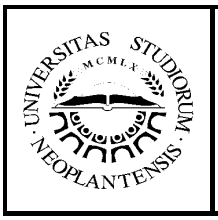


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Марко Говедар**
Број индекса: **РА 183/2017**

Тема рада: **Интеграција EBUTT-D формата титлова у аудио-видео програмском додатку са подршком за контролу у реалном времену**

Ментор рада: **др Илија Башичевић**

Нови Сад, април 2022.

УНИВЕРЗИТЕТ У НОВОМ САДУ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Завршни (Bachelor) рад
Аутор, **АУ**:		Марко Говедар
Ментор, **МН**:		др Илија Башичевић
Наслов рада, **НР**:		Интеграција EBUTT-D формата титлова у аудио-видео програмском додатку са подршком за контролу у реалном времену
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2022.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/34/0/1/44/0/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		Хромијум, Браузер, ebu-tt-d траке са преводом
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У оквиру задатка потребно је извршити интеграцију и омогућити приказ унутар опсега ебу-тт-д формата трака са преводом у оквиру АВ контролног додатка, омогућити контролу приказивања титлова као і промену језика титлова из терминала уређаја (дигитални ТВ пријемник, ТВ уређај).
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан:	Потпис ментора
	Члан, ментор:	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monographic publication
Type of record, **TR**:	Textual printed material
Contents code, **CC**:	Bachelor Thesis
Author, **AU**:	Marko Govedar
Mentor, **MN**:	PhD. Ilija Basicevic
Title, **TI**:	Integration of EBU-TT-D subtitle format in audio-video plugin with the support for real-time control
Language of text, **LT**:	Serbian
Language of abstract, **LA**:	Serbian
Country of publication, **CP**:	Republic of Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2022.
Publisher, **PB**:	Author's reprint
Publication place, **PP**:	Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/34/0/1/44/0/0
Scientific field, **SF**:	Electrical Engineering
Scientific discipline, **SD**:	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, **S**/**KW**:	Chromium, Browser, EBUTT-D subtitles
UC	
Holding data, **HD**:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:	
Abstract, **AB**:	This thesis describes implementation of software for integration and display of IN-BAND EBU-TT-D subtitle formats in the A/V control plugin, implementation of display option of subtitles depending on show/hide option, and change of subtitle language from termianal of device(set-top box, TV device).
Accepted by the Scientific Board on, **ASB**:	
Defended on, **DE**:	
Defended Board, **DB**: President:	
Member:	Menthor's sign
Member, Mentor:	

Захвалност

Захваљујем се Соњи Мандић и институту РТ-РК на пруженој могућности за реализацију овог рада, као и техничком ментору Милошу Пејовићу и Мариу Радоњићу на пруженој помоћи током израде завршног (бечелор) рада.

Посебно желим да се захвалим академском ментору проф. др. Илији на вођењу, стручним и личним саветима током израде завршног рада.

САДРЖАЈ

Списак слика

Слика 23. Дијаграм комуникације између стека и претраживача и тестне апликације

Списак табела

СКРАЋЕНИЦЕ

A/V – *Audio Video* , Аудио Видео

HTML – *Hyper Text Markup Language*, описни језик намењен опису веб страница

XML – *Extensible Markup Language*, проширив мета језик

HbbTv - *Hybrid Broadcast Broadband Tv* , Хибридни широкопојасни стандард заснован на вебу

IPTV – *Internet Protocol Television*, телевизија заснована на интернет протоколу

STB – *Set-top box*, дигитални тв пријемник

DVB – *Digital Video Broadcast*, стандард за дигиталну видео дифузију

CS – *Companion Screen*, додатни мултимедијални екран

API – *Application programming interface*, програмска спрега

CSS – *Cascading style sheets*, језик форматирања помоћу којег се дефинише изглед елемената веб странице

URL-*Uniform Resource Locators,* уједначени локатор ресурса, веб адреса

RPC- *remote procedure call*, процедурално програмирање, омогућавају клијенту повезивање метода од стране послужитеља успостављањем комуникације између клијента и послужитеља

1. Увод

Задатак овог рада је интеграција, подршка и контрола *EBUTT-d* формата трака са преводом користећи *A/V* контролни додатак у реалном времену. Реализовано је у виду пуштања или заустављања текстуалних трака, приказивања и скривања њих, и промене језика.

Рад је реализован проширењем постојећег *Teatro TV Browser-a*, тестне *HTML* странице као и *XML* датотека. О овоме ће бити више речи у наставку.

Једни од великих добитника *Covid 19* пандемије били су дигитална телевизија и платформе за стримовање, који су у том периоду добили огромну гледаност. Индустрија је у замаху и константно се ствара нови садржај. Кључан део тог садржаја чине текстуалне траке. Текстуалне траке су од изузетног значаја, пошто омогућавају публици да гледа садржај без звука, да публика може да гледа садржај на другим језицима, да особе са оштећењем слуха могу да прате радњу боље, или да публика разуме шта се дешава у случају да је лош аудио или видео.

Ова дипломска дисертација се састоји из 7 поглавља.

У овом поглављу је направљен увод у рад.

У другом поглављу се налазе теоријске основе. Састоји се из *HbbTv-a* и *Chormium-a*.

У трећем поглављу се налази концепт решења, као план решавања проблема.

У четвртом поглављу је сама имплементација решења.

У петом поглављу су резултати имплементације и верификација тестова.

У шестом поглављу се налази закључак са идејом о даљим корацима.

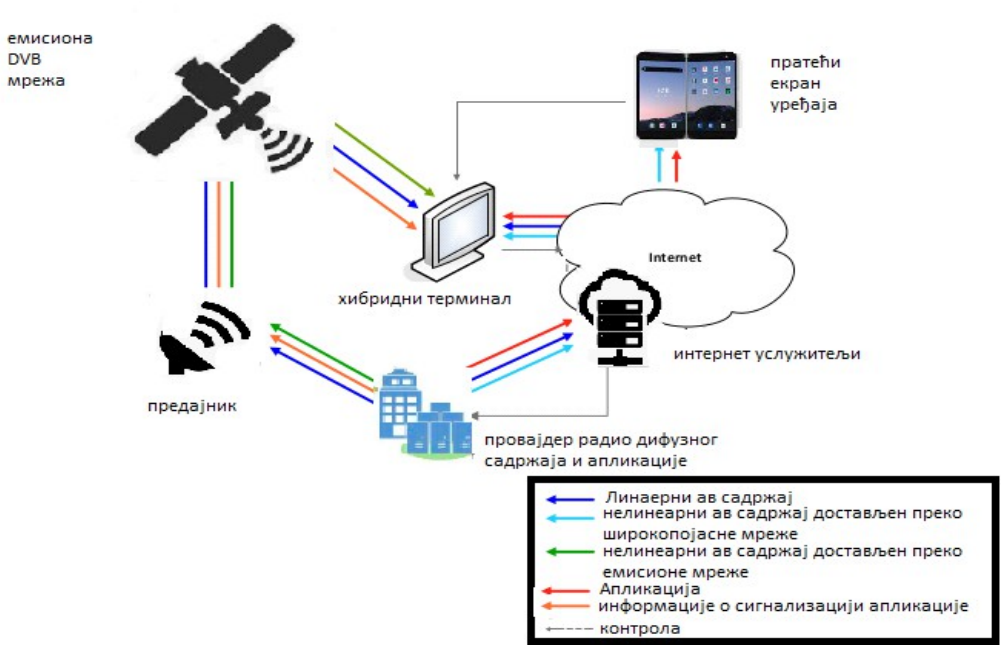
У седмом поглављу је списак литература.

2. Теоријске основе

У овом поглављу ће бити представљени теоријски описи кључних компонената рада.

2.1 *HbbTv* – Преглед и архитектура

HbbTv је уједно и индустријски стандард и промотивна иницијатива за стварањем хибридне дигиталне телевизије чији је циљ да споји емисиону, *IPTV* и широкопојасну испоруку забавних садржаја крајњем кориснику посредством везе између *TV/STB* уређаја и интернета. *HbbTv* има могућност приказивања садржаја дигиталне телевизије са већег броја различитих извора укључујући традиционалну емисиону телевизију, интернет и повезане уређаје у дому. За гледање хибридне дигиталне телевизије потрошачима је потребан хибридни *TV/STB* са различитим улазним прикључцима укључујући и етернет (енг. *Ethernet*) као и бар један улазни степен (енг. *Tuner*) за пријем *TV* сигнала емитерским каналом (енг. *Broadcast*).



Слика 1. Архитектура *HbbTv* телевизије

Хибридни терминал има могућност паралелног повезивања на две мреже. Са једне стране може бити повезан на емисиону (енг. *Broadcast*) *DVB* мрежу. Овим путем хибридни терминал може примати стандардни емисиони *A/V* садржај, *A/V* садржај који се не испоручује у реалном времену (енг. *Non-realtime*), апликативне податке и апликативне сигналне поруке. Чак и када терминал није повезан са широкопојасном (енг. *Broadband*) мрежом, његова веза са емитерским каналом му допушта пријем садржаја везаних за емитерски канал. Додатно, емитерски канал омогућава слање сигналних порука апликацијама.

Поред тога хибридни терминал може се повезати са интернетом посредством широкопојасне мрежне спреге. Ово омогућава двосмерну комуникацију са пружаоцем (енг. *Provider*) апликације. Коришћењем ове спреге терминал може да прими апликативне податке и нелинеарни *A/V* садржај. Хибридни терминал може такође да подржи преузимање *A/V* садржаја ван реалног времена преко ове спреге. Широко појасна мрежна спрега се такође може повезати са *CS* уређајем или другим *HbbTv* терминалима у оквиру исте мреже.

2.2 *Chromium* – веб прегледач

Chromium је *open source* програмски код који служи за веб прегледање, који је већински развила компанија *Google*. Да би се *Chromium* покренуо, формира се *Browser* процес који служи за управљање осталих процеса. Када се формира *Browser* процес покреће се и *Chromium* веб прегледач, али и даље нема могућност приказивања садржаја веб странице. Да би се садржај репродуковао неопходно је да се покрене процес који ће бити задужен за репродукцију садржаја те веб странице. *Chromium* има могућност учитавања више веб страница истовременои свака веб страница ће бити учитана у посебну картицу. Управљање картицама, њихово стварање или уклањање је задатак *Browser* процеса. *Chromium* веб прегледач позива свој скуп *API* функција. Позивом *API* функције *Chromiuma*, *Browser* процес одлучује којој картици је потребно да проследи захтев.

Browser процес се формира како би се покренуо веб прегледач. У овом процесу се обавља иницијализација свих модула потребних за рад *Chromiuma*. *API* функције које *Chromium* обезбеђује позивају се из овог процеса. Након што је веб прегледач покренут, још увек није формирана ни једна картица. Да би се формирала картица и да би се приказао веб садржај, корисник прво мора да затражи од *Broweser* процеса формирање нове картице. У *browser* процесу се управља свим формираним картицама на основу идентификатора картица које овај процес генерише у тренутку стварања картице и памти све док се картица не уколни, тј док се *Renderer* пороцес који представља ту картицу не затвори. У *Renderer* процесу се добавља садржај неке веб странице, обрађује и приказује кориснику. У тренутку када се добије захтев за престанак рада *Chromiuma*, *browser* процес има задатак да прво уради затварање свих картица тако што ће све формиране *Renderer* процесе угасити, затим је неопходно да ослободи све ресуре који су заузети током рада и за крај прекине своје извршавање.

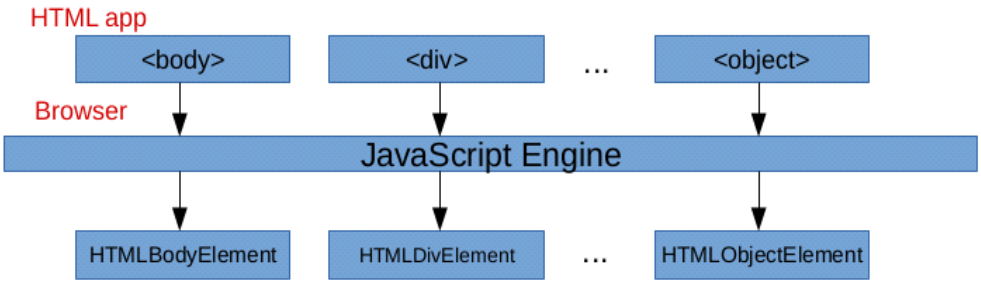
3. Концепт решења

3.1 Компоненте система

Рад се састоји из 3 целине. Прва је *HTML/JavaScript* апликација која је потребна да буде учитана од стране *Teatro TV Browser-a*. Друга представља *XML* документ са тракама које требају бити приказане у оквиру апликације. Трећа целина је *Teatro TV Browser* који се користи за покретање апликације.

