— Report P&O CW 2019–2020 Task 3 —

Department of Computer Science – KU Leuven October 21, 2019

Team 12	
Toon Sauvillers	60h
Seppe Van Steenbergen	54h
Bert Van den Bosch	52h
Frédéric Blondeel	20h

1 Introduction

(Few sentences.) *Clearly state the problem you tackled. The task's assignment defined the functionality you have to offer, not the problem(s) you solved. You have to phrase the exact problem statement, i.e., state the assumptions you made. On top of the problem statement we expect you to state the relevance of the problem and what solution is chosen. Later in §3 you can go into detail as why this is your preferred solution. Throughout the text we expect you to back up claims with (scientific) references.* ***This section is best written after the remainder of the report is finished.***

We are developing a multiscreen casting framework that supports a heterogeneous set of displays. The very first hurdle is the design of a communication protocol that connects all these devices. Devices are connected over the web via a client-server architecture. Clients connect to the server by loading a webpage in their browser. To establish a two-way communication channel, a connection is maintained through sockets [**?**].

Based on these principles we developed a basic multiscreen casting framework that offers the following functionality:

- The ability to transmit complex data such as images.
- Client-side access to the camera.
- Client identification such that clients can receive specific commands.
- Support of two roles, i.e., master and slave.

2 Design schematics and screenshots

(≈ 1 page. Depending on the need for a design overview.) *Design schematics, deployment diagrams, class diagrams, sequence diagrams, . . . Whatever you think we need to understand your design. You could add a little bit of text here but keep that under half a page. You reference the illustrations here from §3 when needed.*

All communication between clients runs through a nodejs server. We use the Socketio [**?**] library to set up a bidirectional communication channel between client and server. A deployment diagram can be found in Figure 1. Clients can take on a specific master or slave role by opening a specific webpage within the browser, i.e., the slaves will surf to `webaddress/slave`, while a client that wants to be the master loads the webpage found at `webaddress/master`.

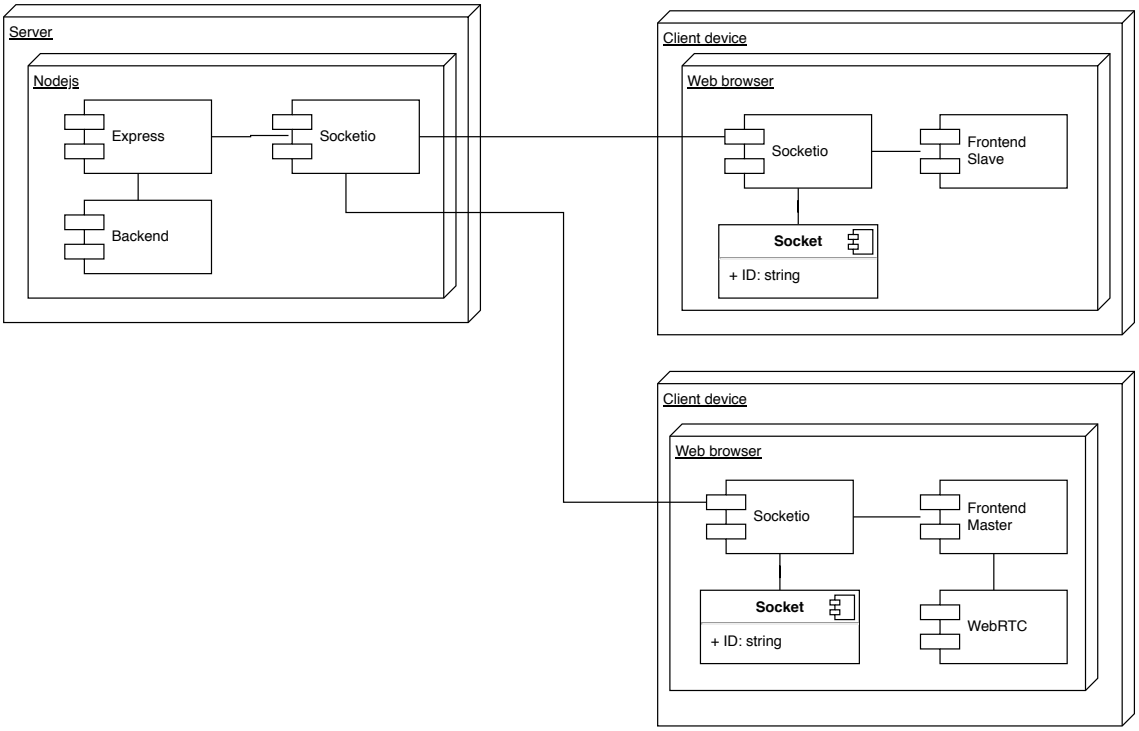

Figure 1: Deployment diagram of the multiscreen casting framework.

