

A  
Project Report On  
**Pico-Projector based Automation**

*Project-2 (2181105)*

Bachelor of Engineering  
in  
Electronics & Communication Engineering

By

***Chaitanya Tejaswi (140080111013)***

Under The Guidance of  
**Dr. Bhargav Goradiya**  
HoD, EC Department.



ELECTRONICS & COMMUNICATION ENGINEERING  
DEPARTMENT  
BVM ENGINEERING COLLEGE  
GUJARAT TECHNOLOGICAL UNIVERSITY  
VALLABH VIDYANAGAR-388120  
Academic Year- 2017-18



# Index

Sr. No.	Topic	Pages
1	Acknowledgement	3
2	Completion Certificate	5
3	Plagiarism Certificate	7
4	Introduction <ul style="list-style-type: none"> <li>• Pico-Projector Technology</li> <li>• Pico-Projector for Embedded Systems</li> </ul>	9 – 14
5	Application: Classroom Automation <ul style="list-style-type: none"> <li>• Raspberry Pi as Server</li> <li>• Real Time Communication using UV4L and WebRTC Standards</li> </ul>	15 – 32
6	Source Code Listing	33 – 70
7	References	71
8	Appendices <ul style="list-style-type: none"> <li>• Periodic Progress Report (PPR): 1-4</li> <li>• AEIOU Canvas</li> <li>• BMC Canvas</li> <li>• BMC Report</li> <li>• Patent Drafting Exercise</li> </ul>	72 +



# Acknowledgement

I am extremely thankful to all my teachers, friends and fellow classmates for their continual support and assistance.

Chaitanya Tejaswi





# GUJARAT TECHNOLOGICAL UNIVERSITY

CERTIFICATE FOR COMPLETION OF ALL ACTIVITIES AT ONLINE PROJECT PORTAL

B.E. SEMESTER VIII, ACADEMIC YEAR 2017-2018

Date of certificate generation : 11 May 2018 (18:03:12)

This is to certify that, *Chaitanya Roop Tejaswi* ( Enrolment Number - 140080111013 ) working on project entitled with *Pico-projector based Automation* from *Electronics & Communication Engineering* department of *BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR* had submitted following details at online project portal.

Periodic Progress Reports (PPR)	Completed
Business Model Canvas (Image)	Completed
Business Model Canvas (Report)	Completed
Patent Drafting Exercise (PDE)	Completed
Final Plagiarism Report	Completed
Final Project Report	Completed

Name of Student : Chaitanya Roop Tejaswi

Name of Guide : HOD\_008\_11

Signature of Student : \_\_\_\_\_

\*Signature of Guide : \_\_\_\_\_

**Disclaimer :**

This is a computer generated copy and does not indicate that your data has been evaluated. This is the receipt that GTU has received a copy of the data that you have uploaded and submitted as your project work.

\*Guide has to sign the certificate, Only if all above activities has been Completed.





# Plagiarism Certificate

PlagScan Results of plagiarism analysis from 2018-05-05 09:05 UTC

Project2\_Report (2181105).docx

14.5%

Date: 2018-05-05 09:03 UTC

\* All sources 53 | 🌐 Internet sources 21

✓ [42]  <https://www.quora.com/How-can-I-control-...s-not-locally-hosted>  
2.0% 1 matches

☒ [43]  [https://www.reddit.com/user/its\\_bananas](https://www.reddit.com/user/its_bananas)  
**1.9%** 4 matches

☒ [44]  <https://www.raspberrypi.org/forums/viewtopic.php?p=1193375>  
2.0% 2 matches


☒ [45]  [https://www.reddit.com/r/raspberry\\_pi/co...om\\_anywhere\\_without/](https://www.reddit.com/r/raspberry_pi/comments/9m0anywhere_without/)  
1.8% 4 matches


✓ [46] [https://yotekai.com/q\\_vnc\\_server/](https://yotekai.com/q_vnc_server/)  
1.6% 4 matches

☑ [47]  <https://gist.github.com/Lauszus/c8ce73f3177d6455c27c>  
1.9% 7 matches

☒ [48]  [https://github.com/jhmoon1211/CATMINT/blob/master/install\\_UV4L.txt](https://github.com/jhmoon1211/CATMINT/blob/master/install_UV4L.txt)  
1.5% 4 matches

☒ [49]  <https://www.raspberrypi.org/forums/viewtopic.php?t=187358>  
0.9% 4 matches

☒ [50]  <https://www.linux-projects.org/uv4l/tutorials/desktop-streaming/>  
0.7% 3 matches


☒ [51]  comlogins.net/vnc-viewer 0.7% 1 matches

☒ [52]  [raspberrypi2maray.blogspot.com/2016/11/tentang-vnc-di-raspberry-pi.html](http://raspberrypi2maray.blogspot.com/2016/11/tentang-vnc-di-raspberry-pi.html)  
0.7% 2 matches

☒ [53]  <https://github.com/Microsoft/Windows-iot...xpander/CS/README.md>  
0.4% 2 matches


☒ [54]  <https://support.realvnc.com/Knowledgebase...n-realvnc-viewer-app>  
0.5% 1 matches

☒ [55] 0.4% 1 matches  
⊕ 2 documents with identical matches


☒ [58]  <https://www.youtube.com/watch?v=A0pXZHAaj3w>  
0.4% 1 matches

✓ [59]  [https://www.reddit.com/r/raspberry\\_pi/co...e\\_a\\_project\\_idea\\_in/](https://www.reddit.com/r/raspberry_pi/comments/9a7p0e/a_project_idea_in/)  
0.3% 1 matches

☒ [60]  <https://www.raspberrypi.org/forums/viewtopic.php?p=899866>  
0.3% 1 matches

✓ [61]  <https://www.raspberrypi.org/products/raspberry-pi-zero>  
0.3% 1 matches

☒ [62]  [https://www.ibm.com/developerworks/bpm/1...oy/1206\\_mcelroy.html](https://www.ibm.com/developerworks/bpm/1...oy/1206_mcelroy.html)  
0.2% 1 matches

✓ [63]  <https://superuser.com/questions/194090/how-to-get-am-on-the-start-menu>  
0.2% 1 matches

☒ [64]  <https://superuser.com/a/1151028>  
0.2% 1 matches

15 pages, 2711 words

⚠ A very light text-color was detected that might conceal letters used to merge words.

PlagLevel: selected / overall

142 matches from 65 sources, of which 65 are online sources.

## Settings

Data policy: *Compare with web sources, Check against my documents*

Sensitivity: *Medium*



# Introduction

## What is a Pico-Projector?

A pico projector is a small hardware device designed to project content from a smartphone, camera, tablet, notebook or memory device onto a wall or other flat surface.



## Applications of a Pico-Projector?



Mobile device projection



Pico projectors



Wearable displays



Smart home displays



Digital signage



Mobile smart TVs

## DLP Technology for Pico-Projectors

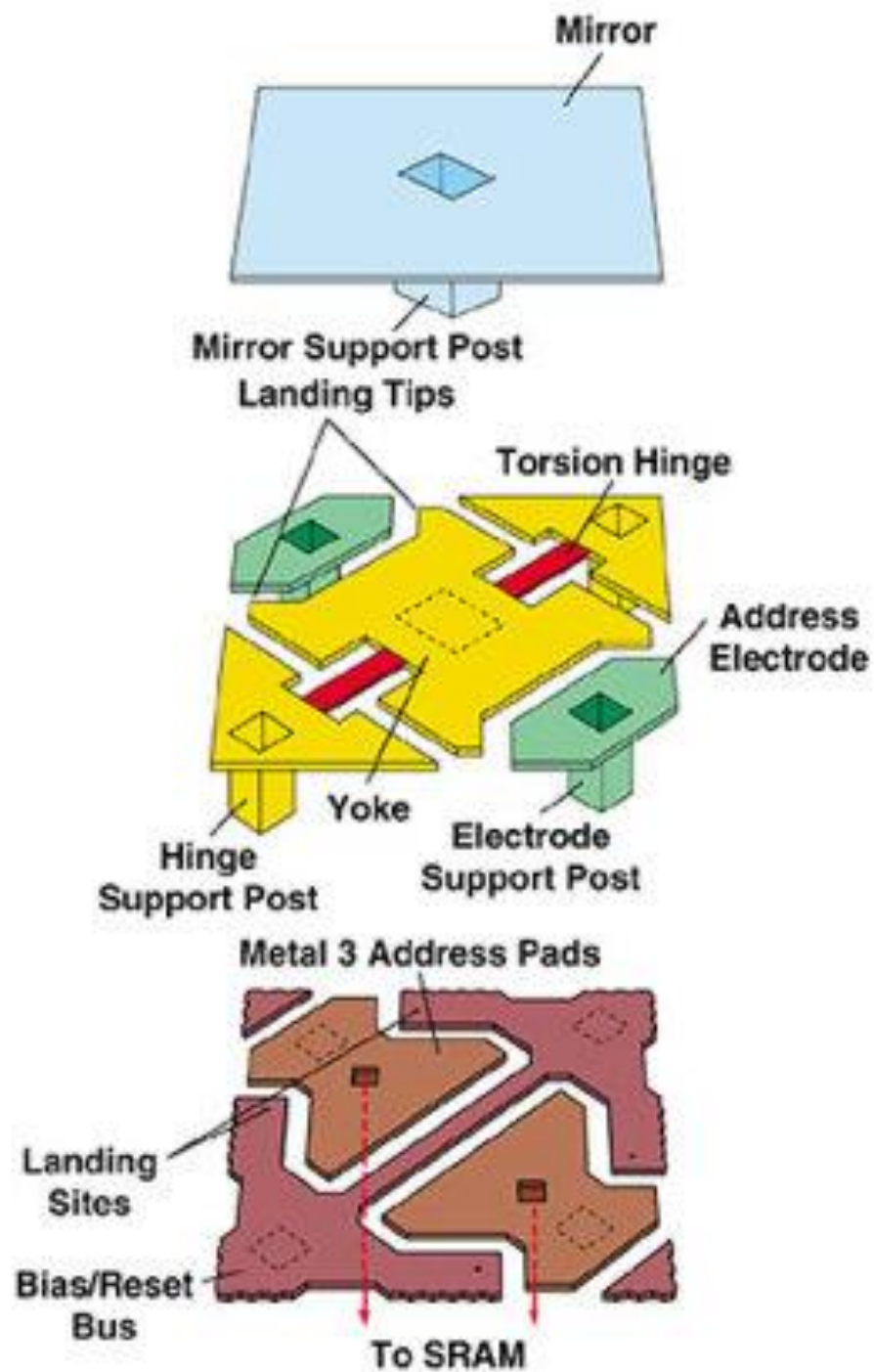
**Digital Light Processing (DLP)** technology is based on an optical semiconductor, called a **Digital Micromirror Device (DMD)**, which uses mirrors made of aluminum to reflect light to make the picture. The DMD is often referred to as the DLP chip.

If you look closely at a DMD, you would see tiny square mirrors that reflect light, instead of the tiny photographs. From far away (or when the light is projected on the screen), you would see a picture.

In addition to the mirrors, the DMD unit includes:

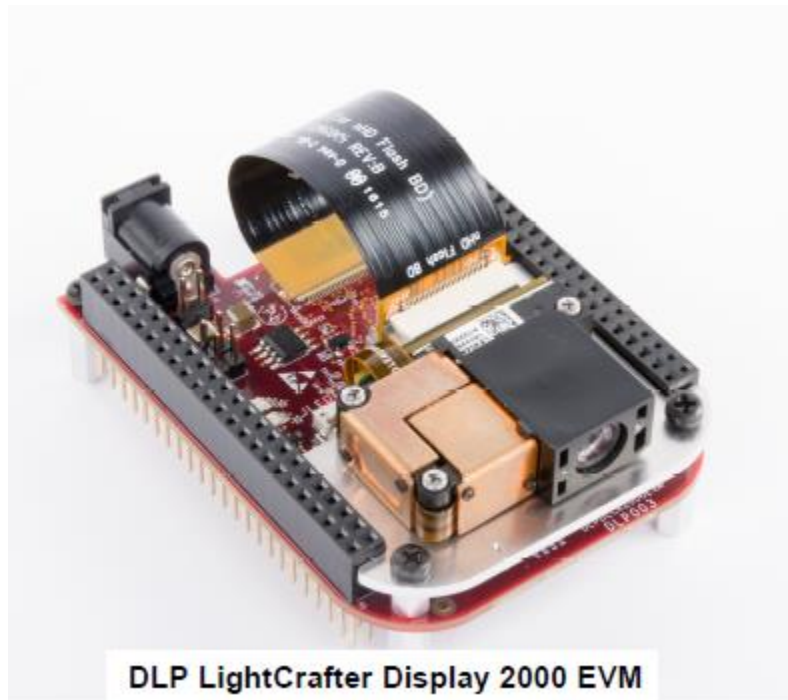
- A CMOS DDR SRAM chip, which is a memory cell that will electrostatically cause the mirror to tilt to the on or off position, depending on its logic value (0 or 1).
- A heat sink.
- An optical window, which allows light to pass through while protecting the mirrors from dust and debris.

A DMD chip is a **micro electro-mechanical system (MEMS)**. MEMS devices combine microscopic machines with the same silicon used in a computer.