3.2 *HTML* документ и *HbbTv* функционалност

Сваки *HTML* елемент има своју представу у оквиру *Browser* дела апликације уз помоћ *JavaScript engine-a*. *JavaScript* engine је програм који извршава *JavaScript* код у *Browser*-у. Он се углавном користи за функционалност веб страница и сваки прегледач има свој *engine*.



Слика 2. *JavaScript engine*

Ознака <*object*> дефинише уграђени објекат унутар *HTML* датотеке. Овај елемент се користи за уграђивање елемената на веб страници. Приликом проширења <*object*> тада долази до могућности коришћења функција унутар *HbbTv-a* према спецификацији.

3.3 Концепт решења

У циљу логичког поједностављања имплементације и лакшег праћења током израде рад је подељен у четири целине, односно четири фазе израде. Свака фаза представља логички заокружену целину. Детаљан опис програмског решења се налази у следећем поглављу.

У првом сегменмту неопходно је упознавање са постојећом документацијом, њена анализа, упознавање са радним окружењем које ће бити коришћено и подешавање истог. Документација укључује званичне *HbbTv* документе у којима је детаљно описан *HbbTv* стандард, стандард за *ebu-tt-d* траке са преводом, као и документација.

Друга фаза тиче се имплементирања *HTML/JavaScript* апликације. Потребно је проширити дату апликацију тако да су подржане уграђене траке са преводом.

Трећа фаза представља парсирање и примену *ebu-tt-d* трака са преводом у оквиру *A/V* контролног додатка. Потребно је преузети траке из апликације и испарсирати из њих податке који ће се слати на даљу обраду. Неопходно је да приказ буде прилагођен тренутном стању *A/V* контролног додатка.

Четврта фаза тиче се увођења контролне функционалности приказа трака, покретања и заустављања трака помоћу догађаја испаљених из *A/V* контролног додатка.

4. Програмско решење

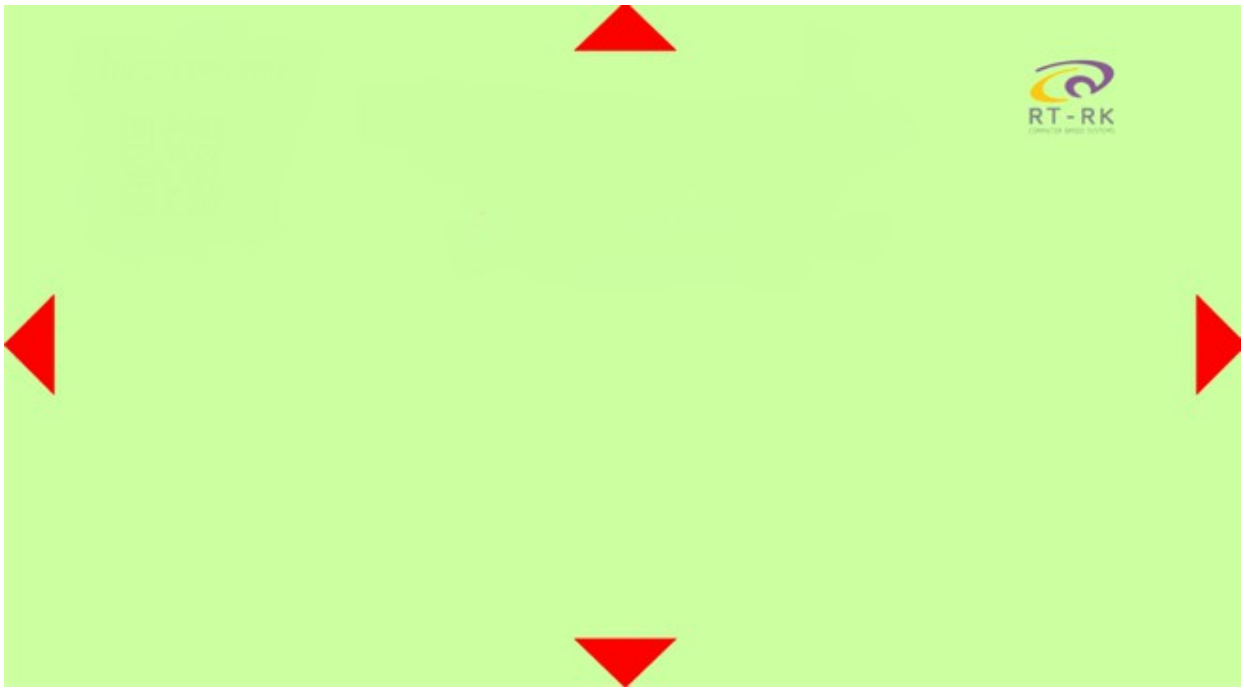
4.1 Развојна HTML/javascript апликација

У складу са захтевима задатка неопходно је било проширити *HTML/javascript* апликацију. Ову *HTML/javascript* апликацију, која се налази на *Apache* веб услужитељу, покрећемо уз помоћ *Teatro TV Browser-a*.

Ова апликација у себи треба да садржи <*object*> елементе, који у себи имају уграђене атрибуте који га шире описују. Атрибут типа (енг. "*type*") је значајан јер он дефинише и одређује ког је типа поменути објекат, у случају додавања *ebu-tt-d* трака са преводом битни су објекти типа type="*video/mp4*", о овоме ће бити више речи у наставку када будемо детаљно обрађивали траке. Поред атрибута типа, битан је и атрибут података (енг. "*data*") јер се у њему налази путања до датотеке од интереса у којој се налази "*mp4*" датотека која треба бити приказана и у склопу које требају бити уграђене траке. Атрибут ид (енг. "*id*") се користи као индетификатор за дати елемент, приликом приступа елементу у *javascript* делу апликације. Остали атрибути који су коришћени у овим објектима су атрибути за стил (енг. "*style*") као што су: видљивост (енг. "*visibility*"), позиција (енг. "*position*"), ширина (енг. "*width*") , висина (енг. "*heigth*").

Траке су додељиване <*object*> елементима у облику параметара. Ти параметри такође у себи садрже атрибуте који га шире описују. Атрибут који говори да је реч о тракама које требају даље да се обрађују је име (енг. "*name*"). За даљу обраду у оквиру *Browser-a* ,о којој обради ће бити више речи мало касније, значајне су траке са именом "*Subtitles*" и "*Captions*". Траке са именом "*Subtitles*" се користе када су траке другог језика у односу на текст садржаја, док "*Captions*" представљају траке које су истог језика као и садржај. Путање до *XML* датотека у којима се налази опис трака које требају бити приказане у овиру објекта уграђене се налазе у атрибуту вредност (енг. "*value*"). За потребе тестирања апликације у <*script*> тагу апликације се добавља жељени елемент позивом функције:

var elem = document.getElementById('mp4video');



Слика 3. Учитана *HTML/javascript* апликација

4.2 Детектовање укупног броја сигнализираних трака са преводом у оквиру стрима, парсирање потребних атрибута и креирање одговарајућег броја веб тескт трака унутар опсега на страни претраживача

```
void HTMLObjectElement::HandleInbandSubtitles() {
    int ret_status;   // error code when receiving message;

    char value_buffer[kValueStringMaxLength];
    ret_status = MEDIA_RPC_API(getPlayerProperty)(mmlib_handler_, (char*)"SUBTITLE_COMPONENT.LENGTH", value_buffer, kValueStringMaxLength);

     if(!ret_status) {
         DBG_WARNING("getPlayerProperty SUBTITLE_COMPONENT.LENGTH returned error - skip checking video");
     } else {

         int num_of_in_band_subtitles = atoi(value_buffer);

         if(num_of_in_band_subtitles > 0) {
             use_subtitle_content_ = true;

             if(!subtitle_renderer_container_)
                 CreateSubtitleVideoElement();

             subtitle_components_.reset(new SubtitleComponentInfo[num_of_in_band_subtitles]);
```

Слика 4. Функција *HandleInbandSubtitles* за парсирање трака, први део

HandleInbandSubtitles у класи *html_object_element* позива функцију из *medialib* класе *MEDIA_RPC_API(getPlayerProperty)*. Њена повратна вредност је истинита у случају да

се функција извршила успешно, а на основу послатог другог аргумента се одређује са чиме ће да се напуни међуспрежник. Потребно је да се добије који је укупан број сигнализираних трака, па се прослеђује као *ID* параметар "*SUBTITLE_COMPONENT.LENGTH*" тој функцији. Врши се провера повратне вредности функције како би се проверило да ли се функција извршила уопште, да у случају да није, не би се трошили ресурси узалудно.

```
int  MEDIA_RPC_API(getPlayerProperty)(unsigned long int handle, char* ID, char* val, int val_len)
{
    DBG_INFO("mmlib getPlayerProperty handle %lu ID %s", handle, ID);
```

Слика 5. *MEDIA_RPC_API* функција, улазни параметри

```
if(strcmp(ID, "SUBTITLE_COMPONENT.LENGTH") == 0)
{
    sprintf(val, "%d", sizeof(sub_component)/8);
}
```

Слика 6. *MEDIA_RPC_API* функција, враћање дужине титл компоненте

Након тога, врши се конверзија из словних података у бројни домен, пошто се у међуспрежнику налазили у *char* типу података. Проверава се затим да ли је број трака у опсегу већи од нуле, и у случају да јесте, поставља се *use_subtitle_content_* застава на истиниту вредност. Она је потребна за рендеровање, у делу кода који је од раније имплементиран. Затим се такође проверава и да ли је већ иницијализован садржајник за рендеровање *subtitle_renderer_container_*, који је припадник класе видео елемент, *HTML_video_element*, како се не би поново стварао при сваком позиву функције. У случају да није иницијализован, мора се позвати функција за стварање видео елемента, *CreateSubtitleVideoElement*.

```
void HTMLObjectElement::CreateSubtitleVideoElement() {

    subtitle_renderer_container_ = MakeGarbageCollected<HTMLVideoElement>(GetDocument());
    subtitle_renderer_container_->SetInlineStyleProperty(CSSPropertyID::kWidth, 100, CSSPrimitiveValue::UnitType::kPercentage, true);
    subtitle_renderer_container_->SetInlineStyleProperty(CSSPropertyID::kHeight, 100, CSSPrimitiveValue::UnitType::kPercentage, true);
    subtitle_renderer_container_->SetDisplayMode(subtitle_renderer_container_->GetVideoMode());

    subtitle_renderer_container_->SetWidth(BoundsInViewport().width());
    subtitle_renderer_container_->SetHeight(BoundsInViewport().height());
}
```

Слика 7. *CreateSubtitleVideoElement* функција за прављење видео елемента

У овој функцији, позивом *MakeGarbageCollected<HTMLVideoElement>(GetDocument())* се прави садржајник. *GarbageCollected* је менаџер за контролу меморије. Он се бави алокацијом и оптуштањем меморије. Он аутоматски руководи меморијом, тако да не мора да се брине о алокацији и отпуштању меморије, или животним веком објекта који користи ту меморију. *Subtitle_renderer_container* је *Member HTMLVideoElement*-а. *Member* су јаки показивачи који показују на алоциране објекте на *HEAP*-у. Сва поља од *Member* класе морају бити праћена у *Trace* методи те класе. То спречава *GarbageCollected* да помисли да тај објекат није жив и да ослободи меморију.

```
Member<HTMLVideoElement> subtitle_renderer_container_;
std::unique_ptr<SubtitleComponentInfo[]> subtitle_components_;
```

Слика 8. Показивачи на видео елемент и на структуру података за информације о тракама

Следеће је додавање *In line Style Property*-а, што је додавање особина *HTML* елементу. Као параметри се користе *CSSPropertyID* да би се пронашле особине које желе да се мењају, а *CSSPrimitiveValue::UnitType* за тип јединица које желе да се додају. Ту се поставља ширина и висина *CSS*-а да се протеже дуж целе дужине HTML-а елемента. Следи постављање висине и ширине самог HTML-а.

