sugarlabs

Sugarizer Measure Activity

Introduction

Name - Adityakumar Sinha

Emails - adityakumar113141@gmail.com (Primary)
adityakumar.sinha@students.iiit.ac.in (University)

Github - aditya113141

IRC (freenode) : aditya113141

LinkedIn
https://www.linkedin.com/in/adityakumar-sinha-485a40193/

Website - https://aditya113141.github.io/

Languages - Hindi (Native) and English (Native)

Location - I am currently living in Vadodara, India. My timezone is UTC+05:30. I would like to work from 9:30 AM to 10:00 PM. In case of emergencies, it can be changed to 8:00 AM to 11:30 PM. I am ready to work even more if the necessity arises.

Degree : Bachelor in Technology

Major : Electronics and Communication Engineering

Institute : International Institute of Information Technology, Hyderabad

Resume : Click here

My Motivation

What is your motivation to take part in Google Summer of Code ?

I want to be a part of Google Summer of Code because of the learning experience that I will get and a chance to network and make friends with cool programmers. This will give me real exposure to the professional coding world and will help me develop as a programmer.

Why did you choose Sugar Labs ?

I came to know about Sugar Labs in December 2020. I researched about the organisation and then came to know about the OLPC initiative and the motive behind the Sugar Project. I have always wanted to contribute in the field of education and working with Sugar Labs will give me an opportunity to work for society.

Why do you want to work on this particular project ?

Being from an Electronics background, I have always been fascinated by signals. I have used an oscilloscope in the lab previously and was curious to explore this activity and find how we can implement it on a software. I have a good knowledge of signals and the underlying concepts like fast fourier transform, filters, inversion, Amplitude scaling, time period scaling, frequency spectrum etc. which are the core ideas of this project. I will love to take this up as my project for Google Summer of Code 2021.

What are your expectations from us during and after completion of the program ?

During the program, I expect the organisation and the mentors to be helpful and friendly like they have always been on the sugar-mail-list . I expect weekly feedback on my work from mentors (apart from the evaluations). After the program ends, my plan is to continue working with Sugar Labs and be a mentor to other participants in the upcoming years.

About My Project

My project is about creating a Sugarizer Measure Activity equivalent to Sugar Measure Activity.

Features from the original activity :-

- Both Waveform and Frequency Spectrum are to be plotted. We will switch between Waveform and Frequency Spectrum using the toolbar buttons.

- Inverting the Waveform.

- Amplitude Scaling - We will make a slider to scale the amplitude of waveform / frequency spectrum.

- Time Scaling - We will make a slider to scale the time period of waveform / frequency spectrum.

- Multi Waveform Display - We will allow multi waveform display. But there won’t be any Multi Spectrum display. (Saying from personal experience on working with oscilloscopes in labs, if we plot two frequency spectrums on the same display, they get mixed up and it is difficult to analyse them. It is better to analyse them separately.)

- Play / Pause the Waveform / frequency spectrum.
- Capture the sample and export it.

New Features :-

- Color Palettes - To change color of waveform / frequency Spectrum

- Sample Audio files - Nursery Rhymes in User's language and instrument themes from Abecedarium.

General Sugarizer features:-

- Cross Platform : The activity will have the same functionality and behaviour on anybrowser (Chrome, Firefox, Safari) and any platform (Android, iOS, Windows, Linux, MacOS) supported by Sugarizer.

- Sugarizer Storage - Load / Save content into the Journal.

- Multi User Environment

- Autosave

- Localisation

- Full Screen mode and Responsiveness

- The User interface of the activity will have Sugarizer look & feel . We will use Sugartoolbar and palette to show the features of the activity .

- Offline availability : All the main features will work smoothly without the requirement of any internet connection which will help to run this activity even in remote areas where internet connection is not available easily.

- Tutorial - An integrated documentation will be integrated to explain each feature of the activity

How will it impact Sugar Labs ?

- The new Sugarizer Measure Activity will familiarise the children with the concepts of sound, waves, signals.

- Children will experiment with their device microphone, try singing, whistling and musical instruments on this activity.

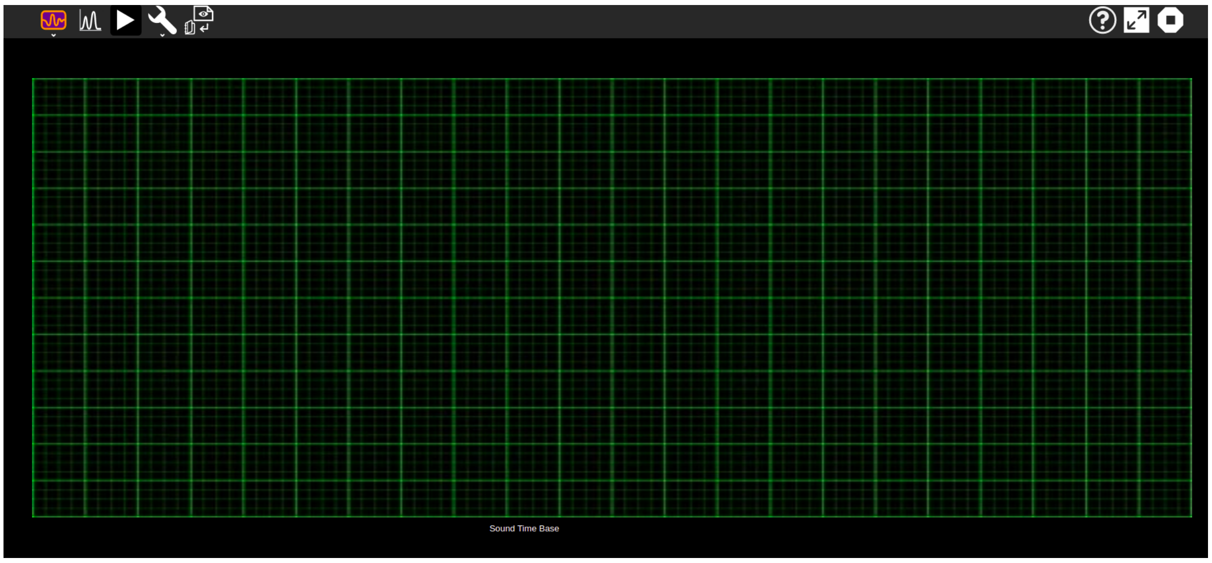
- They will visualise the various properties of sound - Amplitude, waveform, frequency spectrum and this will enhance their knowledge.

- Addition of Measure Activity to Sugarizer will expand Sugar Labs' reach, allowing more children to participate and learn with us.

