# Phewstoc: Integrating Pervasive Solutions for Smart Home Entertainment Systems

Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology, LTU
Luleå, Sweden

Philip Hjortsberg[1], Hampus Holmström[2], Edvin Sladic[3], William Wennerström[4]

[1] phihjo-2@student.ltu.se
[2] hamhol-5@student.ltu.se
[3] edvsla-5@student.ltu.se
[4] wenwil-5@student.ltu.se

***Abstract*—In the smart homes living room, unobtrusive and smart media playback control is an area where no major product has made its breakthrough. There are some key difficulties to overcome before making a truly unobtrusive media playback control without actively involving the user. Phewstoc is an attempt to make an overall solution for the smart living room. Two main difficulties are investigated. Face recognition and authentication and sleep detection, with sleep detection discovered to being the main issue. Phewstoc is a feasible solution for controlling media playback in the living room, but more research is needed on sleep analysis in real time to make a good enough product for end users.**

***Index Terms*—pervasive, face recognition, face detection, sleep detection, kodi, smart tv, smart home.**

## I. Introduction

This project introduces a pervasive solution for smart TV technology. The main purposes regarding this project includes face detection and recognition to authenticate users for access and keeping track if a user is asleep with sleep detection techniques as a power saving functionality for the TV. These are all part of the implemented and integrated pervasive system of Phewstoc.

The objective of this paper is to describe how different libraries and tools were chosen and describe the different limitations, problems and possibilities for reaching a final solution of Phewstoc. The system provides a user friendly, unobtrusive method of controlling the TV with face recognition and sleep detection. Opportunities of saving power become a legitimate reason to provide the system in smart homes.

The first section of the paper describes different approaches and limitations regarding face detection and recognition, OpenFace and Dlib are two libraries that contain functions for face recognition. The second part is sleep detection. This section approaches how different methods of obtaining real time data to analyse if the user is asleep can be used as smoothly and pervasively as possible. The next section describes why Kodi was chosen to be the representative of the media centre in Phewstoc and also interconnections regarding the TV, such as Consumer Electronics Control. The fourth section essentially describes how the system of Phewstoc is designed. Conclusion of the project is discussed in section VI.

## II. Face detection and recognition

### A. Libraries considered

Taking into account the time constraints during the development of Phewstoc it was early decided that the face recognition part of the system should be relying on a open-source available library. Mainly two different libraries where considered.

*1) OpenFace:* OpenFace is a free and open source face recognition library with deep neural networks. It is a very competent library with good documentation and it also has many users. The OpenFace library though has a known issue with architectures other than 64-bit x86[3], which made the implementation for Phewstoc difficult since the first prototype was intended to run on a Raspberry Pi.

*2) Dlib:* Dlib is a C++ library containing many different functions in the area of algorithms and machine learning. The face recognition functions of Dlib are straight forward and easy to implement for Phewstoc's needs. Considering the ease of implementation and the stability of the software this was the chosen library. This solution would not have any constraints as long as the implementation of it works good enough.

### B. Constraints and problems

For any kind off image processing as face detection and recognition the quality of the input stream is a major factor of the performance. A web camera was chosen as the input source for the face detection and recognition program. The reason was the high availability of them at a reasonable price, however the quality of the web camera was found out to be just enough. The distance was compromised, but going for a more powerful camera with higher resolution would mean the

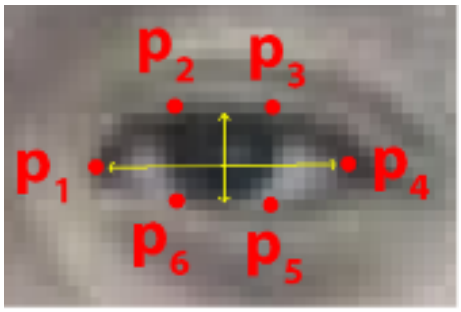


Fig. 1. Six eye landmarks. [1]

system would require heavier processing of the input and the solution being less responsive. The web camera made it by small margins, but compromising on distance for speed was reasonable.