Даље се наставља *HandleInbandSubtitles* функција. Користи се *subtitle_components_* што је јединствени показивач на структуру у коју се смештају информације из испарсираних трака.

```
typedef struct {
  GeneralComponentInfo generalInfo;
  std::string language_bcp_47;
  std::string language_iso_639;
  std::string mode;
  bool isHearingImpaired;
} SubtitleComponentInfo;
```

Слика 9. Структура података у коју се смештају испарсирани подаци

```
char *sub_component [2] = {"2|E-AC3|0|-1|2|||1|1|deu|deu|showing|5|", "2|E-AC3|0|-1|2|||1|2|eng|eng|hidden|5|"};
```

Слика 10. Низ *sub_component* из којег се извлаче стрингови за парсирање

```
for(int i = 0; i < num_of_in_band_subtitles; i++) {
    char property_name[kPropertyNameMaxLength];
    sprintf(property_name, "%s%d", kPropertyStringSubtitleCompItem, i);
    //clear buffer for receiving data
    memset(value_buffer, 0, kPropertyNameMaxLength);
    ret_status = MEDIA_RPC_API(getPlayerProperty)(mmlib_handler_, property_name, value_buffer, kPropertyNameMaxLength);
    if(!ret_status)
    {
      DBG_WARNING("getPlayerProperty %s returned error - skip this component", property_name);
      continue;
    }
    ret_status = StringToSubtitleComp(value_buffer, &subtitle_components_[i]);
    if(ret_status < 0) {
      DBG_WARNING("error while parsing subtitle component string '%s' - skip this component", value_buffer);
      continue;
    }
    DBG_INFO("%d. subtitle track:", i);
    DBG_INFO("\t type: %d", subtitle_components_[i].generalInfo.type);
    DBG_INFO("\t encoding: %s", subtitle_components_[i].generalInfo.encoding.c_str());
    DBG_INFO("\t isEncrypted: %d", subtitle_components_[i].generalInfo.isEncrypted);
    DBG_INFO("\t componentTag: %d", subtitle_components_[i].generalInfo.componentTag);
    DBG_INFO("\t pid: %d", subtitle_components_[i].generalInfo.pid);
    DBG_INFO("\t role: %s", subtitle_components_[i].generalInfo.role.c_str());
    DBG_INFO("\t language_bcp_47: %s", subtitle_components_[i].language_bcp_47.c_str());
    DBG_INFO("\t language_iso_639: %s", subtitle_components_[i].language_iso_639.c_str());
    DBG_INFO("\t mode: %s", subtitle_components_[i].mode.c_str());
    DBG_INFO("\t isHearingImpaired: %d", subtitle_components_[i].isHearingImpaired);
    DBG_INFO("\t label: %s", subtitle_components_[i].generalInfo.label.c_str());
    DBG_INFO("\t isEnabled: %d", subtitle_components_[i].generalInfo.isEnabled);
    DBG_INFO("\t id: %s", subtitle_components_[i].generalInfo.id.c_str());

    const blink::WebString web_label = blink::WebString::FromUTF8(subtitle_components_[i].generalInfo.label);
    const blink::WebString web_language = blink::WebString::FromUTF8(subtitle_components_[i].language_bcp_47);

    const blink::WebString web_id = blink::WebString::FromUTF8(subtitle_components_[i].generalInfo.id);
    WebInbandTextTrack::Kind web_kind = WebInbandTextTrack::Kind::kKindSubtitles;

    DBG_INFO("%d. web text track:", i);
    DBG_INFO("\t web_id: %s", web_id.Utf8().c_str());
    DBG_INFO("\t web_kind: %d", web_kind);
    DBG_INFO("\t web_label: %s", web_label.Utf8().c_str());
    DBG_INFO("\t web_language: %s", web_language.Utf8().c_str());

    WebInbandTextTrackImpl* web_inband_text_track = new WebInbandTextTrackImpl(web_kind, web_label, web_language, web_id, mmlib_handler_);
    subtitle_renderer_container_->AddAVPlayerInbandTextTrack(web_inband_text_track, subtitle_components_[i].mode, subtitle_components_[i].generalInfo.isEnabled);
```

Слика 11. Функција *HandleInbandSubtitles* за парсирање трака, други део

Затим се итерира кроз петљу. Та петља почиње са конкатанацијом стринга *kPropertyStringSubtitleCompItem* и броја итерације у коме се петља налази тренутно. На тај начин се приступа особини плејера која ће нам вратити у међуспрежник из низа са сигнализираним тракама, сигнализирану траку која се налази на том месту.

```
if(strcmp(ID, "SUBTITLE_COMPONENT.ITEM.0") == 0)
{
    sprintf(val, "%s", sub_component[0]);
}
if(strcmp(ID, "SUBTITLE_COMPONENT.ITEM.1") == 0)
{
    sprintf(val, "%s", sub_component[1]);
}
```

Слика 12. *MEDIA_RPC_API* функција, део који враћа стрингове за парсирање

Затим се парсира тај стринг позивом функције *StringToSubtitleComp*.

```
int HTMLObjectElement::StringToSubtitleComp(char* received_string, SubtitleComponentInfo* comp_info) {

  char* param_string = received_string; // pointer pointing to the start of the
                                        // current parameter in the received_string
  int bytes_consumed;

  bytes_consumed = StringToGeneralCompInfo(param_string, &comp_info->generalInfo);

  if(comp_info->generalInfo.type != ComponentChanged::Subtitle) {
    DBG_WARNING("error - component type is %d, expected %d for subtitle", comp_info->generalInfo.type, ComponentChanged::Subtitle);
    return -1;
  }

  // move the pointer to subtitle specific parameters
  param_string += bytes_consumed;

  param_string += GetStringParamFromCompString(param_string, comp_info->language_bcp_47);
  param_string += GetStringParamFromCompString(param_string, comp_info->language_iso_639);
  param_string += GetStringParamFromCompString(param_string, comp_info->mode);
  param_string += GetBoolParamFromCompString(param_string, &comp_info->isHearingImpaired);

  // return number of bytes parsed
  return (int)(param_string - received_string);
}
```

Слика 13. *StringToSubtitleComp* функција која врши парсирање

Функција прима као парамере стринг, који се обрађује, и структуру података *SubtitleCompenentInfo,* у коју се смешта. Прво се прави показивач који показује на почетак параметра који се тренутно парсира. Затим се прави промењива која води рачуна о колико бита је већ испарсирано. Даље се позива функција која извлачи *GeneralCompInfo* из стринга.

```
int HTMLObjectElement::StringToGeneralCompInfo(char* received_string, GeneralComponentInfo* comp_info) {

  char* param_string = received_string; // pointer pointing to the start of the
                                        // current parameter in the received_string

  param_string += GetIntParamFromCompString(param_string, &comp_info->type);  // extracted only for error checking
  param_string += GetStringParamFromCompString(param_string, comp_info->encoding);
  param_string += GetBoolParamFromCompString(param_string, &comp_info->isEncrypted);
  param_string += GetIntParamFromCompString(param_string, &comp_info->componentTag);
  param_string += GetIntParamFromCompString(param_string, &comp_info->pid);
  param_string += GetStringParamFromCompString(param_string, comp_info->role);
  param_string += GetStringParamFromCompString(param_string, comp_info->label);
  param_string += GetBoolParamFromCompString(param_string, &comp_info->isEnabled);
  param_string += GetStringParamFromCompString(param_string, comp_info->id);

  // return number of bytes parsed
  return (int)(param_string - received_string);
}
```

Слика 14. *StringToGeneralCompInfo* функција за парсирање генералих информација

Као и у претходној функцији, и овде се ставља показивач на карактере који прати докле се стигло са обрадом стринга, *param_string*. Позивају се функције које извлаче из стринга информације и стављају их у поља *comp_info* компоненте.

```
int HTMLObjectElement::GetIntParamFromCompString(char* comp_string, int* result) {
  int param_str_len;  // length of the parameter

  param_str_len = (int)strcspn(comp_string, kParamDelimiter);
  comp_string[param_str_len] = '\0';
  if(param_str_len) {
    *result = atoi(comp_string);
  } else {
    *result = -1;
  }
  return param_str_len + 1;   // return number of parsed characters
}
```

Слика 15. *GetIntParamFromCompString* функција за извлачење броја из стринга

За добијање бројног типа користи се *GetIntParamFromCompString* функција. У овој функцији се прелази преко стринга док се не дође до карактера за граничник "|" помоћу функције *strcspn*, са тим да се резултат те функције пребацује у бројну вредност, пошто је потребна дужина. На месту тог граничника се ставља нул терминатор, да би се ставио крај на то место. Ако је дужина тог стринга различита од нуле, смешта се у други аргумент функције вредност тог стринга претворена у у бројну вредност, у супротном, вредност -1. Као повратна вредност функције се враћа број испарсираних карактера плус један за гранични карактер.

```
int HTMLObjectElement::GetStringParamFromCompString(char* comp_string, std::string &result) {
  int param_str_len;    // length of the parameter

  param_str_len = (int)strcspn(comp_string, kParamDelimiter);
  comp_string[param_str_len] = '\0';
  result = comp_string;

  return param_str_len + 1;    // return number of parsed characters
}
```

Слика 16. *GetStringParamFromCompString* функција за извлачење стринга из стринга

По истом принципу и ова функција има промењиву која прати дужину параметра, са *strcspn* се проверава на ком месту се налази параметар за разграничење, који се претвара у бројну вредност да би се добила дужина стринга, и ставља се нул терминатор на крај тог стринга. Тај стринг се ставља у резултат, и повратна вредност је број обрађених карактера плус један за карактер граничник.

```
int HTMLObjectElement::GetBoolParamFromCompString(char* comp_string, bool* result) {
  int param_str_len;    // length of the parameter

  param_str_len = (int)strcspn(comp_string, kParamDelimiter);
  comp_string[param_str_len] = '\0';
  if(param_str_len) {
    *result = atoi(comp_string) != 0;
  } else {
    *result = false;
  }
  return param_str_len + 1;    // return number of parsed characters
}
```

Слика 17. *GetBoolParamFromCompString* функција за извлачење Булове промењиве из стринга

Ова функција тражи први граничник помоћу *strcspn*, али пошто је повратна вредност те функције карактер, пребацује се у бројну вредност. Затим се додаје нул терминатор на крају извученог стринга. У случају да је стринг различит од нуле, у резултат се ставља да је истинит, ако је нула, онда неистинит. Као повратна вредност функције се враћа број испарсираних карактера плус један за параметар граничник.

На основу тих функција из почетног стринга извлачи се тип трака, енкодовање, да ли је тракака енкриптована, ознака компоненте, идентификација процеса, улога стринга, најбоља тренутна пракса за ознаке језика, кодови за репрезентацију имена језика, мод видљивости трака, да ли је укључен мод за особе са оштећеним слухом, лабела, да ли су омогућене траке и идентификација трака.

```
subtitle track: 0
type: 2
encoding: E-AC3
isEncrypted: 0
componentTag: -1
pid: 2
role:
label:
isEnabled: 1
id: 1
language_bcp_47: deu
language_iso_639: deu
mode: showing
isHearingImpaired: 1

subtitle track: 1
type: 2
encoding: E-AC3
isEncrypted: 0
componentTag: -1
pid: 2
role:
label:
isEnabled: 1
id: 2
language_bcp_47: eng
language_iso_639: eng
mode: hidden
isHearingImpaired: 1
```

Слика 18. Испис испарсираних трака функције *HandleInBandSubtitles* на терминалу

```
const blink::WebString web_label = blink::WebString::FromUTF8(subtitle_components_[i].generalInfo.label);
const blink::WebString web_language = blink::WebString::FromUTF8(subtitle_components_[i].language_bcp_47);

const blink::WebString web_id = blink::WebString::FromUTF8(subtitle_components_[i].generalInfo.id);
WebInbandTextTrack::Kind web_kind = WebInbandTextTrack::Kind::kKindSubtitles;

DBG_INFO("%d. web text track:", i);
DBG_INFO("\t web_id: %s", web_id.Utf8().c_str());
DBG_INFO("\t web_kind: %d", web_kind);
DBG_INFO("\t web_label: %s", web_label.Utf8().c_str());
DBG_INFO("\t web_language: %s", web_language.Utf8().c_str());

WebInbandTextTrackImpl* web_inband_text_track = new WebInbandTextTrackImpl(web_kind, web_label, web_language, web_id, mmlib_handler_);
subtitle_renderer_container_->AddAVPlayerInbandTextTrack(web_inband_text_track, subtitle_components_[i].mode, subtitle_components_[i].generalInfo.isEnabled);
```

Слика 19. *HandleInbandSubtitles* функција за парсирање трака, трећи део

Пошто је за прављење текст трака потребан *WebString*, онда се из испарсираних података који су смештени у *subtitle_components_* прави екививалент за *WebString* класу.

web_label се прави од лабеле.