Implementation

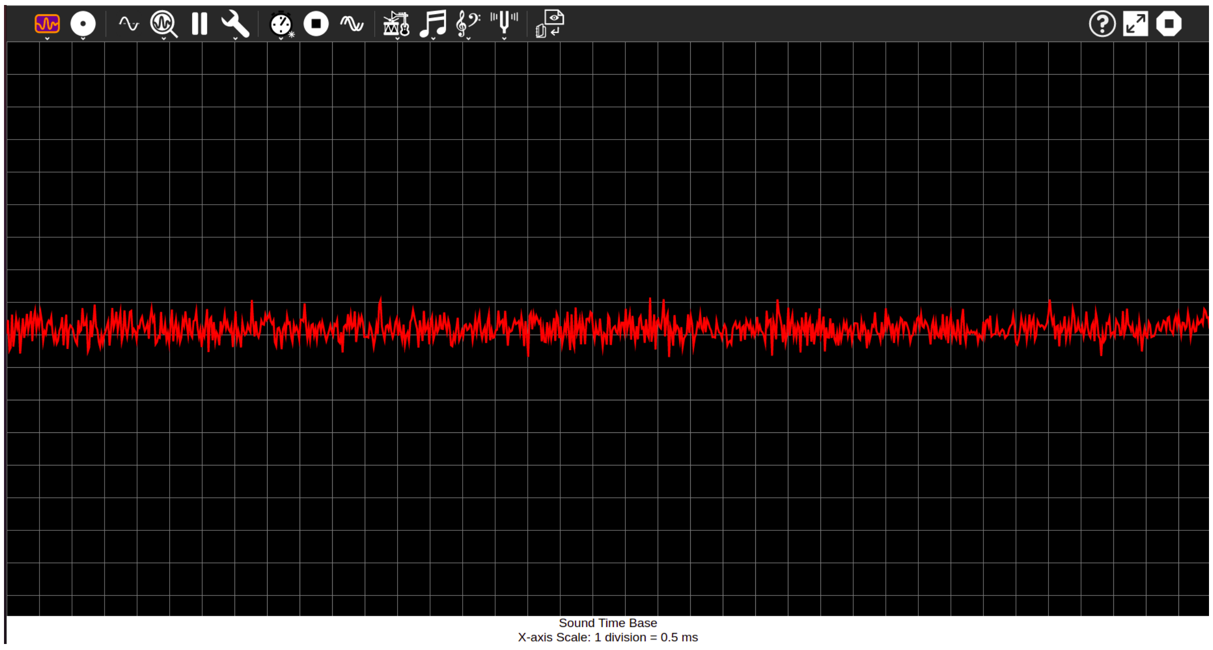
I found two incomplete implementations of Sugarizer Measure activity.

- AbhishekTanwar's version - This version has the implementation of Waveform / Frequency spectrum with time period and amplitude scaling features. Going from the comments on the Pull request highlighted in the link, I understand that this version is not compatible with **Chrome.** Also, the problem with it is the freezing of the waveform / frequency spectrum. This version uses screen capture to take a screenshot of the canvas and display it as an image.



  - I tried understanding the implementation (using p5.js) and found an alternative for pausing the waveform / frequency spectrum. This can be found here. I used **noLoop()** and **loop()** functions. **noLoop()** freezes everything in the canvas and **loop()** resumes it. This alternative can eliminate screen capture and solve the double background issue.
    If you are opening the link, I would suggest you open it on **Firefox**.

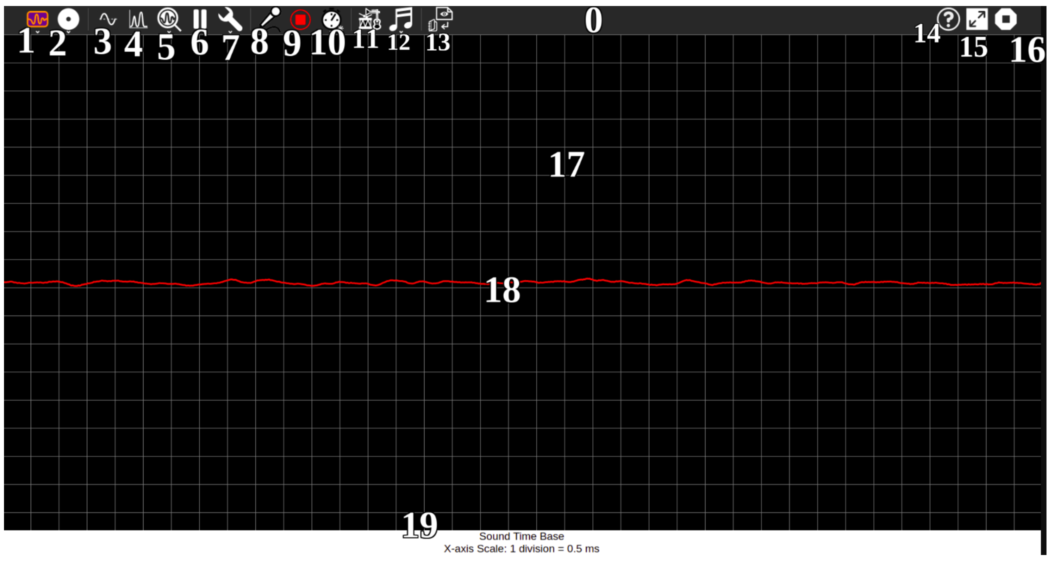
- Sarthak-g's version - This version doesn't use p5.JS . Instead, it uses Web Audio API. It already has the implementation of waveform.



My Planning

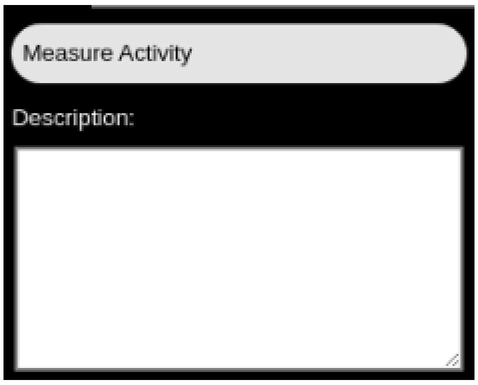
- I will work on Sarthak-g's version.

Proposed UI

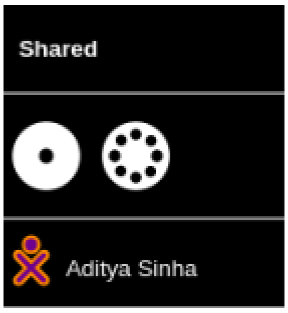


0. **Toolbar**

1. **Activity Button -** Like every other Sugarizer Activity, there is an activity button here with a drop down palette with description.



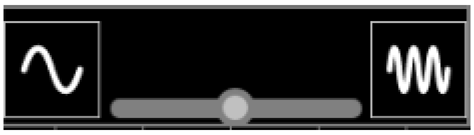
2. **Network Button -** The Network Button will allow us to switch between a single user and multi-user environment.



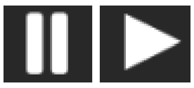
3. **Time-Domain-Button -** To plot the waveform of the Signal

4. **Freq-Domain-Button -** To plot the frequency spectrum of the signal. It will also have a color palette.

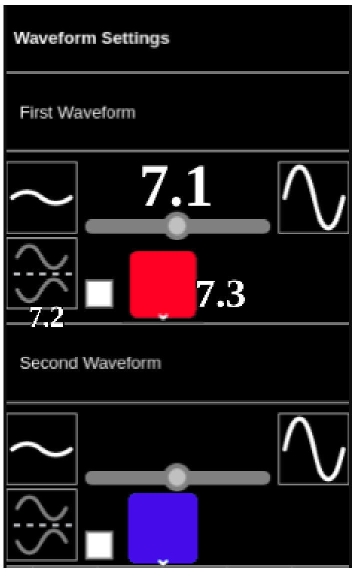
5. **Time Period Scaling -** This will scale the x-axis such that it looks like the time period of waveform and frequency spectrum changes. If we scale down the x-axis the waveform will contract but frequency spectrum will expand and vice versa. This is due to the inverse proportionality between time and frequency.