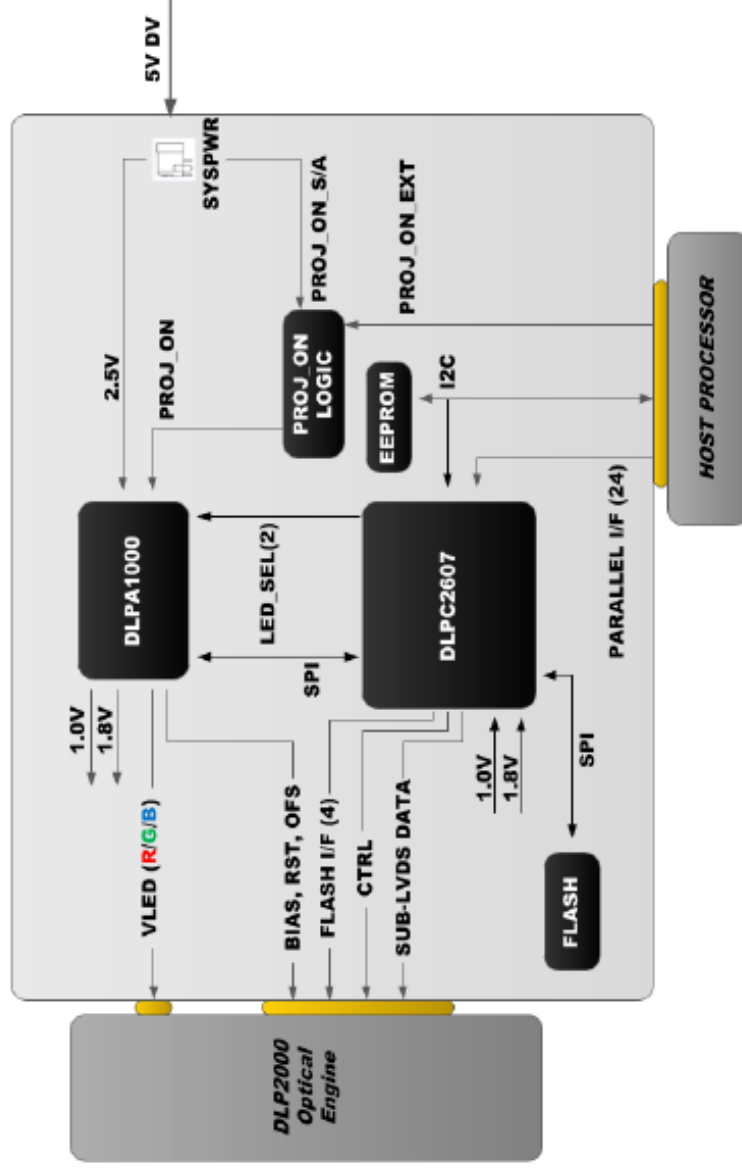
### DMD Architecture

## DLPDLCREVM2000 Pico-Projector Module



The DLP LightCrafter Display 2000EVM is a low-cost platform enabling the use of DLP technology with embedded host processors

The small, compact design enables quick implementation and demonstration in embedded applications.



**DLP LightCrafter Display 2000 EVM Block Diagram**

## External Interface Overview

This EVM communicates with the outside world using a pair of GPIO connectors designated P1 and P2. These connectors can be interfaced with the following sources:

- Parallel video driver (using adapter board)
- Host processor (via direct GPIO)

Some examples of host processors that can be used with this device include the BeagleBone Black and Raspberry Pi. A breakdown of the pinouts on these two connector ports can be found in [Figure 13](#).

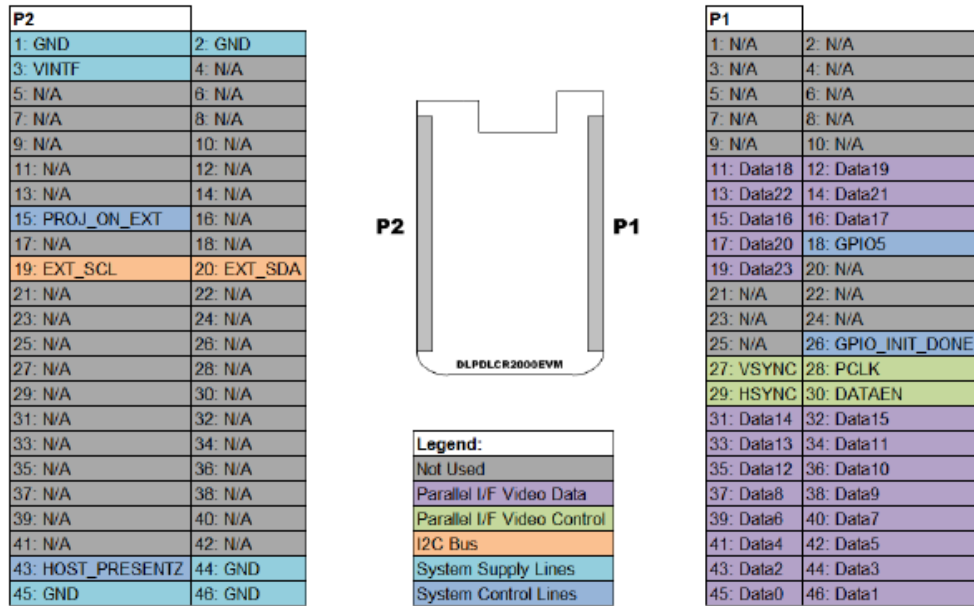


Figure 13 EVM Pinout Diagram



# Application: Classroom Automation

## Abstract:

The aim of this project is to automate everyday tasks with the help of a Pico-projector module.

## Tasks:

Remote Desktop Access (SSH/VNC)

Remote Desktop Sharing

Remote Desktop Mirroring (UV4L)

PDF Presenter #1 (Manual)

- \* PDF presentation with interactive support. To be used by teacher manually.

PDF Presenter #2 (Automatic)

- \* Added Voice-over using TTS support. Subtitles (lecture notes) have to be provided.

PDF Presenter #3 (Lab Assistant Support)

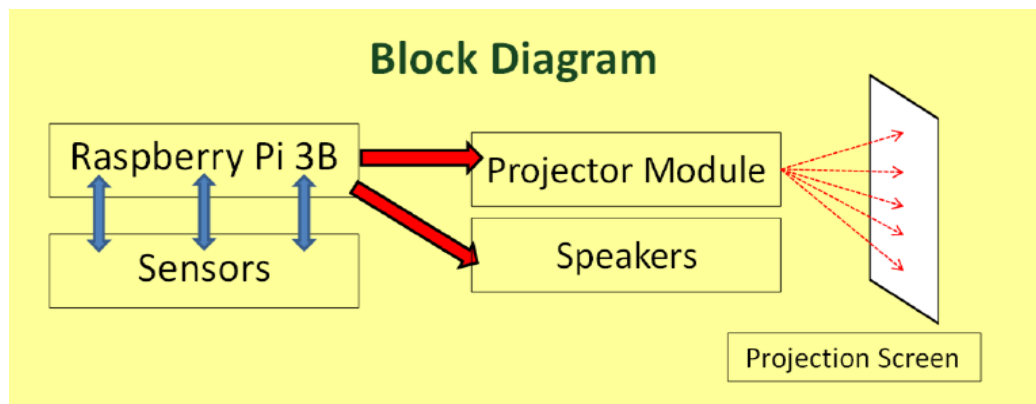
- \* Equipment identification & content retrieval using QR codes.

PDF Presenter #4 (Programming Support)

- \* Using Jupyter Notebooks & Firefox Browser addons

PDF Presenter #5 (Classroom Support)

- \* Student attendance logging, multi-user interaction, test paper evaluation verification.



# Raspberry Pi as Server (NGINX)

NGINX (pronounced *engine x*) is a popular lightweight web server application you can install on the Raspberry Pi to allow it to serve web pages.

Like Apache, NGINX can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

- **Install NGINX**

First install the `nginx` package by typing the following command in to the Terminal:

```
sudo apt-get install nginx
```

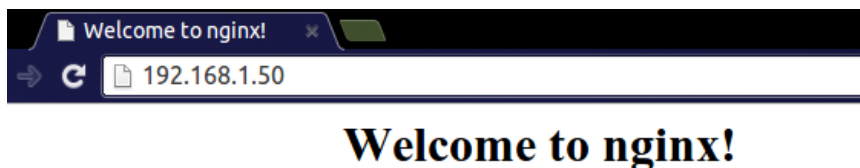
and start the server with:

```
sudo /etc/init.d/nginx start
```

- **Test the web server**

By default, NGINX puts a test HTML file in the web folder. This default web page is served when you browse to `http://localhost/` on the Pi itself, or `http://192.168.1.10` (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type `hostname -I` at the command line (or read more about finding your [IP address](#)).

Browse to the default web page either on the Pi or from another computer on the network and you should see the following:



- **Changing the default web page**

NGINX defaults its web page location to `/var/www/html` on Raspbian. Navigate to this folder and edit or replace `index.nginx-debian.html` as you like. You can confirm the default page location at `/etc/nginx/sites-available` on the line which starts with 'root', should you need to.

# Real Time Communication using UV4L and WebRTC Standards

## UserSpace Video4Linux (UV4L)

UV4L was originally conceived as a modular collection of Video4Linux2-compliant, cross-platform, user space drivers for real or virtual video input and output devices (with absolutely no external difference from kernel drivers).

While still preserving the original intentions, UV4L has evolved over the years and now optionally includes a generic purpose Streaming Server plug-in, especially made for IoT devices, that can serve custom web applications that can make use of a number of standard and modern built-in services for Real-Time Communications such as encrypted, bidirectional data channels, audio and video streaming or conferencing over the web.

UV4L also provides a RESTful API for the developers who want to implement their own custom applications.

uv4l core

uv4l-server

uv4l-raspicam

uv4l-uvc

uv4l-mjpegstream

uv4l-raspidisp

uv4l-xscreen

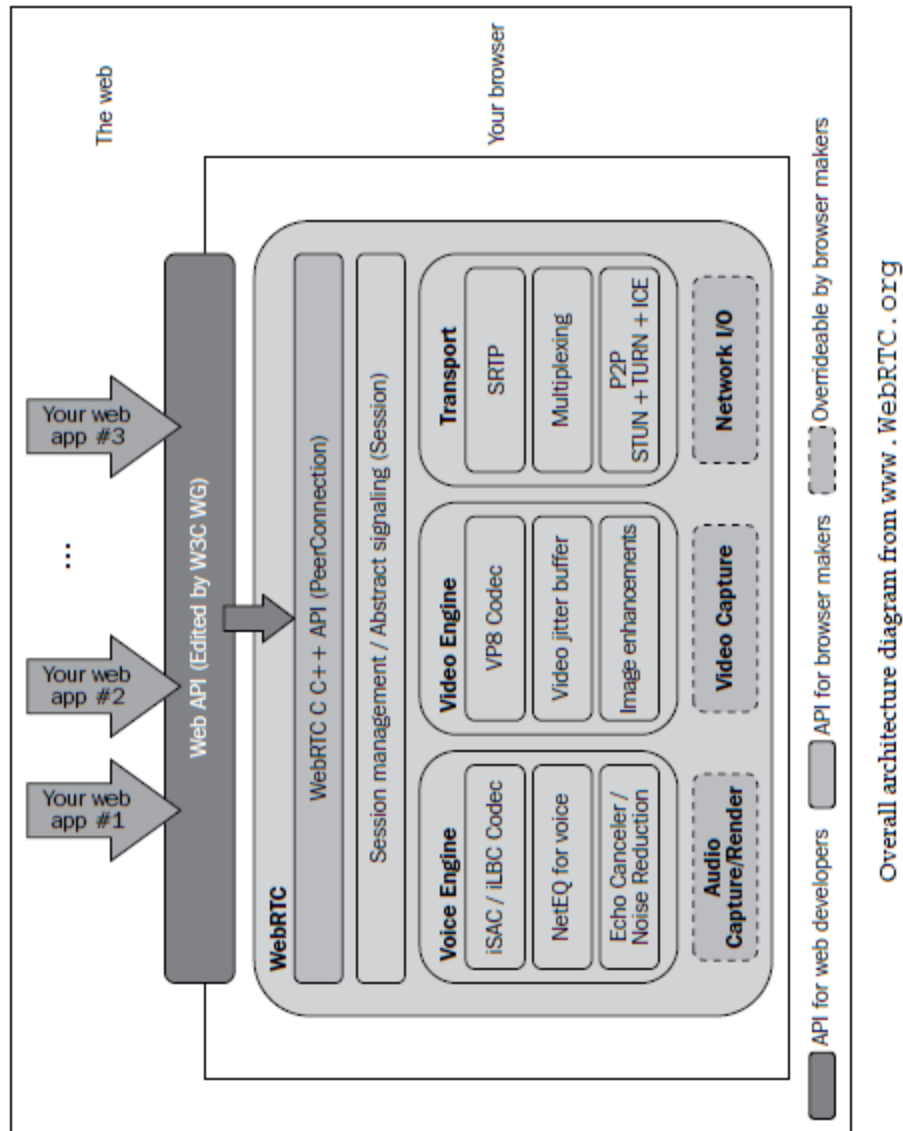
uv4l-dummy

## WebRTC

WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs.

It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps.





## Remote Desktop Access

Sometimes you need to access a Raspberry Pi without connecting it to a monitor. Perhaps the Pi is embedded in something like a robot, or you may want to view some information from it from elsewhere. Maybe you simply don't have a spare monitor! You can find more information on accessing your Raspberry Pi remotely [here](#).

- [IP address](#)
  - How to find your Raspberry Pi's IP address in order to connect to it
- [Access over Internet](#)
  - Remote access to the Pi over the internet using Weaved or Dataplicity
- [VNC](#)

- Remote access to the Pi's graphical interface, viewed in a window on another computer
- [SSH](#)
  - Access the command line of the Pi from another computer
- [SFTP](#)
  - Copy files between your Pi and another computer using SFTP (Secure File Transfer Protocol)
- [SCP](#)
  - Copy files between your Pi and another computer using SCP (Secure Copy Protocol)
- [SSHFS](#)
  - Copy files between your Pi and another computer using SSHFS (Secure Shell Filesystem)
- [rsync](#)
  - Synchronise folders between the Pi and another computer using rsync over SSH
- [FTP](#)
  - Copy files between your Pi and another computer using FTP (File Transfer Protocol)
- [Web Server](#)
  - Set up a website or a web page to display some information about the Pi, using a web browser on another machine, on the network or on the internet.