Затим се прави *web_language* која је најбоља тренутна пракса за језичке тагове на интернету.

web_id представља идентификацију те траке.

web_kind има сврху да прави разлику између тога да ли је трака *subtitle* или *captions*. Разлика између та два је да ли је трака превод са једног језика на други, или је трака само текст језика који се налази у видеу.

Пошто *subtitle_renderer_container_* наслеђује такође и класу *html_media_element*, може да позива функције и из те класе. *AddAVPlayerInbandTextTrack* као аргумент прима *WebInbandTextTrackImpl* тако да се прави нови објекат те класе који прима претходно направљене *WebString*-ове.

```
void HTMLMediaElement::AddAVPlayerInbandTextTrack(WebInbandTextTrack* track, std::string mode, bool isEnabled) {

  AddBrowserHandledTextTrack(track, mode, isEnabled);
}

void HTMLMediaElement::ParseAVPlayerInbandTextTrack(char* subtitleData, int subtitleSize) {

  InBandSubtitleWrapper(subtitleData, subtitleSize);
}

void HTMLMediaElement::ParseAVPlayerFontFamily(char* url, int urlLen, char* fontFamily, int fontFamilyLen) {

  SetFontFamily(url, urlLen, fontFamily, fontFamilyLen);
}

void HTMLMediaElement::ShowHideAVPlayerInbandTextTrack(bool showSubtitle) {

  VisibleInBandSubtitle(showSubtitle);
}

void HTMLMediaElement::ChangeAVPlayerInBandSubtitle() {

  ChangeInBandSubtitleWrapper();
}
```

Слика 20. Декомпозиција функција у класи *HTML_Media_Element*

Ово су функције из класе *html_media_element*. *AddAVPlayerInbandTextTrack* је претходно позвана. Ова функнција као и функције испод ње, о којима ће касније бити више речи, служе као декомопозиција функција. Наиме, њихова улога је само да служе да позивају друге функције, а њихов назив служи да апстракује функције, да боље дочара шта те функције раде.

```
void HTMLMediaElement::AddBrowserHandledTextTrack(WebInbandTextTrack* track, std::string mode, bool isEnabled) {
    LoadableInbandTextTrack* loadable_inband_text_track = LoadableInbandTextTrack::create(track, this, track->mediaPlayerId(), isEnabled);
    AddTextTrack(loadable_inband_text_track);
    loadable_inband_text_track->setModeFromClient(mode.c_str());
    last_loadable_inband_text_track_ = loadable_inband_text_track;
}
```

Слика 21. *AddBrowserHandledTextTrack* функција за прављење траке

AddBrowserHandledTextTrack прима као аргументе *WebInBandTextTrack*, мод траке, и да ли је трака укључена или не. Прво се прави објекат класе *LoadableInbandTextTrack* чијем се конструктору прослеђује *WebInBandTextTrack* коју је примила и ова функција, идентификатор те траке, и да ли је она укључена. Овај објекат се прави зато што функција која је од раније била имплементирана прима као параметар ову класу. Затим се позива та функција *AddTextTrack* и прослеђује јој се претходно направљени објекат.

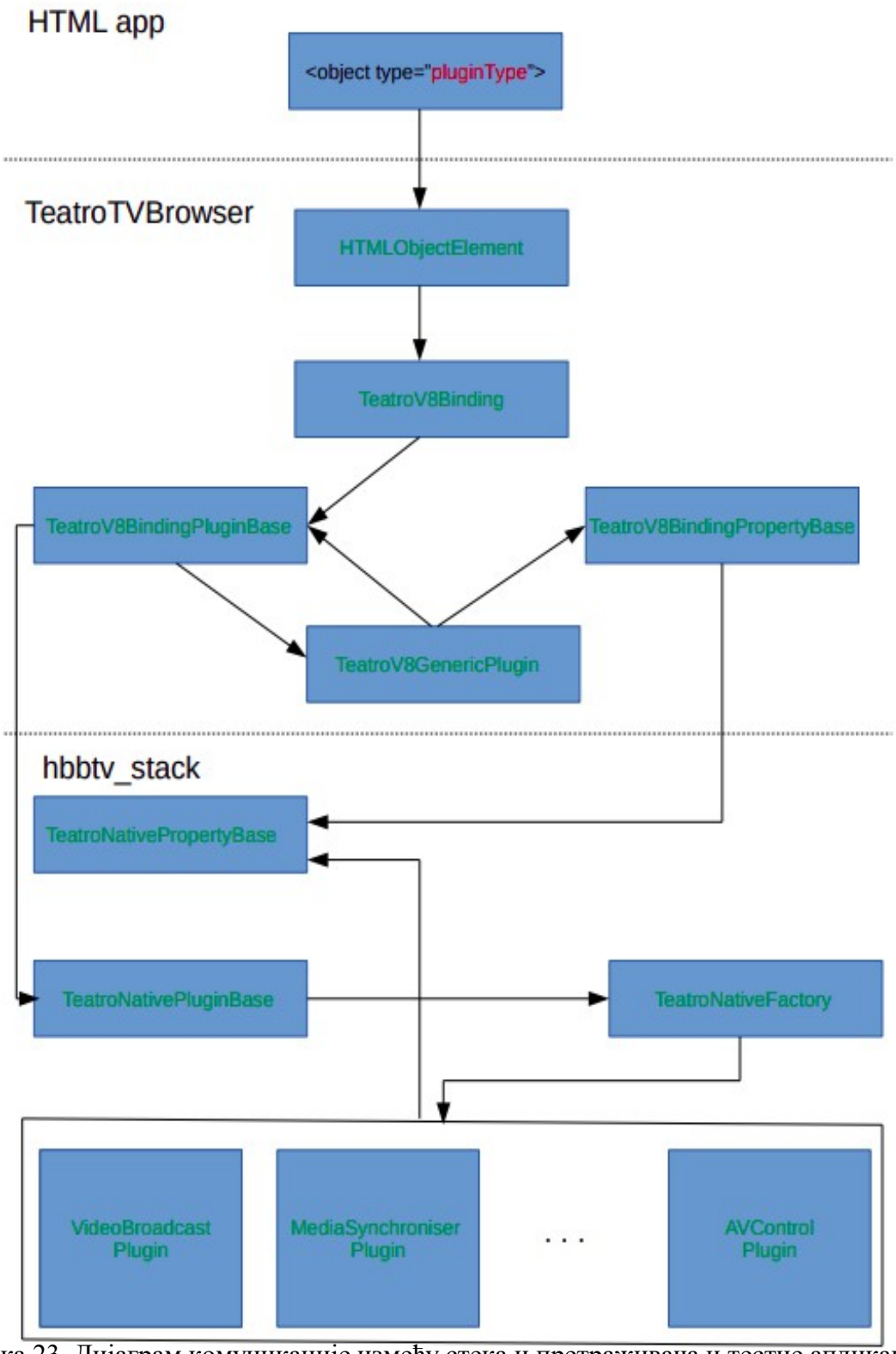
```
void HTMLMediaElement::AddTextTrack(TextTrack* track) {
  for(unsigned int i=0; i<textTracks()->length(); i++){
    if(track->TrackType() == TextTrack::kInBand && compareTracks(track, textTracks()->AnonymousIndexedGetter(i))){
      return;
    }
  }
  if(track->kind() == TextTrack::MetadataKeyword()){
    track->setMode(TextTrack::HiddenKeyword());
  }
  textTracks()->Append(track);
}
```

Слика 22. *AddTextTrack* функција за додавање направљене траке у листу трака

У овој функцији се проверава да ли је трака унутар опсега, и да ли је врста трака *Metadata*. У случају да јесте, то је нешто што кориник не мора да види, и онда се ставља мод те траке у *HiddenKeyword*, да би траке биле сакривене. На крају се додаје та трака међу текст траке.

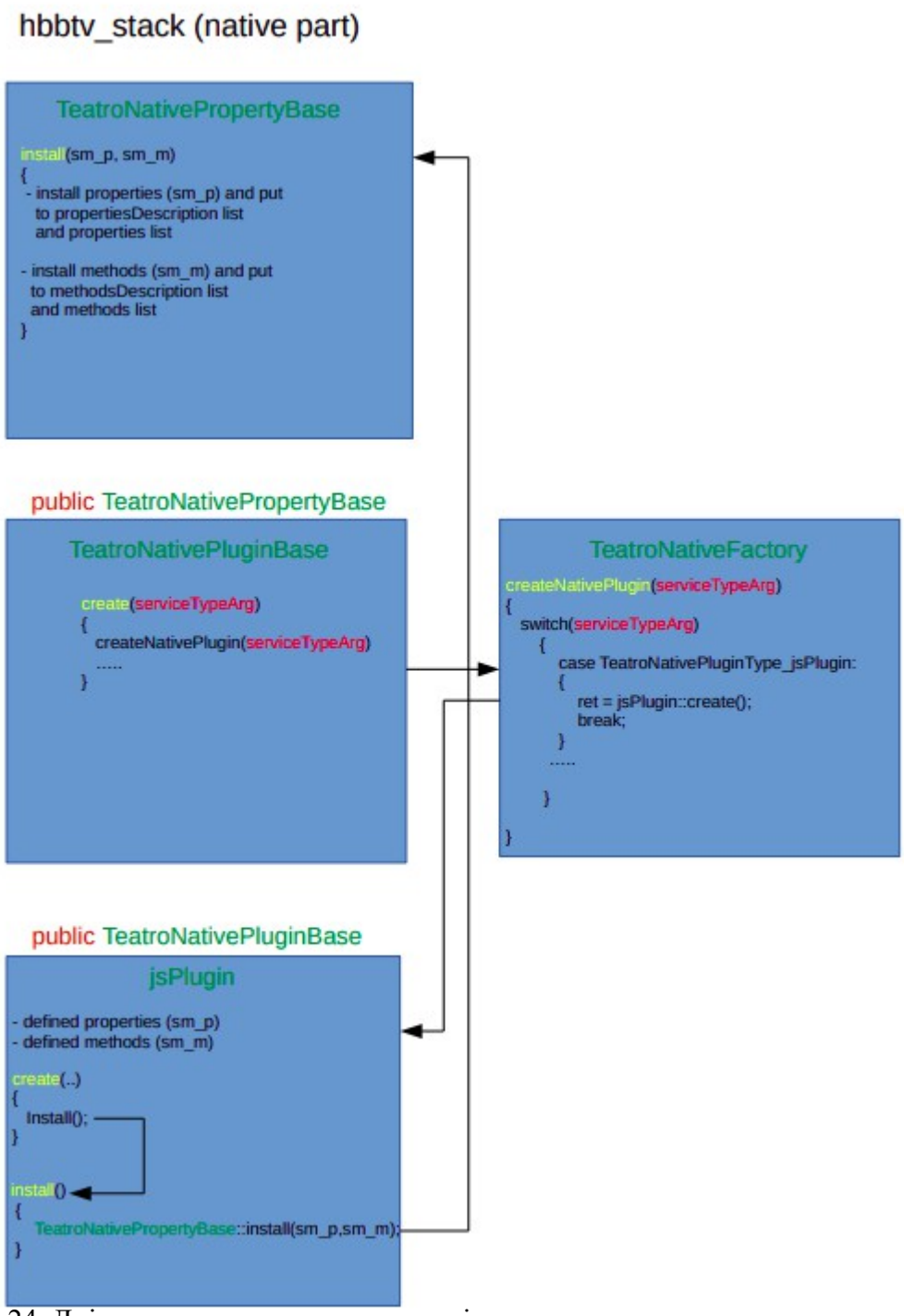
Слика 22. Прошиење тестне апликације приказом укупног броја и језика учитаних трака

4.3 Парсирање и приказ титлова унутар опсега текста у реалном времену



Слика 23. Дијаграм комуникације између стека и претраживача и тестне апликације

На слици је приказана архитектура целокупног претраживача са стек делом и апликацијом. На овом дијаграму се види које класе комуницирају са којим класама и у ком редоследу.



Слика 24. Дијаграм са описом комуникације класа у стеку и описом прављења додатака

На овој слици је приказана детаљнија архитектура стек дела, и на који начин се дефинишу методе и својства у додатку, и на који начин се те методе после прављења инсталирају и додају у листу својства и метода у *native* делу.

