Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2014

Google Glass in Medical Applications

Etudiant-e : Wai-kin Hau

Professeur : Dr. Henning Müller

Déposé, le : 28 juillet 2014

1. Résumé

Le thème de ce travail de Bachelor est de réalisé une application pour les Google Glass pour le domaine médical et plus précisément pour les ambulanciers. Le but est de pouvoir filmer les interventions des ambulanciers et de transmettre en temps réel la vidéo à l'hôpital, l'ambulancier peut communiquer directement avec le personnel médical à l'hôpital et le personnel médical peut aussi envoyer les informations lié au patient et d'afficher sur les Google Glass.

Pour atteindre ce but, il faut créer une application Android qui fonctionne avec les Google Glass, mettre en place une architecture pour la communication entre les Google Glass et l'hôpital et de créer une page web afin que le personnel médical puisse accéder.

L'architecture qui va être utilisé pour ce projet de Bachelor est le WebRTC. C'est une interface de programmation JavaScript qui repose sur une architecture triangulaire puis pair à pair.
L'interface pour le personnel médical doit être simple et agréable à l'utiliser.
Le rapport qui suit montre les différents outils utilisés, le déroulement du développement de ce travail de Bachelor et une synthèse du travail accompli.

Table des matières

2. Introduction

2.1. Contexte

Problème

Dans le domaine médical, chaque minute est très important pour apporter de l'aide au patient. Les ambulanciers sont très souvent confrontés à ce problème lors des interventions extérieures. Lorsqu'ils sont sur le lieu d'intervention, il leur arrive de devoir faire une intervention rapide nécessitant des connaissances médicales plus spécifiques, recueillir des informations du patient comme le groupe sanguin et les allergies ou tout simplement de transmettre l'état de la situation de l'accident afin de préparer les salles opératoires à l'hôpital. Tout ceci demande du temps et ce temps est précieux pour sauver des vies.

Solution existante

Pour répondre au besoin de connaissances médicales plus spécifiques, la Suisse a mis en place un service mobile d'urgence et de réanimation (SMUR) qui apporte les soins d'aide médical urgente en dehors de l'hôpital. Ce service est composé d'un médecin urgence et un infirmier. Sur le lieu de l'intervention, le médecin devient le chef de l'équipe médical sur place. Mais, le service SMUR entraine un coût supplémentaires pour les soins préhospitaliers et créer des absences des médecins qualifiés à l'hôpital.

Solution proposée

Des recherches ont montré que la télémédecine peut réduire le coût des soins préhospitaliers tout en améliorant la sécurité des patients. [1][2]

Pour la télémédecine, on peut utiliser le téléphone ou la radio mais pour les ambulanciers, les mains sont très souvent occupées et ce n'est pas pratique d'expliquer la situation uniquement avec les mots. Grâce à la sortie des lunettes intelligentes, on peut les utiliser pour remplacer les téléphones et la radio. Ces lunettes permet de diffuser les vidéo et l'audio de ce que les ambulanciers voient, de faire une vidéo-conférence avec un médecin de

l'hôpital et le personnel médical de l'hôpital pourra envoyer les informations du patient et de les afficher sur les lunettes directement. Ils peuvent être commandés via la voix donc plus besoin d'utiliser les mains [3] [4].

Dans le contexte de télémédecine, ce projet propose d'utiliser les Google Glass possédant un appareil photo avec 5 MP et d'un enregistrement vidéo en 720p. Il propose aussi un prisme du côté droite de l'œil afin d'afficher l'écran, un pavé tactile pour naviguer, un haut-parleur et un microphone. Il fonctionne sous Android. Il existe déjà des recherches portant sur les Google Glass pour des applications médicales. Ces recherches ont démontré que les Google Glass pourraient être utiles dans divers tâches médicales.

En vertu de la loi suisse, les ambulanciers sont autorisés à prendre des images et enregistrer des vidéos des scènes d'accident sans le consentement exprès des patients tant que les images restent dans le domaine médical. Néanmoins, la communication et le stockage doivent être sécurisés. Aucunes données ne seront stockées en permanence sur les Google Glass.

L'avantage de cette solution

Les avantages de l'utilisation des Google Glass pour les soins préhospitaliers sont les suivants:

- La transmission vidéo permet aux médecins de rester à l'hôpital où ils sont les plus utiles et d'apporter leur aide quand c'est nécessaire. De plus, les médecins pourront préparer les salles d'opération en cas d'accident important.
- Les médecins pourront aider plusieurs équipes d'ambulanciers en même temps ou dans un court laps de temps.
- Les paramètres vitaux des patients sont stockés pour aider l'hôpital à diagnostiquer plus précisément.
- Les ambulanciers peuvent accéder à des connaissances médicales et les antécédents du patient.

Solution similaire à la solution proposée

PRISTINE, une entreprise américaine, développe des applications pour la prochaine génération de télémédecine avec les Google Glass. L'entreprise est fondée en mai 2013. Elle est la première compagnie qui a délivré une solution avec les Glass et il se développe rapidement car il déploie ses solutions dans les hôpitaux à travers le pays. Leur produit permet de filmer avec les Glass et de voir la vidéo en temps réel sur un périphérique Android.

LiveStream est une plate-forme de diffusion en direct sur internet. Il offre une application pour les Google Glass sur leur site pour diffuser en direct la vidéo capturé via les Google Glass.

2.2. Objectifs

L'objectifs de ce travail de Bachelor est de développer une application Android fonctionnant sur Google Glass afin d'aider les ambulanciers dans leur intervention.

Comme fonctionnalité pour l'application, il y aura la possibilité de :

- filmer et partager en temps réel ce que l'ambulancier voit
- communiquer avec un médecin de l'hôpital
- pouvoir afficher les informations du patient envoyé par l'hôpital.

A part l'application Android, il faut mettre en place un serveur afin de faire les échanges entre l'hôpital et les Google Glass et une page web afin que l'hôpital puisse consulter et interagir.

Le travail se compose donc en 3 étapes.

La première est la mise en place du serveur pour l'échange de message avec la technologie WebRTC.

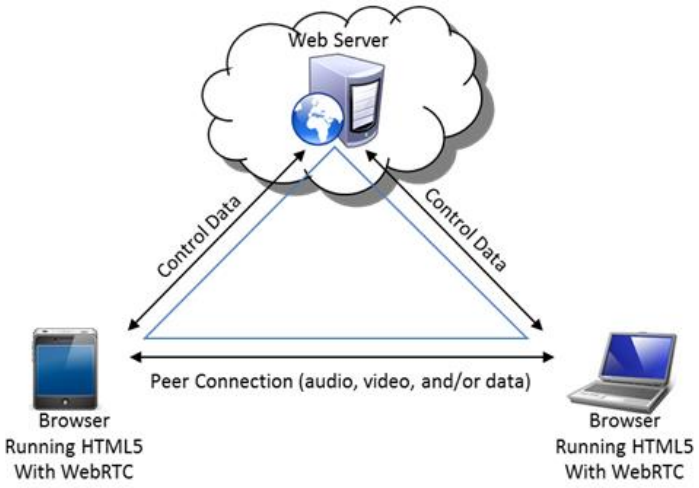

Figure 1 serveur d'échange avec les flux d'échange de donnée

Ensuite, créer une page Web pour le personnel médical à l'hôpital communiquant avec le serveur WebRTC. La page Web doit pouvoir afficher la vidéo des Google Glass, avoir une zone de texte pour envoyer les informations aux Google Glass.

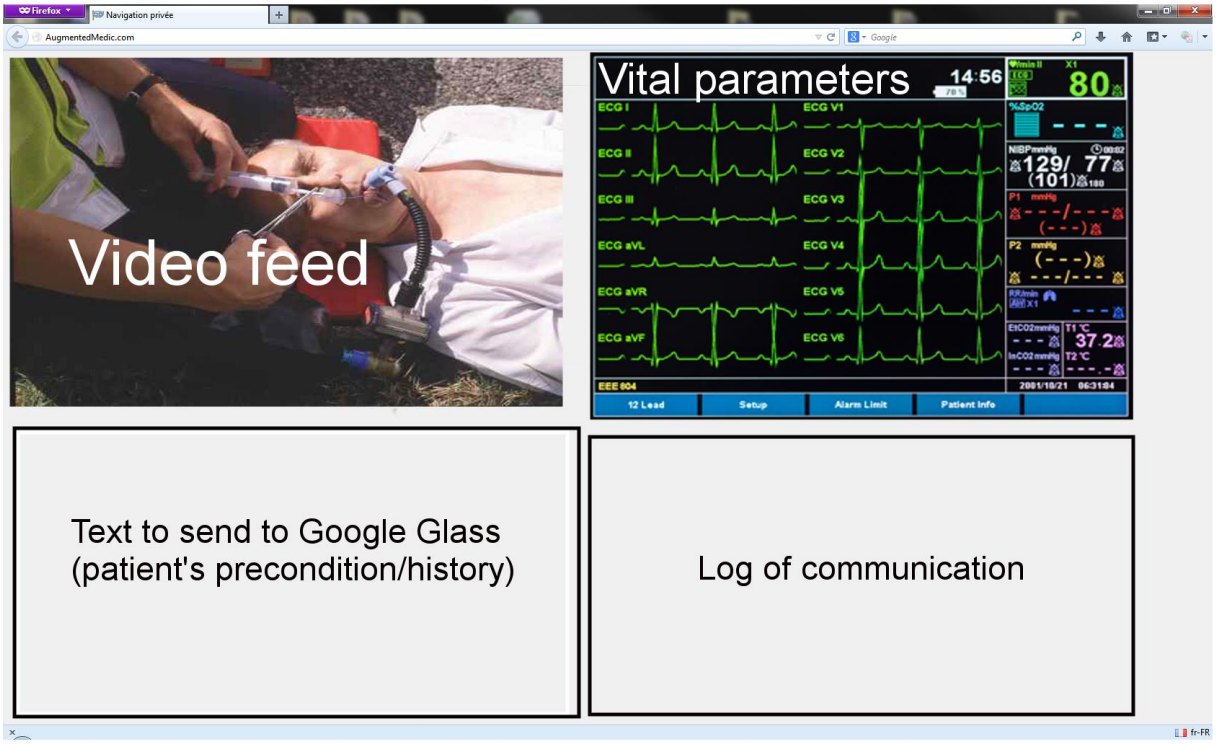

Figure 2 exemple de page Web pour l'hôpital

Et pour finir, le développement de l'application Android. Sur les Google Glass, l'ambulancier doit pouvoir voir ce qu'il est en train de filmer et d'afficher du texte ou d'image envoyé depuis l'hôpital.