Only using a web camera as input does not allow for identifying faces in dark environments. This could be improved upon using additional input such as thermal IR sensors[6] or diodes[7]. Furthermore, to enhance the authentication, depth perception could be added. Both of these improvements was skipped due to the time frame of the project and never researched in depth.

Two common hardware platforms for running a media centre is Raspberry Pi and Intel NUC which was both considered. To decide on a platform, the team ran the chosen face recognition software on both platforms and compared performance. Both platforms sport CPU's with four threads but the computing time was significantly lower (less than 1s vs. about 30s) for the Intel based NUC and therefore chosen as platform.

## III. Sleep detection

To be able to shut of the TV in the case of a sleeping viewer, an accurate, unobtrusive and fast method was needed to determine the consciousness of the viewer.

### A. Computer vision with facial landmarks

A common method to detect blinks of an eye and drowsiness is based on computer vision with facial landmarks. This maps a fixed amount of points to landmarks on a face and can be used to tell emotion, facial expressions and drowsiness amongst some. The intended approach consisted of using a library that used the eye landmarks and the EAR (Eye Aspect Ratio)[1] formula Eq.1

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2 \times ||p_1 - p_4||} \quad (1)$$

to determine whether the viewer had closed it's eyes for a set amount of time and thereby sleeping and turn of the TV. Working libraries was found and tested, problems arose with the intended viewer distance being to long, resulting in the images having to low resolution and the EAR formula being unable to operate alternatively giving unreliable results. The face detection and recognition software did not recognise faces when having an angle approximately more than 45 degrees of the vertical line. Due to previously mentioned constraints, face recognition for sleep detection was not chosen as the method to use. Therefore it was not justified solving the tilted faces issue.

### B. Somnofy

This is a hardware sensor that has the capability to measure pulse and respiration unobtrusively for up to two persons. It's wall mounted or placed on a surface near the place to be monitored, the group had allowance to use one of the early access samples located at the university smart home lab, which at the time is under lock-down and therefore unavailable.

### C. Empatica activity trackers

Empatica activity trackers were at the groups disposal but not useful due to the demand for near real time sensing and reporting. The provided Empatica devices required to be physically connected to a host computer to download the data and hereby deemed as a non viable solution.

### D. Fitbit consumer activity tracker

Fitbit Charge 2 is an activity tracker with pulse and motion sensing capabilities. The data is uploaded to Fitbit services via a host device that synchronises the tracker over a BLE (Bluetooth Low Energy) connection. The data retrievable by the API is not raw data, proprietary data analysis is run by the Fitbit service. This is convenient for this application since it is not a medical application or similar where data needs to be verifiable.

Fitbit API offers a possibility to fetch sleep data, however this feature is only fetch able once the user has woken up. With regards to previous deficit the sleep data is not a viable solution and the group decided to use pulse data as a replacement since this could be used to fetch more recent data in real time. If the host device, in this scenario an Android device has the Fitbit application in focus the data is sent in real time to the Fitbit servers. Due to unknown factors, the data is not available through the API until approximately 17 minutes later. This was however the most appropriate solution in the given time frame and therefore chosen as sleep detection technology.

## IV. Media centre and TV with interconnections

A media centre was implemented in the project as a part of making the solution a smart TV system.

### A. Kodi

The media centre that was chosen for this particular solution was Kodi. Kodi is an open source home theatre software that can be installed on most of the modern operating systems, Alpine Linux was used in this project. The API of Kodi includes useful functionality that was implemented in a client of the project to control and notify users of the smart-TV. The API of Kodi communicates over the JSON-RPC protocol. The client was used to call for single methods in the remote system, Kodi [5].

The client consists of the following functionality:

- The pause function that keeps track if a video is playing, then pausing the video if a recognised face is not detected in a predetermined amount of time.
- The notification function that notifies the user when a recognised face is detected for the first time or if a face is not detected while Kodi is running.

*Constraints:* One approach initially was to sign in the user profile when the specific user was authenticated. Whenever the method of changing a profile was called from JSON-RPC, the connection was lost for a small amount of time. However, this became a problem as the loss of connection causes the client to crash. The functionality of changing profiles was not a high priority and therefore not necessary to include in Phewstoc.