Над објектом који се налази у тестној *HTML* апликацији позива се функција *Testing_subtitles*, која као први параметар добија која ће операција да се врши, а као други параметар прима податке који треба да се обраде. Ова функција се налази у класи *AVControlPlugin*.

Да би ова метода могла да се региструје, инсталира и користи, она мора бити додата као метода *TeatroNativeMethod* у *sm_methods*[ ], као и својства *TeatroNativeProperty* у *sm_properties*[ ].

```
void AVControlPlugin::Testing_subtitles(int event_type, void *data) {

   switch(event_type){
     case 0:
     {
       TTMLSubtitle* ttmlSubtitleData = (TTMLSubtitle*)data;
       if(ttmlSubtitleData->length <= 0)
         return;
       TeatroNativeVariant args[1];
       ttnplUtil::set(ttmlSubtitleData->data, args[0], (int)ttmlSubtitleData->length);
       dispatchEvent("onInBandSubtitleData", args, 1);
     }
       break;
     case 1:
     {
       TTMLFont* ttmlFont = (TTMLFont*)data;
       if(ttmlFont->url_length <= 0 || ttmlFont->font_family_length <= 0)
         return;
       TeatroNativeVariant args[2];
       ttnplUtil::set(ttmlFont->url, args[0], (int)ttmlFont->url_length);
       ttnplUtil::set(ttmlFont->font_family, args[1], (int)ttmlFont->font_family_length);
       dispatchEvent("onSubtitleFontFamily", args, 2);
     }
       break;
     case 2:
     {
       TeatroNativeVariant args[1];
       ttnplUtil::set(true, args[0]);
       dispatchEvent("onInBandSubtitleShowHide", args, 1);
     }
       break;
     case 3:
     {
       TeatroNativeVariant args[1];
       ttnplUtil::set(false, args[0]);
       dispatchEvent("onInBandSubtitleShowHide", args, 1);
     }
       break;
     case 4:
       dispatchEvent("onInBandSubtitleChange", NULL, 0);
       break;

     default:
       break;
   }
```

Слика 25. *Testing_subtitles* функција у *A/V* контролном додатку

У случају да је прослеђена нула као први аргумент, позива се функција за обраду титлова. Подаци који се налазе у другом аргументу се пребацују на показивач на класу *TTMLSubtitle* (*eng .Time Text Markup Language*). Затим се врши провера да ли је дужина титлова већа од нуле, да у случају да није, да се ту излађе из фунцкије. Након тога се прави низ *TeatroNativeVariant* дужине један, пошто се се шаље само један аргумент са подацима. Онда се у *ttnplUtil* постављају ти подаци као први аргумент, смештају се у претходно направљен *TeatroNativeVariant* низ, као други аргумент, и као трећи се шаље која је дужина података. На крају се позива функција *dispatchEvent* , и као први аргумент се поставља назив догађаја „*onInBandSubtitleData*“ који ће бити опапаљен, као други се ставља *TeatroNativeVariant* у који су смештени подаци, и у трећи број аргумента који се прослеђује, што је у овом случају један. Ова функција ће опалити догађај из стека који ће бити ухваћен на страни претраживача у класи *TeatroV8BindingPluginBase* у функцији *fire_from_thread*.

```
void TeatroV8BindingPluginBase::fire_from_thread(std::unique_ptr<hbbtv_fire_calback_arg> data)
{
  struct hbbtv_fire_calback_arg * arg = data.get();

  Event* event;
```

Слика 26. fire_from_thread функција на страни претраживача за хватање догађаја

У наставку ће бити више речи о овоме, а сада остале функционалности функције *Testing_subtitles*. У случају да је послат број један као први аргумент из тестне апликације, врши се промена фонта титлова. Прво се подаци из другог аргумента конвертују у *TTMLFont* класу, затим се врши провера да ли је дужина *URL* већа од нуле, и да ли је дужина фонта већа од нуле, и у случају да није, ту излази из функције. Онда се прави *TeatroNativeVariant* низ са два члана пошто је потребно да се поставе два аргумента, један за *URL* и други за фонт који је послат преко *TtnplUtil*. Први аргумент подаци који ће бити смештени у други аргимент које припадник низа *TeatroNativeVariant*, и трећи аргумент дужина података . На крају се позива функција *dispatchEvent* , и као први аргумент се поставља назив догађаја „*onSubtitleFontFamily*“ који ће бити опапаљен, као други се ставља *TeatroNativeVariant* у који су смештени подаци за *URL* и породицу фонтова, и у трећи број аргумента који се прослеђује, што је у овом случају два.

У случају да се проследе два или три, прави се *TeatroNativeVariant* низ, дужине један који ће бити прослеђен *dispatchEvent*-у, и у њега се ставља као први аргумент у *TtnplUtil* истинита вредност ако је случај 2. или неистинита вредност ако је случај 3. У *dispatchEvent* се као први аргумент прослеђује назив догађаја „*onInBandSubtitlesShowHide*“, претходно направљен аргумент *TeatroNativeVariant* и број један за број аргумената. Случај четири је само испаљивање догађаја „*OnInBandSubtitleChange*“ преко *dispatchEvent* где су остали аргументи *NULL* и 0 пошто се не прави *TeatroNativeVariant* низ јер нема прослеђивање додатних података.

У класи *TeatroV8BindingPluginBase* у функцији *fire_from_thread* се наставља имплементација на страни претраживача.

```
if(!strcmp(arg->eventName, "onInBandSubtitleData")) {
  if(arg->args[0].details.type == TeatroNativeVariantType_String) {
    owner()->ParseInBandSubtitleData(arg->args[0].value.stringValue.value, arg->args[0].value.stringValue.length);
  } else {
    TEATRO_PLUGINS_LOG(TEATRO_LOG_LEVEL_ERROR) << PLUGIN_TAG << "Invalide type of in-band subtitle data";
  }

  return;
}

if(!strcmp(arg->eventName, "onSubtitleFontFamily")) {
  if(arg->args[0].details.type == TeatroNativeVariantType_String && arg->args[1].details.type == TeatroNativeVariantType_String) {
    owner()->ParseSubtitleFontFamily(arg->args[0].value.stringValue.value, arg->args[0].value.stringValue.length, arg->args[1].value.stringValue
  } else {
    TEATRO_PLUGINS_LOG(TEATRO_LOG_LEVEL_ERROR) << PLUGIN_TAG << "Invalide type of in-band subtitle data";
  }

  return;
}

if(!strcmp(arg->eventName, "onSubtitleSettingsChanged")) {
  if(arg->args[0].details.type == TeatroNativeVariantType_Int32 && arg->args[1].details.type == TeatroNativeVariantType_String) {
    owner()->SubtitlesSettingsChanged(arg->args[0].value.intValue, (char*)arg->args[1].value.stringValue.value, arg->args[1].value.stringValue.l
  }
}

if(!strcmp(arg->eventName, "onInBandSubtitleShowHide")) {
  if(arg->args[0].details.type == TeatroNativeVariantType_Bool) {
    owner()->ShowHideInBandSubtitleData(arg->args[0].value.boolValue);
  } else {
    TEATRO_PLUGINS_LOG(TEATRO_LOG_LEVEL_ERROR) << PLUGIN_TAG << "Invalide type of show/hide in-band subtitle data";
  }

  return;
}

if(!strcmp(arg->eventName, "onInBandSubtitleChange")) {
  owner()->ChangeInBandSubtitle();
  return;
}
```

Слика 27. fire_from_thread функција, део где се хватају догађаји

Преко *strcmp* се проверава који је догађај испаљен. *Owner()* предстаља инстанцу *html_object_element*-а, над којим онда постоји приступ свим функцијама те класе. У првом случају врши се провера да ли је име догађаја „*onInBandSubtitleData“*, и ако јесте, врши се провера типа *TeatroNativeVariant* података, да ли је он типа стринг. Ако јесте типа стринг, онда се позива *ParseInBandSubtitleData* из *html_object_element*-а, и прослеђују се стринг и

дужина стринга. У случају да није, шаље се испис логова о грешци.

Ако је испаљен догађај под називом *onSubtitleFontFamily*, врши се провера да ли је тип *TeatroNativeVariant* података стринг у оба послата аргумента. У случају да јесте, позива се функција из *html_object_element*-а *ParseSubtitleFontFamily* за мењање фонта, и шаљу јој се као параметри стринг за *URL* и дужина тог стринга, и стринг са породицом фонта. Ако тип података није био добар, шаље се испис логова о грешци.

За догађај „*OnInBandSubtitleShowHide*“ врши се провера да ли је послати тип података *TeatroNativeVariant*-а Булова промењива. Ако јесте, шаље се вредност Булове промењиве аргумента функцији *html_object_element*-а *ShowHideInBandSubtitleData*, а у супротном се испис логова о грешци. У случају да догађај са називом „*onInBandSubtitleChange*“ позива се функција *ChangeInBandSubtitle* из *html_object_element*-а.

```
void HTMLObjectElement::ParseInBandSubtitleData(const char* data, int size) {
    if(size <= 0)
      return;
    int subtitle_size = size;
    char *subtitle_data = (char*)malloc(subtitle_size);
    strncpy(subtitle_data, data, size);
    subtitle_renderer_container_->ParseAVPlayerInbandTextTrack(subtitle_data, subtitle_size);
}
```

Слика 28. *ParseInBandSubtitleData* функција

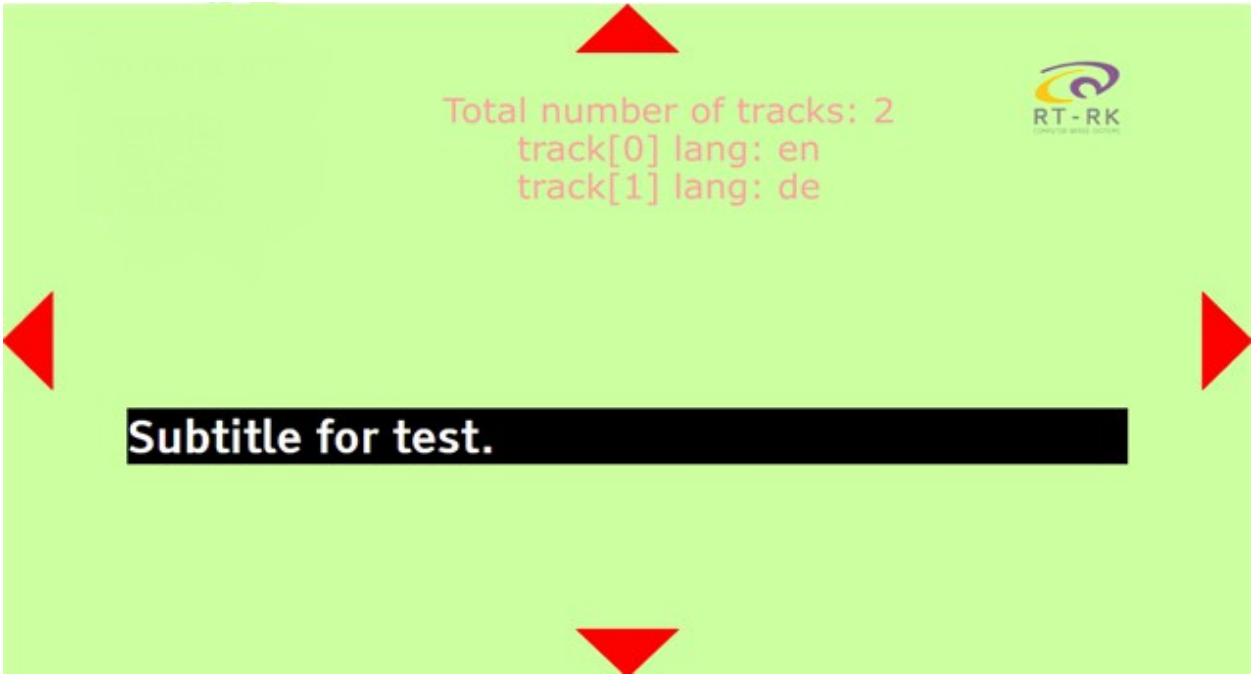
„*OnInBandSubtitleData*“ догађај окида ову функцију. Она прима као параметре податке, и њихову величину. Врши се провера да ли је величина већа од нуле. Пошто функција *ParseAVPlayerInbandTextTrack* прима обичан тип карактера, а не константне карактере, прави се низ карактера за који се динамчки алоцира меморија са *malloc*-ом у дужини који је послат као аргумент. Затим се врши копирање из података послатих преко првог аргумента у направљени низ преко *strncpy*. Након тога се то прослеђује до *html_media_element*-а преко функције *ParseAVPlayerInbandTextTrack* над *subtitle_renderer_container_* објектом. Ова функција је раније поменута декомопозиција функција, и та функција служи да се позове *InBandSubtitleWrapper* у *html_media_element*-у.

```
void HTMLMediaElement::InBandSubtitleWrapper(char* subtitleData, int subtitleSize) {
  LoadableInbandTextTrack* loadable_inband_text_track = LoadableInbandTextTrack::getActiveTrack();

  if(loadable_inband_text_track)
    loadable_inband_text_track->inBandSubtitleWrapper(subtitleData, subtitleSize);
  else if(last_loadable_inband_text_track_)
    last_loadable_inband_text_track_->inBandSubtitleWrapper(subtitleData, subtitleSize);
  else {
    DBG_ERROR("No available text track");
  }
}
```