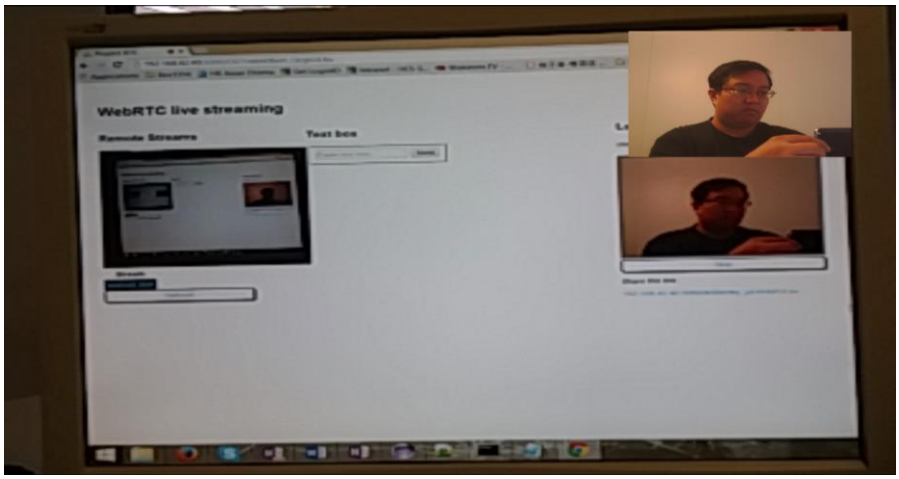

Figure 3 exemple de vue des Google Glass en train de filmer

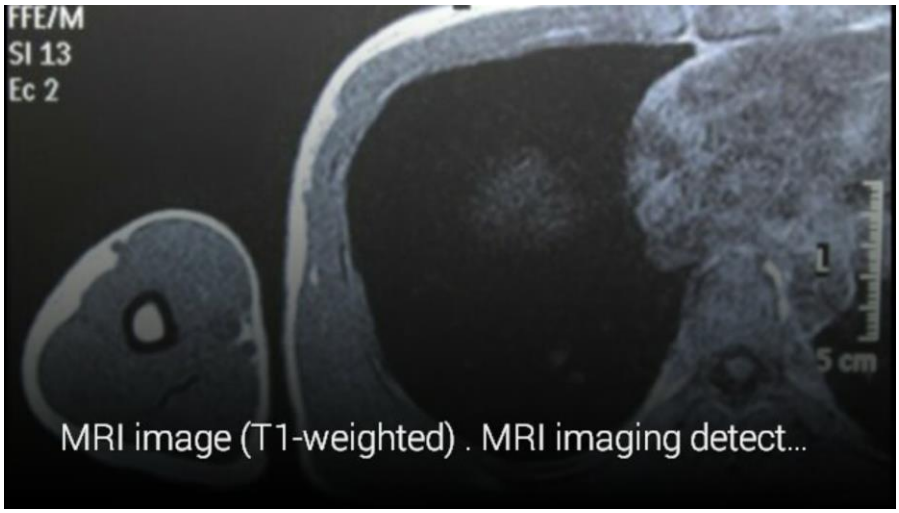

Figure 4 exemple de vue d'une image envoyée depuis l'hôpital

2.3. Motivation personnelle

Ce travail de Bachelor est un projet très intéressant et très utile dans l'avenir. Il se peut que le

projet soit utilisé dans tous les hôpitaux de la Suisse et il facilitera le travail des ambulanciers et des médecins.

Dans le cadre de mes études en tant qu'informaticien de gestion à la HES-SO Valais, j'ai pu apprendre les bases sur le développement Android et grâce à ce projet, je pourrai approfondir cette matière qui est pour moi, une compétence très recherché dans le monde du travail.

De plus, c'est intéressant de pouvoir travailler avec les Google Glass qui sont sorti cette année. Je pense qu'il y aura beaucoup de projet développement pour ces lunettes connectées.

3. Etude de la technologie

3.1. Analyse des lunettes connectés

Le but du travail de Bachelor est de développer un outil sur le Google Glass mais il est intéressant de pouvoir comparer s'il y a d'autres lunettes ou équipements qui pourraient aussi bien convenir à ce genre d'utilisation.

Google Glass

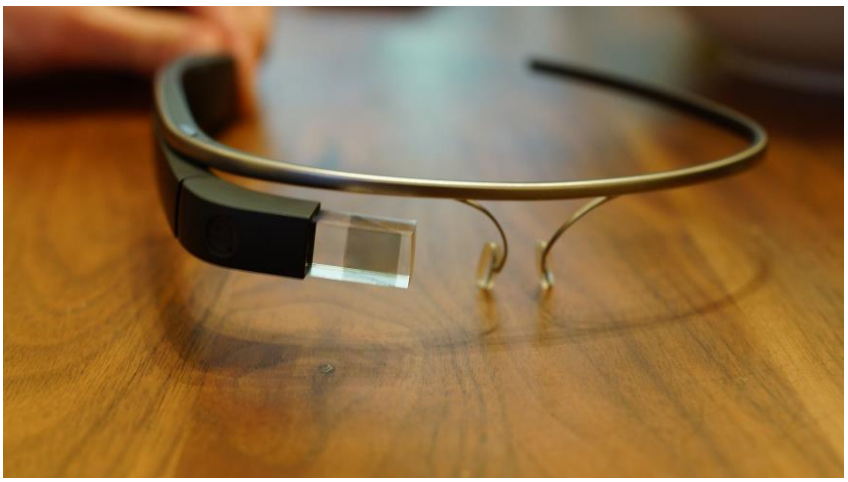

Figure 5 : Google Glass avec le projecteur et le prisme à gauche de l'image

Google Glass est développé par Google. Ces lunettes sont sorties en 2013 avec la version explorer. Elles coûtent environ 1500.- CHF. Elles sont équipées d'une caméra frontale qui peut prendre des photos avec une résolution de 5 MP et des vidéos en 720p HD, d'un micro qui permet de donner les commandes aux lunettes et de pouvoir discuter avec un interlocuteur, d'un pavé tactile pour commander plus rapidement et aisément les lunettes et d'un prisme qui réfléchit la lumière du projecteur. (Pour plus d'information, veuillez se référer au chapitre 4.1)

Vuzix M100

Figure 6 : Vuzix Glasses M100 avec la branche pour l'oreille et l'écran se trouve au bout de la branche principale

Vuzix est une entreprise américaine spécialisé dans les affichages des périphériques et de la réalité augmenté. Leur nouveau produit le Vuzix M100 se porte comme une oreillette ou avec une monture de lunette. Il possède un système de kit main libre et une caméra de 1080p qui permet de communiquer et de filmer.

Recon Jet

Figure 7 : Recon Jet avec la caméra et les boutons de commande à gauche de l'image

Recon Instrument a pour but de fabriquer des lunettes pour les sportifs afin d'afficher divers données directement à l'utilisateur via les lunettes. Il est produit en quantité limité pour l'instant et il offre la possibilité de développer des applications.

GlassUp

Figure 8 : GlassUp avec le projecteur qui se trouve à l'intérieur des lunettes

Une paire de lunette connecté qui peut afficher les différents applications du téléphone Android ou iOS mais il est obligé d'avoir une connexion avec le téléphone.

MetaPro

Figure 9 : MetaPro, la caméra se trouve au centre des lunettes avec un cordon sur un des branches

MetaPro est le premier à avoir intégré une interface holographique. Il est possible de gérer grâce à la reconnaissance des gestes et des mains. Il sortira en septembre 2014. Le produit étant récent, il n'est pas possible de savoir s'il est compatible avec le projet mais néanmoins, il possède une caméra frontal et une carte son.

Atheer Labs

Figure 10 : Atheer One avec 2 caméras à l'intérieur et un capteur de profondeur pour la réalité augmentée

Atheer One est une paire de lunette basé sur la réalité augmentée. Il permet de communiquer entre les lunettes et de transférer des documents. À part un SDK qui laisse aux développeurs de créer des applications avec la réalité augmentée, il peut aussi intégrer les

applications Android. Il est prévu de sortir en été 2015.

3.2. Comparaison de différentes lunettes connectées

Nom	Google Glass	Vuzix m100	Recon Jet	GlassUp	MetaPro	Atheer One
Développeur	Google	Vuzix	Recon Instrument	GlassUp	META	Atheer Labs
Prix ($)	1500	999	599	399	3650	500
Système	Android	Vuzix (compatible avec Android)	?	Android	?	Atheer Labs (compatible avec Android
CPU	1.2GHz	1.2GHz	1.0GHz		i5	
Mémoire	1GB	1GB			4GB	
Stockage	16GB	4GB	8GB		128GB	
Affichage	640x360	4 in	7 in	320x240	1280x720	23 in
Son	oui	oui	oui	oui	oui	oui
Micro	oui	oui	oui	non	non	
Commande vocale	oui	oui		non	non	non
Commande tactile	oui	oui	oui	oui	(commande par geste)	(commande par geste)
Camera	Photo 5MP Vidéo 720p	Photo 5MP Vidéo 1080p	Photo Vidéo 720p		Vidéo	Vidéo
Connexion	Wi-Fi, Bluetooth	Wi-Fi, Bluetooth	Wi-Fi, Bluetooth	Bluetooth	Wi-Fi, Bluetooth	
Poids	50g		60g	65g	180g	70g
Batterie (heure allumé 100%)	0.33	1		0.66	0.33	

3.3. Analyse de l'architecture de communication

WebRTC

Figure 11 logo WebRTC

Description

WebRTC veut dire Web Real-Time Communication. C'est une interface de programmation JavaScript qui est actuellement au stade expérimental. Le but est de pouvoir communiquer entre deux applications sans installer un programme. Il permet le partage des fichiers, la voix sur IP et la vidéoconférence. Il est basé sur une architecture triangulaire puis pair à pair (connexion direct) dans laquelle le serveur permet de mettre en relation deux pairs qui veulent échanger des flux de données ou de médias.

Le WebRTC étant récent, il peut être utilisé que sur les navigateurs comme Firefox, Chrome et Opera.

Architecture

L'API est basée sur architecture triangulaire impliquant deux pairs ou serveur centrale. Le serveur est l'intermédiaire afin de coordonner les échanges entre les pairs jusqu'à ce que la connexion devienne du pair à pair.

Les requêtes de communication se font par http ou par WebSocket. Ensuite, grâce au JavaScript, il permet de maintenir la connexion. Pour les flux de donnée, il utilise des standards existants comme STUN, ICE, TURN, DTLS ou encore SRTP.