6. **Play / Pause Button** - If the button is in Pause State, the waveform / frequency spectrum will be stationary on canvas and can be captured and exported as pdf / image.



7. Waveform Settings



7.1 - Slider for changing the Amplitude of Waveform / Frequency Spectrum

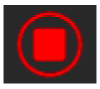
7.2 - Invert Option. If the whitebox is ticked, then the waveform / is inverted

7.3 - Color Palette

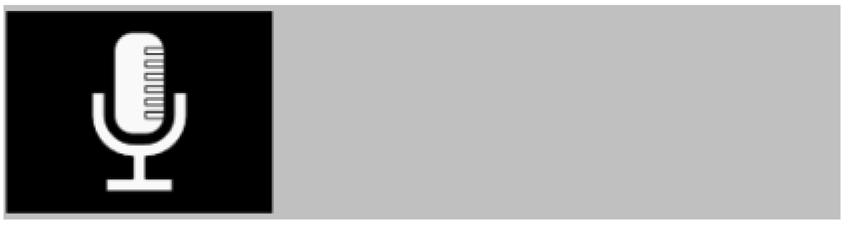
It will be disabled when the user will be in Frequency Spectrum mode.

8. **Mic Button** - To take audio input from device microphone and plot real time waveform / frequency spectrum. It will have a Disable option.

9. **Record Button** - To record audio from the mic.

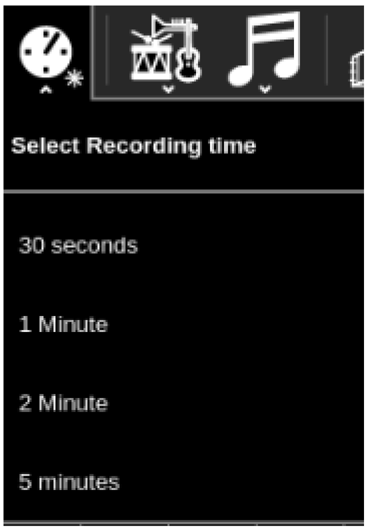


The recorded audio is going to be displayed as:-



It will have a disable option.

10. **Recording Time -** To select the recording duration



Select Recording time

30 seconds

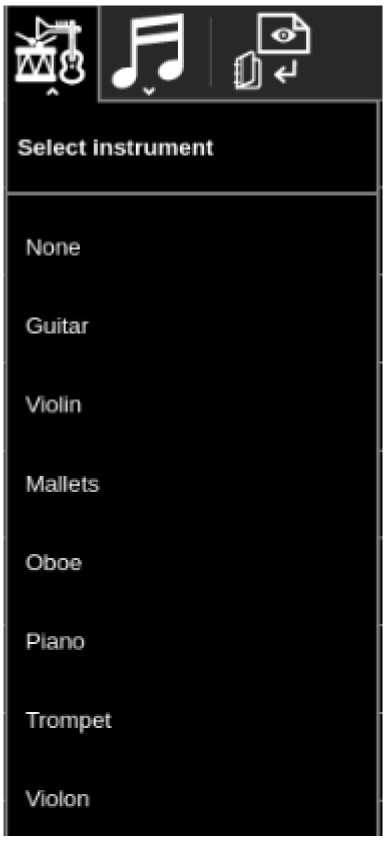
1 Minute

2 Minute

5 minutes

Default Value will be 30 seconds
It will be enabled only when the user selects the Record Button.

11. **Sample Instruments -** Sample Audio files imported from Abecedarium. They can be played inside the activity along with the display of their waveform / frequency spectrum.


Select instrument

None

Guitar

Violin

Mallets

Oboe

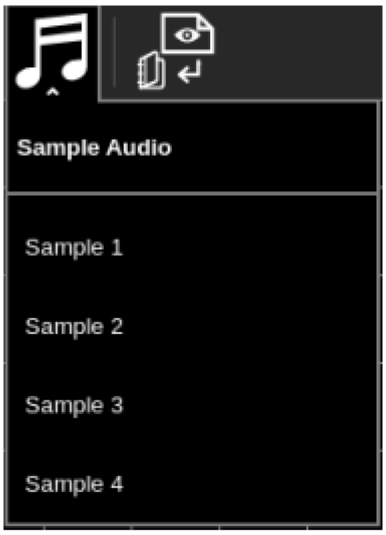
Piano

Trompet

Violon

,

12. **Sample Audio -** Nursery Rhymes in user’s language. Can be played inside the activity along with the display of their waveform / frequency spectrum.


Sample Audio

Sample 1

Sample 2

Sample 3

Sample 4

13. **Waveform / Frequency Spectrum Capture -** Capture waveform as pdf/image

14. **Tutorial Button -** Like every other activity, we will be having a tutorial for it.

15. **Full screen Button**

16. **Stop Button**

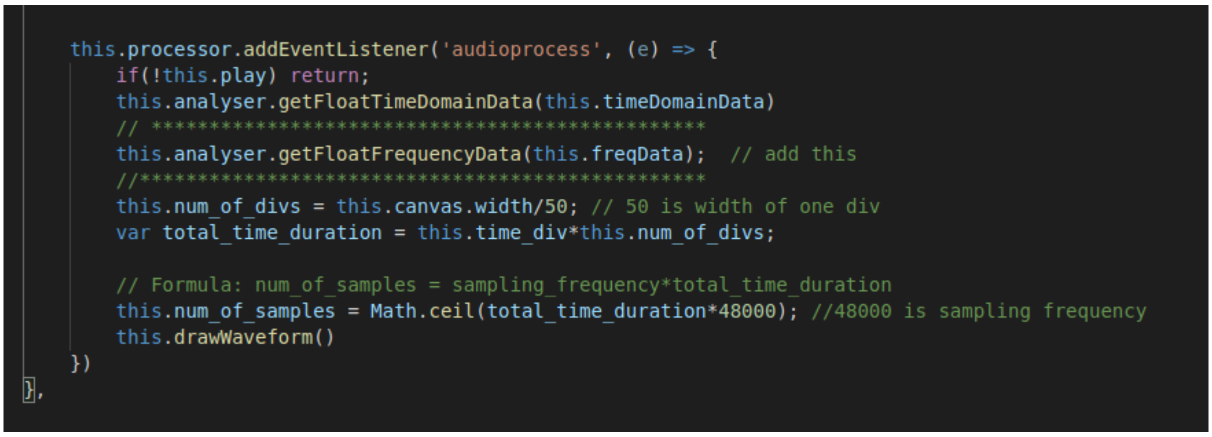
17. **Canvas -** Waveforms / Frequency Spectrum are plotted here.

18. **Waveform -** A waveform displayed in image. Similarly we can have frequency spectrum too.