Слика 29. InBandSubtitleWrapeer функција

Овде се прави показивач на објекат класе *LoadableInbandTextTrack* над којим позивамо функцију за добијање тренутне активне траке *getActiveTrack* из те класе. Затим се врши провера да ли та трака постоји, ако постоји, позива се над тим објектом *inBandSubtitleWrapper* из *LoadableInbandTextTrack* класе и прослеђују се подаци са титловима и величином тих података. Ако не постоји, врши се провера да ли постоји глобална промењива *last_loadable_inband_text_track* са последњом учитаном траком, и ако постоји, онда се позива *inBandSubtitleWrapper* из *LoadableInbandTextTrack* класе и прослеђују се подаци са титловима и величином тих података из аргумената. Ако не постоји, врши се испис логова да не постоји ни једна доступна текст трака. *InBandSubtitleWrapper* је функција која позива функције за рендеровање текст трака и парсирање послатих титлова до *ebutt_subtitle_engine* који је већ био имплементиран и његове функције *parse* која врши парсирање и приказ титлова.



Слика 30. Проширење тестне апликације учитавањем текстних трака са преводом

4.4. Контролне функционалности текстуалних трака унутар опсега

4.4.1 Приказивање и не приказивање текстуалних трака

На основу раније регистрованог догађаја „*OnInBandSubtitleShowHide*“ из *A/V* контролног додатка позива се функција из *html_object_element*-а *ShowHideInBandSubtitleData*, који позива функцију *ShowHideAVPlayerInbandTextTrack* из *html_media_element* класе над објектом *subtitle_renderer_container_* који наслеђује ту класу.

```
void HTMLObjectElement::ShowHideInBandSubtitleData(bool show_subtitle) {
    subtitle_renderer_container_->ShowHideAVPlayerInbandTextTrack(show_subtitle);
}
```

Слика 31. *ShowHideInBandSubtitleData* функција

Ово је декомпозицију функција, и позива функцију *VisibleInBandSubtitle* која прима послату вредност Булове промењиве.

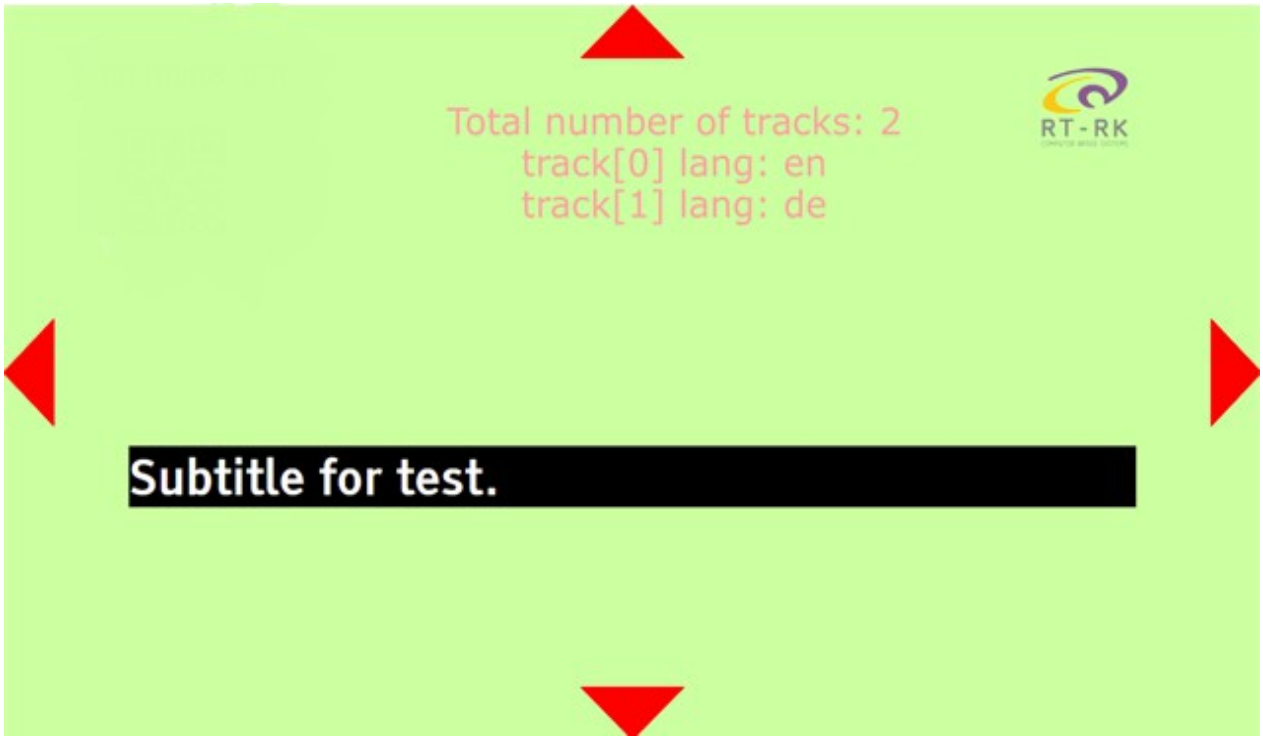
```
void HTMLMediaElement::VisibleInBandSubtitle(bool isVisible) {
  LoadableInbandTextTrack* loadable_inband_text_track = LoadableInbandTextTrack::getActiveTrack();

  if(loadable_inband_text_track){
      if(isVisible)
        loadable_inband_text_track->setModeFromClient(TextTrack::ShowingKeyword());
      else
        loadable_inband_text_track->setModeFromClient(TextTrack::HiddenKeyword());
  }
  else if(last_loadable_inband_text_track_) {
      if(isVisible)
        last_loadable_inband_text_track_->setModeFromClient(TextTrack::ShowingKeyword());
      else
        last_loadable_inband_text_track_->setModeFromClient(TextTrack::HiddenKeyword());
  }
  else {
      DBG_ERROR("No available text track");
  }
}
```

Слика 32. *VisibleInBandSubtitle* функција за приказивање и скривање текстуалних трака

Прво се прави показивач на *LoadableInbandTextTrack* и у њега се смешта тренутна трака која је активна. Затим се проверава да ли постоји та трака, и у случају да постоји, у зависности од аргумента који је послат функцији, се титлови приказују или не приказују помоћу функције *setModeFromClient*. У случају да не постоји тренутно активна трака, проверава се да ли постоји последња учитана текст трака, и ако постоји, на основу прослеђеног аргумента се траке приказују или не приказују помоћу функције *SetModeFromClient.*

Слика 33. Проширење тестне апликације са активираном опцијом за скривање текст трака



Слике 34. Проширење тестне апликације са активираном опцијом приказивања текст трака

4.4.2. Промена језика, заустављање текстуалних трака

```
case 5:
{
  SubtitleSettingsParams* settingsParams =  (SubtitleSettingsParams*)data;
  TeatroNativeVariant args[2];
  ttnplUtil::set(params.enabled, args[0]);
  ttnplUtil::set(params.language, args[1], params.language_length);
  dispatchEvent("onSubtitleSettingsChanged", args, 2);
}
  break;
default:
  break;
```

Слика 35. Регистровање догађаја за промену језика у *Testing_subtitles* функцији у *A/V* додатку

Догађај „*onSubtitleSettingsChanged*“ који је регистрован догађај у оквиру *A/V control plugin*-а под случај пет у *Testing_Subtitles* функцији. Почиње тако што примљене податке пребацује у *SubtitleSettingsParams**. Након тога прави низ од два члана структуре *TeatroNativeVariant*. *ttnplUtil* функција поставља у један члан тог низа да ли су текст траке омогућене, а у други члан име језика и дужина тог имена. Затим се позива функција *dispatchEvent* којој се прослеђује име догађаја, аргументи и број аргумената.

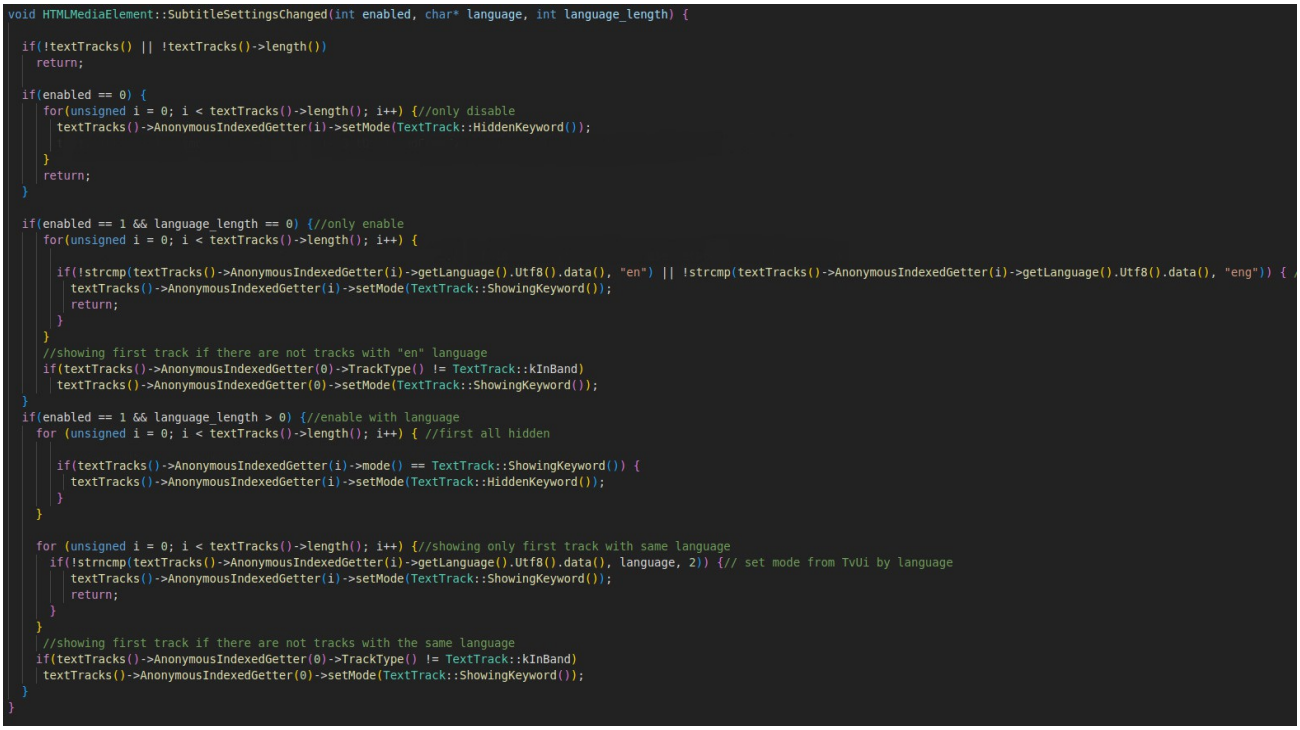
Овај догађај бива ухваћен на страни претраживача у *TeatroV8BindingPluginBase* класи *fire_from_thread* функцији, и даље позива *SubtitleSettingsChanged* функцију над *html_object-element*-ом.

```
void HTMLObjectElement::SubtitlesSettingsChanged(int enabled, char* language, int language_length) {
  if(subtitle_renderer_container_) {
    subtitle_renderer_container_->CallSubtitleSettingsChanged(enabled, language, language_length);
    if(enabled == 1) {
      if(!load_timer_.IsActive())
        load_timer_.StartRepeating(TimeDelta::FromMilliseconds(1), FROM_HERE);
    } else if(enabled == 0) {
      if(load_timer_.IsActive())
        load_timer_.Stop();
    }
  }
}
```

