di face recognition e controllabile tramite gesture, grazie ad un sensore di gesture recognition. Questo specchio viene definito "smart", non solamente grazie alle funzionalità sopracitate, ma per la presenza di alcune caratteristiche di base, come la rappresentazione di alcune informazioni utili all'utente come l'ora, il meteo, le news, il calendario e molte altre. Questa configurazione è del tutto personalizzabile, permettendo all'utente di impostare il Magic Mirror come più ritiene utile.

Capitolo 2

Fasi della realizzazione del Progetto

2.1 Ideazione

Il mio obiettivo era creare un magic mirror con più funzionalità utili possibili, in modo da renderlo un oggetto utile all'interno della casa a 360°. Mi sono domandato quali caratteristiche avrebbero reso questo specchio ancora più smart e, dopo aver trovato le risposte, ho strutturato il tutto in due fasi:

- Fase hardware: scelta, acquisto e assemblaggio dei componenti
- Fase software: installazione e configurazione delle librerie, packages e moduli necessari e programmazione dei moduli aggiuntivi

Ho deciso di sviluppare questo progetto secondo un modello prototype andando ad intervallare le due fasi, precedentemente citate, implementando una soluzione base, che con il passare delle iterazioni viene arricchita e perfezionata.

2.2 Fase Hardware

In questa fase ho dovuto, per prima cosa, fare un analisi dei costi relativi alla realizzazione del Magic Mirror, stilando una lista degli oggetti necessari, per poi documentarmi online sul miglior rapporto tra prezzo e rapidità nella consegna. Nella maggior parte dei casi ho acquistato i componenti da Amazon, grazie all'ottimo servizio di consegna e a prezzi comunque ragionevoli. Se avessi voluto ridurre ulteriormente i costi a fronte, però, di un tempo di attesa maggiore avrei potuto acquistare gli stessi componenti su siti come Aliexpress. Alcuni oggetti necessari alla realizzazione finale, come la cornice e il display, li ho recuperati, da articoli, di mio possesso, ormai in disuso e adattati alla situazione. In un secondo momento ho assemblato i componenti, step by step, in relazione allo sviluppo software.

2.3 Fase Software

Questa parte del progetto consiste nell'installazione, programmazione e configurazione di tutte le parti software utili al corretto funzionamento del Magic Mirror. Partendo dall'installazione e configurazione dell'OS necessario al funzionamento della RaspberryPi, proseguendo con i moduli relativi alla versione base di Magic Mirror, presente su GitHub, concludendo con l'implementazione delle ulteriori caratteristiche di cui ho voluto dotare il dispositivo, come la face recognition, l'assistente vocale Alexa, e l'inserimento di un bot Telegram. Oltre a quanto descritto, ho inserito ulteriori moduli per rendere il dispositivo più funzionale e utile negli ambiti elencati nei paragrafi precedenti. Un esempio di spiccata importanza è l'integrazione del modulo relativo al riconoscimento delle gesture e al conseguente svolgimento di determinate funzioni dello specchio.

Capitolo 3

Hardware

3.1 Componenti

Il primo step da me svolto è stato stilare l'elenco di ciò che mi sarebbe servito per la realizzazione del progetto d'esame, che con il passare delle iterazioni *fase hardware - fase software*, si è arricchito di alcuni componenti che inizialmente non avevo considerato. La lista conclusiva del necessario è la seguente:

- RaspberryPi 3B+
- MicroSD 32GB
- Cavo USB-Micro-B
- Alimentatore USB
- Display
- Alimentatore Display
- Cavo HDMI (se servisse anche adattatore HDMI - VGA)
- Vetro delle dimensioni del display
- Pellicola a specchio
- Cornice in grado di contenere il display
- RaspberryPi Camera Module
- Gesture Sensor 3D
- Mini Microfono USB
- Speaker Jack 3,5mm o Bluetooth

A questa lista bisogna chiaramente aggiugnere tutto il materiale necessario al montaggio come viti, staffe d'acciaio e fascette.