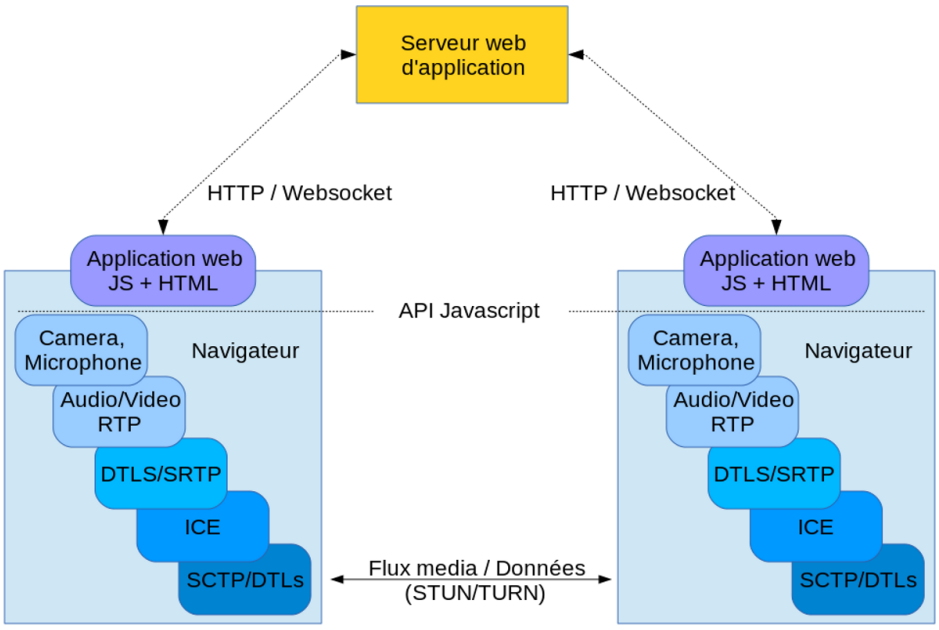

Figure 12 : L'architecture du WebRTC avec les échanges en http/Websocket avec le serveur et l'échange entre les pairs via le flux media et donnée

Voici une image explicative du processus

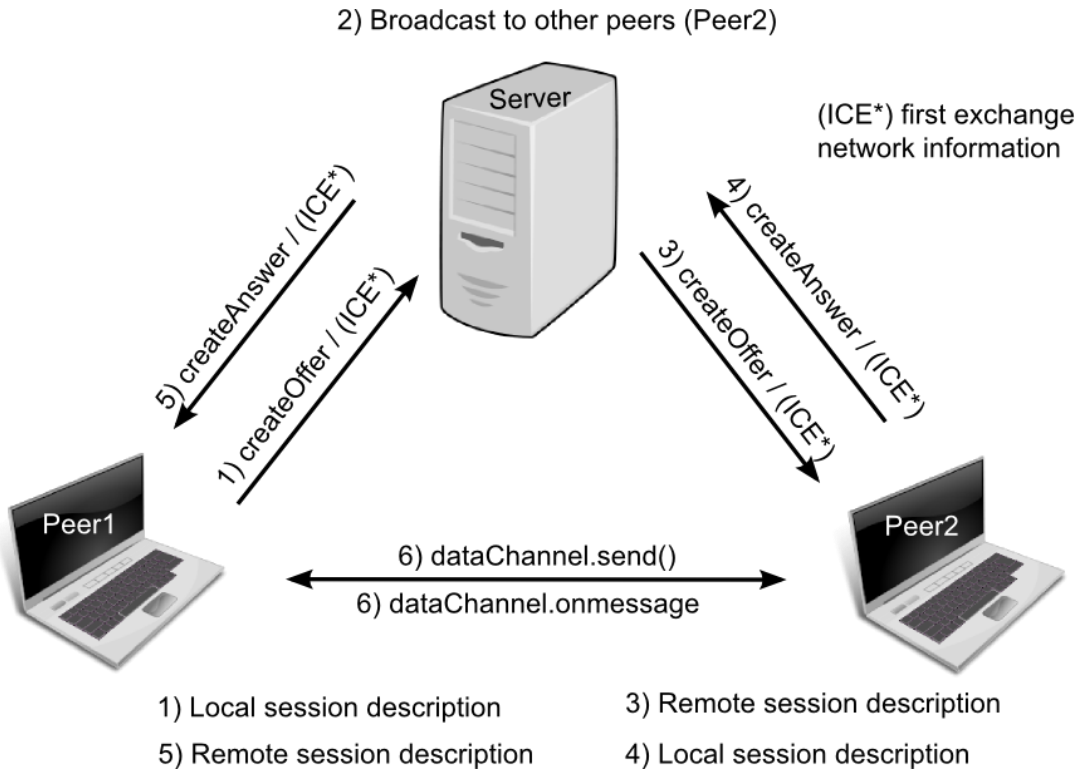

Figure 13 Processus d'une connexion WEBRTC

Etablissement d'une connexion entre deux clients utilisant WebRTC :

1 : Peer1 demande au serveur une connexion avec Peer2. (createOffer)

2 : Le serveur relaie la demande de Peer1 aux autres Peer qui sont connectés.

3 : Peer2 reçoit la demande de Peer1

4 : Si Peer2 accepte la demande, il renvoie au serveur une réponse. (createAnswer)

5 : Peer1 reçoit la réponse de Peer2

6 : Peer1 et Peer2 établi une connexion bidirectionnel.

HTTP Live Streaming

Description

Http Live Streaming est une architecture pour les périphériques iOS. Elle permet d'envoyer en direct ou une vidéo préenregistrée d'un site internet à un périphériques iOS. L'architecture est composée de 3 éléments : un serveur « élément », un distributeur « élément » et d'un logiciel client.

Cet architecture fonctionne avec les appareils iOS 3.0 ou plus récent et pour le navigateur avec Safari 4.0.

Architecture

Il est composé de 3 éléments :

1. Le serveur « élément » : il est responsable de prendre le flux d'entrée de la vidéo et de l'encoder. Ensuite, il digitalise et l'encapsule pour pouvoir être envoyé.

2. Le distributeur « élément » : c'est un simple web serveur. Il est responsable de faire le transfert de la vidéo à la bonne personne avec les ressources associées.

3. Le logiciel client : Il est responsable de télécharger les ressources et de les assembler afin que l'appareil puisse lire la vidéo.

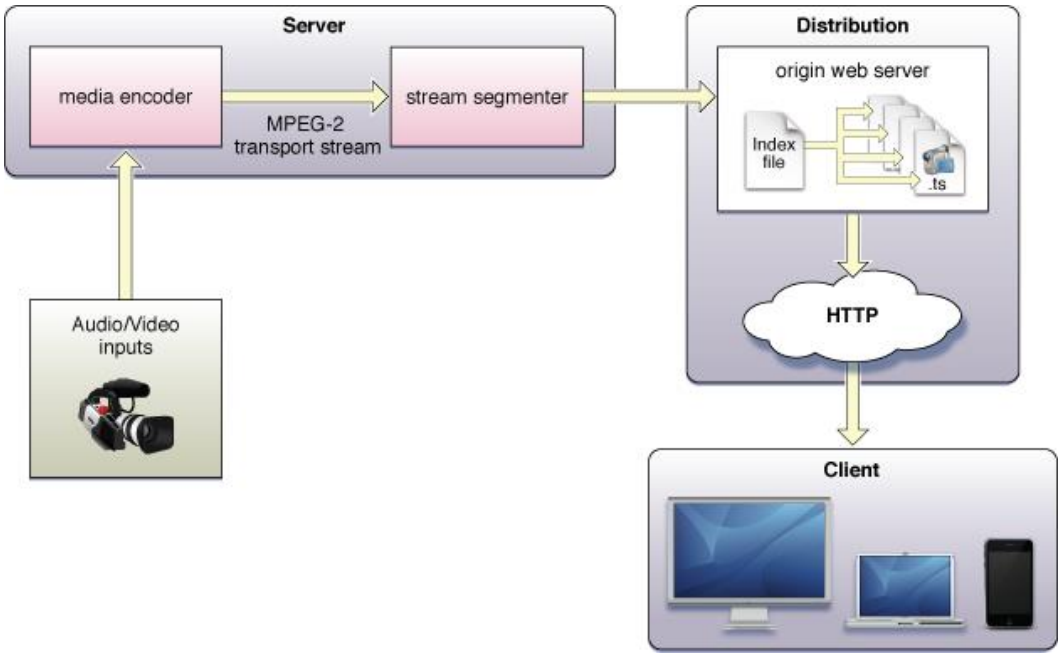

Figure 14 Http Live Streaming architecture

3.4. Analyse de l'architecture pour l'envoi de donnée

DataChannel (WebRTC)

Figure 15 logo Data Channel

DataChannel fait partie du WebRTC. Il permet l'envoi de donnée entre deux navigateurs, l'échange des données de jeu ou le transfert de fichier. Comme pour la communication, l'architecture s'initie avec une communication avec le serveur comme relaie et ensuite, il

devient une connexion entre pairs.

Il peut configurer de 2 manières le moyen de transfert de donnée :

UDP : un protocole de transfert qui ne garantit pas l'arrivée des données. Il ne possède pas d'en-tête pour les données qui permet un transfert beaucoup plus rapide.

TCP : un protocole de transfert qui contrôle l'arrivée de donnée au destinataire. Il possède un en-tête qui permet de vérifier l'intégralité des données mais ce protocole est plus lent.

Les données sont encrypté avec le Datagramme Transport Layer Security (DTLS) qui fait partie du WebRTC.

Hangout

Figure 16 logo hangout

Hangout est une plateforme de messagerie instantanée et de visioconférence développé par Google. L'architecture est basé sur le client-serveur. Il permet de communiquer de deux à dix utilisateurs en même temps. L'historiques des conversations sont conservés sur le serveur et permet de synchronisés sur plusieurs appareils.

Les Google Glass ont directement cette application. Il faut se connecter avec un compte de Google+.

SMS / MMS (Short Message Service / Multimedia Messaging Service)

Le SMS est aussi un moyen de transférer du texte via le réseau GSM et le MMS pour les

multimédias.

Les données sont alors envoyées directement sur le téléphone. Les données sont alors stocké dans la mémoire du téléphone ce qui libère de la mémoire des Google Glass.

Les Google Glass récupèrent les données et les affichent depuis le téléphone.

Mirror API

Mirror API est fourni par Google. Il est basé sur l'architecture REST. Il permet de faire certaines requêtes sur le serveur Google pour les Glass. Les informations sont stockées dans les serveurs de Google et ensuite, il transmette aux Glass sous forme de card. Une card peut contenir un texte, une image ou une vidéo. Il utilise OAuth 2.0 pour l'authentification et l'accès aux données sur le serveur.

4. Choix des outils et solutions

4.1. Equipement

L'équipement était déjà choisi dans le projet, c'est donc les Google Glass.

Les avantages des Glass sont sa légèreté avec environ 50 grammes, l'utilisation d'un prisme pour afficher l'écran qui laisse donc la transparence du champ de vision, et il possède un pavé tactile qui facilite le contrôle des Glass en cas de problème liée à la reconnaissance vocale.