Слика 36. *SubtitlesSettingsChanged* функција

SubtitleSettingsChanged функција проверава да ли постоји *subtitle_render_container_* и у случају да да, над њим се позива функција *CallSubtitleSettingsChanged* и прослеђују јој се примљени аргументи, да ли се омогућује рад титловима, име језика и његова дужина. У случају да је омогућен рад титловима, активира се тајмер, ако није до сада био активан и то преко

*load_timer_* инстанце објекта *TimeDelta* и то да се понавља сваких милисекунд. У случају да је послато да није омогућен рад текстуалних трака, у случају да је тајмер укључен, он се зауставља. *CallSubtitleSettingsChanged* функција је декомпозиција функција, и она позива *SubtitleSettingsChanged* функцију у *html_media_element*-у.



```
void HTMLMediaElement::SubtitleSettingsChanged(int enabled, char* language, int language_length) {

  if(!textTracks() || !textTracks()->length())
    return;

  if(enabled == 0) {
    for(unsigned i = 0; i < textTracks()->length(); i++) {//only disable
      textTracks()->AnonymousIndexedGetter(i)->setMode(TextTrack::HiddenKeyword());

    }
    return;
  }

  if(enabled == 1 && language_length == 0) {//only enable
    for(unsigned i = 0; i < textTracks()->length(); i++) {

      if(!strcmp(textTracks()->AnonymousIndexedGetter(i)->getLanguage().Utf8().data(), "en") || !strcmp(textTracks()->AnonymousIndexedGetter(i)->getLanguage().Utf8().data(), "eng")) {
        textTracks()->AnonymousIndexedGetter(i)->setMode(TextTrack::ShowingKeyword());
        return;
      }
    }
    //showing first track if there are not tracks with "en" language
    if(textTracks()->AnonymousIndexedGetter(0)->TrackType() != TextTrack::kInBand)
      textTracks()->AnonymousIndexedGetter(0)->setMode(TextTrack::ShowingKeyword());
  }
  if(enabled == 1 && language_length > 0) {//enable with language
    for (unsigned i = 0; i < textTracks()->length(); i++) { //first all hidden

      if(textTracks()->AnonymousIndexedGetter(i)->mode() == TextTrack::ShowingKeyword()) {
        textTracks()->AnonymousIndexedGetter(i)->setMode(TextTrack::HiddenKeyword());
      }
    }

    for (unsigned i = 0; i < textTracks()->length(); i++) {//showing only first track with same language
      if(!strncmp(textTracks()->AnonymousIndexedGetter(i)->getLanguage().Utf8().data(), language, 2)) {// set mode from TvUi by language
        textTracks()->AnonymousIndexedGetter(i)->setMode(TextTrack::ShowingKeyword());
        return;
      }
    }
    //showing first track if there are not tracks with the same language
    if(textTracks()->AnonymousIndexedGetter(0)->TrackType() != TextTrack::kInBand)
      textTracks()->AnonymousIndexedGetter(0)->setMode(TextTrack::ShowingKeyword());
  }
}
```

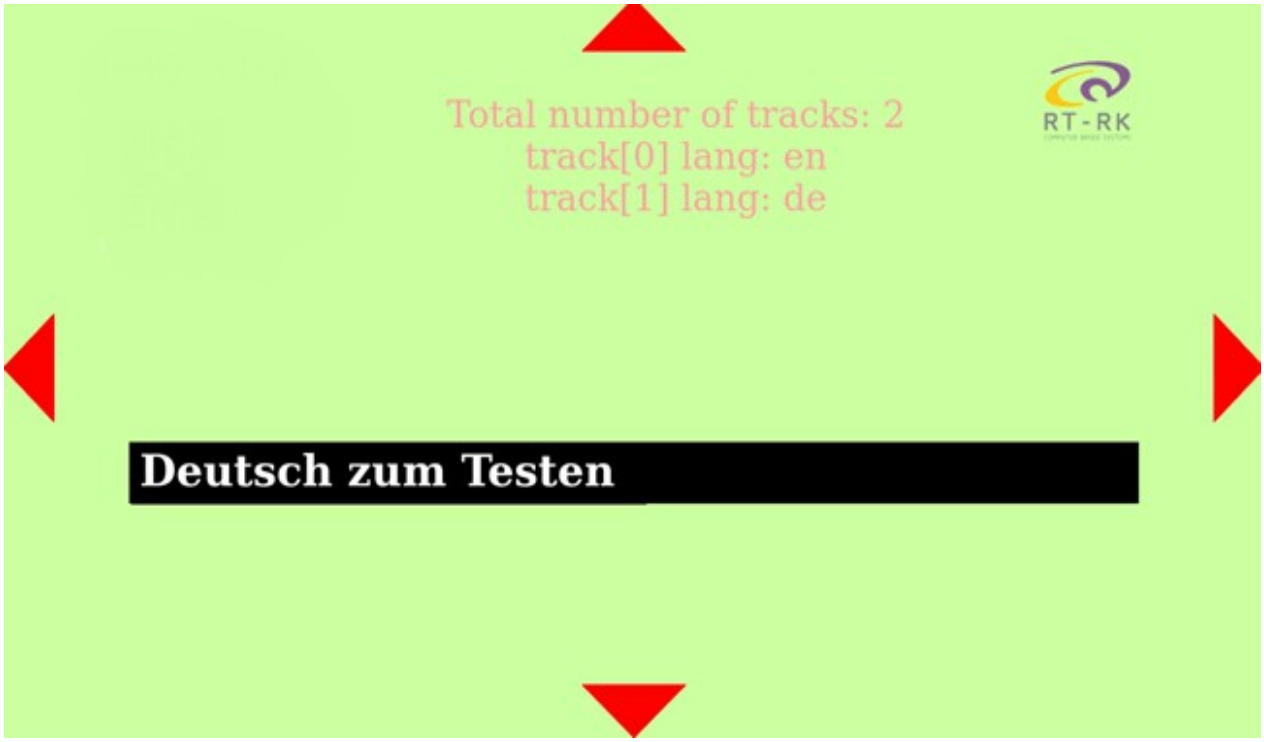
Слика 37. *SubtitleSettingsChanged* функција за омућавање заустављање и промену језика трака

Ова функција прво проверава да ли постоје траке у листи трака, да се изађе из функције у случају да нема.

Ако је послато као први аргумент да су титлови онемогућени, пролази се кроз листу учитаних трака, поставља се да су невидљиве и затим се излази из функције. Ако су послате да су само траке омогућене, без промене језика, проверава се да ли постоје траке на енглеском језику и ако постоје, траке се постављају да буду видљиве и излази се из функције. У случају да не постоји трака на енглеском, узима се прва трака која постоји и поставља се да буде видљива.

Ако су омогућени титлови и ако је дужина послатог језика већа од нуле, пролази се кроз листу трака и прво се сакривају све траке. Затим се поново пролази кроз све траке које се налазе у листи трака, и пореде се са називом послатог језика, и само та трака која се подудара ставља се да се приказује и излази се из функције. Ако се ни једна трака не

подудара са послатим стрингом, онда се показује прва трака која је доступна.

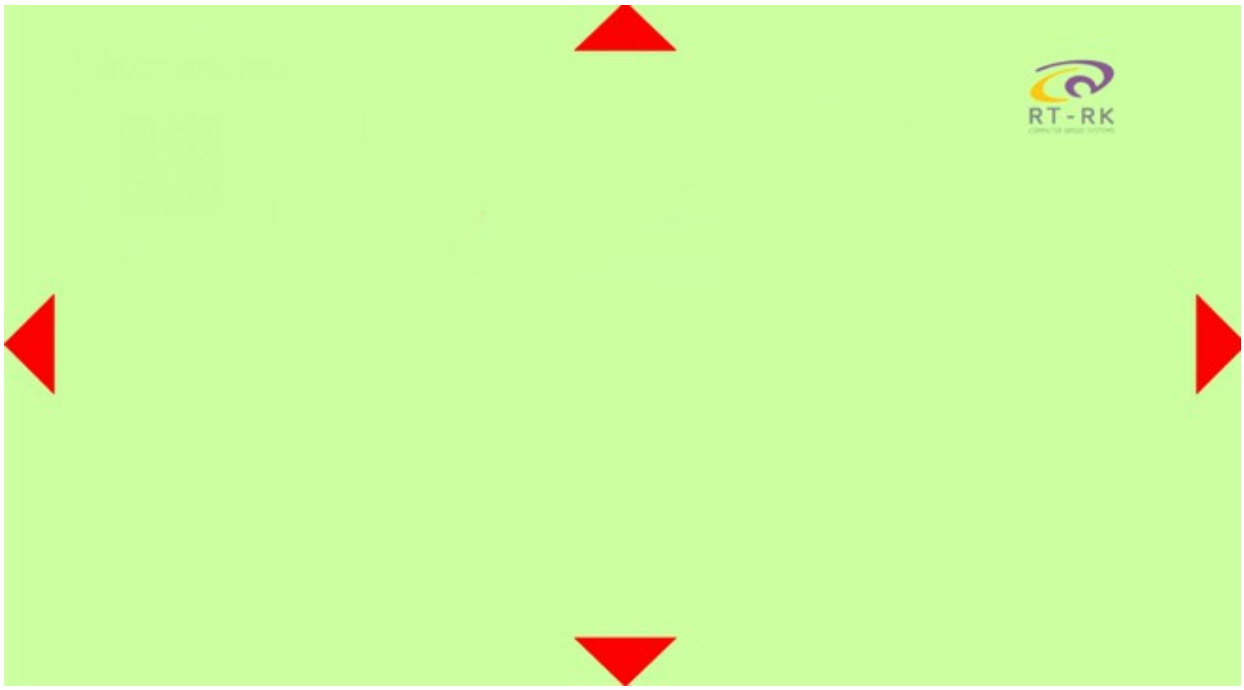


Слика 38. Проширење тестне апликације додатком функије за промену језика трака

5. Резултати

У овом поглављу биће представљена табела са резултатима тестирања и верификацијом, као и кратко описана остварена функционалност.

5.1. Учитана тестна *javascript* апликација помоћу *Teatro Tv Browser*-а.

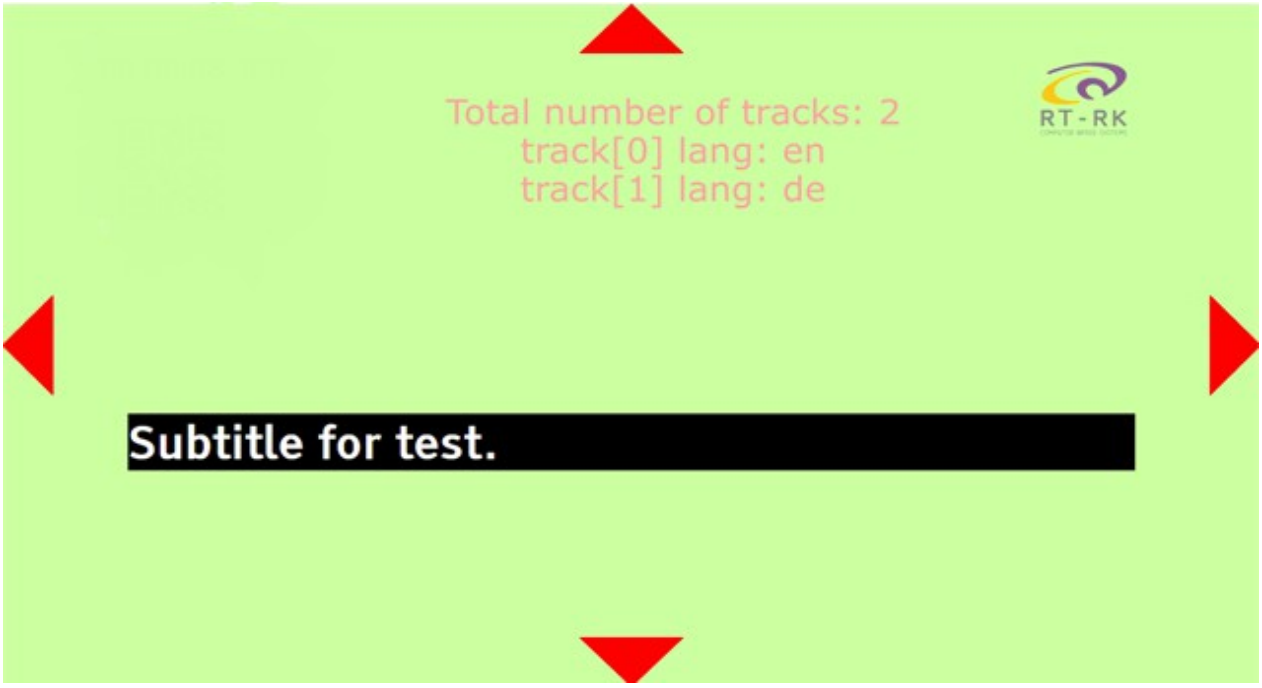


Слика 39. Резултат учитавања тестне апликације помоћу *Teatro Tv Browser*-а

5.2. Проширење тестне апликације из претходног задатка и користећи одговарајуће методе и својства *A/V* контролног додатка и исписивање укупног броја као и језике сигнализираних трака.