3.2 Analisi dei costi

In seguito ho analizzato i costi di cui mi sarei dovuto sobbarcare per la realizzazione di questo Magic Mirror. Fortunatamente alcuni componenti sono riuscito a procurarmeli a costo 0, possedendoli già o riutilizzando oggetti ormai in disuso, per poi adattarli a ciò di cui avevo bisogno. Per questo motivo non ho dovuto comprare la RaspberryPi 3B+ e tutto ciò ad essa legato, come cavo USB, alimentatore e MicroSD. Lo stesso vale per lo spekaer Bluetooth e il display, ottenuto

smontando un vecchio monitor, il quale era ormai in disuso da diversi anni, il suo alimentatore e il cavo HDMI, dotato di adattatore VGA. Anche la cornice l'ho realizzata modificandone una molto antica, l'ho intagliata, riassemblata e verniciata in modo tale che si adattasse perfettamente al mio display. I restanti componenti da acquistare erano i seguenti:

COMPONENTE	PREZZO (€)
RaspberryPi Camera Module	10.99
Gesture Sensor 3D	20.99
Mini Microfono USB	8.09
Pellicola a specchio	0.00
Vetro delle dimensioni del display	5.00
Totale	45.07

3.3 Assemblaggio

Avendo voluto relizzare finiscamente il Magic Mirror, ho dovuto procedere con l'assemblaggio dei componenti nel seguente modo:

1. Montaggio della cornice
2. Applicazione della pellicola a specchio sul vetro
3. Inserimento del vetro nella cornice
4. Inserimento del display nella cornice
5. Fissaggio del display alla cornice tramite staffe
6. Inserimento RaspberryPi Camera Module nella cornice
7. Organizzazione dei cavi tramite fascette
8. Inserimento RaspberryPi 3B+
9. Fissaggio Gesture Sensor

Il risultato finale è quello rappresentato in foto, in Figura 3.1 si può notare l'aspetto anteriore del costrutto, mentre in Figura 3.2 la parte posteriore, sulla quale ho applicato una copertura di plastica, in modo da proteggere l'utilizzatore da alcuni spiacevoli pericoli, riguardanti alcuni fili scoperti.

Figura 3.1: Lato Anteriore

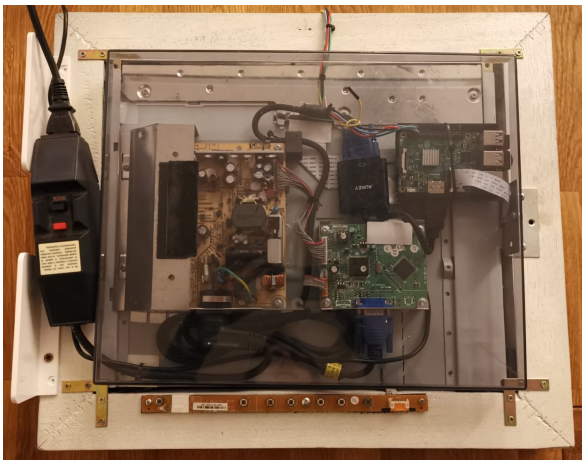

Figura 3.2: Lato Posteriore

Capitolo 4

Software

Dopo aver installato Raspbian, come OS, su RaspberryPi 3B+, sono passato alla configurazione di MagicMirror^2, una piattaforma open source modulare, che grazie all'utilizzo di Electron come wrapper, permette di visualizzare a tutto schermo la pagina, personalizzabile con i vari moduli, che verrà mostrata sullo schermo. Questa è un operazione chiave perchè MagicMirror^2 è la base su cui sono andato a implementare le restanti caratteristiche.

4.1 Installazione e configurazione MagicMirror^2

Ci sono diversi modi per installarlo, io ho utilizzato il seguente, che è quello suggerito dalla guida ufficiale:

1. Scaricamento e installazione dell'ultima versione di Node.js

```
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
sudo apt install -y nodejs
```

2. Clonazione della repository GitHub riguardante MagicMirror^2

```
git clone https://github.com/MichMich/MagicMirror
```

3. Spostamento all'interno della repository e installazione dell'applicazione

```
cd MagicMirror
npm install
```

4. Creazione del file config da modificare, partendo da un file di esempio

```
cp config/config.js.sample config/config.js
```

Il MagicMirror ora può essere lanciato eseguendo il comando:

```
npm run start
```