Another problem of the API connection is the fact that Kodi needs to be signed in to a profile in order communicate with JSON-RPC. The Kodi instance was therefore set up to automatically sign in to a default user.

### B. TV

To turn on and off the TV it was required that the TV supported the CEC, Consumer Electronics Control protocol. This protocol allows for several commands to be issued to the TV, such as changing source and power state. Also the computer that runs the media centre had to support sending CEC signals. In the beginning of the project a Raspberry Pi 3 was used as it provides CEC natively. It was discovered that the Raspberry Pi 3 was too slow to analyse the video stream, so an Intel NUC was used instead. The Intel NUC does not support CEC natively, but has a CEC HDMI header that allows for connecting an ad-hoc adapter. A Pulse-Eight CEC header adapter was used that communicated with the CEC library that Pulse-Eight provides, libCEC.

The CEC implementation LG provides, Simplink, does not support turning off the TV. Turning on is supported, however. It was discovered that the TV had an energy saving function that would turn off the display if the output brightness was turned down. This feature was (ab)used by providing auxiliary command hooks that are run when the TV should turn on or off. In this case, a tool for configuring display modes and properties, `xrandr`, was used to decrease and increase the brightness.

## V. System design

The Phewstoc software suite encompasses two applications, the sleep detection web service and the Phewstoc controller that runs on the same machine as Kodi. The sleep detection web service was written in Go, while the Phewstoc controller was written in Python. Python was used because of the ease of interacting with the `face_recognition` Python module. While Go was used as the authors had prior experience with writing HTTP web servers in the language. The system is illustrated in fig. 2.

### A. Sleep detection service

The sleep detection service is a web server exposed to the internet, serving sleep status requests for the Phewstoc software. This is accomplished by the server querying the Fitbit web API for the latest pulse measurements and if the average pulse is below set threshold, the sleep detection service answers with a HTTP status code representing `asleep`.

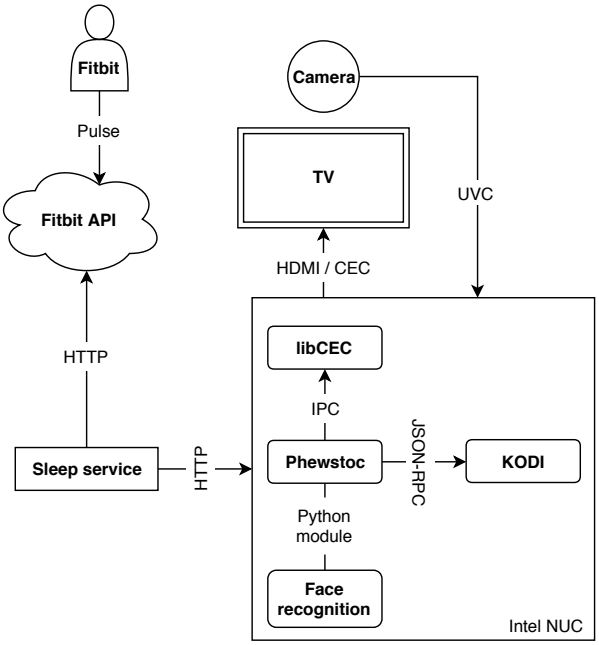


Fig. 2. Overview of the Phewstoc system

The service is built as a Fitbit Personal App which means that only the creator of the app has access to the data stored by Fitbit. Therefore, only one user is supported and the service should be seen as a proof of concept. The sleep status is polled every minute by the Phewstoc software, which executes the necessary actions described below in section V-C.

### B. Face recognition

The Phewstoc software runs on an Intel NUC that receives a video stream from a camera that is placed above the TV. The video stream is retrieved using OpenCV and feeds each frame into the `face_recognition` Python package. First, the `face_recognition` package tries to find faces in the current frame. Multiple faces are handled and for each face a thread is spawned by Phewstoc that tries to recognise the provided face, again with the `face_recognition` package. The thread that returns a recognised face first is used, the rest are discarded.