### VNC (Virtual Network Computing)

Sometimes it is not convenient to work directly on the Raspberry Pi. Maybe you would like to work on it from another device by remote control.

VNC is a graphical desktop sharing system that allows you to remotely control the desktop interface of one computer (running VNC Server) from another computer or mobile device (running VNC Viewer). VNC Viewer transmits the keyboard and either mouse or touch events to VNC Server, and receives updates to the screen in return.

You will see the desktop of the Raspberry Pi inside a window on your computer or mobile device. You'll be able to control it as though you were working on the Raspberry Pi itself.



VNC Connect from RealVNC is included with Raspbian. It consists of both VNC Server, which allows you to control your Raspberry Pi remotely, and VNC Viewer, which allows you to control desktop computers remotely from your Raspberry Pi should you want to.

You must enable VNC Server before you can use it: instructions for this are given below. By default, VNC Server gives you remote access to the graphical desktop that is running on your Raspberry Pi, as though you were sitting in front of it.

However, you can also use VNC Server to gain graphical remote access to your Raspberry Pi if it is headless or not running a graphical desktop. For more information on this, see [Creating a virtual desktop](#), further below.

### Enabling VNC Server

On your Raspberry Pi, run the following commands to make sure you have the latest version of VNC Connect:

```
sudo apt-get update
sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer
```

Now enable VNC Server. You can do this graphically or at the command line.

#### Enabling VNC Server graphically

On your Raspberry Pi, boot into the graphical desktop.

Select **Menu > Preferences > Raspberry Pi Configuration > Interfaces**.

Ensure VNC is Enabled.

#### Enabling VNC Server at the command line

You can enable VNC Server at the command line using `raspi-config`:

```
sudo raspi-config
```

Now, enable VNC Server by doing the following:

Navigate to **Interfacing Options**.

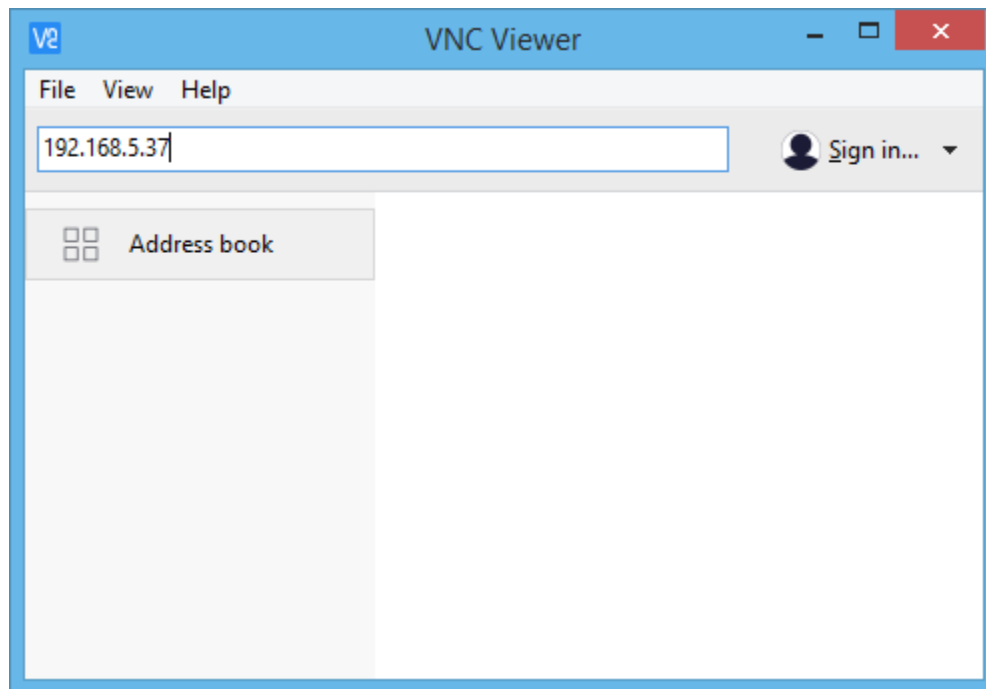
Scroll down and select **VNC > Yes**.

### Connecting to your Raspberry Pi with VNC Viewer

There are two ways to connect to your Raspberry Pi. You can use either or both, depending on what works best for you.

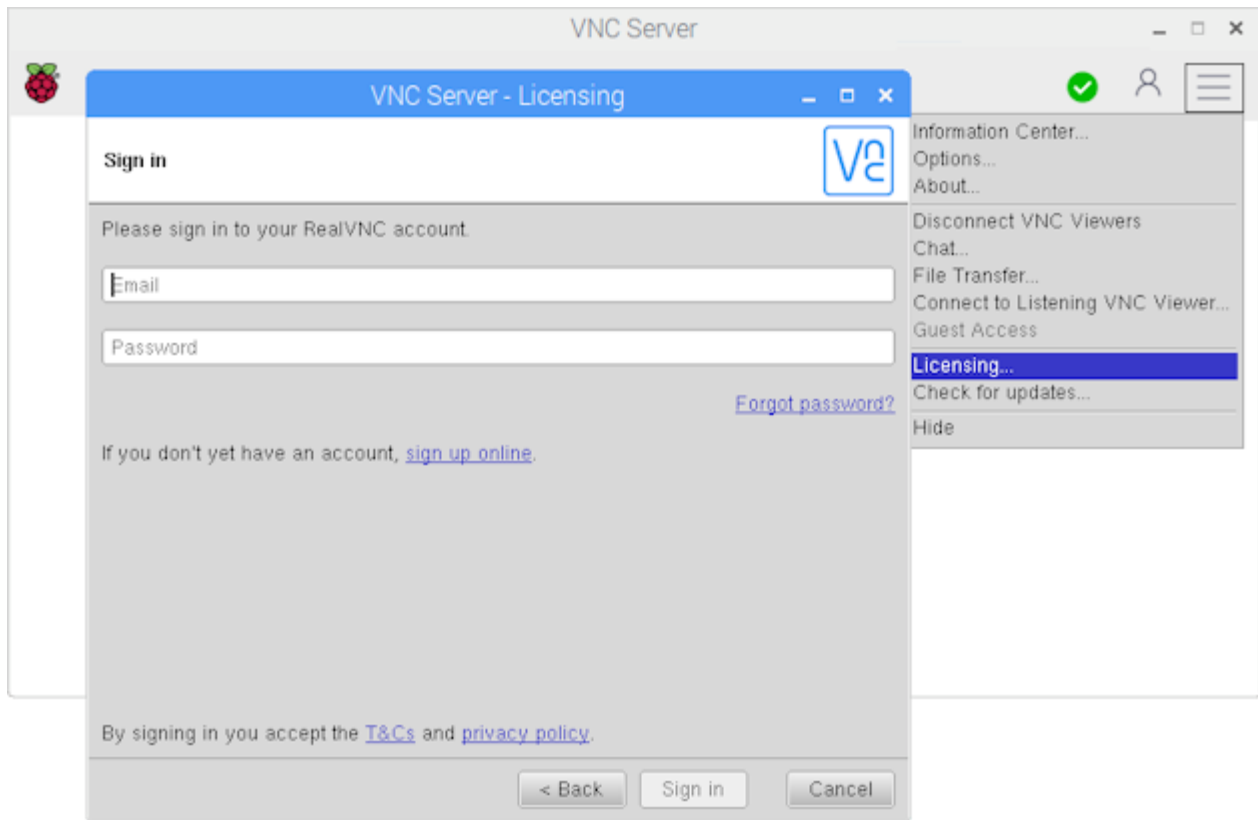
#### *Establishing a direct connection*

1. Direct connections are quick and simple providing you're joined to the same private local network as your Raspberry Pi. For example, this might be a wired or wireless network at home, at school, or in the office).
2. On your Raspberry Pi (using a terminal window or via SSH) use these instructions or run `ifconfig` to discover your private IP address.
3. On the device you'll use to take control, download VNC Viewer. For best results, use the compatible app from RealVNC.
4. Enter your Raspberry Pi's private IP address into VNC Viewer:



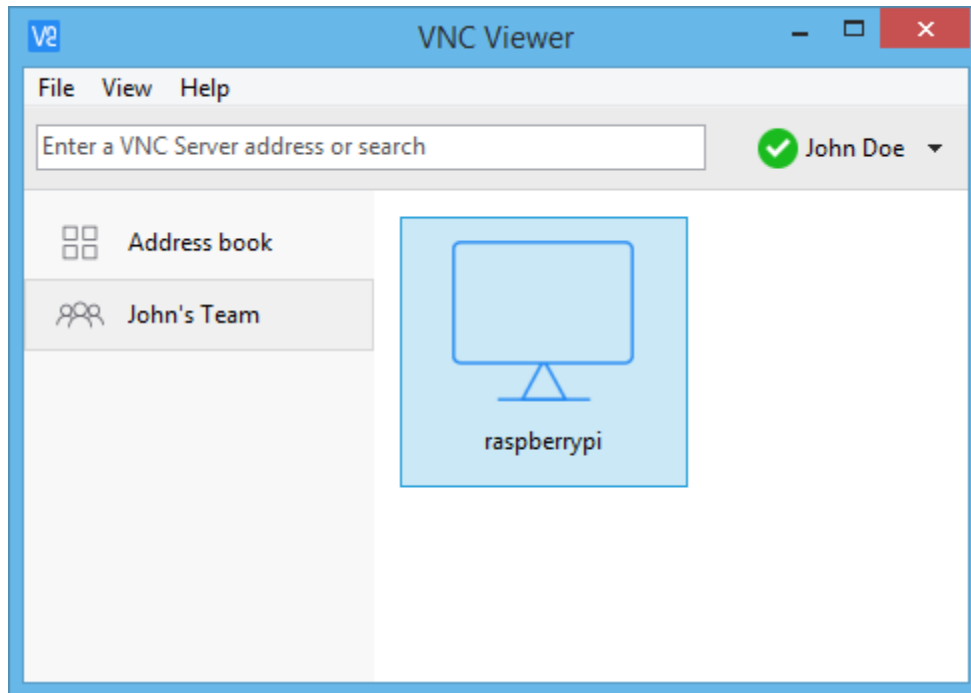
### ***Establishing a cloud connection***

1. You are entitled to use RealVNC's cloud service for free, provided that remote access is for educational or non-commercial purposes only.
2. Cloud connections are convenient and encrypted end-to-end. They are highly recommended for connecting to your Raspberry Pi over the internet. There's no firewall or router reconfiguration, and you don't need to know the IP address of your Raspberry Pi, or provide a static one.
3. Sign up for a RealVNC account here: it's free and it only takes a few seconds.
4. On your Raspberry Pi, sign in to VNC Server using your new RealVNC account credentials:



5. On the device you'll use to take control, download VNC Viewer. You must use the compatible app from RealVNC.
6. Sign in to VNC Viewer using the same RealVNC account credentials, and then either tap or click to connect to your Raspberry Pi:





### ***Authenticating to VNC Server***

1. To complete either a direct or cloud connection, you must authenticate to VNC Server.
2. If you're connecting from the compatible VNC Viewer app from RealVNC, enter the user name and password you normally use to log in to your user account on the Raspberry Pi. By default, these credentials are pi and raspberry.
3. If you're connecting from a non-RealVNC Viewer app, you'll first need to downgrade VNC Server's authentication scheme, specify a password unique to VNC Server, and then enter that instead.
4. If you are in front of your Raspberry Pi and can see its screen, open the VNC Server dialog on your Raspberry Pi, select Menu > Options > Security, and choose VNC password from the Authentication dropdown.
5. Or if you're configuring your Raspberry Pi remotely from the command line, then to make the changes for Service Mode (the default configuration for the Raspberry Pi):
  - Open the `/root/.vnc/config.d/vncserver-x11` config file.
  - Replace `Authentication=SystemAuth` with `Authentication=VncAuth` and save the file.
  - In the command line, run `sudo vncpasswd -service`. This will prompt you to set a password, and will insert it for you in the right config file for VNC Server running in Service Mode.
  - Restart VNC Server.

## Remote Desktop Sharing

The core of the UV4L software falls in the *middleware* category. It consists of a series of highly configurable *drivers*, an optional *Streaming Server* component providing a [RESTful API](#) for custom development and various *extensions* for the server that cooperate together. The Streaming Server also provides the necessary *web UI* for the end-users to try or use all the key functionalities directly. For maximum efficiency, each instance of UV4L runs as a single system process which exploits the underlying hardware natively whenever possible. Here is a more detailed [list of features](#).

Below we will see how to install all the components to get the best from UV4L, with particular focus on the driver for the Raspberry Pi camera boards for the sake of explanation, although all other drivers can be optionally installed in a similar way.

This same driver has been extended to support the **TC358743 HDMI to MIPI chipset converter** on all Raspberry Pi boards (this chipset is present on the [B101 HDMI to CSI-2 Bridge](#), for example). However, instructions on how to enable the TC358743 on boards different from *Raspberry Pi 3* (e.g. *Zero*, *ZeroW*, *CM3*, etc...) will be provided [upon request](#) only.

If you are running *Raspbian Wheezy* or *Raspbian Jessie*, open a terminal and type the following commands:

```
$ curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc | sudo apt-key add -
```

On *Raspbian Jessie* add this line:

```
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ jessie main
```

Finally, we are ready to update the system and to fetch and install the packages:

```
$ sudo apt-get update
$ sudo apt-get install uv4l uv4l-raspicam
```

The above two commands will upgrade UV4L to the most recent version, if it's already installed.

If you want the driver to be loaded at boot, also install this optional package:

```
$ sudo apt-get install uv4l-raspicam-extras
```

As a convenience, the above package will install a service script for starting, stopping or restarting the driver at any time, for example:

```
$ sudo service uv4l_raspicam restart
```

When (re)starting the service, *uv4l* will be instructed to parse the configuration file */etc/uv4l/uv4l-raspicam.conf* to get the default values for the driver and the server options. You can edit that file to add, remove or change the default values. This same service is started at boot.