4.1.1 Moduli aggiuntivi

La cosa più semplice per modificare l'aspetto con cui MagicMirror appare di default è andare a modificare il foglio di stile dal quale riceve le impostazioni estetiche *(css/main.css)*. Qui io ho modificato i parametri *color* degli oggetti *.dimmed* e *.normal* impostandoli, rispettivamente, a *#EE7F11*, un arancione vivace, e *#0080FF*, un blu quasi puro. Nonostante questo, la parte fondamentale per la personalizzazione del nostro specchio è presente nel file di configurazione *(config/config.js)*. Qui potremo andare a modificare le impostazioni dei moduli installati di default ed inserirne di nuovi. Come detto in precedenza, essendo un progetto OpenSource molte persone hanno sviluppato dei loro moduli che hanno poi reso disponibili alla comunità. I moduli che ho installato sono i seguenti:

- MMM-Page-Selector
- MMM-AirQuality
- MMM-CoinMarketCap
- MMM-GroveGestures
- MMM-APOD

Ogni modulo ha le sue peculiarità e i suoi metodi di installazione, perfettamente descritti nelle corrispondenti repository di GitHub. È inoltre necessario ricordare che questi moduli li ho scelti sulla base delle miei bisogni, infatti ne sono presenti molti altri, per qualsiasi necessità.

4.1.2 Configurazione moduli

Ogni modulo utilizzato deve essere inserito all'interno del file config, impostando correttamente, secondo i propri bisogni, i parametri richiesti nelle rispettive guide.

Moduli di default

Clock
Questo modulo permette di visualizzare l'orario sul nostro MagicMirror, permettendoci di visualizzare anche il sorgere e il tramontare del sole e della luna, impostando le latitudini della nostra posizione.

```
{
  module: "clock",
  position: "top_left",
  config: {
    showSunTimes: true,
    showMoonTimes: true,
    lat: 46.0637,
    lon: 13.24458,
    locationID: "5128581",
    appid: "YOUR_OPENWEATHER_API_KEY" //ID from http://bulk.openweathermap.org/
    sample/city.list.json.gz
  }
}
```

Calendar

Il modulo Calendar mostra uno o più calendari in formato .ical precedentemente scelti.

```
{
  module: "calendar",
  header: "Calendario",
  position: "top_left",
  config: {
    calendars: [
      {
        symbol: "calendar-check",
        url: "https://calendar.google.com/calendar/ical/it.italian%23holiday%40
    group.v.calendar.google.com/public/basic.ics"  },
      {
        symbol: "calendar-alt",
        url: "YOUR_.ical_CALENDAR"
      }
    ]
  }
}
```

Compliments

Questo modulo visualizza iterativamente alcune frasi personalizzabili. Esse possono variare, per esempio, in base al giorno e all'ora.

```
{
  module: "compliments",
  position: "lower_third",
  config: {
    compliments: {
      anytime: [
        "Hey amico!"
      ],
      morning: [
        "Buongiorno, sar  una super giornata!",
        "Divertiti oggi!",
        "Dormito bene?"
      ],
      afternoon: [
        "Hai ancora molto da fare oggi!",
        "Stai alla grande!",
        "Ti vedo in forma oggi!"
      ],
      evening: [
        "Sono sicuro che    stata una fantastica giornata!",
        "Hai dato il massimo!",
        "Ti aspetta un meritato riposo"
      ],
      "....-01-01": [
        "Happy new year!"
      ],
      "....-05-25": [
        "Buon compleanno!"
      ]
    }
  }
}
```

Current Weather e Weather Forecast
I due moduli in questione, impostando una città, permettono di visualizzare l'attuale meteo, oltre a quello dei giorni seguenti.

```
{
  module: "currentweather",
  position: "top_right",
  config: {
    location: "Udine",
    locationID: "3165072", //ID from http://bulk.openweathermap.org/sample/city
    .list.json.gz;
    appid: "YOUR_OPENWEATHER_API_KEY"
  }
},
{
  module: "weatherforecast",
  position: "top_right",
  header: "Weather Forecast",
  config: {
    location: "Udine",
    locationID: "3165072", //ID from http://bulk.openweathermap.org/sample/city
    .list.json.gz;
    appid: "YOUR_OPENWEATHER_API_KEY"
  }
}
```