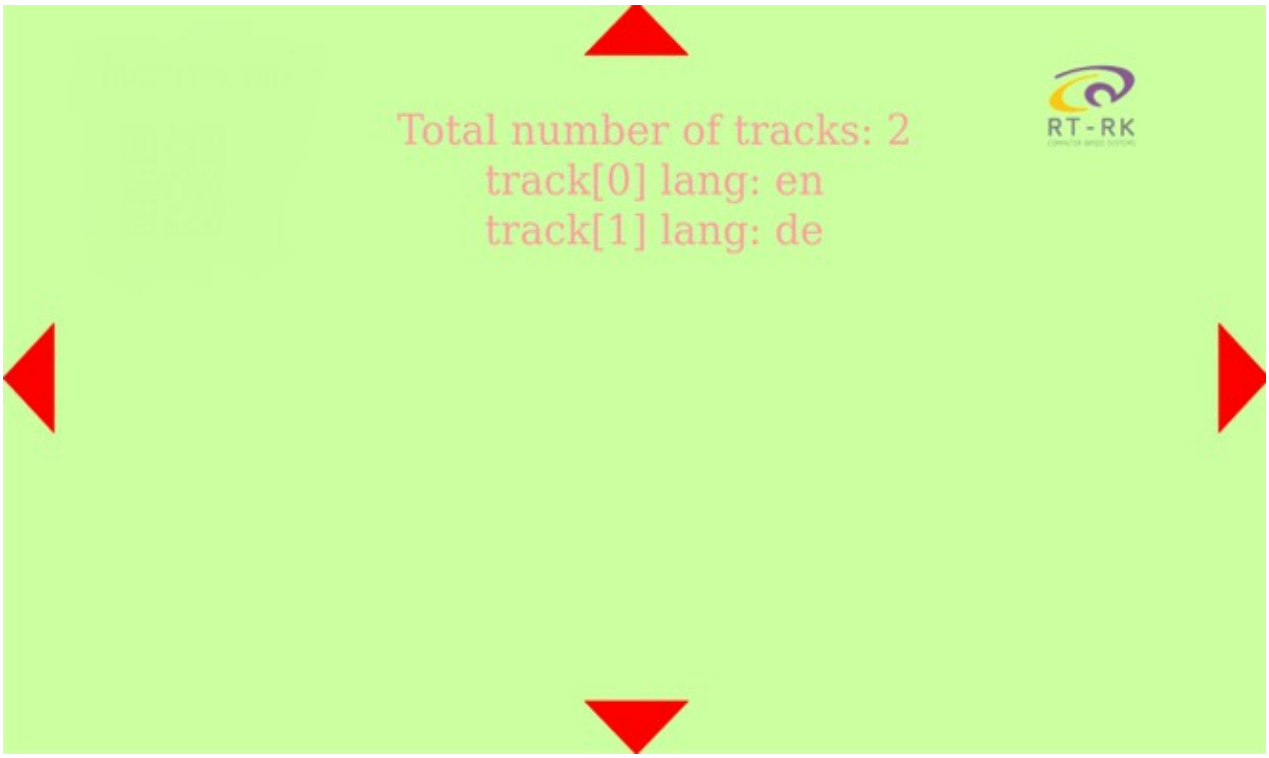
Слика 40. Резултат парсирања сигнализираних трака и приказа на тестној апликацији

5.3. Подршка и приказ *ebu-tt-d* формата титлова у опсегу у оквиру формата АВ контролног додатка у реалном времену.

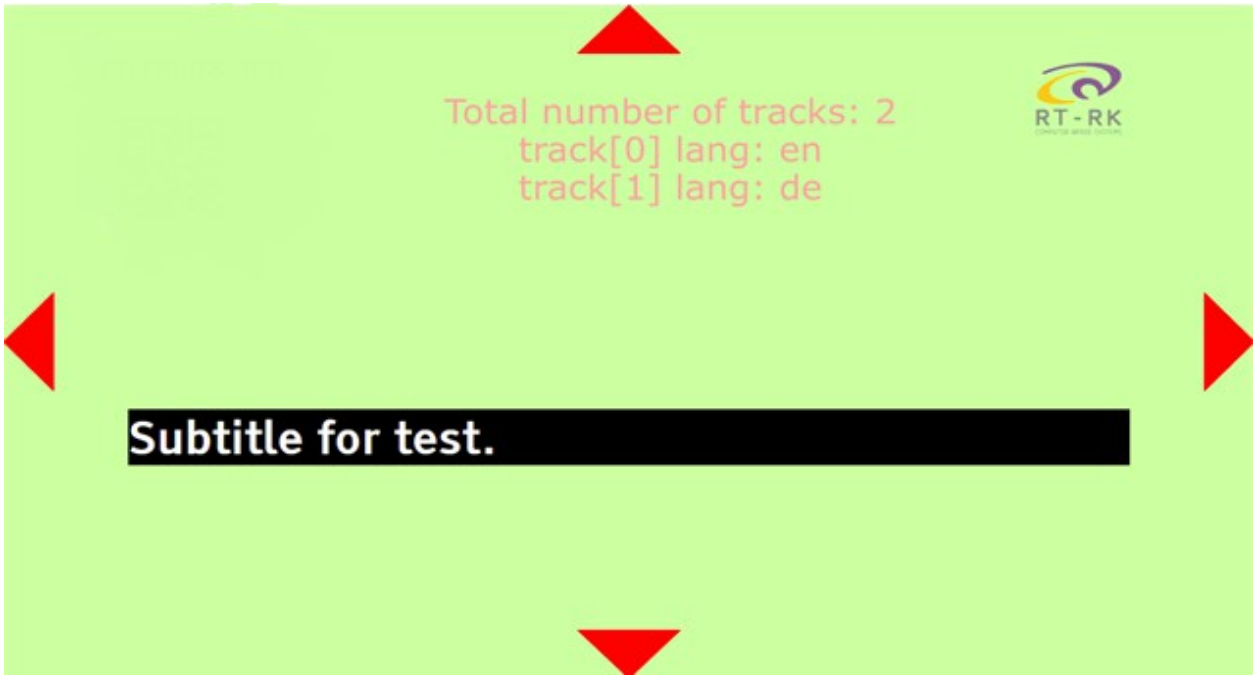


Слика 41. Резултат учитавања *XML* трака и њихов приказ на тестној апликацији

5.4. Подршка функционалности приказа и скривања текст трака из терминала уређаја

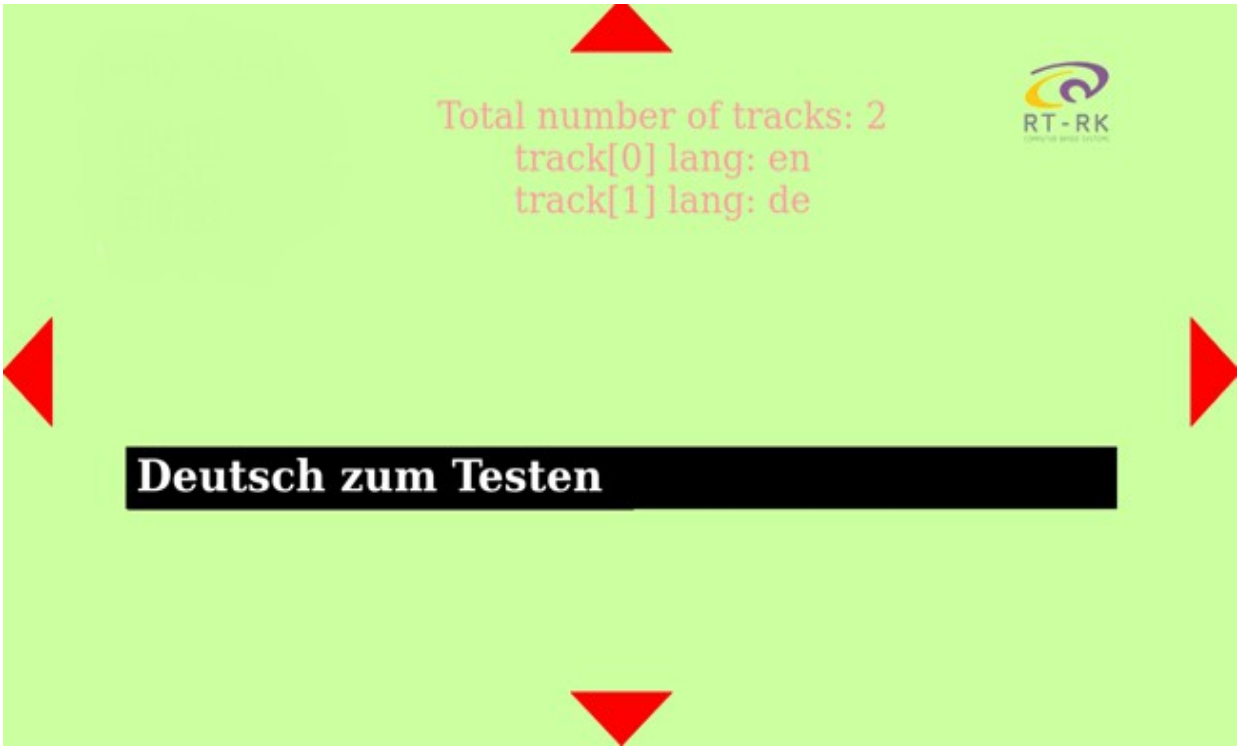


Слика 42. Резултат убацивања подршке за функционалност сакривања текстуалних трака



Слика 43. Резултат убацивања подршке за функционалност приказивања текстуалних трака

5.5 Промена језика трака са преводом



Слика 44. Резултат убацивања подршке за функционалност промене језика

Верификација *HbbTv* тестова је извршена на *Sony TPV Tv 32-inch LCD* плочи.

Назив теста	Учитавање	Парсирање	Тачан временски интервал приказивања	Проценат успешности теста
org.hbbtv_SUB0026	Да	Да	Да	100/100
org.hbbtv_SUB0028	Да	Да	Да	100/100
org.hbbtv_SUB0029	Да	Да	Да	100/100
org.hbbtv_SUB0210	Да	Да	Да	100/100
org.hbbtv_SUB0220	Да	Да	Да	100/100
org.hbbtv_SUB0290	Да	Да	Да	100/100
org.hbbtv_SUB0350	Да	Да	Да	100/100
org.hbbtv_SUB0370	Да	Да	Да	100/100
org.hbbtv_SUB0600	Да	Да	Да	100/100
org.hbbtv_SUB0610	Да	Да	Да	100/100

Табела 1. Резултати верификације тестова на *Sony TPV Tv 32-inch LCD* плочи

6. Закључак

У овом раду је реализована програмска интеграција и подршка у опсегу *EBUTT-D* формата титлова коришћењем *A/V* контролног додатка у реалном времену. Подржане су функционалности пуштања и заустављања трака са преводом у реалном времену, као и опције приказивања и сакривања трака и промена језика.

Решење је тестирано на *Sony TPV Tv 32“ LCD*.

Даљи рад се може вршити у правцу проширивања функционалности у виду мењања фонта титлова, промене величине трака, промене боје позадине поља трака или оптимизације постојећег кода.

7. Литература

[1] https://www.computerworld.com/article/3261009/googles-*Chromium*-browser-explained.html

[2] https://tech.ebu.ch/publications/tech3380, **EBU-TT-D SUBTITLING FORMAT**

[3] https://www.hbbtv.org/wp-content/uploads/2020/10/HbbTV-SPEC-00525-HbbTV-SPEC-00515-008-hbbtv203_2020_10_14.pdf, **HbbTv 2.0.3 Specification**

[4] https://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.04.01_60/ts_102796v010401p.pdf, **Hybrid Broadcast Broadband Tv**

[5] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/track