Before the Phewstoc software can analyse any faces it has to be provided with images of the users. Each image of the user is encoded into landmarks that is recognised by the `dlib` library. To speed up this encoding during start up each face encoder is threaded. The encodings are also cached to the file system with their corresponding MD5 checksums as file names. These face encoding caches are re-used and the encoding step can be skipped, greatly improving the start up speed.

### C. TV control

Each event from either the sleep detection polling service (`Sleep` event) or the face recognition service (`Face` event) is fed into the TV controller. The TV controller uses a scheduler where a `Pause` and `Turn Off` event is scheduled to run after a provided time interval.

Every time a new positive detection event from either `Sleep` or `Face` is received the currently scheduled `Pause` and `Turn Off` events are removed and re-added, i.e. they are postponed by their set time intervals. Also, for every such positive detection event, if the TV is turned off, a CEC signal is sent over HDMI that turns on the TV and a notification is sent to Kodi that greets the detected user.

The `Pause` event's time interval should be set lower than `Turn Off`. As the name may suggest, the `Pause` event pauses the current playback on the Kodi media centre and sends a notification that is hasn't detected anyone sitting in front of the TV, while the `Turn Off` event sends a CEC signal over HDMI that turns off the TV.

Additionally, auxiliary command hooks that are run on either an `Turn Off` or `Turn On` event are provided. These were used to mitigate the problem that the TV used (LG, with Simplink for CEC) didn't provide any way to turn off the TV using CEC as noted in section IV-B.

## VI. Conclusion

There are plenty of work that can be improved from a future perspective. One area of improvement is the sleep detection part. For an optimal solution the data would have to be gathered in real-time and be precise. There exist such devices, but not available as consumer electronics. The authors wished to use devices that were available for regular consumers.

If the room where the system is used is dark, then the web camera will not be able to distinguish well between the users. A simple solution to this problem would be to enrich the hardware with ordinary IR diodes which would enable the camera to work in the dark without disturbing the users.

The media centre Kodi was chosen as it is open source and has a lot of potential when it comes to building functionality with its JSON-RPC library. The problems that arose regarding the API was however not optimal for implementing Kodi as the media platform to the project. It takes a lot of effort to make things work and bugs were encountered frequently. Since there is no other open source media platform that was of any use to the project, Kodi was probably the best way to go.

The power saving functionality is highly relevant to today's society from the environmental standpoint. Even though improvements in Phewstoc is necessary to become more consumer friendly, the idea is still sustainable for further developments.


## Acknowledgement

The authors would like to thank Kåre Synnes, the supervisor of the research and development team. He contributed with suggestions, encouragement and materials throughout the project.


## Links

- Project website: https://hampusholmstrom.github.io/M7012E-phewstoc/
- Project repository: https://github.com/Hampusholmstrom/M7012E-phewstoc

## REFERENCES


[1] Tereza Soukupova and Jan Cechy. (February 3–5, 2016). *Real-Time Eye Blink Detection using Facial Landmarks*. Rimske Toplice, Slovenia, https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

[2] OpenFace. (n.d.). *Face recognition with deep neural networks*. GitHub.com. Fetched 2019-03-18, from: https://cmusatyalab.github.io/openface/

[3] OpenFace. (October 29, 2015). *Architecture issues with Torch.load*. GitHub.com. Fetched 2019-03-18, from: https://github.com/cmusatyalab/openface/issues/42

[4] Dlib (Mars 10, 2019). Fetched 2019-03-18, from: http://dlib.net/

[5] Kodi (25 January 2019). *JSON-RPC API*. Fetched 2019-03-15, from: https://kodi.wiki/view/JSON-RPC_API

[6] M. Krišto, M. Ivašić-Kos. (n.d.). *An Overview of Thermal Face Recognition Methods*. Department of Informatics, University of Rijeka, Rijeka, Croatia. Fetched 2019-03-18, from: https://bib.irb.hr/datoteka/941518.5074-An-Overview-of-Thermal-Face-Recognition-Methods-v2.pdf

[7] S. Li, R. Chu, S. Liao, L. Zhang. (2007). *Illumination Invariant Face Recognition Using Near-Infrared Images.* IEEE transactions on pattern analysis and machine intelligence. 29. 627-39. 10.1109/TPAMI.2007.1014.