Newsfeed
Questo modulo mostra sullo specchio le principali notizie di un feed RSS scelto in precedenza.

```
{
  module: "newsfeed",
  position: "bottom_bar",
  config: {
    feeds: [
      {
        title: "New York Times",
        url: "https://rss.nytimes.com/services/xml/rss/nyt/HomePage.xml"
      }
    ],
    showSourceTitle: true,
    showPublishDate: true,
    broadcastNewsFeeds: true,
    broadcastNewsUpdates: true
  }
}
```

Moduli da me scelti

I seguenti moduli, come detto in precedenza, sono stati da me scelti, sulla base dei miei gusti personali e delle mie esigenze. Il progetto MagicMirror[2] è estremamente modellabile e personalizzabile grazie ad una vastità di moduli differenti disponibili, di conseguenza sarebbe stato possibile includerne diversi altri, esaudendo le richieste più disparate.

MMM-Page-Selector

Questo modulo permette di creare, sullo stesso specchio, diverse pagine, rendendo possibile l'inserimento di un numero maggiore di moduli, oltre alla divisione di essi in relazione al loro contesto. Inoltre, come è possibile vedere nel codice sottostante, è possibile far esguire delle azioni al modulo, come il cambio di pagina, attraverso l'emissione di alcune notifiche. Questo svolge un ruolo fondamentale nell'implementazione delle gesture.

```
{
  module: "MMM-Page-Selector",
  position: "top_center",
  config: {
      defaultPage: "main",
      displayTitle: true,
      selectPageNotif: ["SELECT_PAGE"],
      incrementPageNotif: ["PAGE_UP"],
      decrementPageNotif: ["PAGE_DOWN"],
    persistentPages: true,
    },
    autoChange: {
    interval: 100
  }
}
```

Al termine del file di configurazione però bisognerà inserire il posizionamento dei vari moduli nelle pagine create.

```
pages: {
  "main": {
  "calendar": "top_left",
  "compliments": "lower_third",
                "weatherforecast": "top_right",
                "newsfeed": "bottom_bar"

  },
  "second": {
    "MMM-AirQuality": "lower_third",
  "MMM-CoinMarketCap": "top_left",
  "MMM-APOD": "top_right"
  }
},
exclusions: {
    "updatenotification": "top_bar",
  "clock": "top_left",
  "currentweather": "top_right",
  "MMM-GroveGestures": "bottom_right"
}
```

MMM-AirQuality

Il modulo MMM-AirQuality permette di verificare in tempo reale la qualità dell'aria di una data città, precedentemente impostata.

```
{
  module: 'MMM-AirQuality',
  position: 'top_center', // you may choose any location
  config: {
      location: 'italy/friuli-venezia-giulia/udine/v.cairoli/' // the location
    to check the index for
```

```
  }
}
```

MMM-CoinMarketCap

Il modulo, come suggerisce il suo nome, fornisce, tramite le API di CoinMarketCap, le quotazioni in tempo reale delle criptovalute selezionate.

```
{
     module: 'MMM-CoinMarketCap',
  position: "top_left",
  header: "Cryptocurrencies",
  config: {
    apiKey: 'YOUR_COINMARKETCAP_KEY',
    currencies: ['bitcoin', 'ethereum', 'polkadot'],
    view: 'graphWithChanges',
      conversion: 'EUR',
     }
}
```