19. **Footer with information about Graph Scale**

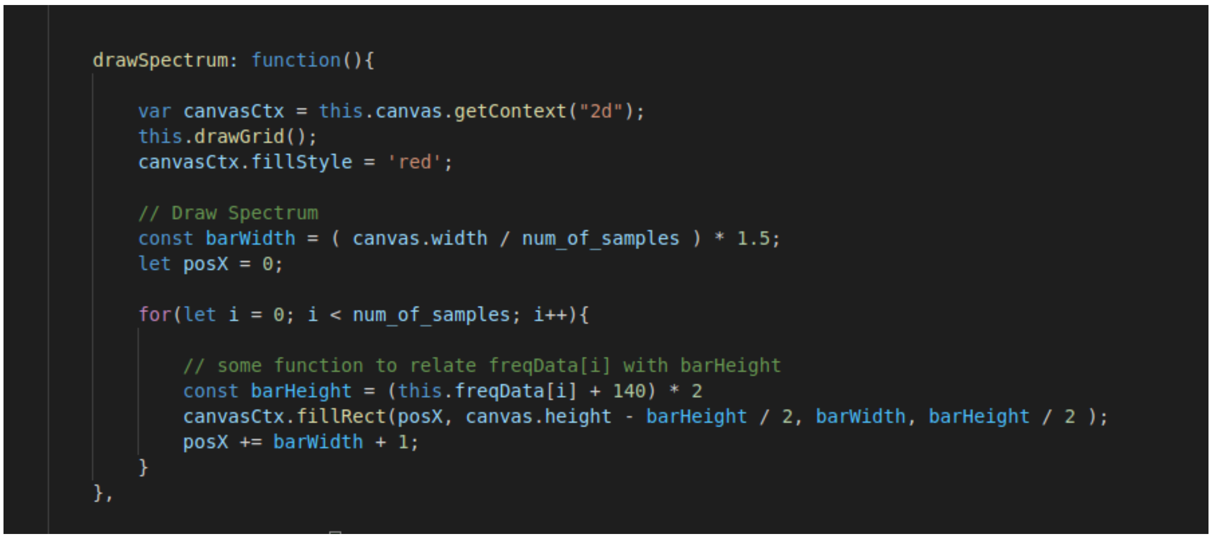
Frequency Spectrum

- We can use getFloatFrequencyData () and fill the freqData[] array, similar to filling time data from getFloatTimeDomainData() into timeDomainData [] array.



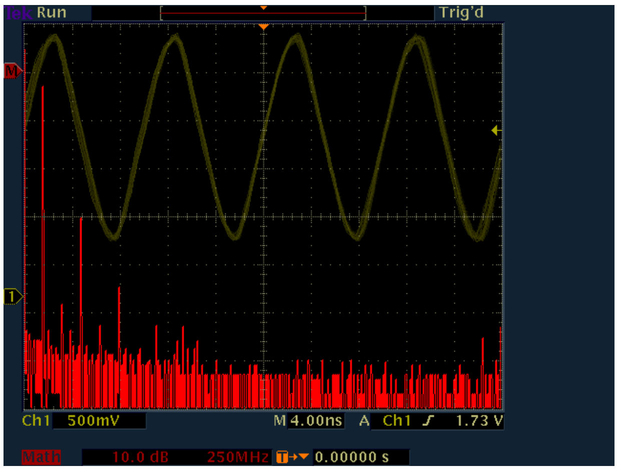
```
    this.processor.addEventListener('audioprocess', (e) => {
        if(!this.play) return;
        this.analyser.getFloatTimeDomainData(this.timeDomainData)
        // ************************************************
        this.analyser.getFloatFrequencyData(this.freqData);  // add this
        //*************************************************
        this.num_of_divs = this.canvas.width/50; // 50 is width of one div
        var total_time_duration = this.time_div*this.num_of_divs;

        // Formula: num_of_samples = sampling_frequency*total_time_duration
        this.num_of_samples = Math.ceil(total_time_duration*48000); //48000 is sampling frequency
        this.drawWaveform()
    })
},
```

- Using the freqData[], we will construct frequency spectrum using this method



```
drawSpectrum: function(){

    var canvasCtx = this.canvas.getContext("2d");
    this.drawGrid();
    canvasCtx.fillStyle = 'red';

    // Draw Spectrum
    const barWidth = ( canvas.width / num_of_samples ) * 1.5;
    let posX = 0;

    for(let i = 0; i < num_of_samples; i++){

        // some function to relate freqData[i] with barHeight
        const barHeight = (this.freqData[i] + 140) * 2
        canvasCtx.fillRect(posX, canvas.height - barHeight / 2, barWidth, barHeight / 2 );
        posX += barWidth + 1;
    }
},
```

We will construct blocks of very thin width which will look like this:-



Snap of original oscilloscope

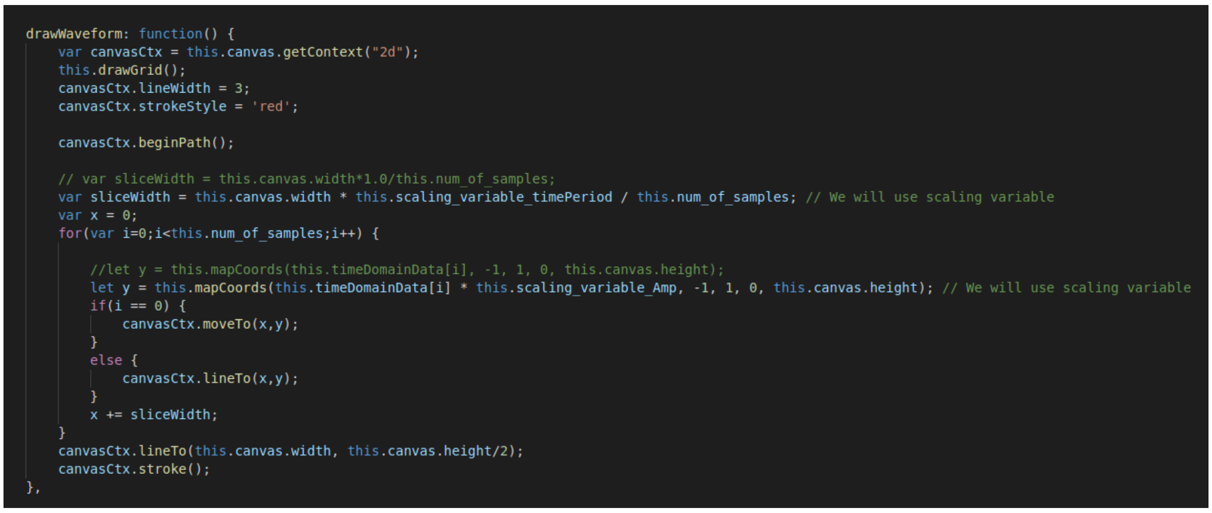
Amplitude and Time Period Scaling

- To change the amplitude of the waveform / frequency spectrum, we need to multiply the elements of freqData[ ] and timeDomainData [ ] by some number. In this way the waveform / frequency spectrum will scale up or scale down.