If you are using the TC358743, the *uv4l-tc358743-extras* package has to be installed for it to work:

```
$ sudo apt-get install uv4l-tc358743-extras
```

The above package will automatically install the *uv4l-raspicam-extras* package and some other helper programs. Before using the TC358743 make sure that both the *Camera* interface and the *I2C* bus are enabled in the *raspi-config* system tool and that the line *tc358743=yes* is present or uncommented in the configuration file */etc/uv4l/uv4l-raspicam.conf*.

Now the UV4L core component and the Video4Linux2 driver for the CSI Camera Board are installed. If you occasionally get unexpected errors from the driver, make sure the camera is enabled and enough memory is reserved for the *GPU* (not less than 128MB is suggested) from this menu:

```
$ sudo raspi-config
```

Also consider updating the firmware with the following command:

```
$ sudo rpi-update
```

For detailed informations, options, etc... about the modules installed type accordingly:

```
$ man uv4l
$ man uv4l-raspicam
```

To get the list of all available options:

```
$ uv4l --help --driver raspicam --driver-help
```

If you have not installed the optional *uv4l-raspicam-extras* package (which provides a convenient script for starting *uv4l* with the settings taken from a configuration file) and want to quickly test *uv4l*, load it manually:

```
$ uv4l --driver raspicam --auto-video_nr --width 640 --height 480 --encoding
jpeg
```

and take a JPEG snapshot from the camera:

```
$ dd if=/dev/video0 of=snapshot.jpeg bs=11M count=1
```

For a list of other use cases click [here](#).

To manually terminate a running driver, close all the applications accessing the device and kill the corresponding *uv4l* process:

```
$ pkill uv4l
```

Apart from the driver for the Raspberry Pi Camera Board, the following Streaming Server front-end and drivers can be optionally installed:

```
$ sudo apt-get install uv4l-server uv4l-uvc uv4l-xscreen uv4l-mjpegstream  
uv4l-dummy uv4l-raspisp
```

for which the manual pages are available:

```
$ man uv4l-server  
$ man uv4l-uvc  
$ man uv4l-xscreen  
$ man uv4l-mjpegstream  
$ man uv4l-dummy  
$ man uv4l-raspisp
```

The *WebRTC* extension for the Streaming Server is also available with two alternative packages depending on the Raspberry Pi model in use. If you have a *Raspberry Pi 1*, *Compute Module 1*, *Zero* or *Zero W* (Wireless), type:

```
$ sudo apt-get install uv4l-webrtc-armv6
```

Otherwise, if you have any other model (e.g. Raspberry Pi 2 or 3), type:

```
$ sudo apt-get install uv4l-webrtc
```

As the Streaming Server is able to serve and run any custom web applications, an optional package containing the source code of some of the demos mentioned in the examples (e.g. [this one](#)) is available. The files will be installed in */usr/share/uv4l/demos/*:

```
$ sudo apt-get install uv4l-demos
```

Note that some browsers may no longer exploit many of the WebRTC functionalities over HTTP for security reasons. You will need to configure secure *HTTPS* in the Streaming Server instead. To do this, you must provide a **password-less private key** and a valid **certificate** via the *-ssl-private-key-file* and the *-ssl-certificate-file* server options. A private key and a self-signed certificate can be generated as follows:

```
$ openssl genrsa -out selfsign.key 2048 && openssl req -new -x509 -key  
selfsign.key -out selfsign.crt -sha256
```

Once you have installed and eventually configured the HTTP(S) Streaming Server module as shown above, make sure to reload *uv4l* for it to notice and start the server. Afterwards you can

access the server with the browser at the default address and port *https://raspberrypi:8080/* (where *raspberrypi* has to be replaced with the actual hostname or IP address of your RaspberryPi and the protocol can be either *http* or *https*).

## Screen and audio mirroring to the Raspberry Pi

In this example we will see how to mirror the PC or laptop desktop's screen and audio-in (e.g. microphone) to the Raspberry Pi display and speakers with Firefox or Chrome. No special software or browser plug-in needs to be installed on the PC or laptop as a web application with the basic functionalities is provided by the *UV4L Streaming Server* and gets downloaded by the browser by simply visiting the appropriate URL. As opposite to many other screen sharing applications, no *X Server* is required on the Raspberry Pi, as UV4L makes use of the display natively.

It's of course possible for you to customize or install your own screen sharing web application, but this is out of scope now (see [this](#) example if you are interested).

### Requirements:

1. download and install the *UV4L* software on your Raspberry Pi. For the purposes of this simple example, you will only really need the following packages: ***uv4l***, ***uv4l-dummy***, ***uv4l-server***, ***uv4l-webRTC***. Please refer to the [installation instructions](#).
2. **HTTPS** must be enabled in the *UV4L Streaming Server* for the browser to share the desktop. To do this, you will need a *SSL private key* and a valid *SSL certificate*. If you do not have them already, you can generate them by running the following command on the Raspberry Pi:

```
sudo bash -c "openssl genrsa -out /etc/ssl/private/selfsign.key 2048 &&
openssl req -new -x509 -key /etc/ssl/private/selfsign.key -out
/etc/ssl/private/selfsign.crt -sha256"
```

3. Open Firefox on your PC or laptop, type ***about:config*** in the address bar and press *enter*. In the **Search** field, type ***media.getusermedia.screensharing.enabled*** and toggle its value to ***true*** with a double-click. Now search for ***media.getusermedia.screensharing.allowed\_domains*** and add your Raspberry Pi *hostname* or *IP address* or domain name to the list in the string (e.g. *192.168.1.2*).

Now you are ready. For example, assuming you have a Full-HD display attached to the Raspberry Pi, open a terminal on the Raspberry Pi and run a new instance of *uv4l* with a command like this (in one line):

```
uv4l --driver dummy --auto-video_nr --enable-server --server-option '--use-ssl=yes' --server-option '--ssl-private-key-file=/etc/ssl/private/selfsign.key' --server-option '--ssl-certificate-file=/etc/ssl/private/selfsign.crt' --server-option '--enable-webrtc-video=no' --server-option '--enable-webrtc-audio=no' --server-option '--webrtc-receive-video=yes' --server-option '--webrtc-receive-audio=yes' --server-option '--webrtc-renderer-window=0 0 1920 1080' --server-option '--webrtc-renderer-fullscreen=yes' --server-option '--webrtc-receive-datachannels=no' --server-option '--port=9000'
```

At this point the UV4L Streaming Server should be running and listening to the HTTPS port 9000 for incoming connections. Open a tab in Firefox and enter the URL of the web page that allows you to mirror the desktop screen (or even a just window) and optionally cast the microphone, i.e. `https://raspberrypi:9000/stream/webrtc`, where “raspberrypi” has to be replaced with the hostname or IP address of your Raspberry Pi in the network (on the first access you may be asked to trust the self-signed certificate if you have generated them as shown above).

The web page should look like this:

---

**WebRTC two-way Audio/Video/Data Intercom & Recorder**

remote

local

Pause/Resume
Mute/Unmute
Fullscreen
Start Recording

**Remote peer options**

Video ☐ force use of hardware codec for 1280x720 30 fps, 4kpbs, min:800 max:4000 start:1200 ▼

NOTE: if your browser does not support the hardware codec yet, try Firefox with the codec plugin enabled or a recent version of Chrome.

▶ **Recorded Audio/Video stream**

**Cast local Audio/Video sources to remote peer**

Audio ☐ microphone/other input ☐ echo cancellation

Video ☐ camera ☐ screen ☐ window ☐ application

NOTE: except for camera, to enable screen, window or application casting open about:config URL and set media.getusermedia.screensharing.enabled to true and permanently add the current domain to the list in media.getusermedia.screensharing.allowed\_domains.

NOTE: if you want to cast music, for better audio quality disable echo-cancellation, and aec, noise-suppression, agc in the browser configuration (about:config).

▶ **Data Channels**

▼ **Advanced options**

Remote Peer/Signalling Server Address 192.168.1.2:9000

Optional ICE Servers (STUN/TURN):

Call
Hangup

[home control panel](#)

To

start casting all have to do is to check the two “Audio: microphone/other input” and the “Video: screen” checkboxes under the “Cast local Audio/Video sources to remote peer” section and click on the green button “Call”. A browser pop-up will then appear asking you to select the screen and audio input device and to allow sharing the media sources:


[home control panel](#)



## Summary of Steps

### # OS Installation

#### \* Basic Setup

sudo raspi-config

\* Set Username-Password, Interfacing options (Enable SSH,VNC,Camera,Remote-GPIO), Timezone, Permissions, etc.

#### \* Configure WiFi - Select Network, then find RPi3B's IP using:

ifconfig

#### \* (optional) Time & Date Synchronization

sudo apt-get install ntpdate

sudo timedatectl set-ntp True

reboot

#### \* Update & Upgrade installed packages

sudo apt-get update && sudo apt-get dist-upgrade

reboot

### # Installation for ARM (Raspberry Pi) - Raspbian\_Stretch OS

#### \* Add UV4L to package listing

curl http://www.linux-projects.org/listing/uv4l\_repo/lpkey.asc | sudo apt-key add -

sudo nano /etc/apt/sources.list

#### \* Append this line

deb http://www.linux-projects.org/listing/uv4l\_repo/raspbian/stretch stretch main

sudo apt-get update

#### \* Fetch & Install packages

sudo apt-get install uv4l uv4l-raspicam uv4l-raspicam-extras

reboot

#### \*Update RPi3B firmware

sudo apt-get install rpi-update

sudo rpi-update

reboot

#### \* Test: UV4L Quick Test - JPEG snapshot from camera

uv4l --driver raspicam --auto-video\_nr --width 640 --height 480 --encoding jpeg

dd if=/dev/video0 of=snapshot.jpeg bs=11M count=1

#### \* Test: Manually terminate a running driver. But first close all relevant running applications

pkill uv4l

### \*\* Streaming over WebRTC

#### \* Streaming Server Frontend & Drivers

sudo apt-get install uv4l-server uv4l-uvic uv4l-xscreen uv4l-mjpegstream uv4l-dummy uv4l-raspidisp

uv4l-raspidisp-extras

- \* Download WebRTC Server Extension package  
`sudo apt-get install uv4l-webrtc`
- \* Download demos to /usr/share/uv4l/demos/  
`sudo apt-get install uv4l-demos`
- \* Module to broadcast AV content to users of a Jitsi-Meet conference  
`sudo apt-get install uv4l-xmpp-bridge`
- \* Configure HTTPS communication by generating Private Key & Self-Signed Certificate  
`sudo openssl genrsa -out selfsign.key 2048 && openssl req -new -x509 -key selfsign.key -out selfsign.crt -sha256`
- \* Test: Server can be accessed by browser at `https://<RPI-IP-or-hostname>:8080/`

## # Screen & Audio Mirroring

[Firefox]

- \* Enable Screensharing & Add RPi3B's IP to the list

`about:config`

`media.getusermedia.screensharing.enabled`

`media.getusermedia.screensharing.allowed_domains`

[RPi3B]

- \* Run UV4L Streaming Server - Listening at Port 9000 (assuming HDMI output)

```
uv4l --driver dummy --auto-video_nr --enable-server --server-option '--use-ssl=yes' --server-option '--ssl-private-key-file=/etc/ssl/private/selfsign.key' --server-option '--ssl-certificate-file=/etc/ssl/private/selfsign.crt' --server-option '--enable-webrtc-video=no' --server-option '--enable-webrtc-audio=no' --server-option '--webrtc-receive-video=yes' --server-option '--webrtc-receive-audio=yes' --server-option '--webrtc-renderer-window=0 0 1920 1080' --server-option '--webrtc-renderer-fullscreen=yes' --server-option '--webrtc-receive-datachannels=no' --server-option '--port=9000'
```

- \*\* Use 1024x576, ...

[Firefox]

- \* Cast desktop screen from browser

`https://<RPI-IP-or-hostname>:9000/stream/webrtc`

- \* Configure AV inputs

Audio: microphone/other input

Video: screen

[RPi3B]

- \* Terminate the server driver

`pkill uv4l`

## # Video Calling

`appr.tc`

## Source Code Listing

- **Basic Audio Call**
- **Video Call with Chat**
- **Video Chat with File Sharing**
- **Face Detection using OpenCV.js**
- **UV4L Streaming Server Applet**



## 01 – Basic Audio Call

```
<!DOCTYPE html>
<html>
<head>
<script>

var webrtc_capable = true;
var rtc_peer_connection = null;
var rtc_session_description = null;
var get_user_media = null;
var connect_stream_to_src = null;
var stun_server = "stun.l.google.com:19302";

if (navigator.getUserMedia) { // WebRTC 1.0 standard compliant browser
  rtc_peer_connection = RTCPeerConnection;
  rtc_session_description = RTCSessionDescription;
  get_user_media = navigator.getUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    // https://www.w3.org/Bugs/Public/show_bug.cgi?id=21606
    media_element.srcObject = media_stream;
    media_element.play();
  };
} else if (navigator.mozGetUserMedia) { // early firefox webrtc implementation
  rtc_peer_connection = mozRTCPeerConnection;
  rtc_session_description = mozRTCSessionDescription;
  get_user_media = navigator.mozGetUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    media_element.mozSrcObject = media_stream;
    media_element.play();
  };
  stun_server = "74.125.31.127:19302";
} else if (navigator.webkitGetUserMedia) { // early webkit webrtc implementation
  rtc_peer_connection = webkitRTCPeerConnection;
  rtc_session_description = RTCSessionDescription;
  get_user_media = navigator.webkitGetUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    media_element.src = webkitURL.createObjectURL(media_stream);
  };
} else {
  alert("This browser does not support WebRTC - visit WebRTC.org for more info");
  webrtc_capable = false;
}
</script>
```

```

<script>

var call_token; // unique token for this call
var signaling_server; // signaling server for this call
var peer_connection; // peer connection object

function start() {
  // create the WebRTC peer connection object
  peer_connection = new rtc_peer_connection({ // RTCPeerConnection configuration
    "iceServers": [ // information about ice servers
      { "url": "stun:"+stun_server }, // stun server info
    ]
  });

  // generic handler that sends any ice candidates to the other peer
  peer_connection.onicecandidate = function (ice_event) {
    if (ice_event.candidate) {
      signaling_server.send(
        JSON.stringify({
          type: "new_ice_candidate",
          candidate: ice_event.candidate ,
        })
      );
    }
  };

  // play remote audio streams when they arrive using local <audio> MediaElement
  peer_connection.onaddstream = function (event) {
    connect_stream_to_src(event.stream, document.getElementById("remote_audio"));
    // hide placeholder and show remote audio interface
    document.getElementById("loading_state").style.display = "none";
    document.getElementById("open_call_state").style.display = "block";
  };

  // setup stream from the local microphone
  setup_audio();

  // setup generic connection to the signaling server using the WebSocket API
  signaling_server = new WebSocket("ws://192.168.22.58:1234");

  if (document.location.hash === "" || document.location.hash === undefined) { // you are the Caller

    // create the unique token for this call
    var token = Date.now()+"-"+Math.round(Math.random()*10000);
    call_token = "#"+token;
  }
}