MMM-GrooveGesture

Questo è l'altro modulo chiave per l'utilizzo di gesture. Permette di interfacciarci con il Gesture Sensor e, sulla base di ciò che rileva, invia delle notifiche agli altri moduli, permettendoci, come in questo caso, di cambiare pagina, cambiare notizia, far scomparire e ricomparire tutti i moduli.

```
{
    module: "MMM-GroveGestures",
    position: "bottom_right",
    config: {
    autoStart: true, //When Mirror starts, recognition will start.
    verbose:false, // If set as `true`, useful messages will be logged.
    recognitionTimeout: 1000, //Gesture sequence will be ended after this time
    from last recognized gesture.
    cancelGesture: "WAVE", //If set, You can cancel gesture sequence with this
    gesture.
    visible: true, //Recognized gesture sequence will be displayed on position

    idleTimer: 0, // `0` for disable, After this time from last gesture, onIdle
     will be executed.
      onIdle: { // See command section
        moduleExec: {
      module: [],
      exec: (module, gestures) => {
          module.hide(1000, null, {lockstring:"GESTURE"})
        }
        }
    },
    onDetected: {
        notificationExec: {
        notification: "GESTURE_DETECTED",
        },
    },

    gestureMapFromTo: { //When your sensor is installed with rotated direction,
     you can calibrate with this.
        "Up": "UP",
        "Down": "DOWN",
        "Left": "LEFT",
        "Right": "RIGHT",
        "Forward": "FORWARD",
        "Backward": "BACKWARD",
        "Clockwise": "CLOCKWISE",
        "anti-clockwise": "ANTICLOCKWISE",
        "wave": "WAVE"
    },

    defaultNotification: "GESTURE",
    pythonPath: "/usr/bin/python", // your python path
```

```
    defaultCommandSet: "default",
    commandSet: {
        "default": {
        "FORWARD-BACKWARD": {
            notificationExec: {
              notification: "PAGE_UP",
              payload: null
            }
        },
        "LEFT-RIGHT": {
            notificationExec: {
              notification: "PAGE_DOWN",
              payload:null,
            }
        },
        "CLOCKWISE": {
            moduleExec: {
              module: [],
              exec: (module, gestures) => {
                  module.hide(1000, null, {lockstring:"GESTURE"})
              }
            }
        },
        "ANTICLOCKWISE": {
            moduleExec: {
              module: [],
              exec: (module, gestures) => {
                  module.show(1000, null, {lockstring:"GESTURE"})
              }
            }
        },
        "LEFT": {
            notificationExec: {
              notification: "ARTICLE_PREVIOUS",
              payload: null,
            }
        },
        "RIGHT": {
            notificationExec: {
              notification: "ARTICLE_NEXT",
              payload: null,
            }
        },
        },
    },
    }
}
```

MMM-MMM-APOD

È un modulo che permette di visualizzare alcune fantastiche immagini scattate dalla NASA. Un ottimo modo per rimanere aggiornati sul viaggio di Perseverance su Marte.

```
{
  module: "MMM-APOD",
  position: "top_left",
  config: {
    appid: "YOUR_NASA_API_KEY", // NASA API key (api.nasa.gov),
    maxMediaWidth: 300,
    maxMediaHeight: 800
  }
}
```

4.2 Implementazione funzioni aggiuntive

Ho voluto inserire alcune caratteristiche aggiuntive a questo specchio, in modo da renderlo il più smart possibile. Tutta l'implementazione da me realizzata è stata fatta in Python. Le librerie, con le rispettive versioni, da me utilizzate sono contenute all'interno del file **requirements**.

4.2.1 Face Recognition

Ho implementato questa funzionalità in modo tale che il display si accendesse, solamente, quando una persona autorizzata venisse inquadrata dal Camera Module, collegato alla RaspberryPi 3B+. In questo modo avrei ridotto gli sprechi, causati da un display sempre acceso, oltre a dare accesso solamente alle persone precedentemente inserite nell'apposito dataset. Date le ridotte capacità di calcolo del dispositivo, l'algoritmo di face recognition implementato non utilizza le tecniche di Deep Learning più avanzate, ma utilizzando dei semplici algoritmi di Machine Learning riesce a performare discretamente.

Per la realizzazione di questa proprietà ho realizzato due file:

- *encode_faces.py* che mi permette di codificare le facce, presenti all'interno della cartella dataset, in vettori di 128 elementi, salvandoli nel file *encodings.pickle*. Questo grazie all'utilizzo di un leggero algoritmo pretrainato di Machine Learning, chiamato Haar Cascades, disponibile attrverso la libreria OpenCV.
- *main.py* che rappresenta il file principale, in cui confronto ciò che il Camera Module mi comunica con gli encodings precedentemente elaborati. Qualora il programma riconoscesse un volto noto, il programma accenderebbe il display per 100 secondi, svolgendo altre funzioni descritte in seguito.