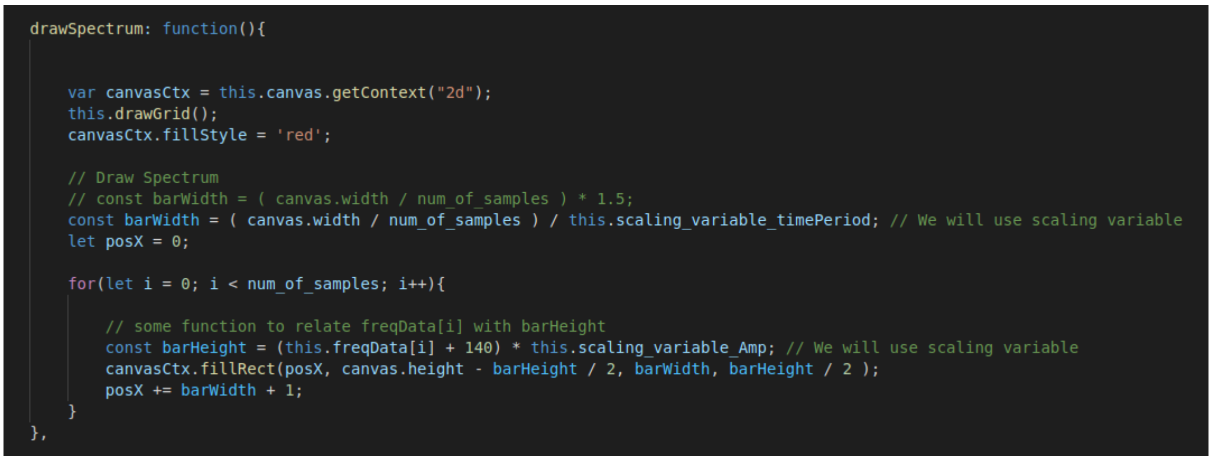
- To change the Time period of the waveform / frequency spectrum, we need to change the :-

    - sliceWidth in case of waveform
    - BarWidth in case of Frequency Spectrum

We can keep a scaling variable for this.



```
drawWaveform: function() {
    var canvasCtx = this.canvas.getContext("2d");
    this.drawGrid();
    canvasCtx.lineWidth = 3;
    canvasCtx.strokeStyle = 'red';

    canvasCtx.beginPath();

    // var sliceWidth = this.canvas.width*1.0/this.num_of_samples;
    var sliceWidth = this.canvas.width * this.scaling_variable_timePeriod / this.num_of_samples; // We will use scaling variable
    var x = 0;
    for(var i=0;i<this.num_of_samples;i++) {

        //let y = this.mapCoords(this.timeDomainData[i], -1, 1, 0, this.canvas.height);
        let y = this.mapCoords(this.timeDomainData[i] * this.scaling_variable_Amp, -1, 1, 0, this.canvas.height); // We will use scaling variable
        if(i == 0) {
            canvasCtx.moveTo(x,y);
        }
        else {
            canvasCtx.lineTo(x,y);
        }
        x += sliceWidth;
    }
    canvasCtx.lineTo(this.canvas.width, this.canvas.height/2);
    canvasCtx.stroke();
},
```

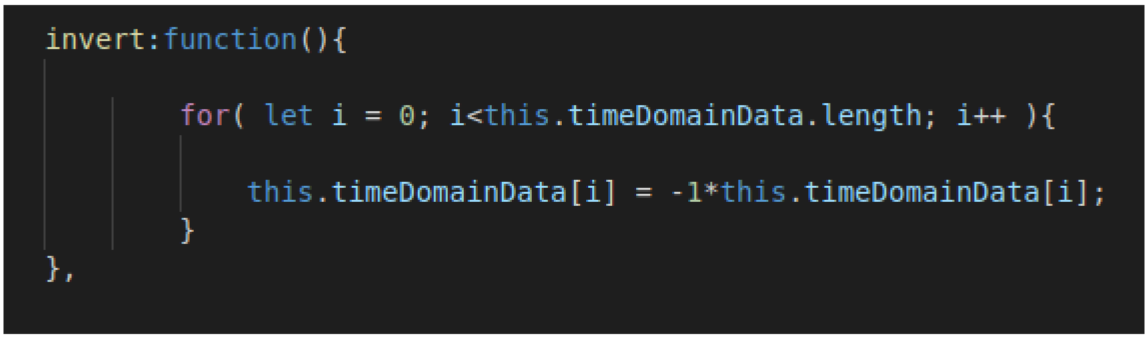


```
drawSpectrum: function(){


    var canvasCtx = this.canvas.getContext("2d");
    this.drawGrid();
    canvasCtx.fillStyle = 'red';

    // Draw Spectrum
    // const barWidth = ( canvas.width / num_of_samples ) * 1.5;
    const barWidth = ( canvas.width / num_of_samples ) / this.scaling_variable_timePeriod; // We will use scaling variable
    let posX = 0;

    for(let i = 0; i < num_of_samples; i++){

        // some function to relate freqData[i] with barHeight
        const barHeight = (this.freqData[i] + 140) * this.scaling_variable_Amp; // We will use scaling variable
        canvasCtx.fillRect(posX, canvas.height - barHeight / 2, barWidth, barHeight / 2 );
        posX += barWidth + 1;
    }
},
```

- The value of scaling variables will change according to the sliders.

Inverting the waveform

- For **inverting** the waveforms, we will simply multiply the elements of timeDomainData [ ] by -1.



```
invert:function(){

        for( let i = 0; i<this.timeDomainData.length; i++ ){

            this.timeDomainData[i] = -1*this.timeDomainData[i];
        }
},
```

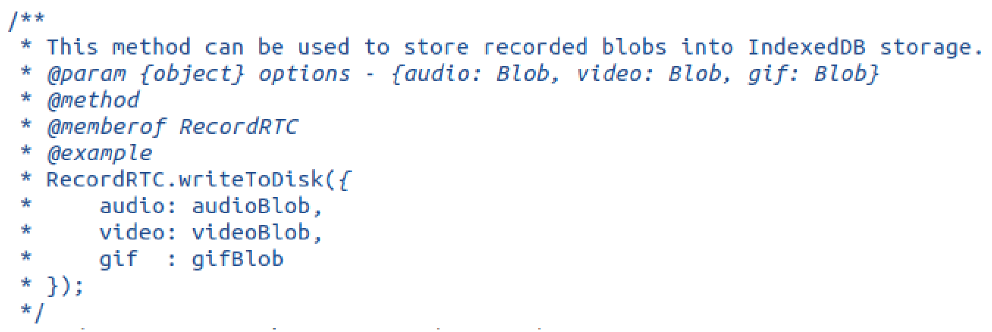
Recording

- For recording the audio we can use the audio recording feature from createHelper script of Record Activity. It uses the Record RTC library. For playing the audio, we can again take help from createHelper script as described here. It uses the HTML DOM Audio object.

- I am planning to create a separate VueJs component **SugarRecord.** I took inspiration from the **SugarSpeak** component which uses the meSpeak library. I will integrate the **SugarRecord** component in my project. The **SugarRecord** component can be used in future with other activities too. It will be based on Record RTC library.

- The recording time will be adjusted according to the selected interval.