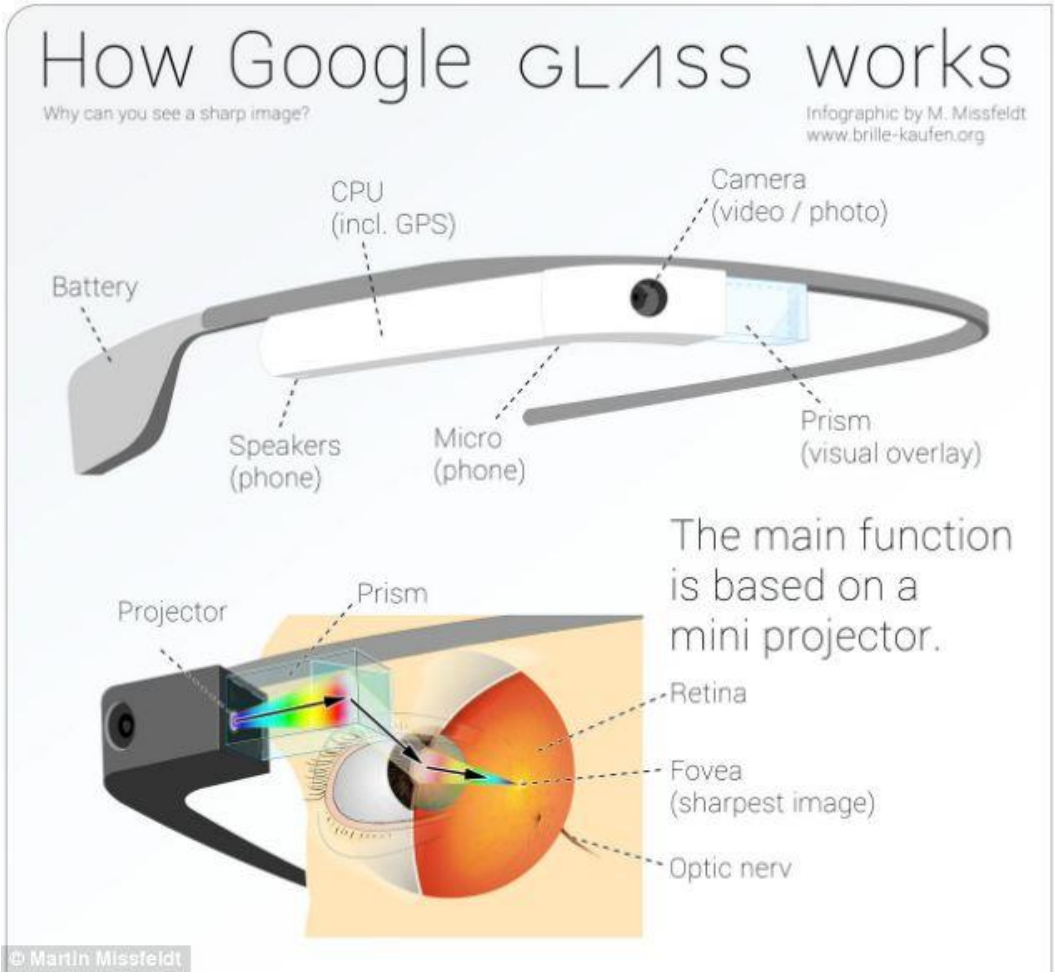

Figure 17 fonctionnement des Google Glass

Pour les inconvénients, il y a l'autonomie environ 20 minutes à 100% qui pourrait être un peu

court pour certaines interventions mais il serait possible d'en rajouter une batterie de externe pour prolonger l'autonomie. Les Glass sont potentiellement fragiles mais d'après les ambulanciers, ils sont convenables pour les interventions. Et pour finir, ils sont obligés d'avoir un point d'accès ou une connexion Bluetooth avec le téléphone afin d'avoir une communication avec l'extérieur.

Voici une fiche technique du Google Glass et une liste des composants :

Features	
	Google Glass 16GB
Operating System	Android 4.0.4 Ice Cream Sandwich
Display	640x360
Battery	570 mAh
Camera	5 MP
Connectivity & Sensors	WiFi/Bluetooth, GPS, Accelerometer, Compass, Gyroscope
NAND	16 GB
SDRAM	1GB DDR2 SDRAM
Processor	Texas Instruments OMAP4430
Baseband + Transceiver	None

Cost*	
	Google Glass 16GB
Teardown Date	April-2014
Display/Touchscreen & Glass	$3.00
Battery	$1.14
Camera	$5.66
Connectivity	$10.79
NAND	$8.18
SDRAM	$4.68
Processor	$13.96
BB+XCR	$0.00
Power Mgmt/Audio	$3.52
Non-Electric	$13.63
Other	$11.32
Supporting Materials	$1.75
Assembly & Test	$2.15
Total	$79.78

Figure 18 fiche technique et liste des composants

(http://www.objetconnecte.net/quel-est-le-prix-reel-lunettes-google-glass/)

Il faut prendre en considération de ces remarques, que le Google Glass utilisé pour le travail Bachelor est la version Explorer qui est un prototype et non le produit commercial avec la version XE18.11.

4.2. Architecture de communication

L'architecture de communication utilisée pour ce projet est le WebRTC.

Comparer à http Live Streaming, il y a moins d'élément à implémenter et aussi, on n'est pas restreint à des produits Apple surtout que les Google Glass fonctionnent sous Android.

L'avantage du WebRTC est la mise en place de son architecture car il ne demande pas une

grande charge de travail du côté du serveur et vu que les clients se connectent en direct entre eux, il n'y aura moins de temps d'attente de réponse.

L'inconvénient est que cette architecture est basé sur le JavaScript donc faut l'adapter pour les périphériques Android et pour les Google Glass. Le WebRTC est encore au stade brouillon, il se peut qu'il y ait encore des bugs dans cette architecture ou des éléments qui ne fonctionnent pas et en plus il est récent, il n'y a pas beaucoup de référence sur l'utilisation du WebRTC avec Android.

4.3. Architecture pour l'envoi de donnée

L'architecture que j'ai utilisée pour l'affichage du texte sur Google Glass est DataChannel du WebRTC et le Mirror API.

Parce que la connexion entre les clients est déjà basée sur le WebRTC. Le texte transmis via cette architecture ne passe pas par un serveur alors que les autres solutions présenté ci-dessus, à part l'architecture SMS/MMS, sont stockés sur le serveur puis retransmis au téléphone ou au Glass.

L'inconvénient est qu'il est dépendant de la connexion avec le WebRTC. Si la connexion avec le WebRTC n'est pas disponible, il n'est pas possible d'envoyer les messages et le message n'est pas retransmis plus tard. C'est pourquoi la deuxième architecture que j'ai choisie est le Mirror API de Google qui n'a pas besoin d'une connexion avec les Glass et si les Glass n'ont pas de connexion, le message est retransmis dès qu'ils ont de nouveau le réseau. Autre inconvénient du Mirror API est que leur librairie client est toujours en cours de développement.

4.4. Outils de développement

Eclipse

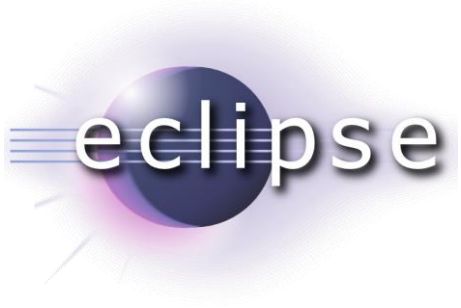

Figure 19 logo eclipse

Eclipse est un environnement de développement qui s'appuie principalement sur le JAVA. Il a été choisi parce qu'il est extensible, il permet d'intégrer le SDK Android et l'interface est simple à utiliser. Il va permettre de développer le client pour le prototype sur Android et les Glass

J'ai choisi éclipse parce que j'ai déjà développé avec ce logiciel pendant les études à la HES-SO Valais et en même temps, il était recommandé par M. Antoine Widmer.

Node.js

Figure 20 logo node.js

C'est une plateforme logicielle libre en JavaScript orienté vers les applications réseau. Et il contient une bibliothèque de serveur http qui peut être utilisé comme un serveur web. Il va être utilisé pour la partie du serveur du projet.

J'ai choisi le node.js car il est simple à utiliser, il ne demande pas une installation d'un logiciel

externe comme Apache et Lighttpd. Il peut être combiné avec le Framework Socket.io qui permet d'établir des connexions socket.

GDK

Le GDK, acronyme de Glass Development Kit, est un module d'extension de l'Android SDK. Il est destiné à développer des applications qui s'exécutent directement sur les Glass en ayant accès aux capteurs et à l'environnement Android. Grâce à GDK, on peut programmer les capteurs tels que le détecteur du pavé tactile, les cards et la reconnaissance vocale.

L'utilisation du GDK est indispensable pour la création d'une application qui fonctionne sur les Glass.

4.5. Langages utilisés

Javascript

Figure 21 logo JavaScript

Le JavaScript est un langage de programmation principalement utilisé pour les pages Web interactives mais aussi pour le côté serveur. C'est un langage orienté objet. Il permet de faire des améliorations au langage HTML en permettant d'exécuter des commandes du côté du client. Ce langage a été créé en 1995 par Brendan Eich pour Netscape Communication Corporation.

Le JavaScript est utilisé dans le serveur via le node.js et pour la page Web à cause de la connexion avec le WebRTC.

HTML

Figure 22 logo html

Le HTML, acronyme de HyperText Markup Language, est le format de données conçu pour représenter les pages Web. Il permet de structurer le contenu des pages. Il est souvent utilisé conjointement avec le JavaScript et les feuilles de style en cascade (CSS).