```

```

// set location.hash to the unique token for this call
document.location.hash = token;

signaling_server.onopen = function() {
  // setup caller signal handler
  signaling_server.onmessage = caller_signal_handler;

  // tell the signaling server you have joined the call
  signaling_server.send(
    JSON.stringify({
      token:call_token,
      type:"join",
    })
  );
}

document.title = "You are the Caller";
document.getElementById("loading_state").innerHTML = "Ready for a call...ask your friend to
visit:<br/><br/>" + document.location;

} else { // you have a hash fragment so you must be the Callee

  // get the unique token for this call from location.hash
  call_token = document.location.hash;

  signaling_server.onopen = function() {
    // setup caller signal handler
    signaling_server.onmessage = callee_signal_handler;

    // tell the signaling server you have joined the call
    signaling_server.send(
      JSON.stringify({
        token:call_token,
        type:"join",
      })
    );

    // let the caller know you have arrived so they can start the call
    signaling_server.send(
      JSON.stringify({
        token:call_token,
        type:"callee_arrived",
      })
    );
  }
}

```

```

    document.title = "You are the Callee";
    document.getElementById("loading_state").innerHTML = "One moment please...connecting your
call...";
}

}

/* functions used above are defined below */

// handler to process new descriptions
function new_description_created(description) {
    peer_connection.setLocalDescription(
        description,
        function () {
            signaling_server.send(
                JSON.stringify({
                    token:call_token,
                    type:"new_description",
                    sdp:description
                })
            );
        },
        log_error
    );
}

// handle signals as a caller
function caller_signal_handler(event) {
    var signal = JSON.parse(event.data);
    if (signal.type === "callee_arrived") {
        peer_connection.createOffer(
            new_description_created,
            log_error
        );
    } else if (signal.type === "new_ice_candidate") {
        peer_connection.addIceCandidate(
            new RTCIceCandidate(signal.candidate)
        );
    } else if (signal.type === "new_description") {
        peer_connection.setRemoteDescription(
            new rtc_session_description(signal.sdp),
            function () {
                if (peer_connection.remoteDescription.type == "answer") {
                    // extend with your own custom answer handling here
                }
            },
        ),
    }
}

```



```

        log_error
    );
} else {
    // extend with your own signal types here
}
}

// handle signals as a callee
function callee_signal_handler(event) {
    var signal = JSON.parse(event.data);
    if (signal.type === "new_ice_candidate") {
        peer_connection.addIceCandidate(
            new RTCIceCandidate(signal.candidate)
        );
    } else if (signal.type === "new_description") {
        peer_connection.setRemoteDescription(
            new rtc_session_description(signal.sdp),
            function () {
                if (peer_connection.remoteDescription.type == "offer") {
                    peer_connection.createAnswer(new_description_created, log_error);
                }
            },
            log_error
        );
    } else {
        // extend with your own signal types here
    }
}

// setup stream from the local microphone
function setup_audio() {
    get_user_media(
        {
            "audio": true, // request access to local microphone
            "video": false // don't request access to local camera
        },
        function (local_stream) { // success callback
            // add local microphone stream to peer_connection ready to be sent to the remote peer
            peer_connection.addStream(local_stream);
        },
        log_error
    );
}

// generic error handler
function log_error(error) {

```

```

    console.log(error);
}

</script>
<style>
html, body {
    padding: 0px;
    margin: 0px;
    font-family: "Arial", "Helvetica", sans-serif;
}
#loading_state {
    position: absolute;
    top: 45%;
    left: 0px;
    width: 100%;
    font-size: 20px;
    text-align: center;
}
#open_call_state {
    display: none;
}
#remote_audio {
    position: absolute;
    top: 0px;
    left: 0px;
    width: 1024px;
    height: 768px;
    background: #999999;
}
</style>
</head>
<body onload="start()">
    <div id="loading_state">
        loading...
    </div>
    <div id="open_call_state">
        <audio id="remote_audio"></audio>
    </div>
</body>
</html>

```

## 02 – Video Call with Chat

```
<!DOCTYPE html>
<html>
<head>
<script>

var webrtc_capable = true;
var rtc_peer_connection = null;
var rtc_session_description = null;
var get_user_media = null;
var connect_stream_to_src = null;
var stun_server = "stun.l.google.com:19302";

if (navigator.getUserMedia) { // WebRTC 1.0 standard compliant browser
  rtc_peer_connection = RTCPeerConnection;
  rtc_session_description = RTCSessionDescription;
  get_user_media = navigator.getUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    // https://www.w3.org/Bugs/Public/show_bug.cgi?id=21606
    media_element.srcObject = media_stream;
    media_element.play();
  };
} else if (navigator.mozGetUserMedia) { // early firefox webrtc implementation
  rtc_peer_connection = mozRTCPeerConnection;
  rtc_session_description = mozRTCSessionDescription;
  get_user_media = navigator.mozGetUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    media_element.mozSrcObject = media_stream;
    media_element.play();
  };
  stun_server = "74.125.31.127:19302";
} else if (navigator.webkitGetUserMedia) { // early webkit webrtc implementation
  rtc_peer_connection = webkitRTCPeerConnection;
  rtc_session_description = RTCSessionDescription;
  get_user_media = navigator.webkitGetUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    media_element.src = webkitURL.createObjectURL(media_stream);
  };
} else {
  alert("This browser does not support WebRTC - visit WebRTC.org for more info");
  webrtc_capable = false;
}
</script>
```

```

<script>

var call_token; // unique token for this call
var signaling_server; // signaling server for this call
var peer_connection; // peer connection object

function start() {
  // create the WebRTC peer connection object
  peer_connection = new rtc_peer_connection({ // RTCPeerConnection configuration
    "iceServers": [ // information about ice servers
      { "url": "stun:"+stun_server }, // stun server info
    ]
  });

  // generic handler that sends any ice candidates to the other peer
  peer_connection.onicecandidate = function (ice_event) {
    if (ice_event.candidate) {
      signaling_server.send(
        JSON.stringify({
          token: call_token,
          type: "new_ice_candidate",
          candidate: ice_event.candidate ,
        })
      );
    }
  };

  // display remote video streams when they arrive using local <video> MediaElement
  peer_connection.onaddstream = function (event) {
    connect_stream_to_src(event.stream, document.getElementById("remote_video"));
    // hide placeholder and show remote video
    document.getElementById("loading_state").style.display = "none";
    document.getElementById("open_call_state").style.display = "block";
  };

  // setup stream from the local camera
  setup_video();

  // setup generic connection to the signaling server using the WebSocket API
  signaling_server = new WebSocket("ws://localhost:1234");

  if (document.location.hash === "" || document.location.hash === undefined) { // you are the Caller

    // create the unique token for this call
    var token = Date.now()+"-"+Math.round(Math.random()*10000);
    call_token = "#"+token;
  }
}

```

```

// set location.hash to the unique token for this call
document.location.hash = token;

signaling_server.onopen = function() {
  // setup caller signal handler
  signaling_server.onmessage = caller_signal_handler;

  // tell the signaling server you have joined the call
  signaling_server.send(
    JSON.stringify({
      token:call_token,
      type:"join",
    })
  );
}

document.title = "You are the Caller";
document.getElementById("loading_state").innerHTML = "Ready for a call...ask your friend to
visit:<br/><br/>" + document.location;

} else { // you have a hash fragment so you must be the Callee

  // get the unique token for this call from location.hash
  call_token = document.location.hash;

  signaling_server.onopen = function() {
    // setup caller signal handler
    signaling_server.onmessage = callee_signal_handler;

    // tell the signaling server you have joined the call
    signaling_server.send(
      JSON.stringify({
        token:call_token,
        type:"join",
      })
    );

    // let the caller know you have arrived so they can start the call
    signaling_server.send(
      JSON.stringify({
        token:call_token,
        type:"callee_arrived",
      })
    );
  }
}

```

```

    document.title = "You are the Callee";
    document.getElementById("loading_state").innerHTML = "One moment please...connecting your
call...";
}

// setup message bar handlers
document.getElementById("message_input").onkeydown = send_chat_message;
document.getElementById("message_input").onfocus = function() { this.value = ""; }
}

/* functions used above are defined below */

// handler to process new descriptions
function new_description_created(description) {
    peer_connection.setLocalDescription(
        description,
        function () {
            signaling_server.send(
                JSON.stringify({
                    token:call_token,
                    type:"new_description",
                    sdp:description
                })
            );
        },
        log_error
    );
}

// handle signals as a caller
function caller_signal_handler(event) {
    var signal = JSON.parse(event.data);
    if (signal.type === "callee_arrived") {
        peer_connection.createOffer(
            new_description_created,
            log_error
        );
    } else if (signal.type === "new_ice_candidate") {
        peer_connection.addIceCandidate(
            new RTCIceCandidate(signal.candidate)
        );
    } else if (signal.type === "new_description") {
        peer_connection.setRemoteDescription(
            new rtc_session_description(signal.sdp),
            function () {

```

```

        if (peer_connection.remoteDescription.type == "answer") {
            // extend with your own custom answer handling here
        }
    },
    log_error
);
} else if (signal.type === "new_chat_message") {
    add_chat_message(signal);
} else {
    // extend with your own signal types here
}
}

// handle signals as a callee
function callee_signal_handler(event) {
    var signal = JSON.parse(event.data);
    if (signal.type === "new_ice_candidate") {
        peer_connection.addIceCandidate(
            new RTCIceCandidate(signal.candidate)
        );
    } else if (signal.type === "new_description") {
        peer_connection.setRemoteDescription(
            new rtc_session_description(signal.sdp),
            function () {
                if (peer_connection.remoteDescription.type == "offer") {
                    peer_connection.createAnswer(new_description_created, log_error);
                }
            },
            log_error
        );
    } else if (signal.type === "new_chat_message") {
        add_chat_message(signal);
    } else {
        // extend with your own signal types here
    }
}

// add new chat message to messages list
function add_chat_message(signal) {
    var messages = document.getElementById("messages");
    var user = signal.user || "them";
    messages.innerHTML = user+": "+signal.message+"<br/>\n"+messages.innerHTML;
}

// send new chat message to the other browser
function send_chat_message(e) {

```

```

if (e.keyCode == 13) {
  var new_message = this.value;
  this.value = "";
  signaling_server.send(
    JSON.stringify({
      token: call_token,
      type: "new_chat_message",
      message: new_message
    })
  );
  add_chat_message({ user: "you", message: new_message });
}
}

// setup stream from the local camera
function setup_video() {
  get_user_media(
    {
      "audio": true, // request access to local microphone
      "video": true // request access to local camera
    },
    function (local_stream) { // success callback
      // display preview from the local camera & microphone using local <video> MediaElement
      connect_stream_to_src(local_stream, document.getElementById("local_video"));
      // add local camera stream to peer_connection ready to be sent to the remote peer
      peer_connection.addStream(local_stream);
    },
    log_error
  );
}

// generic error handler
function log_error(error) {
  console.log(error);
}

</script>
<style>
html, body {
  padding: 0px;
  margin: 0px;
  font-family: "Arial", "Helvetica", sans-serif;
}
#loading_state {
  position: absolute;
  top: 45%;

```



```
left: 0px;
width: 100%;
font-size: 20px;
text-align: center;
}
#open_call_state {
  display: none;
}
#local_video {
  position: absolute;
  top: 10px;
  left: 10px;
  width: 160px;
  height: 120px;
  background: #333333;
}
#remote_video {
  position: absolute;
  top: 0px;
  left: 0px;
  width: 1024px;
  height: 768px;
  background: #999999;
}
#chat {
  position: absolute;
  top: 0px;
  left: 1024px;
  height: 768px;
  width: 300px;
  background: #CCCCCC;
}
#messages {
  overflow: auto;
  position: absolute;
  top: 20px;
  left: 0px;
  height: 733px;
  width: 295px;
  padding-top: 15px;
  padding-left: 5px;
  font-size: 14px;
}
#message_input {
  position: absolute;
  top: 5px;
```

```
left: 5px;
height: 20px;
width: 280px;
padding-left: 5px;
background: #FFFFFF;
}
</style>
</head>
<body onload="start()">
  <div id="loading_state">
    loading...
  </div>
  <div id="open_call_state">
    <video id="remote_video"></video>
    <video id="local_video"></video>
    <div id="chat">
      <div id="messages"></div>
      <input type="text" id="message_input" value="Type here then hit enter..."></input>
    </div>
  </div>
</body>
</html>
```