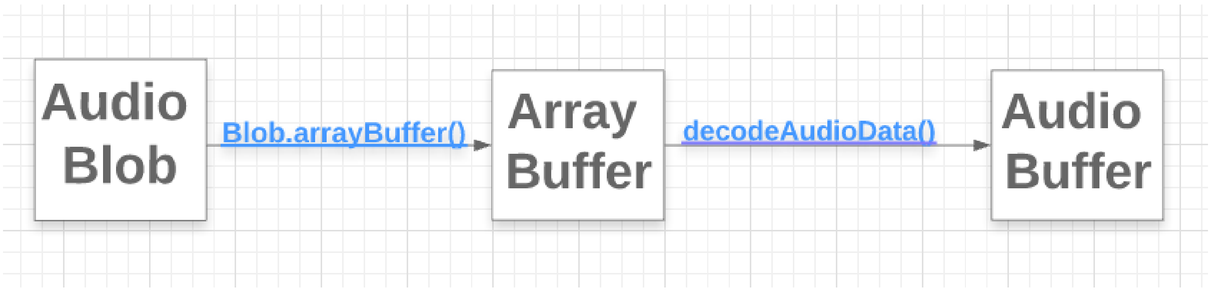
- The Record RTC library stores audio data as audioblob.



```
/**
 * This method can be used to store recorded blobs into IndexedDB storage.
 * @param {object} options - {audio: Blob, video: Blob, gif: Blob}
 * @method
 * @memberof RecordRTC
 * @example
 * RecordRTC.writeToDisk({
 *     audio: audioBlob,
 *     video: videoBlob,
 *     gif  : gifBlob
 * });
 */
```

Snap from RecordRTC.js

- We will convert **audioblob** to an **arraybuffer** using Blob.arrayBuffer(). After this, we will convert **arraybuffer** to **audiobuffer** using decodeAudioData() and then plot the audiobuffer in the canvas as shown in the next section.

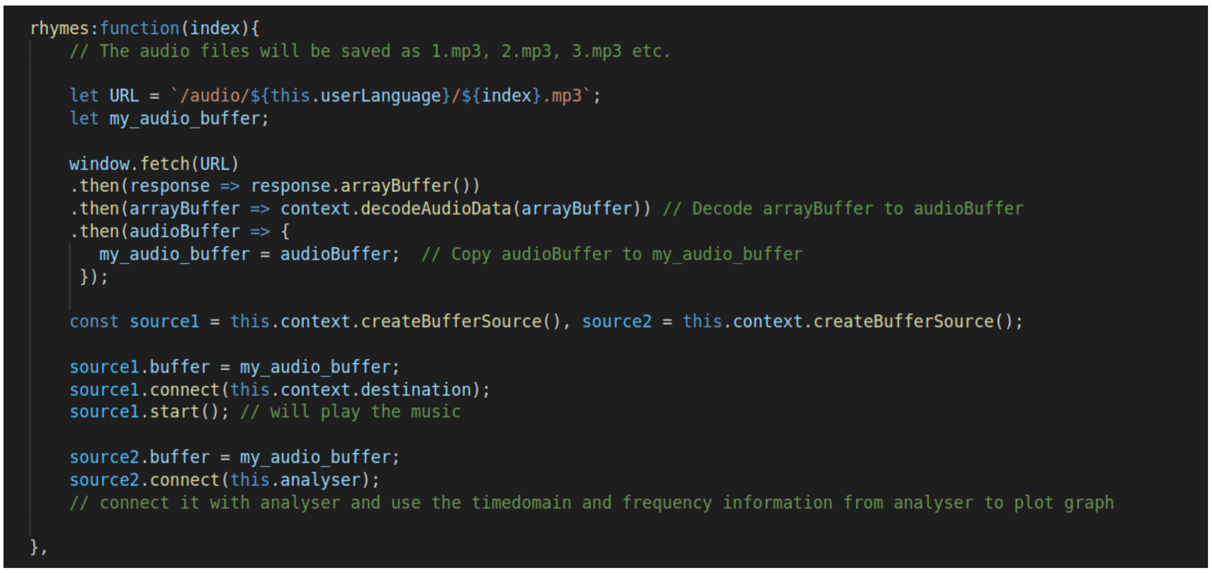


Sample Audio

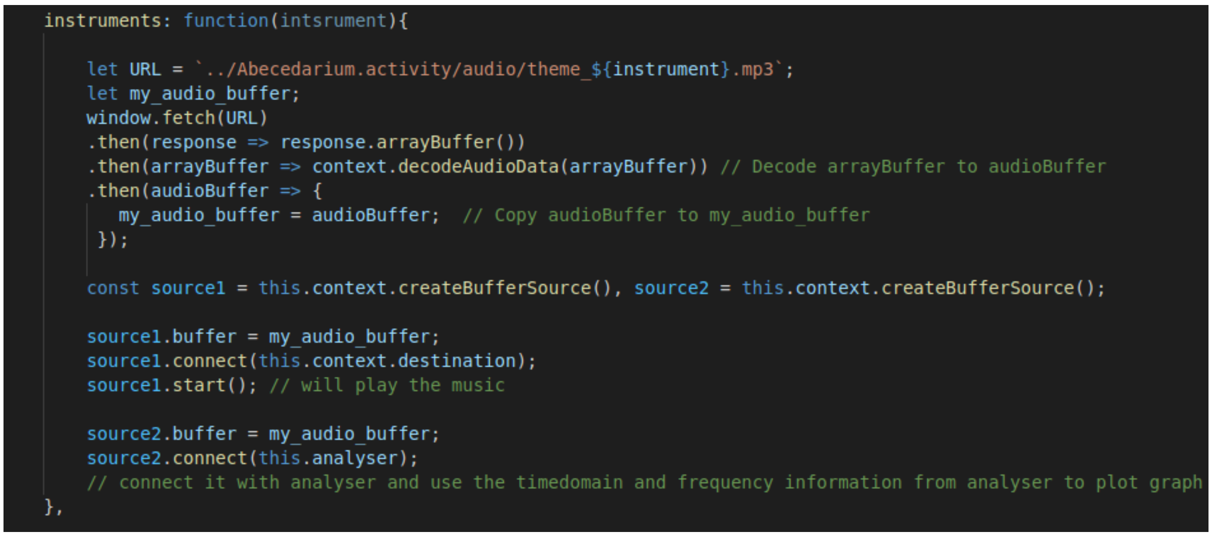
- We will include sample audio files like nursery rhymes in various languages. Depending upon the user's language settings, we will display them a set of sample audio files.

- We will also include the instrument themes from Abecedarium Activity.