Structure d'un document HTML

Source HTML	Modèle du document

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
 <head>
  <title>
   Exemple de HTML
  </title>
 </head>
 <body>
  Ceci est une phrase avec un <a href="cible.html">hyperlien</a>.
  <p>
   Ceci est un paragraphe où il n'y a pas d'hyperlien.
  </p>
 </body>
</html>
```

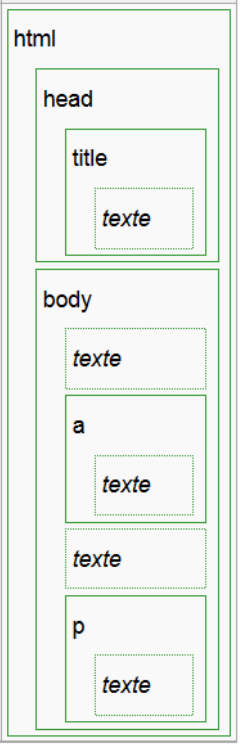

Figure 23 explication de la structure d'un document HTML avec le code Source à gauche et le modèle à droite

Le HTML est utilisé pour la page Web de l'hôpital. Le choix de ce langage est justifié par l'utilisation courant des développeurs Web.

CSS

Figure 24 logo CSS

Le CSS, acronyme de Cascading Style Sheets, forme un language informatique qui décrit la présentation des documents en HTML et XML. Il est utilisé couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs depuis les années 2000.

Le CSS, est utilisé pour la mise en forme de la page HTML de l'hôpital pour qu'il soit plus esthétique niveau visuel et en même temps gardé un format de présentation centralisé.

4.6. Librairies utilisés

Socket.IO

C'est une librairie JavaScript pour les applications Web en temps réel. Il permet d'établir une communication bidirectionnelle entre les clients et le serveur Web. Il comporte deux parties : une librairie pour le côté client qui s'exécute dans le navigateur et l'autre pour le serveur avec Node.js. Il utilise principalement le protocole WebSocket.

Libjingle (ou Jingle)

Libjingle est une librairie Jabber/XMPP libre écrite en C++ développé par Google. Il permet de créer des sessions audio et vidéo. Il utilise le Real-time Transport Protocol (RTP) pour

l’échange du flux multimédia. Le WebRTC utilise certain des composants de libjingle.

EJS (Embedded JavaScript)

EJS est une libraire qui permet de mieux organisé le code et de les séparer en plusieurs fichiers. Il combine les données et un modèle pour produire du HTML. Lors des erreurs, il est plus facile à trouver la line et le message d’erreur et il fonctionne avec tous les navigateurs

5. Outils complémentaires pour l'application

Dans cette section, on peut voir les différents programmes ou outils qui permettent de compléter certaines fonctionnalités.

Send to Google Glass (Chrome WebStore)[1]

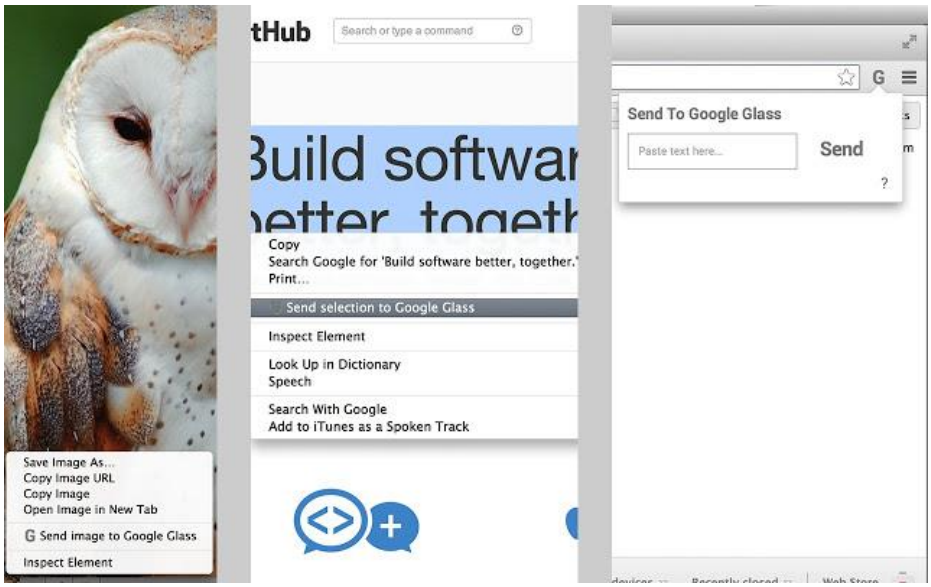

Figure 25 aperçu de l'utilisation du send to Google Glass (A gauche et au centre, une sélection du texte ou de l'image et l'envoie grâce au menu contextuel. A droite un popup du navigateur avec un textbox et un bouton pour l'envoi)

C'est une extension du navigateur Chrome et elle permet d'envoyer du texte via un popup sur le navigateur directement sur les Glass. Elle envoie au Glass qui est lié avec le même compte qui se trouve sur le navigateur. Et elle peut envoyer les éléments sélectionnés dans le navigateur et de l'envoyer via le menu contextuel. Les Glass qui reçoivent les informations envoyés via cette extension peuvent utiliser les fonctionnalités suivantes : lire à haute voix, supprimer et partager.

[1] https://chrome.google.com/webstore/detail/send-to-google-glass/ifhgjbjejfocglfphkdecifccicemfll (consulté le 22.07.2014)

Send to Google Glass (Android)[2]

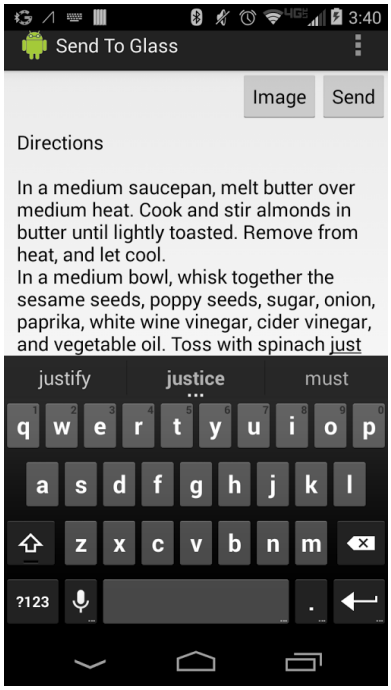

Figure 26 aperçu du send to Google Glass

Une application Android qui permet d'envoyer du texte et de l'image. Il se connecte au compte qui est lié au téléphone. Comme pour celui de l'extension, les Glass peuvent utiliser les fonctionnalités : lire à haute voix, supprimer et partager.

[2] https://play.google.com/store/apps/details?id=com.jiayao.glass.sender&hl=fr (consulté le 22.07.2014)

Screencastify[3]

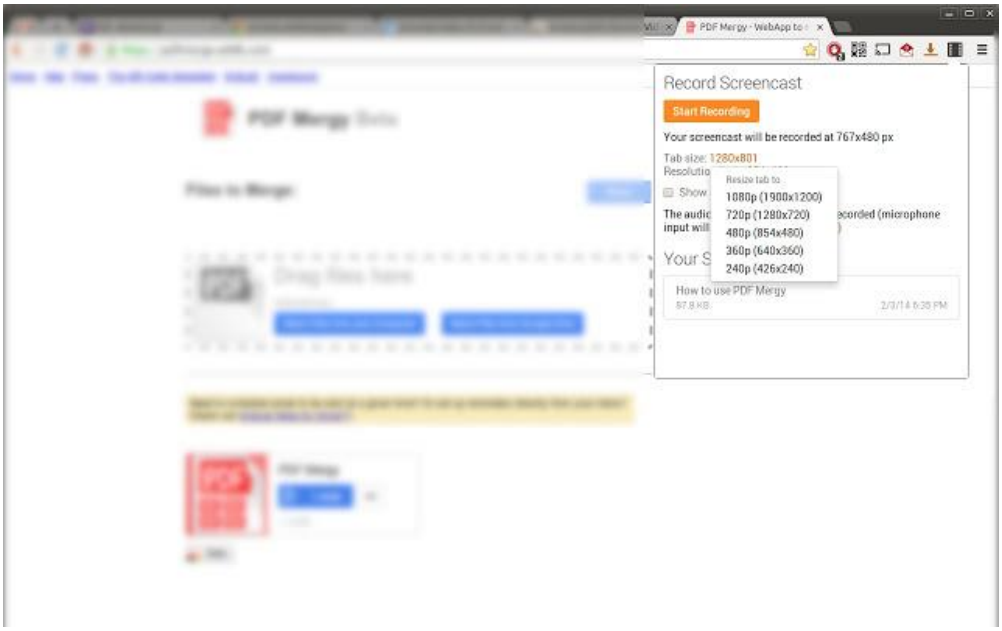

Figure 27 aperçu du Screencastify avec le choix de la résolution de la vidéo

Screencastify est une extension de Chrome. Il permet de faire des captures du navigateur en vidéo. Il enregistre aussi le son d'entrée tel que le microphone. Il peut être utile en cas de besoin d'archiver les conversations.

[3] https://chrome.google.com/webstore/detail/screencastify-screen-vide/mmeijimgabbpbgpdklnllpncmdofkcpn?hl=fr (consulté le 22.07.2014)

FastStone Capture[4]

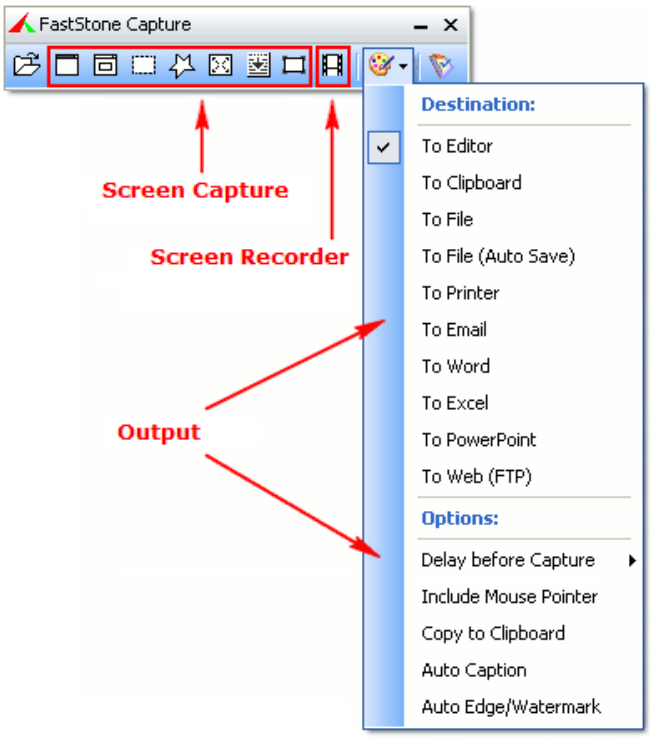

Figure 28 interface de FastStone

C'est un logiciel qui permet de capturer tout ce qui se passe à l'écran. Il est possible d'ajouter des cadres, des effets de transitions et bien plus encore. L'insertion de texte ou de formes géométriques dans les captures est aussi au programme. Il peut être utilisé pour archiver les conversations du navigateur si on n'utiliser pas le Google Chrome et il possède plus de fonctionnalités.

[4] http://www.faststone.org/FSCaptureDetail.htm (consulté le 22.07.2014)

6. Résultat

Le développement du prototype s'est arrêté le 18 juillet 2014. La version sur smartphone fonctionne parfaitement mais pour les Glass, il y a encore beaucoup de problème comme le surchauffe qui fait arrêter les Glass après quelques minutes d'utilisation, la fluidité de l'image filmé par les Glass et l'affichage de la vidéo distant. Les problèmes sont détaillés ci-dessous dans la rubrique 6.3 Problèmes rencontrés

6.1. Interface de l'application

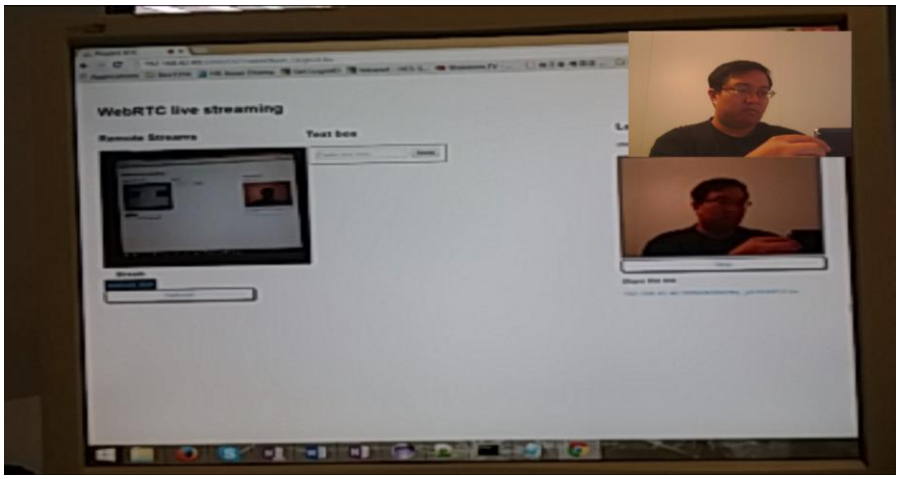

Figure 29 aperçu sur le smartphone qui est la même que sur les Glass

L'interface du Google Glass montre en arrière-plan la vidéo de la caméra et en haut à gauche, la vidéo à distance. S'il y a plusieurs personnes qui se connectent, leur vidéo se retrouve en dessous de la première vidéo.

6.2. Fonctionnalités de l'application

L'application permet :

- Envoyer un lien pour la connexion

- Envoyer la vidéo et l'audio
- Recevoir les vidéos des personnes qui sont connectés

6.3. Interface du site

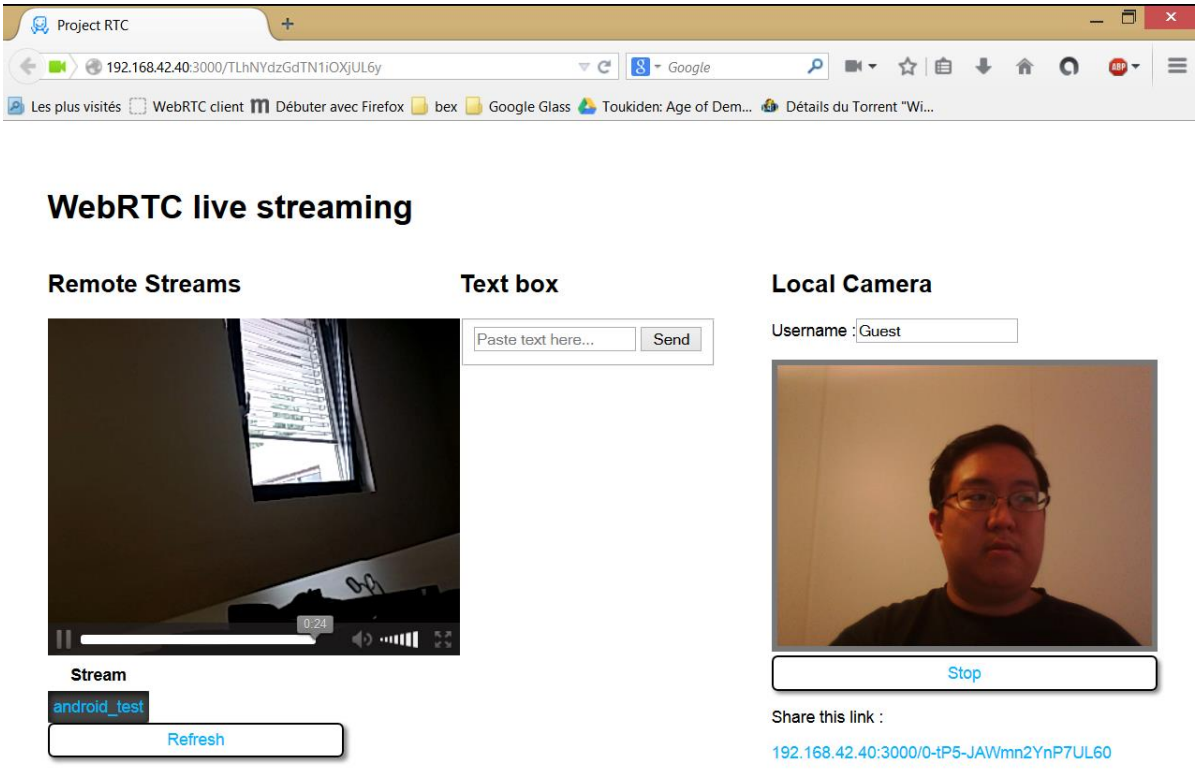

Figure 30 l'interface du site avec la vidéo à distance à droite,
un zone pour le texte au centre et la caméra de la webcam à droite

Pour la page Web, la présentation est assez simple. À gauche, on voit la vidéo filmée par les Google Glass. En dessous de la vidéo, dans la partie Stream, la liste des périphériques qu'on peut regarder avec un bouton qui permet de rafraîchir cette liste. Au centre de la page Il y a une zone de texte qui permet d'envoyer du texte. L'affichage du texte s'affiche en dessous. Et pour finir, à droite de la page Web, on retrouve notre webcam avec la possibilité de donner un nom, un bouton START pour démarrer la webcam. En-dessous, on trouve un lien qui permet de partager aux autres pour qu'il puisse voir votre webcam.

6.4. Fonctionnalités de la page Web

La page Web permet de :

- Recevoir la vidéo du Google Glass
- Envoyer du texte (uniquement navigateur à navigateur)
- Envoyer la vidéo au Google Glass

6.5. Problèmes rencontrés

Les problèmes reportés ci-dessous sont ceux qui ne sont pas encore résolu à la fin du développement. Pour voir tous les problèmes rencontrés, veuillez se référer l'annexe.

Problème de compatibilité avec la librairie Libjingle

L'application Android fonctionne sur les smartphones mais les Google Glass ne supportent pas cette libraire à cause qu'il y a un problème avec le moteur vidéo et audio. Ces deux moteurs appartiennent à la libraire Libjingle qui permet le formatage et le transfert de la vidéo et du son. Grâce à la mise à jour des Google Glass, l'application fonctionne mais il y a encore beaucoup d'erreur lors du lancement de l'application.