We rely on webrtc [**?**] to collect a client-side video stream. This module offers the necessary tools for serialization, i.e., as a base64 encoded string. Serialization is required for transmission [**?**]. Server-side the backend code will process the image. In our demo we broadcast the image by sending it to all slaves. Slaves and masters can be reached by the server, as we use the concept of a namespace [**?**]. A namespace can be used to group clients with a similar role, and broadcast to them.

A socket is assigned a unique identifier by Socketio. This identifier can be used to emit messages from the server to a specific client. A list of connected clients is available through the API of Socketio [**?**].

3 Algorithms

(≈ 1 page. Depending on necessity) *When you design or use an algorithm you should be able to explain how well it performs and if it satisfies the requirements. Often there will be alternatives or system parameters that need to be chosen. We want to know why you chose a specific set of parameters, or why you chose an approach over another. We supply some example text of last year.*

3.1 Scherm identificatie met barcodes

3.1.1 Algemene uitleg

Voor de verschillende schermen te identificeren wordt gebruik gemaakt van een kleuren barcode. De barcode bestaat uit een herhalend patroon van 5 unieke kleuren telkens gevolgd door een witte lijn. Door het gebruik van deze witte lijn weet het algoritme waar het patroon eindigt en de volgende sequentie terug opnieuw begint. De 5 kleuren zijn

speciaal gekozen om zo ver mogelijk van elkaar verwijderd te zijn in het HSL spectrum. Op deze manier is de kans op foute detectie zo klein mogelijk gemaakt. Voor de identificatie van de slaves wordt er dus een unieke combinatie van deze 5 kleuren weergegeven. Deze vormt dan een unieke vijfcijferige code die gelinkt kan worden aan de bijhorende slave. Dit zorgt ervoor dat we in theorie een totaal van 5! (= 120) verschillende schermen op één moment kunnen detecteren. Herhaling van het patroon heeft als resultaat dat bij overlap het scherm nog steeds geïdentificeerd kan worden. Het algoritme zal van links naar rechts over de pixels, die zich in het midden van het scherm bevinden, itereren (complexiteit N). Vervolgens worden de HSL waarden van deze pixels bekeken om de overeenkomstige kleur van elke pixel te achterhalen. Wanneer een HSL waarde binnen de gewenste range valt, wordt het overeenkomstig cijfer opgeslaan in een lijst. Bij het bereiken van een witte lijn weet het algoritme dat het aan het einde van het patroon is. Wanneer dit het geval is wordt het inlezen van de volledige vijfcijferige code gecontroleerd op volledigheid. Zo niet wordt de lijst leeg gehaald en zoekt het algoritme verder. Het herhalend patroon is dus essentieel aan het correct inlezen van de barcode. Bijgevolg werd het aantal herhalingen van het patroon bekeken in een aantal testen. Uit deze testen werd een aantal van 7 herhalingen als optimaal resultaat bekomen. Een groter aantal herhalingen stemt overeen met meer kansen op een mogelijke detectie, maar stemt ook overeen met een kleinere oppervlakte per herhaling. Deze kleinere oppervlakte is dan weer nadelig voor detectie.

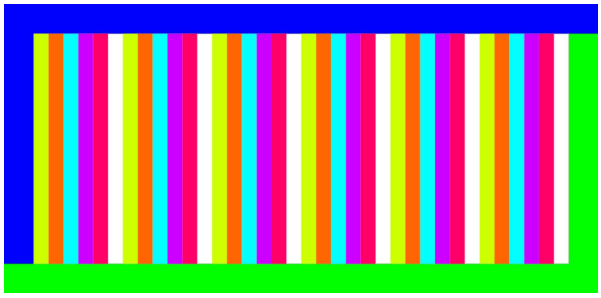

Figure 2: Voorbeeld van een barcode met 6 herhalingen.

3.1.2 Data set

Tijdens de testfase van het algoritme werden een aantal foto's genomen. Deze foto's verschilden in afstand tot het scherm, het scherm zelf (kleuren kunnen afwijken van scherm tot scherm) en het aantal herhalingen van het patroon (barcode).

3.1.3 Experiments

3.2 Image

3.2.1 Algemene uitleg

3.2.2 Data set

3.2.3 Experiments

4 Conclusion and prospects

(≈ 1/4 page.) *This section contains your evaluation of the development. How well did you succeed to complete this task? What can be improved?*

We designed a basic framework for screen casting. We offer all the functionality demanded in task 2. We achieved our goal by relying on basic socket.io functionality, as well as built-in features of front facing html5 technology such as webrtc. We did not discuss in detail the sending of commands to change background colors etc. We offer all functionality for this problem, the core challenge here was socket identification.

In the current state of our project multiple masters can coexist. We do not see this as a problem at the moment. From a security perspective this can be problematic depending on the actual usage scenario of the final application.

A User licenses

For every software package and all images/illustrations/pictures used we want to see that you have looked up its user license and comply with it. Please provide a list here. This may also be a good location to write about what software license you are thinking for your finished project.