### 03 – Video Chat with File Sharing

```
<!DOCTYPE html>
<html>
<head>
<script>

var webrtc_capable = true;
var rtc_peer_connection = null;
var rtc_session_description = null;
var get_user_media = null;
var connect_stream_to_src = null;
var stun_server = "stun.l.google.com:19302";

if (navigator.getUserMedia) { // WebRTC 1.0 standard compliant browser
  rtc_peer_connection = RTCPeerConnection;
  rtc_session_description = RTCSessionDescription;
  get_user_media = navigator.getUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    // https://www.w3.org/Bugs/Public/show_bug.cgi?id=21606
    media_element.srcObject = media_stream;
    media_element.play();
  };
} else if (navigator.mozGetUserMedia) { // early firefox webrtc implementation
  rtc_peer_connection = mozRTCPeerConnection;
  rtc_session_description = mozRTCSessionDescription;
  get_user_media = navigator.mozGetUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    media_element.mozSrcObject = media_stream;
    media_element.play();
  };
  stun_server = "74.125.31.127:19302";
} else if (navigator.webkitGetUserMedia) { // early webkit webrtc implementation
  rtc_peer_connection = webkitRTCPeerConnection;
  rtc_session_description = RTCSessionDescription;
  get_user_media = navigator.webkitGetUserMedia.bind(navigator);
  connect_stream_to_src = function(media_stream, media_element) {
    media_element.src = webkitURL.createObjectURL(media_stream);
  };
} else {
  alert("This browser does not support WebRTC - visit WebRTC.org for more info");
  webrtc_capable = false;
}
</script>
```

```

<script>

var call_token; // unique token for this call
var signaling_server; // signaling server for this call
var peer_connection; // peer connection object
var file_store = []; // shared file storage

function start() {
  // create the WebRTC peer connection object
  peer_connection = new rtc_peer_connection({ // RTCPeerConnection configuration
    "iceServers": [ // information about ice servers
      { "url": "stun:"+stun_server }, // stun server info
    ]
  });

  // generic handler that sends any ice candidates to the other peer
  peer_connection.onicecandidate = function (ice_event) {
    if (ice_event.candidate) {
      signaling_server.send(
        JSON.stringify({
          token: call_token,
          type: "new_ice_candidate",
          candidate: ice_event.candidate ,
        })
      );
    }
  };

  // display remote video streams when they arrive using local <video> MediaElement
  peer_connection.onaddstream = function (event) {
    connect_stream_to_src(event.stream, document.getElementById("remote_video"));
    // hide placeholder and show remote video
    document.getElementById("loading_state").style.display = "none";
    document.getElementById("open_call_state").style.display = "block";
  };

  // setup stream from the local camera
  setup_video();

  // setup generic connection to the signaling server using the WebSocket API
  signaling_server = new WebSocket("ws://localhost:1234");

  if (document.location.hash === "" || document.location.hash === undefined) { // you are the Caller

    // create the unique token for this call
    var token = Date.now()+"-"+Math.round(Math.random()*10000);

```

```

call_token = "#" + token;

// set location.hash to the unique token for this call
document.location.hash = token;

signaling_server.onopen = function() {
    // setup caller signal handler
    signaling_server.onmessage = caller_signal_handler;

    // tell the signaling server you have joined the call
    signaling_server.send(
        JSON.stringify({
            token: call_token,
            type: "join",
        })
    );
}

document.title = "You are the Caller";
document.getElementById("loading_state").innerHTML = "Ready for a call...ask your friend to
visit:<br/><br/>" + document.location;

} else { // you have a hash fragment so you must be the Callee

    // get the unique token for this call from location.hash
    call_token = document.location.hash;

    signaling_server.onopen = function() {
        // setup caller signal handler
        signaling_server.onmessage = callee_signal_handler;

        // tell the signaling server you have joined the call
        signaling_server.send(
            JSON.stringify({
                token: call_token,
                type: "join",
            })
        );

        // let the caller know you have arrived so they can start the call
        signaling_server.send(
            JSON.stringify({
                token: call_token,
                type: "callee_arrived",
            })
        );
    }
}

```

```

    }

    document.title = "You are the Callee";
    document.getElementById("loading_state").innerHTML = "One moment please...connecting your
call...";
}

// setup message bar handlers
document.getElementById("message_input").onkeydown = send_chat_message;
document.getElementById("message_input").onfocus = function() { this.value = ""; }

// setup file sharing
if (!(window.File && window.FileReader && window.FileList && window.Blob)) {
    document.getElementById("file_sharing").style.display = "none";
    alert("This browser does not support File Sharing");
} else {
    document.getElementById("file_add").onclick = click_file_input;
    document.getElementById("file_input").addEventListener("change", file_input, false);
    document.getElementById("open_call_state").addEventListener("dragover", drag_over, false);
    document.getElementById("open_call_state").addEventListener("drop", file_input, false);
}
}

/* functions used above are defined below */

// handler to process new descriptions
function new_description_created(description) {
    peer_connection.setLocalDescription(
        description,
        function () {
            signaling_server.send(
                JSON.stringify({
                    token:call_token,
                    type:"new_description",
                    sdp:description
                })
            );
        },
        log_error
    );
}

// handle signals as a caller
function caller_signal_handler(event) {
    var signal = JSON.parse(event.data);
    if (signal.type === "callee_arrived") {

```

```

peer_connection.createOffer(
    new_description_created,
    log_error
);
} else if (signal.type === "new_ice_candidate") {
    peer_connection.addIceCandidate(
        new RTCIceCandidate(signal.candidate)
    );
} else if (signal.type === "new_description") {
    peer_connection.setRemoteDescription(
        new rtc_session_description(signal.sdp),
        function () {
            if (peer_connection.remoteDescription.type == "answer") {
                // extend with your own custom answer handling here
            }
        },
        log_error
    );
} else if (signal.type === "new_chat_message") {
    add_chat_message(signal);
} else if (signal.type === "new_file_thumbnail_part") {
    store_file_part("thumbnail", signal.id, signal.part, signal.length, signal.data);
    if (file_store[signal.id].thumbnail.parts.length == signal.length) {
        document.getElementById("file_list").innerHTML =
get_file_div(signal.id)+document.getElementById("file_list").innerHTML;
        document.getElementById("file-img-"+signal.id).src = file_store[signal.id].thumbnail.parts.join("");
    }
} else if (signal.type === "new_file_part") {
    store_file_part("file", signal.id, signal.part, signal.length, signal.data);
    update_file_progress(signal.id, file_store[signal.id].file.parts.length, signal.length);
} else {
    // extend with your own signal types here
}
}

// handle signals as a callee
function callee_signal_handler(event) {
    var signal = JSON.parse(event.data);
    if (signal.type === "new_ice_candidate") {
        peer_connection.addIceCandidate(
            new RTCIceCandidate(signal.candidate)
        );
    } else if (signal.type === "new_description") {
        peer_connection.setRemoteDescription(
            new rtc_session_description(signal.sdp),
            function () {

```

```

        if (peer_connection.remoteDescription.type == "offer") {
            peer_connection.createAnswer(new_description_created, log_error);
        }
    },
    log_error
);
} else if (signal.type === "new_chat_message") {
    add_chat_message(signal);
} else if (signal.type === "new_file_thumbnail_part") {
    store_file_part("thumbnail", signal.id, signal.part, signal.length, signal.data);
    if (file_store[signal.id].thumbnail.parts.length == signal.length) {
        document.getElementById("file_list").innerHTML =
get_file_div(signal.id)+document.getElementById("file_list").innerHTML;
        document.getElementById("file-img-"+signal.id).src = file_store[signal.id].thumbnail.parts.join("");
    }
} else if (signal.type === "new_file_part") {
    store_file_part("file", signal.id, signal.part, signal.length, signal.data);
    update_file_progress(signal.id, file_store[signal.id].file.parts.length, signal.length);
} else {
    // extend with your own signal types here
}
}

// add new chat message to messages list
function add_chat_message(signal) {
    var messages = document.getElementById("messages");
    var user = signal.user || "them";
    messages.innerHTML = user+": "+signal.message+"<br/>\n"+messages.innerHTML;
}

// send new chat message to the other browser
function send_chat_message(e) {
    if (e.keyCode == 13) {
        var new_message = this.value;
        this.value = "";
        signaling_server.send(
            JSON.stringify({
                token:call_token,
                type: "new_chat_message",
                message: new_message
            })
        );
        add_chat_message({ user: "you", message: new_message });
    }
}

```



```

// setup stream from the local camera
function setup_video() {
  get_user_media(
    {
      "audio": true, // request access to local microphone
      "video": true // request access to local camera
    },
    function (local_stream) { // success callback
      // display preview from the local camera & microphone using local <video> MediaElement
      connect_stream_to_src(local_stream, document.getElementById("local_video"));
      // add local camera stream to peer_connection ready to be sent to the remote peer
      peer_connection.addStream(local_stream);
    },
    log_error
  );
}

// file sharing html template
function get_file_div(id) {
  return '<div id="file-'+id+'" class="file"><div id="file-progress-'+id+'"
class="file_progress"></div></div>';
}

// initiate manual file selection
function click_file_input(event) {
  document.getElementById('file_input').click();
}

// prevent window from reloading when file dragged into it
function drag_over(event) {
  event.stopPropagation();
  event.preventDefault();
}

// handle manual file selection or drop event
function file_input(event) {
  event.stopPropagation();
  event.preventDefault();
  var files = undefined;
  if (event.dataTransfer && event.dataTransfer.files !== undefined) {
    files = event.dataTransfer.files;
  } else if (event.target && event.target.files !== undefined) {
    files = event.target.files;
  }
  if (files.length > 1) {

```

```

    alert("Please only select one file at a time");
} else if (!files[0].type.match('image.*')) {
    alert("This demo only supports sharing image files");
} else if (files.length == 1) {
    var kb = (files[0].size/1024).toFixed(1);
    var new_message = "Sending file...<br><strong>" + files[0].name + "</strong> (" + kb + "KB)";
    signaling_server.send(
        JSON.stringify({
            token: call_token,
            type: "new_chat_message",
            message: new_message
        })
    );
    add_chat_message({ user: "you", message: new_message });
    document.getElementById("file_list").innerHTML =
    get_file_div(file_store.length) + document.getElementById("file_list").innerHTML;
    var reader = new FileReader();
    reader.onload = (function(file, id) {
        return function(event) {
            send_file(file.name, id, event.target.result);
        }
    })(files[0], file_store.length);
    reader.readAsDataURL(files[0]);
}
}

// send selected file
function send_file(name, file_id, data) {
    var default_width = 160;
    var default_height = 120;
    var img = document.getElementById("file_img_src");
    img.onload = function() {
        var image_width = this.width;
        var target_width = default_width;
        var image_height = this.height;
        var target_height = default_height;
        var top = 0;
        var left = 0;
        if (image_width > image_height) {
            var ratio = target_width/image_width;
            target_height = image_height*ratio;
            top = (default_height-target_height)/2;
        } else if (image_height > image_width) {
            var ratio = target_height/image_height;
            target_width = image_width*ratio;
            left = (default_width-target_width)/2;

```

```

    } else {
        left = (default_width-default_height)/2;
        target_width = target_height;
    }
    var canvas = document.getElementById("file_thumbnail_canvas");
    canvas.width = default_width;
    canvas.height = default_height;
    var cc = canvas.getContext("2d");
    cc.clearRect(0,0,default_width,default_height);
    cc.drawImage(img, left, top, target_width, target_height);
    var thumbnail_data = canvas.toDataURL("image/png");
    document.getElementById("file-img-"+file_id).src = thumbnail_data;
    send_file_parts("thumbnail", file_id, thumbnail_data);
    send_file_parts("file", file_id, data);
}
img.src = data;
}

// break file into parts and send each of them separately
function send_file_parts(type, id, data) {
    var message_type = "new_file_part";
    if (type == "thumbnail") {
        message_type = "new_file_thumbnail_part";
    }
    var slice_size = 1024;
    var parts = data.length/slice_size;
    if (parts % 1 > 0) {
        parts = Math.round(parts)+1;
    }
    for (var i = 0; i < parts; i++) {
        var from = i*slice_size;
        var to = from+slice_size;
        var data_slice = data.slice(from, to);
        store_file_part(type, id, i, parts, data_slice);
        signaling_server.send(
            JSON.stringify({
                token:call_token,
                type: message_type,
                id: id,
                part: i,
                length: parts,
                data: data_slice
            })
        );
    }
}

```

```

// store individual file parts in the local file store
function store_file_part(type, id, part, length, data) {
  if (file_store[id] === undefined) {
    file_store[id] = {};
  }
  if (file_store[id][type] === undefined) {
    file_store[id][type] = {
      parts: []
    };
  }
  if (file_store[id][type].length === undefined) {
    file_store[id][type].length = length;
  }
  file_store[id][type].parts[part] = data;
}

// show the progress of a file transfer
function update_file_progress(id, parts, length) {
  var percentage = Math.round((parts/length)*100);
  if (percentage < 100) {
    document.getElementById("file-progress-"+id).innerHTML = percentage+"%";
    document.getElementById("file-img-"+id).style.opacity = 0.25;
  } else {
    document.getElementById("file-progress-"+id).innerHTML = "";
    document.getElementById("file-img-"+id).style.opacity = 1;
  }
}

// show the full file
function display_file(event) {
  var match = event.target.id.match("file-img-(.*)");
  var file = file_store[match[1]].file;
  if (file.parts.length < file.length) {
    alert("Please wait - file still transferring");
  } else {
    window.open(file.parts.join(""));
  }
}

// generic error handler
function log_error(error) {
  console.log(error);
}

</script>

```

```

<style>
html, body {
  padding: 0px;
  margin: 0px;
  font-family: "Arial", "Helvetica", sans-serif;
}
#loading_state {
  position: absolute;
  top: 45%;
  left: 0px;
  width: 100%;
  font-size: 20px;
  text-align: center;
}
#open_call_state {
  display: none;
}
#local_video {
  position: absolute;
  top: 10px;
  left: 10px;
  width: 160px;
  height: 120px;
  background: #333333;
}
#remote_video {
  position: absolute;
  top: 0px;
  left: 0px;
  width: 1024px;
  height: 768px;
  background: #999999;
}
#chat {
  position: absolute;
  top: 0px;
  left: 1024px;
  height: 768px;
  width: 300px;
  background: #CCCCCC;
}
#messages {
  overflow: auto;
  position: absolute;
  top: 20px;
  left: 0px;