The audio files will be imported using **Fetch API**. The audio file will be imported as arrayBuffer and will be decoded to audioBuffer using **decodeAudioData()**



```
rhymes:function(index){
    // The audio files will be saved as 1.mp3, 2.mp3, 3.mp3 etc.

    let URL = `/audio/${this.userLanguage}/${index}.mp3`;
    let my_audio_buffer;

    window.fetch(URL)
    .then(response => response.arrayBuffer())
    .then(arrayBuffer => context.decodeAudioData(arrayBuffer)) // Decode arrayBuffer to audioBuffer
    .then(audioBuffer => {
       my_audio_buffer = audioBuffer;  // Copy audioBuffer to my_audio_buffer
     });

    const source1 = this.context.createBufferSource(), source2 = this.context.createBufferSource();

    source1.buffer = my_audio_buffer;
    source1.connect(this.context.destination);
    source1.start(); // will play the music

    source2.buffer = my_audio_buffer;
    source2.connect(this.analyser);
    // connect it with analyser and use the timedomain and frequency information from analyser to plot graph

},
```



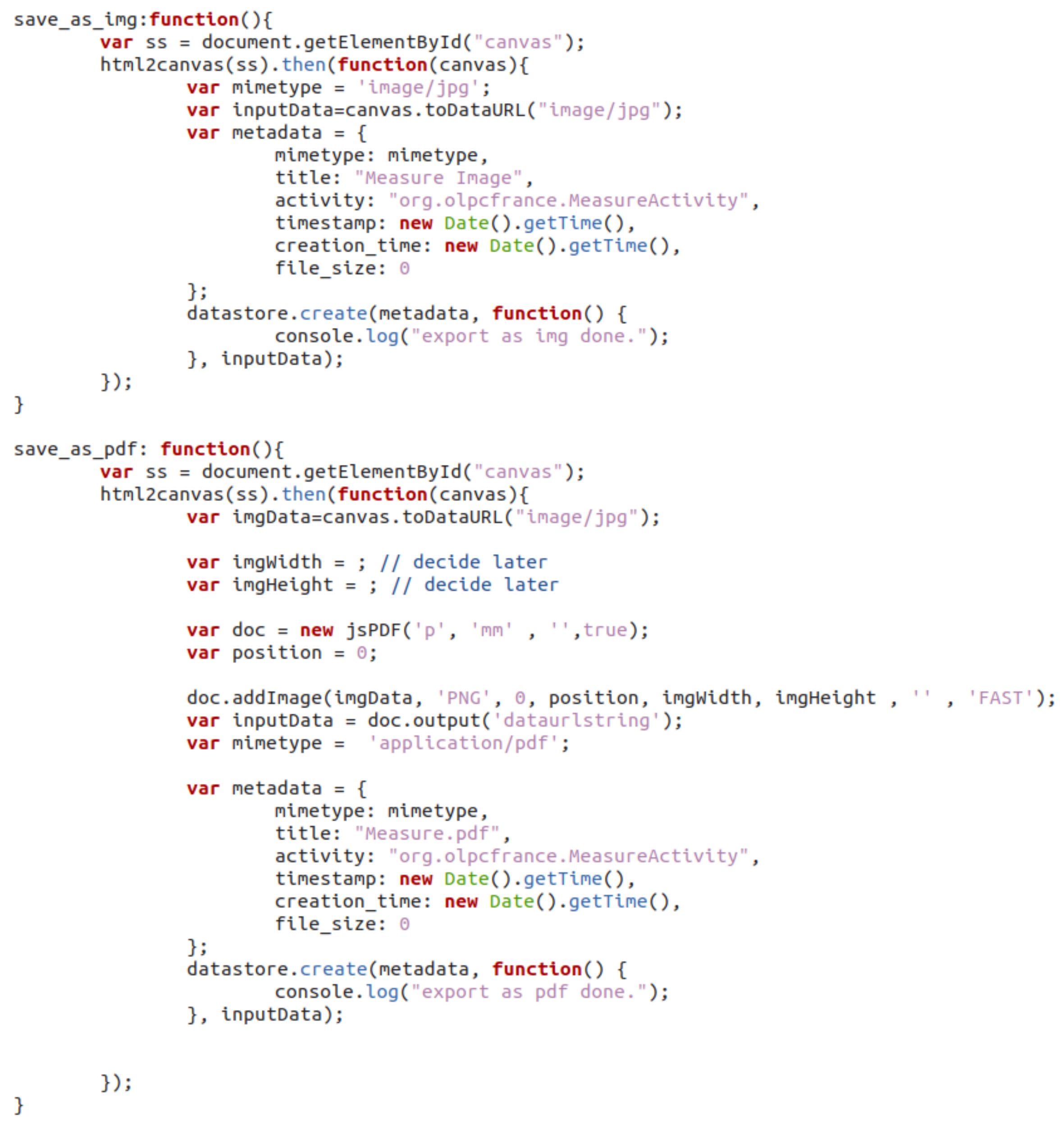
```
instruments: function(intsrument){

    let URL = `../Abecedarium.activity/audio/theme_${instrument}.mp3`;
    let my_audio_buffer;
    window.fetch(URL)
    .then(response => response.arrayBuffer())
    .then(arrayBuffer => context.decodeAudioData(arrayBuffer)) // Decode arrayBuffer to audioBuffer
    .then(audioBuffer => {
       my_audio_buffer = audioBuffer;  // Copy audioBuffer to my_audio_buffer
     });

    const source1 = this.context.createBufferSource(), source2 = this.context.createBufferSource();

    source1.buffer = my_audio_buffer;
    source1.connect(this.context.destination);
    source1.start(); // will play the music

    source2.buffer = my_audio_buffer;
    source2.connect(this.analyser);
    // connect it with analyser and use the timedomain and frequency information from analyser to plot graph
},
```

Color Palette

- We will add color palette from Sugar-Web library which will allow the users to change the **strokeStyle/fillStyle** of the Canvas.

Export Features

- We use **html2canvas** and **jspdf** libraries for export features. The export features will be available only when the waveform / frequency is paused, otherwise it will be disabled.



```
save_as_img:function(){
        var ss = document.getElementById("canvas");
        html2canvas(ss).then(function(canvas){
                var mimetype = 'image/jpg';
                var inputData=canvas.toDataURL("image/jpg");
                var metadata = {
                        mimetype: mimetype,
                        title: "Measure Image",
                        activity: "org.olpcfrance.MeasureActivity",
                        timestamp: new Date().getTime(),
                        creation_time: new Date().getTime(),
                        file_size: 0
                };
                datastore.create(metadata, function() {
                        console.log("export as img done.");
                }, inputData);
        });
}

save_as_pdf: function(){
        var ss = document.getElementById("canvas");
        html2canvas(ss).then(function(canvas){
                var imgData=canvas.toDataURL("image/jpg");

                var imgWidth = ; // decide later
                var imgHeight = ; // decide later

                var doc = new jsPDF('p', 'mm' , '',true);
                var position = 0;

                doc.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight , '' , 'FAST');
                var inputData = doc.output('dataurlstring');
                var mimetype =  'application/pdf';

                var metadata = {
                        mimetype: mimetype,
                        title: "Measure.pdf",
                        activity: "org.olpcfrance.MeasureActivity",
                        timestamp: new Date().getTime(),
                        creation_time: new Date().getTime(),
                        file_size: 0
                };
                datastore.create(metadata, function() {
                        console.log("export as pdf done.");
                }, inputData);


        });
}
```

- After the export is done, a popup will appear using the SugarPopup Library.

Localization

- We will directly use the SugarL10n Library for localisation of all the components.
- It is based upon the webL10n Javascript library.

Tutorial Integration

- For tutorial integration, we will directly use SugarTutorial Library.
- It is based upon Bootstrap Tour Library.