I passaggi svolti per rendere il tutto funzionante sono i seguenti:

1. Codifica delle facce presenti nella cartella dataset

```
python encode_faces.py --dataset dataset --encodings encodings.pickle
 --detection-method hog
```

2. Esecuzione del programma principale

```
python main.py --cascade haarcascade_frontalface_default.xml --
encodings encodings.pickle
```

4.2.2 Assistente vocale Alexa

Un'ulteriore caratteristica, a parer mio, rilevante, in modo rendere quest'oggetto ancora più utile, in un contesto abitativo moderno, è la presenza di un assistente vocale, come Alexa o Google Assistant. Ho preferito quello di casa Amazon, grazie a una più facile personalizzazione. Ho installato l'SDK di Alexa, seguendo i passaggi descritti all'interno della documentazione relativa al suo utilizzo su RaspberryPi. Per il corretto funzionamento del dispositivo bisogna dotare la RasberryPi 3B+ di un microfono USB e uno speaker connesso tramite jack da 3,5mm o Bluetooth, come nel mio caso.
L'assistente vocale verrà poi lanciato all'interno del file *main.py*, tramite lo script bash *mm2.sh*, precedentemente creato e già all'interno della repository di GitHub, nel quale è contenuto la riga di comando per la corretta esecuzione di Alexa.
Il nostro assistente vocale sarà personalizzabile e configurabile, o attraverso la CLI (Command Line Interface), eseguendo il comando contenuto nello script bash sopracitato, oppure tramite l'app Alexa scaricabile dai principali store Android ed Apple.

4.2.3 Bot Telegram

Per controllare l'esecuzione dello specchio da remoto, e aggiungere qualche altra caratteristica, ho inserito all'interno del file *main.py* un bot Telegram. Questo mi permette di iniziare e concludere l'esecuzione dello specchio comunicando tramite Telegram al bot le rispettive istruzioni *start* e *close*. All'interno del file *main.py* è possibile settare un elenco di codici chat che possono comunciare con lo specchio. Un'ulteriore funzione che ho implementato è l'invio delle foto scattate dal Camera Module, nelle quali è stata riconosciuta una faccia, sia tra quelle note, che quelle riconosciute come *Unknown*.

4.3 File main.py

Questo file rappresenta, come si evince dal suo nome, il cuore per l'esecuzione del nostro specchio "smart". Una volta eseguito, come descritto precedentemente, viene creato un bot Telegram, il quale si mette in ascolto dei comandi *start* e *close*. Se questi comandi vengono digitati in una delle chat incluse nella whitelist, svolgono determinate funzioni. Il comando start esegue in sequenza lo script bash *mm2.sh*, che attiva l'assistente vocale Alexa, e *mm.sh* in cui viene eseguito MagicMirror^2. Al contempo viene spento il display, il quale verrà riabilitato solamente quando il Camera Module riconoscerà un volto noto e, come detto in precedenza, il bot telegram invierà tale foto ai contatti nella whitelist. Dopo 100 secondi, se non viene rilevata nessun volto noto, verrà spento nuovamente il display. Invece, quando il bot riceverà il comando close, grazie all'esecuzione asincrona della funzione di callback del comando start, terminerà tutti i programmi lanciati dai sopracitati script bash, lasciando il bot in ascolto.

Capitolo 5

Conclusioni

L'implementazione di queste ultime caratteristiche sarebbe stata possibile tramite alcuni packages realizzati da terze parti per il progetto MagicMirror^2, come alcuni di quelli che ho utilizzato anche io, per altre funzioni. Ho preferito realizzarli da zero per avere un maggiore controllo sulle funzioni che questi packages ottemperavano, riuscendo così ad avere una maggiore personalizzazione, secondo quali sono i miei interessi e scopi. Così facendo sono riuscito a far interagire i vari componenti al meglio, rendendo semplici anche futuri aggiornamenti e implementazioni di nuove features.
È stato molto piacevole realizzare interamente questo progetto, sia dal lato pratico, che dal punto di vista software, permettendomi di costruire un prodotto funzionante e utile all'interno del mio ambiente casalingo. Userò questo prodotto nel quotidiano, implementando, con il passare del tempo, nuove funzionalità.

La demo è possibile visualizzarla a questo link.