```

```
height: 733px;
width: 295px;
padding-top: 15px;
padding-left: 5px;
font-size: 14px;
}
#message_input {
  position: absolute;
  top: 5px;
  left: 5px;
  height: 20px;
  width: 280px;
  padding-left: 5px;
  background: #FFFFFF;
}
#file_sharing {
  position: absolute;
  top: 140px;
  left: 10px;
  height: 628px;
  width: 160px;
}
#file_input {
  display: none;
}
#file_add {
  position: absolute;
  padding: 0px;
  height: 120px;
  width: 160px;
  text-align: center;
}
#file_list {
  overflow: auto;
  position: absolute;
  padding: 0px;
  top: 130px;
  height: 488px;
  width: 180px;
}
.file {
  width: 160px;
  height: 120px;
  padding: 0px;
  padding-bottom: 10px;
  margin: 0px;
```

```

}
.file_img {
  width: 160px;
  height: 120px;
}
.file_progress {
  color: #333333;
  font-family: "Helvetica", "Arial", sans-serif;
  font-size: 30px;
  font-weight: bold;
  position: relative;
  text-align: center;
  top: -80px;
  width: 160px;
}
#file_img_src {
  display: none;
}
#file_thumbnail_canvas {
  display: none;
  width: 160px;
  height: 120px;
}
</style>
</head>
<body onload="start()">
  <div id="loading_state">
    loading...
  </div>
  <div id="open_call_state">
    <video id="remote_video"></video>
    <video id="local_video"></video>
    <div id="chat">
      <div id="messages"></div>
      <input type="text" id="message_input" value="Type here then hit enter..."></input>
    </div>
    <div id="file_sharing">
      <input type="file" id="file_input"></input>
      <div id="file_add">
        
      </div>
      <div id="file_list">
      </div>
      <img id="file_img_src" />
      <canvas id="file_thumbnail_canvas"></canvas>
    </div>

```

```
</div>  
</body>  
</html>
```



## 04 – Face Detection with OpenCV.js

### 1. facedetect.js

```
var faceCascade;

showImage = function (mat, canvas) {
    var data = mat.data(); // output is a Uint8Array that aliases directly into the Emscripten heap
    var channels = mat.channels();
    var channelSize = mat.elemSize1();
    var ctx = canvas.getContext('2d');

    ctx.clearRect(0, 0, c.width, c.height);
    var imdata = ctx.createImageData(mat.cols, mat.rows);

    for (var i = 0, j = 0; i < data.length; i += channels, j += 4) {
        imdata.data[j] = data[i];
        imdata.data[j + 1] = data[i + 1 % channels];
        imdata.data[j + 2] = data[i + 2 % channels];
        imdata.data[j + 3] = 255;
    }

    ctx.putImageData(imdata, 0, 0);
};

function detectFace(canvas) {
    if (!faceCascade) {
        console.log("Creating the Face cascade classifier");
        faceCascade = new cv.CascadeClassifier();
        faceCascade.load('../test/data/haarcascade_frontalface_default.xml');
    }

    var ctx = canvas.getContext('2d');
    var input = ctx.getImageData(0, 0, canvas.width, canvas.height);
    var img = cv.matFromArray(input, 24); // 24 for rgba
    var imgGray = new cv.Mat();
    var imgColor = new cv.Mat(); // Opencv likes RGB

    cv.cvtColor(img, imgGray, cv.ColorConversionCodes.COLOR_RGBA2GRAY.value, 0);
    cv.cvtColor(img, imgColor, cv.ColorConversionCodes.COLOR_RGBA2RGB.value, 0);

    var faces = new cv.RectVector();
    var s1 = [50, 50];
    var s2 = [0, 0];
    faceCascade.detectMultiScale(imgGray, faces, 1.3, 4, 0, s1, s2);

    for (var i = 0; i < faces.size(); i += 1) {
```

```

    var faceRect = faces.get(i);
    fx = faceRect.x;
    fy = faceRect.y;
    fw = faceRect.width;
    fh = faceRect.height;
    var p1 = [fx, fy];
    var p2 = [fx + fw, fy + fh];
    var color = new cv.Scalar(255, 0, 0);
    cv.rectangle(imgColor, p1, p2, color, 2, 8, 0);
    faceRect.delete();
    color.delete();
}

showImage(imgColor, canvas);

img.delete();
imgColor.delete();
faces.delete();
imgGray.delete();
}

```

## 2. main.js

```

(function () {
    var signalObj = null;

    window.addEventListener('DOMContentLoaded', function () {
        var isStreaming = false;
        var start = document.getElementById('start');
        var stop = document.getElementById('stop');
        var video = document.getElementById('v');
        var canvas = document.getElementById('c');
        var ctx = canvas.getContext('2d');
        var effect = document.getElementById('effect');
        var isEffectActive = false;

        start.addEventListener('click', function (e) {
            var address = document.getElementById('address').value;
            var protocol = location.protocol === "https:" ? "wss:" : "ws:";
            var wsurl = protocol + '//' + address;

            if (!isStreaming) {
                signalObj = new signal(wsurl,
                    function (stream) {
                        console.log('got a stream!');
                        //var url = window.URL || window.webkitURL;
                    }
                );
            }
        });
    });
}

```

```

        //video.src = url ? url.createObjectURL(stream) : stream; // deprecated
        video.srcObject = stream;
        video.play();
    },
    function (error) {
        alert(error);
    },
    function () {
        console.log('websocket closed. bye bye!');
        video.srcObject = null;
        //video.src = ""; // deprecated
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        isStreaming = false;
    },
    function (message) {
        alert(message);
    }
    );
}
}, false);

stop.addEventListener('click', function (e) {
    if (signalObj) {
        signalObj.hangup();
        signalObj = null;
    }
}, false);

// Wait until the video stream can play
video.addEventListener('canplay', function (e) {
    if (!isStreaming) {
        canvas.setAttribute('width', video.videoWidth);
        canvas.setAttribute('height', video.videoHeight);
        isStreaming = true;
    }
}, false);

// Wait for the video to start to play
video.addEventListener('play', function () {
    // Every 33 milliseconds copy the video image to the canvas
    setInterval(function () {
        if (video.paused || video.ended) {
            return;
        }
        var w = canvas.getAttribute('width');
        var h = canvas.getAttribute('height');

```

```

    ctx.fillRect(0, 0, w, h);
    ctx.drawImage(video, 0, 0, w, h);
    if (isEffectActive) {
        detectFace(canvas);
    }
    }, 33);
}, false);

effect.addEventListener('click', function () {
    isEffectActive = !isEffectActive;
}, false);
});
})();

```

### 3. signalling.js

```

RTCPeerConnection = window.RTCPeerConnection || /*window.mozRTCPeerConnection || */
window.webkitRTCPeerConnection;
RTCSessionDescription = /*window.mozRTCSessionDescription || */ window.RTCSessionDescription;
RTCIceCandidate = /*window.mozRTCIceCandidate || */ window.RTCIceCandidate;

function signal(url, onStream, onError, onClose, onMessage) {
    if ("WebSocket" in window) {
        console.log("opening web socket: " + url);
        var ws = new WebSocket(url);
        var pc;

        ws.onopen = function () {
            /* First we create a peer connection */
            var config = {"iceServers": [{"urls": ["stun:stun.l.google.com:19302"]}]};
            var options = {optional: []};
            pc = new RTCPeerConnection(config, options);

            pc.onicecandidate = function (event) {
                if (event.candidate) {
                    var candidate = {
                        sdpMLineIndex: event.candidate.sdpMLineIndex,
                        sdpMid: event.candidate.sdpMid,
                        candidate: event.candidate.candidate
                    };
                    var request = {
                        what: "addIceCandidate",
                        data: JSON.stringify(candidate)
                    };
                    ws.send(JSON.stringify(request));
                } else {

```

```

        console.log("end of candidates.");
    }
};

if ('ontrack' in pc) {
    pc.ontrack = function (event) {
        onStream(event.streams[0]);
    };
} else { // onaddstream() deprecated
    pc.onaddstream = function (event) {
        onStream(event.stream);
    };
}

pc.onremovestream = function (event) {
    console.log("the stream has been removed: do your stuff now");
};

pc.ondatachannel = function (event) {
    console.log("a data channel is available: do your stuff with it");
    // For an example, see https://www.linux-projects.org/uv4l/tutorials/webrtc-data-channels/
};

/* kindly signal the remote peer that we would like to initiate a call */
var request = {
    what: "call",
    options: {
        force_hw_vcodec: true,
        vformat: 30 /* 640x480 */
    }
};
console.log("send message " + JSON.stringify(request));
ws.send(JSON.stringify(request));
};

ws.onmessage = function (evt) {
    var msg = JSON.parse(evt.data);
    var what = msg.what;
    var data = msg.data;

    console.log("received message " + JSON.stringify(msg));

    switch (what) {
        case "offer":
            var mediaConstraints = {
                optional: [],
            }

```

```

    mandatory: {
      OfferToReceiveAudio: true,
      OfferToReceiveVideo: true
    }
  };
  pc.setRemoteDescription(new RTCSessionDescription(JSON.parse(data)),
    function onRemoteSdpSuccess() {
      pc.createAnswer(function (sessionDescription) {
        pc.setLocalDescription(sessionDescription);
        var request = {
          what: "answer",
          data: JSON.stringify(sessionDescription)
        };
        ws.send(JSON.stringify(request));
      }, function (error) {
        onError("failed to create answer: " + error);
      }, mediaConstraints);
    },
    function onRemoteSdpError(event) {
      onError('failed to set the remote description: ' + event);
      ws.close();
    }
  );

  var request = {
    what: "generateIceCandidates"
  };
  ws.send(JSON.stringify(request));
  break;

case "answer":
  break;

case "message":
  if (onMessage) {
    onMessage(msg.data);
  }
  break;

case "iceCandidates":
  var candidates = JSON.parse(msg.data);
  for (var i = 0; candidates && i < candidates.length; i++) {
    var elt = candidates[i];
    let candidate = new RTCIceCandidate({sdpMLineIndex: elt.sdpMLineIndex, candidate:
elt.candidate});
    pc.addIceCandidate(candidate,

```

```

        function () {
            console.log("IceCandidate added: " + JSON.stringify(candidate));
        },
        function (error) {
            console.error("addIceCandidate error: " + error);
        }
    );
}
break;
}
};

ws.onclose = function (event) {
    console.log('socket closed with code: ' + event.code);
    if (pc) {
        pc.close();
        pc = null;
        ws = null;
    }
    if (onClose) {
        onClose();
    }
};

ws.onerror = function (event) {
    onError("An error has occurred on the websocket (make sure the address is correct)!");
};

this.hangup = function() {
    if (ws) {
        var request = {
            what: "hangup"
        };
        console.log("send message " + JSON.stringify(request));
        ws.send(JSON.stringify(request));
    }
};

} else {
    onError("Sorry, this browser does not support Web Sockets. Bye.");
}
}

```





# References

## \* Pico-Projector Technology

Introduction to DLP Technology

<https://electronics.howstuffworks.com/dlp1.htm>

<http://www.ti.com/dlp-chip/overview.html>

<http://www.ti.com/dlp-chip/technical-documents.html>

DLPDLCR2000EVM Evaluation Module

<http://www.ti.com/tool/DLPDLCR2000EVM>

<http://www.ti.com/lit/pdf/dlpu049>

## \* UV4L & WebRTC Standards

<https://www.linux-projects.org/uv4l/>

<https://github.com/ShubhamCpp/RaspiCam-UV4L>

<https://webrtc.org/>

<https://codelabs.developers.google.com/codelabs/webrtc-web/>

[https://developer.mozilla.org/en-US/docs/Web/API/WebRTC\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API)

## \* Practical Considerations

<https://www.elektormagazine.com/news/prevent-overheating-problems-on-your-raspberry-pi-3>

<https://elinux.org/images/1/1b/RPiThermalTest-8-20-2012.pdf>

<https://www.raspberrypi.org/forums/viewtopic.php?t=138383>

<http://www.ti.com/lit/pdf/dlpa068>

# Appendix

- Periodic Progress Report (PPR): 1-4
- AEIOU Canvas
- BMC Canvas
- BMC Report
- Patent Drafting Exercise

[Print](#)[Back](#)

College : BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR

StudentName : Chaitanya Roop Tejaswi

EnrollmentNo : 140080111013

Department : Electronics & Communication Engineering

MobileNo : 9427140794

Discipline : BE

Email : crtejaswi13@gmail.com

Semester : Semester 8

## PPR Details

Periodic Progress Report : First PPR

Project : Pico-projector based Automation

Status : Reviewed

1. What Progress you have made in the Project ?

Decided on the software & hardware stack. Listed the components. Referred the corresponding literature.

2. What challenge you have faced ?

Peripheral Selection.

3. What support you need ?

None.

4. Which literature you have referred ?

Online documentation. Hardware datasheets. Relevant patents.

## Comments

Comment by Internal Guide :

None

Comment by External Guide :

None

Comment by HOD :

None

Comment by Principal :

None

Comment by University Admin :

None

[Print](#)[Back](#)

College : BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR

StudentName : Chaitanya Roop Tejaswi

EnrollmentNo : 140080111013

Department : Electronics & Communication Engineering

MobileNo : 9427140794

Discipline : BE

Email : crtejaswi13@gmail.com

Semester : Semester 8

## PPR Details

Periodic Progress Report : Second PPR

Project : Pico-projector based Automation

Status : Reviewed

1. What Progress you have made in the Project ?

Implemented Server running on Raspberry Pi 3.

2. What challenge you have faced ?

Video lag. Frequent OS crashes.

3. What support you need ?

None.