```
libjingle         Error(webrtcvideoengine.cc:1265): webrtc: (voe_audio_processing_impl.cc ↵
                  :990): virtual int webrtc::VoEAudioProcessingImpl::SetTypingDetectionSt ↵
                  atus(bool): not supported
```

Figure 31 message d'erreur d'éclipse sur la libraire du libjingle

Google APIs Client librairie en cours de développement[5]

Google recommande d'utiliser leur librairie officielle pour le développement avec l'API Mirror. Il existe pour de nombreuses langues mais pour le JavaScript, il est toujours en cours de développement. La librairie API Mirror contient une autre librairie qui est l'OAuth2. OAuth2 permet l'authentification et l'autorisation de certaines méthodes.

La surchauffe des Google Glass

Après avoir pu démarrer l'application sur les Glass, il y a un problème de surchauffe après 2-3 minutes d'utilisation. L'application se termine et les Glass affichent un message comme l'image ci-dessous.

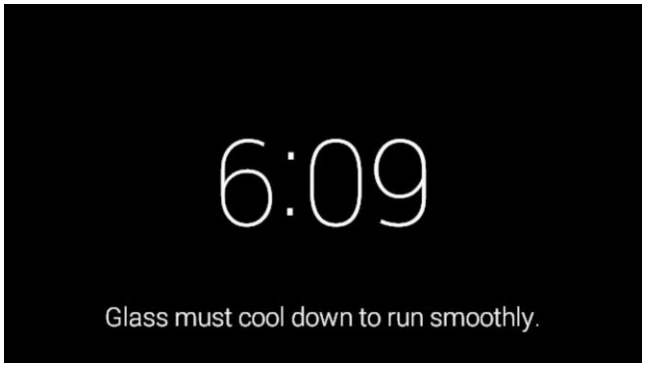

Figure 32 écran Glass avec message d'avertissement

L'affichage de la vidéo de la page web sur les Google Glass

La Glass ne capte pas le flux vidéo envoyé depuis l'hôpital. Sur l'écran de la Glass, on peut voir un carré vert dans l'emplacement réservé à la vidéo de l'hôpital.

[5] https://code.google.com/p/google-api-javascript-client/ (consulté le 22.07.2014)

7. Conclusion

7.1. Analyse des résultats obtenus

Malgré les problèmes rencontrés, l'application fonctionne. Il respecte les principales fonctionnalités qu'on avait décidées au début du projet. La page Web diffère légèrement du mockup qu'on avait décidé mais vu que j'ai utilisé la technologie ejs, il est simple d'ajouter d'autres fonctionnalités au page Web. Il reste tout de même beaucoup de chose à modifier et à améliorer qui va être expliqué dans les chapitres suivants.

7.2. Avantages et inconvénients

Vu que la version sur le Google Glass ne fonctionne pas complètement, les avantages et les inconvénients sont faits sur l'application qui fonctionne sur le smartphone

L'avantage qu'offre cette application est que le porteur de Google Glass n'a pas besoin de faire une commande pour la connexion. Il suffit de lancer une application et un lien sera transmis à l'hôpital et dès qu'il ouvre le lien, la connexion se fait automatiquement et le porteur de Google Glass peut voir la personne s'est connectée.

L'inconvénient est le fait que lorsque l'hôpital envoie une image ou un texte. Le porteur de Glass est obligé d'arrêter la transmission vidéo et de consulté le texte ou l'image à cause que API Mirror stocke une image sur leur serveur et qu'il transfère par la suite au Google Glass en créant un carte d'évènement.

7.3. Opinion personnelle et critique

Je suis assez déçu de mon travail parce que je n'ai pas réussi à faire fonctionner l'application sur les Google Glass de la même manière que sur les smartphones.

Je pense qu'il est trop tôt pour développer des applications sur les Google Glass. Parce qu'il n'y a pas beaucoup de référence pour s'appuyer dessus et certains éléments ne sont pas supportés par les Google Glass comme pour le problème de la libraire Libjingle et la librairie API Mirror Client qui n'est pas encore totalement achevée. Au niveau du matériel, pour la version « Explorer », il manque de la puissance car on peut voir la différence de fluidité entre la Glass et le Smartphone et en même temps, la Glass surchauffe trop rapidement qui ne permet pas d'une utilisation prolongée.

Du côté du WebRTC, cette technologie est toujours au stade preliminaire. On peut le constater avec l'instabilité des connexions et aussi la compatibilité avec certains navigateurs.

Comme point positif de ce travail de Bachelor est la possibilité de travailler avec ces nouvelles technologies que je pense qu'il va être très utilisé dans un futur proche. Le WebRTC pour son côté de pair à pair qui demande pas une connexion avec le serveur et il réduit le délai d'attente des vidéos. Les Google Glass déjà pour son côté tendance dans l'informatique qui va accroître un nombre considérable d'utilisateurs et pour son côté pratique et utile qui va être utilisé dans certains métiers comme pour la police, l'hôpital, les pompier,…

Un autre point positif, tous les développements dans ce projet est une nouvelle expérience pour moi. J'ai pu apprendre à utiliser le node.js pour faire simuler un serveur, le JavaScript pour la connexion du WebRTC et le développement des Glass. Je n'avais jamais fait auparavant à part le développement Android que j'ai eu des cours à l'HES-SO Valais.

7.4. Améliorations

L'interface du page Web peut être mieux faite. On peut ajouter un log de communication afin de savoir si la connexion est toujours actif et s'il y a une coupure, de pouvoir voir la raison de cela ou encore d'ajouter un image des machines médicales pour avoir un suivi de la patient.

Pour l'application, il faut modifier afin qu'elle puisse gérer en cas de coupure de connexion pour qu'elle se rétablisse automatiquement. Pour régler le problème de la surchauffe et de l'autonomie de la batterie, au démarrage de l'application, il crée une connexion avec

uniquement le son. L'utilisateur pourra ensuite activer la caméra par la suite avec le pavé tactile du Glass.

On peut créer une application Android pour le smartphone pour transférer les données des machines médicales à la page web et au Google Glass.

Il faudra garder un œil sur le produit MindRDR[6] car il pourrait être utile pour l'utilisation des Google Glass via la pensée.

7.5. Utilisation dans d'autres contextes

Ce projet peut être utilisé dans d'autres contextes à part les interventions des ambulanciers.

Dans le domaine médical, il serait possible d'utiliser lors de l'opération chirurgicale. Le chirurgien peut demander une assistance par un chirurgien plus expérimenté et peuvent se communiquer en temps réel du problème. Sinon, on peut aussi adapter l'application afin qu'il cherche dans une base de donnée le dossier du patient lors des visites de chambre, des procédures médicales et etc.

Dans le domaine de l'intervention, le projet peut être utile aux pompiers. Le centre de contrôle pourra donner directement les ordres sur les Glass des pompiers et savoir le déroulement de l'opération. Les pompiers peuvent recevoir sur leurs Glass le plan du bâtiment. La police qui n'a plus besoin d'appeler au quartier général pour vérifier l'identité d'une personne.

[6] http://mindrdr.thisplace.com/static/index.html (consulté le 22.07.2014)

8. Remerciement

Je tiens à remercier chaleureusement ici toutes les personnes qui ont contribués, de près ou de loin, à la réalisation de mon travail de Bachelor.

Je porte une attention particulière à :

Dr. Henning Müller, pour m'avoir suivi et encadrer durant la réalisation de ce travail.

Dr. Antoine Widmer, pour l'assistance technique.

9. Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

Dr. Henning Müller, professeur chargé du suivi

Dr. Antoine Widmer, assistant à la HES-SO Valais

Martigny, le 27 juillet 2014

Wai-kin Hau

10. Références

[1] Brunetti, N. D.; Dellegrottaglie, G.; Lopriore, C.; Di Giuseppe, G.; De Gennaro, L.; Lanzone, S. and Di Biase, M. “Prehospital Telemedicine Electrocardiogram Triage for a Regional Public Emergency Medical Service: Is It Worth It? A Preliminary Cost Analysis,” Clinical Cardiology, Wiley Periodicals, Inc., 37, pp. 140-145, 2014.

[2] Hsieh, J.-C.; Li, A.-H.; Yang,C.-C. “Mobile, Cloud, and Big Data Computing: Contributions, Challenges, and New Directions in Telecardiology”. International Journal of Environmental Research and Public Health, 10, 6131-6153, 2013.

[3] Muensterer, O.; Lacher, M.; Zoeller, C.; Bronstein, M. and Kbler, J., “Google glass in pediatric surgery: An exploratory study,” International Journal of Surgery, vol. 12, no. 4, pp. 281 – 289, 2014.

[4] Albrecht, U.-V.; von Jan, U.; Kuebler., J.; Zoeller, C.; Lacher, M.; Muensterer, O.; Ettinger, M.; Klintschar, M. and Hagemeier, L., “Google glass for documentation of medical findings: Evaluation in forensic medicine,” Journal of Medical Internet Research, vol. 16, no. 2, pp. e53, Feb 2014.

11. Sources

11.1. Bibliographie

Programming Google Glass, by: Eric Redmond, ISBN: 9781937785796, date: 2013-12-30

11.2. Documents

AugmentedMedic – Using Google Glass to enhance pre-hospital care – d'Antoine Widmer

11.3. Webographie

Introduction

SMUR

http://www.smur.ch/index.php (consulté le 07.07.14)

http://fr.wikipedia.org/wiki/Service_mobile_d%27urgence_et_de_r%C3%A9animation (consulté le 07.07.14)

PRISTINE

http://pristine.io/ (consulté le 07.07.14)

Livestream

http://new.livestream.com/producer/glass (consulté le 07.07.14)

Analyse de lunettes connectées

YouTube – les meilleurs lunettes intelligentes 2014

https://www.youtube.com/watch?v=djrsNxyuBfQ (consulté le 12.05.2014)

Google Glass

http://fr.wikipedia.org/wiki/Google_Glass (consulté le 12.05.2014)

Vuzix M100

http://en.wikipedia.org/wiki/Vuzix (consulté le 12.05.2014)

http://www.vuzix.com/consumer/products_m100/ (consulté le 12.05.2014)

Recon Jet

http://www.reconinstruments.com/products/jet/ (consulté le 12.05.2014)

http://branchez-vous.com/2013/09/26/recon-jet-un-concurrent-a-google-glass-destine-aux-sportifs/ (consulté le 12.05.2014)

GlassUp

http://www.glassup.net/ (consulté le 12.05.2014)

MetaPro

https://www.spaceglasses.com/ (consulté le 12.05.2014)

Atheer Labs

https://www.atheerlabs.com/ (consulté le 12.05.2014)

Architecture de communication

WebRTC

http://fr.wikipedia.org/wiki/WebRTC (consulté le 18.05.2014)

http://www.webrtc.org/ (consulté le 18.05.2014)

https://developer.mozilla.org/fr/docs/WebRTC (consulté le 18.05.2014)

http://chimera.labs.oreilly.com/books/1230000000545/ch18.html#_audio_and_video_engines (consulté le 24.06.2014)

HTTP Live Streaming

https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/stre

amingmediaguide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html (consulté le 24.06.2014)

JavaScript

http://fr.wikipedia.org/wiki/JavaScript(consulté le 1.06.2014)

http://www.commentcamarche.net/contents/577-javascript-introduction-au-langage-javascript (consulté le 1.06.2014)

Architecture pour l'envoi de donnée

Hangout

http://www.google.com/+/learnmore/hangouts/?hl=fr (consulté le 1.07.2014)

http://fr.wikipedia.org/wiki/Google_Hangouts (consulté le 1.07.2014)

