QuickSight

Bite sized information in the corner of your eye

Innehållsförteckning

Requirements

Raspberry pi 3b

- Software installed: Git, Unclutter, Pip3
- Python libraries installed: feedparser, PIL, contextmanager, xmltodict
- google-api python client: oauth2client

LCD flatscreen monitor

2 way mirror

Geetech voice recognition module p803

Design

To get optiomal readability on the mirror the contrast between background and text needs to be high. A black background with white text is the easiest to see through the reflective surface of the mirror. Other colours for the text could be used but in our case we found that black on white provided the best visibility.

The mirror can be scaled up or down to suit your preference and widgets can be placed in corners or spread out across the entire mirror surface depending on how you want your mirror to look.

Technical

The interface of the mirror is made in python's standard GUI package tkinter. Every widget is packaged in a frame and is then placed in one or more larger frames that fill the screen. The frames that contain the widgets consist of 2 larger frames that divide the screen in a top and bottom half. We have chosen to call these ous "billboard frames". We also use one smaller frame that takes up a small central position to the furthest left of the screen. All widgets are then put inside their own frames and are added to the billboard frames.

Startup script

We utilize a startup script named mmstart.sh to start the QuickSight program on startup. Because tk interface is dependent on a GUI to run we needed to run the script after the raspian GUI has started. Therefore we added the script to the bottom of "./.config/lxsession/LXDE-pi/autostart".

Clock widget

The clock widget uses the time library in Python to read the system time, date and day-of-week on the device it is running on. Therefore, even without internet connection you will be able to see the time on your QuickSight.

Weather widget

The weather data uses an API from the "Dark sky" weather service. The service is free and allows up to 1000 requests/day from a single api key. You do need to register an account with them first. We opted to remove the extended forecast from our code since we felt it cluttered the screen but left in the option to turn it on again if the program is to be run on a larger screen.

News widget

The news widget uses the feedparser library to gather headlines from a rss feed and places them in a list. Google has since the start of 2018 changed the way their rss news feed works and you now need to use news.google and ad the tags of topics you are intereseted in. Once you have specified the tags for the news you want to see you can turn that feed into an rss feed and use the link in the news class.

To get the rss link from the google link: https://news.google.com/?hl=sv&gl=SE&ceid=SE:sv

simply add rss to the url as shown: https://news.google.com/rss/?hl=sv&gl=SE&ceid=SE:sv&ned=sv_se

Calender widget

The calender widget uses google calendar and the google dev pages code example on how to access your calendar. This requires some setup as you need to activate the api connection to the calender you wish to use. As well as create a credentials.json file and a token.json from google and make sure your code has access to said files. The calender is formated as a list of dictionaries with one list entry per object in your calender you choose to get. Googles example code then converts this information into a string and prints it to screen. We changed the code so instead of printing to screen we update the calender frame label for events with the string.

Message widget

The message widget uses the urllib library to read raw text from a text file. We opted to change the tet file encoding to utf-16 to add support for swedish characters in the text message. Currently we

are using the storage cloud service "Dropbox" and read from a direct read-only link to the text file and display the text in the file on screen.

To get the text from the text file we need it to be in raw text. We tried different cloud services for this but dropbox was the easiest one to get to display the file in raw text mode.

Timetable widget

The time table widget uses API from trafiklab. A free to use developer initiative where you create projects and get access to API keys for several different travel services. For access to Östgötatrafikens timetables we used API keys for Resrobot.

To get timetable information for the stop we wanted we needed to find the stop-id of the bus stop we want to depart from. Since we see this used mainly in commuting or other often made trips between the same points we added the stop id of the destination so we get suggestions for all departures thet get us to that point from our selected departure point.

Much like the calender widget the data we gather is delivered in a dictionary from which we then extract the information we want and put in a string to display in the tkinter frame.

Voice-command widget

The voice command widget functions by recognizing pre recorded commands and delivering a hexadecimal string back depending on the order the commands were recorded. It is therefore the order of the commands, not what is said that dictate what code is returned from the module.

The voice commands does not create and remove widgets when called uppon, instead each widget is created when tthe program starts and is continuously updated with current information even ehen not viewed. We use the voice command widget to display the other widgets or to hide them by changing the widgets colours.

The only exception to this is the news widget where we have to construct and destruct the headlines frames whenever you eant to hide or display the news. This is because the "News" class creates a frame for each individual headline instead of keeping them i one frame. We chose to not use this method on the other widgets because it is much slower than keeping the widgets active and hiding/displaying them.

User manual

Setup

Place the mirror on a wall or table with access to an electrical outlet. Plug in the screen and raspberry pi power cable. QuickSight will automatically start once the device has started.

Widgets

Clock – Displays current time, date and day of the week. Default widget that is always on.

Weather - Displays current temperature and weather conditions. Default widget that is always on.

Calender – Displays the next 3 upcoming events from the calender linked to the mirror.

News – Displays the top 3 headlines of the selected newsfeed.

Message – Displays a message of up to 100 characters in the middle of the screen.

Timetable – Displays the next 3 upcoming departures from a selected bus stop.

Voice commands

All voice commands toggle different settings, the same command is used to display and to hide a widget

Quicksight – This command hides all applications, including clock and weather, making your smart mirror look like an ordinary mirror. If called again the mirror will display all the widgets that were visible before calling the command

Kalender – This command displays/hides the calender in the bottom right corner of the mirror

Nyheter – This command displays/hides the news headlines in the bottom left corner of the mirror

Meddelande – This command displays/hides the current message in the middle of the screen. Please not that updating the message will make it automatically appera on the screen even if this command is not called

Tidtabell – This command displays/hides the timetable widget on the left side of the screen.

Links

Smart mirror source repository released under MIT license, this is the base code we built on: https://github.com/HackerShackOfficial/Smart-Mirror/

Weather service API: https://darksky.net/dev

Calender sync: https://developers.google.com/calendar/

Traffic API: https://www.trafiklab.se/

News API: https://news.google.com/?hl=sv&gl=SE&ceid=SE:sv add rss to the url as shown: https://news.google.com/rss/?hl=sv&gl=SE&ceid=SE:sv&ned=sv_se