4. Which literature you have referred ?

Online documentation. User Guides. Relevant datasheets.

## Comments

Comment by Internal Guide :

None

Comment by External Guide :

None

Comment by HOD :

None

Comment by Principal :

None

Comment by University Admin :

None

[Print](#)[Back](#)

College : BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR

StudentName : Chaitanya Roop Tejaswi

EnrollmentNo : 140080111013

Department : Electronics & Communication Engineering

MobileNo : 9427140794

Discipline : BE

Email : crtejaswi13@gmail.com

Semester : Semester 8

## PPR Details

Periodic Progress Report : Third PPR

Project : Pico-projector based Automation

Status : Reviewed

1. What Progress you have made in the Project ?

Implemented Streaming server on Raspberry Pi 3. Programmed WebRTC sockets to stream text, audio & video content locally.

2. What challenge you have faced ?

Inadequate documentation. Frequent OS crashes.

3. What support you need ?

None.

4. Which literature you have referred ?

Online documentation. Relevant tutorials and programming manuals.

## Comments

Comment by Internal Guide :

None

Comment by External Guide :

None

Comment by HOD :

None

Comment by Principal :

None

Comment by University Admin :

None

[Print](#)[Back](#)

College : BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR

StudentName : Chaitanya Roop Tejaswi

EnrollmentNo : 140080111013

Department : Electronics & Communication Engineering

MobileNo : 9427140794

Discipline : BE

Email : crtejaswi13@gmail.com

Semester : Semester 8

## PPR Details

Periodic Progress Report : Forth PPR

Project : Pico-projector based Automation

Status : Reviewed

1. What Progress you have made in the Project ?

Implemented Classroom Automation codes to be run locally & remotely on the Raspberry Pi Server. Pixel Processing using OpenCV.js.

2. What challenge you have faced ?

Inadequate documentation. Frequent OS crashes.

3. What support you need ?

Fixing end-kmplementations. Deciding on some specific applications of image processing using RPi Server. Deciding on what sensors to use to sense & respond to system either directly (to RPi, wirelessly) or indirectly (through a controller, such as 8-bit AVR).

4. Which literature you have referred ?

Online documentation. Relevant datasheets & user guides.

## Comments

Comment by Internal Guide :

None

Comment by External Guide :

None

Comment by HOD :

None

Comment by Principal :

None

Comment by University Admin :

None

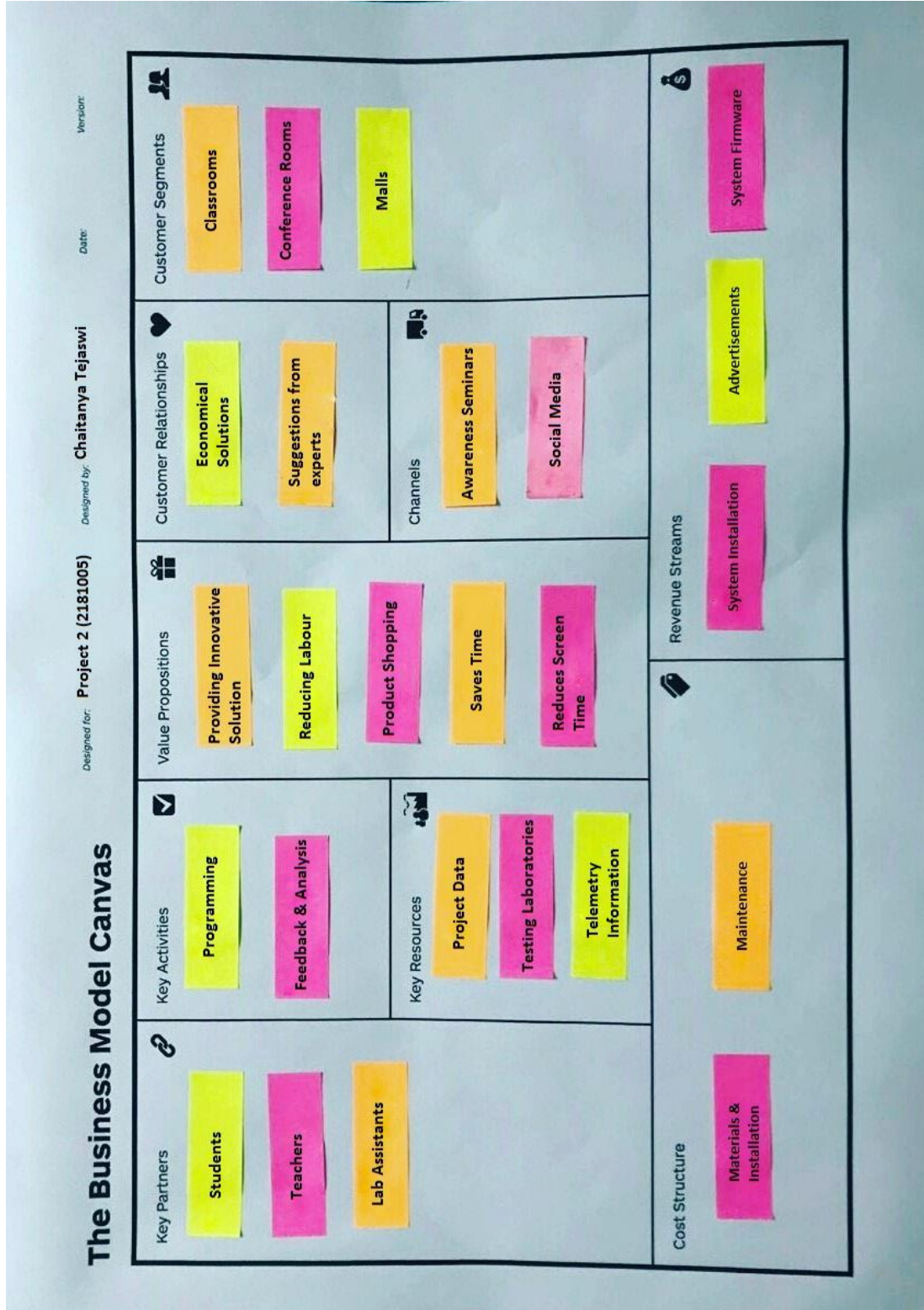
# AEIOU Summary

AEIOU Summary:		Group ID: 1906011013	Date: 14-12-2015	Version: 1.1
Domain Name: ELECTRONICS AND COMMUNICATION				
Environment:	Interactions:	Users:	Objects:	Activities:
<b>SCHOOL</b> <ul style="list-style-type: none"> <li>Well equipped lab</li> <li>Good infrastructure</li> <li>Good surroundings</li> <li>Classroom built with students</li> </ul> <b>MALL</b> <ul style="list-style-type: none"> <li>Well lit room</li> <li>Display items</li> <li>Air-conditioned</li> <li>Arranged good with students</li> </ul> <b>FACTORY</b> <ul style="list-style-type: none"> <li>Smoky fumes</li> <li>Hot air</li> <li>Smells of various</li> <li>Noise from running machinery</li> </ul> <b>OFFICE</b> <ul style="list-style-type: none"> <li>Air-conditioned</li> <li>Well arranged</li> <li>Well equipped</li> <li>Lighting</li> <li>Check-out</li> </ul> <b>THEATRE</b> <ul style="list-style-type: none"> <li>Completed hall</li> <li>Recreation</li> <li>High ceiling</li> <li>Excellent seating</li> </ul> <b>NEWS ROOM</b> <ul style="list-style-type: none"> <li>Running case on the</li> <li>Any/News relation</li> <li>Working on the street</li> </ul> <b>MUSEUM</b> <ul style="list-style-type: none"> <li>Various info</li> <li>IT - room</li> <li>Well lit</li> <li>Chipsy painted artifacts</li> </ul>	<b>SCHOOL</b> <ul style="list-style-type: none"> <li>Students sharing doubts from their professors</li> </ul> <b>MALL</b> <ul style="list-style-type: none"> <li>Students parking designers</li> <li>Customers buying daily ration supplies</li> </ul> <b>OFFICE</b> <ul style="list-style-type: none"> <li>Issues raised by</li> <li>Officers visit them</li> <li>How this</li> </ul> <b>THEATRE</b> <ul style="list-style-type: none"> <li>People watching movies in multiple</li> <li>People buying books at the</li> </ul> <b>NEWS ROOM</b> <ul style="list-style-type: none"> <li>Communities using</li> <li>GPS Navigation</li> <li>Pedestrians crossing the street</li> </ul>	<b>SCHOOL</b> <ul style="list-style-type: none"> <li>Professors</li> <li>Students</li> <li>Lab assistants</li> </ul> <b>FACTORY</b> <ul style="list-style-type: none"> <li>Workers assembling</li> <li>pieces of graded iron</li> <li>for making out parts of a car</li> </ul> <b>THEATRE</b> <ul style="list-style-type: none"> <li>People watching movies in multiple</li> <li>People playing games in changing zone</li> </ul> <b>NEWS ROOM</b> <ul style="list-style-type: none"> <li>Daily commuters travelling to work, stuck in traffic jam</li> </ul>	<b>SCHOOL</b> <ul style="list-style-type: none"> <li>Networks</li> <li>Reference books</li> <li>Frontpage</li> </ul> <b>MALL</b> <ul style="list-style-type: none"> <li>Various clothes</li> <li>Books</li> <li>Various</li> <li>Various</li> </ul> <b>OFFICE</b> <ul style="list-style-type: none"> <li>Professors</li> <li>Students</li> <li>Lab assistants</li> </ul> <b>FACTORY</b> <ul style="list-style-type: none"> <li>Workers assembling</li> <li>pieces of graded iron</li> <li>for making out parts of a car</li> </ul> <b>THEATRE</b> <ul style="list-style-type: none"> <li>People watching movies in multiple</li> <li>People playing games in changing zone</li> </ul> <b>NEWS ROOM</b> <ul style="list-style-type: none"> <li>Daily commuters travelling to work, stuck in traffic jam</li> </ul>	<b>SCHOOL</b> <ul style="list-style-type: none"> <li>Students reading books, doing well performing parties</li> </ul> <b>FACTORY</b> <ul style="list-style-type: none"> <li>People watching movies in multiple</li> <li>People playing games in changing zone</li> </ul> <b>THEATRE</b> <ul style="list-style-type: none"> <li>People watching movies in multiple</li> <li>People playing games in changing zone</li> </ul> <b>NEWS ROOM</b> <ul style="list-style-type: none"> <li>Daily commuters travelling to work, stuck in traffic jam</li> </ul>





# BMC Canvas





# BMC Report

## 1. Key Partners

Students: Primary user for the system. They will use the product for most of applications.

Teachers: They will handle server-side control over the system. Although not the primary user, they will certainly be the ones under control.

Lab Assistants: Apart from teachers, lab assistants will make use of the product for practical lab activities.

Technical Educators: Educators dealing with technical content will be aided by this product in providing access to audio-video content with ease.

Shopkeepers: The flow and the processes behind selling stock items have been explained by them.

## 2. Key Activities

Designing: The main part of how the flow of the project should be and how the final output should be is done in this step.

Interfacing: Which all components should be added and to be interfaced with the controller.

Programming: Which software is needed to use for the system to work.

Result Analysis: Exact working is being done or not is seen in this section.

## 3. Value Proposition

Providing Innovative Solution: It would be a unique solution to reduce the problems of the villagers.

Reducing Labor/ Man Work: Whole System would be automated hence there won't be requirement of any other person.

Effective Product Shopping: Quantity would be exact of how much is being demanded by the customer.

Saves Time: Time consumption would be totally reduced.

Eradicates Corruption: Corruption would be eradicated as there is no need for human interference.

## 4. Customer Relationships

Economical Solutions: The system would be available in the market at very low price.

Suggestions from Experts: If any change is to be made for the betterment then it is welcome.

## 5. Customer Segments: Places where the system can be implemented are mentioned below:

Classrooms

Conference Rooms

Malls

Home

Movie Theatres

## 6. Key Resources: The main factors for the flow of the system has been mentioned below:

Project Data

Testing Laboratories

Telemetry Information

## 7. Channels: The medium for awareness of the system which would be beneficiary for them

Advertisements

Awareness Seminars

Social Media

## 8. Cost Structure: The exact costing for the system goes by-

Materials Cost

Installation Cost

Equipment Maintenance Cost

Packaging Cost

## 9. Revenue Streams:

Installation of System

Advertisement



## Patent Drafting Exercise

PDE Details

<http://projects.gtu.ac.in/Student/StudentActivity/PDEDetails?enc=z1Jph...>

College : BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR  
Department : Electronics & Communication Engineering  
Discipline : BE  
Semester : Semester 8  
Project Name : Pico-projector based Automation  
Team ID : 25549

### Form 1 – APPLICATION FOR GRANT OF PATENT

Applicants :

Sr. No	Name	Nationality	Address	Mobile No.	Email Id
1	Chaitanya Roop Tejaswi	Indian	Electronics & Communication Engineering , BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR , Gujarat Technological University.	9427140794	crtejaswi13@gmail.com

Inventors :

Sr. No	Name	Nationality	Address	Mobile No.	Email Id
1	Chaitanya Roop Tejaswi	Indian	Electronics & Communication Engineering , BIRLA VISHVAKARMA MAHAVIDYALAYA, VALLABH VIDYANAGAR , Gujarat Technological University.	9427140794	crtejaswi13@gmail.com

I/We, the applicant(s) hereby declare(s) that:

Following are the attachments with the applications :

### Form 2 - PROVISIONAL/COMPLETE SPECIFICATION

1 . Title of the project/invention :

Pico-projector based Automation

2. Preamble to the description :

Provisional

3. Description

a) Field of Project / Invention / Application :

Digital Light Processing;

Embedded Systems;

Home Automation.

b) Prior Art / Background of the Project / Invention :

The invention relates to the field of digital projection display, particularly to automation using DLP micro projector.

To,  
The Controller of Patents,  
The Patent Office,  
At Mumbai