SMS/MMS

http://fr.wikipedia.org/wiki/Short_Message_Service (consulté le 1.07.2014)

http://fr.wikipedia.org/wiki/MMS (consulté le 1.07.2014)

GDK

http://www.goglasses.fr/google-glass/developpement/quelle-est-la-difference-entre-lapi-mirror-et-le-gdk-pour-les-google-glass (consulté le 21.07.2014)

https://developers.google.com/glass/develop/gdk/ (consulté le 21.07.2014)

Mirror API

https://developers.google.com/glass/develop/index (consulté le 1.07.2014)

Outils de développement

Eclipse

http://www.eclipse.org/ (consulté le 1.07.2014)

http://fr.wikipedia.org/wiki/Eclipse_%28projet%29 (consulté le 1.07.2014)

Langages utilisés

Node.js

http://nodejs.org/ (consulté le 1.07.2014)

http://fr.wikipedia.org/wiki/Node.js (consulté le 1.07.2014)

HTML

http://fr.wikipedia.org/wiki/Hypertext_Markup_Language (consulté le 1.07.2014)

CSS

http://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade (consulté le 1.07.2014)

Librairies utilisés

Socket.IO

http://socket.io/ (consulté le 1.07.2014)

http://en.wikipedia.org/wiki/Socket.IO (consulté le 1.07.2014)

Libjingle

http://fr.wikipedia.org/wiki/Jingle_%28protocole%29 (consulté le 1.07.2014)

http://www.webrtc.org/reference/webrtc-internals (consulté le 1.07.2014)

EJS

http://embeddedjs.com (consulté le 1.07.2014)

12. Glossaire

Android	Il s'agit d'un système d'exploitation mobile qui est open source.
API	Il s'agit d'un ensemble de class, de méthode ou de fonction qui permet d'utiliser les fonctionnalités d'un programme externe.
DTLS	Il fournit une sécurisation des échanges basés sur des protocoles en mode datagramme.
GS	Il s'agit d'une norme pour la téléphonie mobile.
ICE	Il permet d'établir des connexions directs avec un autre à en utilisant les protocoles tels que STUN et TURN.
Interface holographique	Il s'agit d'une image représenté en 3D apparaissant comme « suspendue en l'air ».
iOS :	Il s'agit d'un système d'exploitation mobile pour les produits Apple.
IP	Il s'agit d'un protocole internet permettant un service d'adressage unique pour l'ensemble des terminaux connectés.
Jabber/XMPP	Il s'agit d'un protocole de messagerie instantanée construit sur le protocole Extensible Messaging and Presence Protocol.
Java	Il s'agit d'un langage de programmation orienté objet pouvant s'exécuter sur n'importe quelle plateforme grâce à une machine virtuelle
Logiciel libre	Il s'agit d'un logiciel qui permet aux utilisateurs d'utiliser, modifier et même distribuer le programme.

Machine Virtuelle	Il permet de précompilé un code afin qu'il puisse exécuter sur une plateforme donnée.
Mockup	Il s'agit d'une maquette pour représenter l'interface d'un projet informatique.
Réalité augmenté	Il permet de mettre la superposition d'un objet en 3D ou 2D à la perception que nous avons de la réalité et ceci en temps réel.
REST	Il s'agit d'un style d'architecture pour concevoir des applications ou des services Web.
SDK	Il s'agit d'un kit de développement qui est un ensemble d'outils permettant aux développeurs de créer des applications.
SRTP	Il permet d'apporter le chiffrement, l'authentification et l'intégrité des messages et la protection pour les données RTP
STUN	Il s'agit d'un protocole client-serveur permettant à un client UDP situé derrière un routeur de découvrir son adresse IP.

13. Table des illustrations

14. Annexes

14.1. Gestion du projet

14.1.1. Déroulement

Ce projet de Bachelor s'est déroulé entre le 28 avril 2014 et le 28 juillet 2014. Le temps consacré à la réalisation est de 400 heures. Ce projet correspond donc aux exigences lié à un travail de Bachelor de la HES-SO Valais

14.1.2. Planification

Avec M. Müller et M. Widmer, la planification était relativement libre parce qu'il dépendait beaucoup de l'avancement de chaque étape. Mais, on peut ressortir les grandes lignes des différentes étapes et les semaines qui sont consacrés.

Etapes	**Semaines**
Recherche	2
Mise en place WebRTC	2
Création application Android (Smartphone)	2
Adaptation application sur Glass	2
Rédaction du Rapport	4

14.1.3. Suivi

En accord avec M. Müller et M. Widmer, il est décidé de se voir une fois par semaine pour avoir un suivi sur l'avancement du projet. A chaque séance, on discute ce qu'il y a été fait, les problèmes rencontrés et une petite planification pour la semaine suivante.

14.1.4. Cahier des charges

Le cahier des charges est répartir en 4 parties :

- Général
- Server
- Site
- Glass

En vert représente les fonctionnalités qui sont réalisés.

En orange représente fonctionnalité existant via un outil ou un logiciel

En rouge, ceux qui sont incomplet ou pas réalisé.

Général

Sécurisation des flux de données

Les données passent à travers la méthode PeerConnection du WebRTC. Sans l'accès à cette méthode, les données qui s'échangent ne peuvent pas être reçues. Pour la page Web, il y a le chiffrement utilisé par les navigateurs avec le DTLS.

Prototype intégré les fonctionnalités ci-dessus

Une partie des fonctionnalités sont intégrés dans le prototype mais la connexion en WebRTC et les Google Glass ne sont pas encore très stable. Il manque aussi la possibilité d'envoyer un texte au Google Glass

Serveur

Etablir la connexion WebRTC

Le serveur est fait avec le node.js. Le serveur reçoit les demandes et le relaie à ceux qui sont connectés au serveur.

Glass

Envoie de la vidéo

Il fonctionne. Les Glass montrent une image que la caméra voit.

Recevoir la vidéo depuis le site

La réception des vidéos venant du site fonctionne avec le smartphone. Il peut communiquer avec 2 personnes en même temps. Pour les Google Glass, un petit carré vert prend place à l'endroit de la vidéo montrant que la connexion est faite mais il n'arrive pas à récupérer le flux vidéo.

Recevoir les informations du patient de l'hôpital

Il existe déjà des outils permettant de faire cela comme le send-to-glass qui utilise l'OAuth2 de Google. Mais avec cette solution, les données sont stockées sur le serveur de Google. J'ai donc mis en place le DataChannel de WebRTC, il fonctionne entre navigateur mais pour les Glass, il n'est pas encore implémenté. J'ai aussi mis en place le socketIO qui fonctionne avec les Glass mais une connexion avec le serveur est obligatoire.

Web

Interface simple

La page est lancée via un lien envoyé depuis les Glass. La connexion se fait automatiquement dès le moment que l'utilisateur accepte de partager les ressources vidéo et audio de son appareil.

Affichage de la vidéo du Google Glass

La partie gauche de la page Web est réservé à l'affichage de la vidéo du Glass avec la possibilité de mettre en plein écran et de supprimer le son.

Transfert du texte au Google Glass

Le texte peut être transmis d'un navigateur à un autre mais sur l'application, il n'est pas

encore implémenté.

14.2. Déroulement du projet

14.2.1. Avant développement (semaine 1 à 2)

Description

Les deux premières semaines, j'ai dû faire des recherches sur les spécificités des Google Glass et de son développement. Ensuite, j'ai cherché les différents moyens de communication possible pour ce projet. M. Henning Müller m'a proposé de commencé par développé une application Android en utilisant la caméra de téléphone.

A part le développement, j'ai fait le cahier des charges.

Problème rencontré

L'application Android avait un problème de rafraîchissement de la caméra. Le problème a pu être résolu en spécifiant la fréquence souhaitée.

14.2.2. Développement WebRTC Server (semaine 3 à 4)

Description

Après une discussion avec M. Antoine Widmer, j'ai décidé de commencer à développer la partie du serveur du WebRTC parce que sans ce serveur aucun connexion sera possible et je ne pourrais pas tester l'application que je vais créer. Pour ce faire, je me suis inspiré des codes WebRTC codelab (https://bitbucket.org/webrtc/codelab/src).

En parallèle, j'ai modifié l'application avec la caméra pour qu'il puisse fonctionner avec les Google Glass.

A part le développement, j'ai regardé les licences de différentes ressources.

Problème rencontré

J'ai essayé de faire un menu pour l'application du Google Glass mais il ne fonctionne pas.

14.2.3. Développement Application Android avec WebRTC (semaine 6 à 8)

Description

Le WebRTC fonctionne avec les différents navigateurs comme Firefox et Chrome et ça fonctionne aussi pour ceux du téléphone. Il faut développer maintenant une application Android qui utiliser le WebRTC.

Problème rencontré

Au début, j'ai essayé d'utiliser le flux vidéo créé par le MediaRecorder qui est une méthode de base sur l'Android. Mais j'ai pu remarquer que le flux vidéo accepté par le WebRTC est de type MediaStream. J'ai cherché une alternative et j'ai trouvé un projet de Pierre Chabardes (https://github.com/pchab/ProjectRTC) qui a réussi à faire le OpenGl ES de l'Android pour faire fonctionner. Mais son projet ne fonctionnait pas à cause du cryptage des navigateurs que j'ai résolu.

Remarque

J'ai pris environ 1 semaine de temps pour préparer les examens de la HES-SO Valais

14.2.4. Développement Application Android sur les Glass + envoie du texte (semaine 9 à 11)

Description

L'application Android avec un smartphone fonctionne parfaitement avec la page Web. Maintenant, il faut adapter cette application Android pour que les Google Glass puissent l'utiliser.

En même temps, il faut mettre en place sur la page Web et le serveur les éléments pour la création du DataChannel afin d'envoyer du texte via la page Web.

A part le développement, j'ai rédigé la première version du rapport du travail de Bachelor

Problème rencontré

Les Google Glass ne supportent pas la librairie Libjingle qui est important pour le fonctionnement du WebRTC à cause du moteur vidéo et audio. Grâce à la mise à jour du Google Glass, l'application peut se lancer mais j'ai pu rapidement voir un nouveau problème pour le développement. Les Google Glass surchauffent trop rapidement après 2 à 3 minutes d'utilisation ceux qui empêche de développer en continue sur les Glass.

14.2.5. Finalisation du rapport (semaine 12 à 13)

Description

Il reste encore 2 semaines avant de rendre le TB. J'ai décidé d'utiliser la majeure partie de ce temps pour compléter le travail de Bachelor et de le finaliser.

14.3. Codes

14.3.1. Serveur

Le serveur est écrit en Javascript en utilisant le Node.js et le SocketIO

App.js

Le fichier principal est app.js qui est le code pour démarrer le serveur.

```
var express = require('express')
  , http = require('http')
  , path = require('path')
  , streams = require('./app/streams.js')();

var app = express()
  , server = http.createServer(app)
  , io = require('socket.io').listen(server);

// all environments
app.set('port', 3000);
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.use(express.favicon(__dirname + '/public/images/favicon.ico'));
app.use(express.logger('dev'));
app.use(express.bodyParser());
app.use(express.methodOverride());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));
```

Figure 33 code pour démarrer le serveur

Cette partie du code permet de paramétrer le serveur. Le premier bloc indique les modules qu'on a besoin tels que le http et le path. Ensuite, il y a la création du serveur en utilisant le module http et la dernière ligne du bloc, on indique l'utilisation du socketIO qui écoute via le serveur. Pour finir, le dernier bloc représente les paramètres pour configurer le serveur comme le port d'écoute à 3000.

A part app.js, il y a 3 autres fichiers JavaScript qui permettent de le compléter.

Route.js

Il permet d'afficher les éléments textes qui sont fixe de la page Web comme le titre de la page et il affiche aussi dès qu'il reçoit une nouvelle personne qui se connecte.

SocketHandler.js

```
io.sockets.on('connection', function(client) {
  console.log('-- ' + client.id + ' joined --');
  client.emit('id', client.id);

  client.on('message', function (details) {
    var otherClient = io.sockets.sockets[details.to];

    if (!otherClient) {
      return;
    }
      delete details.to;
      details.from = client.id;
      otherClient.emit('message', details);
  });

  client.on('readyToStream', function(options) {
    console.log('-- ' + client.id + ' is ready to stream --');

    streams.addStream(client.id, options.name);
  });
```

Figure 34 une partie du code socketHandler

La deuxième partie du code est utilisé dès le moment qu'un client se connecte sur le serveur. Il lit ensuite les messages transmis sur le socket. Dès qu'un des messages correspond à la valeur défini dans le socket.on, il commence à exécuter la fonction. Quand le nombre de client qui sont connecté dans la salle est de moins de 2, il exécute la méthode socket.join() qui permet d'ajouter le client dans la salle et de pouvoir communiquer avec d'autres clients. A fin de chaque fonction, une réponse est renvoyée au client grâce à la méthode socket.emit(). Le log est utilisé pour avoir un suivi de l'exécution.

Stream.js

Il gère le flux vidéo de la page grâce au identifiant stocké dans le socket.

14.3.2. Page Web pour l'hôpital

Index ejs

Index.ejs est composé de plusieurs fichiers ejs. Le layoutTop est composé des scripts qui sont utilisés, le localCam et le remoteStream est fait pour afficher les vidéo et le textbox est composé d'une zone de texte et d'un bouton pour envoyer. Le code utilisé est relativement standard.

RTCclient.js

C'est le fichier qui va gérer toute la partie WebRTC. Au moment de la connexion la méthode addPeer démarre et crée l'objet PeerConnection qui permet de faire une connexion direct avec un autre client.

```
function addPeer(remoteId) {
console.log("addPeer methode start");
  var peer = new Peer(config.peerConnectionConfig, config.peerConnectionConstraints);

  peer.pc.onicecandidate = function(event) {
  peer.pc.onaddstream = function(event) {
  peer.pc.onremovestream = function(event) {
  peer.pc.oniceconnectionstatechange = function(event) {
  peerDatabase[remoteId] = peer;

  return peer;
}
```

Figure 35 la méthode addPeer du fichier RTCclient.js

Ensuite, il y a des méthodes qui permettent d'instancier la création d'une connexion avec un autre. SessionDescription contient des informations du client pour établir la connexion.

```
function offer(remoteId) {
console.log("offer methode start");

  var pc = peerDatabase[remoteId].pc;
  pc.createOffer(
    function(sessionDescription) {
      pc.setLocalDescription(sessionDescription);
      send('offer', remoteId, sessionDescription);
    },
    function(error) {
}
```

Figure 36 méthode pour l'envoie de l'offre

RTCViewModel.js

Il permet de faire le lien entre le fichier index.ejs et le RTCclient.js. Comme exemple, Il transmet le flux vidéo du WebRTC jusqu'à à la balise vidéo de la page Web.

14.3.3. Application Android

Pour l'application Android, il se compose de 4 fichiers Java.

RTCActivity.java

C'est le fichier qui permet de démarrer l'application et qui gère les différents états de l'appareil.

WebRtcClient.java

Il contient tous les méthodes liées à la connexion du WebRTC. Pour créer l'objet PeerConnection, on utilise la librairie Libjingle. Le principe reste le même que celui de la page Web de l'hôpital.

VideoStreamView.java

Ce fichier est fourni par la libraire Libjingle. Il gère l'affichage des vidéos à l'écran. Il utilise le principe de dessiner la capture vidéo sur l'écran avec OpenGL.

FramePool.java

Ce fichier est fourni par la libraire Libjingle. Il gère le taux d'image pour l'affichage de la vidéo.