Responsiveness and Full Screen

- The activity will be fully responsive and will work on any browser (Chrome, Firefox, Safari) and any platform (Android, iOS, Windows, Linux, MacOS) supported by Sugarizer.

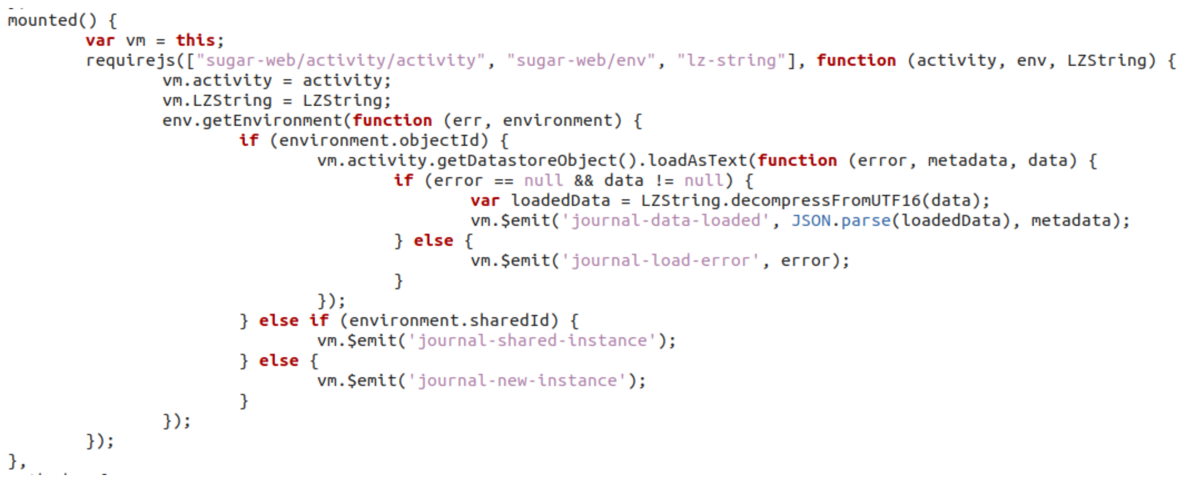
- Will use CSS for responsiveness and will implement a full screen Button in the toolbar.

Journal Handling

- We will be using SugarJournal Library to handle the datastore.

- The context for the activity will be recorded audios from microphone and the state of color palette, Amplitude Scale, Time Period scale and the invert state.

- The SugarJournal provides four events that we will be using



```
mounted() {
	var vm = this;
	requirejs(["sugar-web/activity/activity", "sugar-web/env", "lz-string"], function (activity, env, LZString) {
		vm.activity = activity;
		vm.LZString = LZString;
		env.getEnvironment(function (err, environment) {
			if (environment.objectId) {
				vm.activity.getDatastoreObject().loadAsText(function (error, metadata, data) {
					if (error == null && data != null) {
						var loadedData = LZString.decompressFromUTF16(data);
						vm.$emit('journal-data-loaded', JSON.parse(loadedData), metadata);
					} else {
						vm.$emit('journal-load-error', error);
					}
				});
			} else if (environment.sharedId) {
				vm.$emit('journal-shared-instance');
			} else {
				vm.$emit('journal-new-instance');
			}
		});
	});
},
```

Snap from SugarJournal.js

    - Journal-data-loaded: Fired when objectId is present and data from previous context is loaded.
    - Journal-shared-instance: Fired when Multi-User environment is enabled.
    - Journal-new-instance: Fired when objectId is null and a new instance is created.
    - Journal-load-error: Fired when any error occurred while handling journal

MultiWaveform Mode

- In this mode, more than one waveform will be plotted at the same time.
- The frequency spectrum mode will be disabled
- In single-user mode, the user can plot sample audios or sample audio along with Mic input. The sample audio will only be displayed as waveform on canvas and nothing will be played as audio output.
- The use of MultiWaveform Mode in Multi-User environment is explained in the next section.

Multi-User Environment

- We will use SugarPresence library to handle Multi-User Environment.
- When using a Multi-User environment, the features - frequency spectrum, Recording, Sample audios are going to be disabled.

- The host will share the canvas with guest users. They will speak on their individual mics and the waveforms will be displayed on common canvas in MultiWaveform mode.

- Each user can modify the Amplitude, inversion state and the color of their waveform. The time period scale will be fixed (according to host's settings)

- On receiving data from the server, we will create two actions:
  - Init : The canvas will be shared from host to guest users.
  - Update: The contents of canvas will get updated in real time for every user whenever an user gives audio input through their mic.

Cordova Plugins

- To support Android/iOS platform, we will be using two Cordova Plugins in our project.

  - **Cordova-plugin-media-capture** - It will be part of SugarRecord component. We will use it for recording and playing the audio. Earlier it had some issues but it was fixed by Lionel and Sarthak-g while updating to Cordova10.
  - **Cordova-plugin-audioinput** - We will use it to capture audio input through the device microphone. I have tested it on my Android Device and it is working fine.

Note - I have finished the VueJS tutorial.You can find my work on Pawn Activity here.

Timeline

During the coding period, I have vacation in my college and I am not doing any other internship and will devote my complete time towards my Google Summer of Code Project

Duration	Tasks
May 17 - May 23 **(Community Bonding Period)**	● Bonding with mentors and colleagues ● Explore Sugar Labs and get to know about its work culture
May 24 - May 30 **(Community Bonding Period)**	● Find rhymes and sample audio in all the languages supported by Sugarizer which we will be using with our activity.
May 31 - June 6 **(Community Bonding Period)**	● Explore the technologies and Libraries that we will be using. ● Discuss implementation ideas with Mentors.
Coding Period Finally Begins	
June 7 - June 13	● Create SugarRecord component for VueJS template. ● Test SugarRecord component by integrating it in activities which are already made from VueJS template. ● Example - Vote Activity, Tangram Activity, Curriculum

	Activity
June 14 - June 20	● Work on Frequency Spectrum - Make a basic Frequency Spectrum Plotter (without any Amplitude and time Period Scaling feature.) ● Test the basic Frequency Spectrum Plotter
June 21 - July 1	● Add Amplitude and Time Period Scaling Features ● Test Amplitude and Time Period Scaling features on both waveform and frequency spectrum ● Add Inverting Feature ● Test Inverting Feature on both waveform and frequency spectrum ● Add Color palettes ● Test the Color palette - Change in waveform / frequency spectrum color. ● Add MultiWaveform Feature ● Test the MultiWaveform Features

	○ Waveforms are of different Color ○ Amplitude Scaling of waves individually ○ Inverting of Waves individually
July 2 - July 11	● Implement the Record feature for Measure Activity completely. ● Test the Record feature for various time intervals. ● Play the recorded audio along with the display of its waveform / frequency spectrum on Canvas. ● Finalize all the previous work before evaluation.
July 12 - July 18	**EVALUATIONS** ● The SugarRecord component works perfectly. ● The Measure Activity has fully working Waveform and Frequency Spectrum with following features:- ○ Amplitude Scaling

	○ Time Scaling ○ Inversion ○ MultiWaveform ● The color palettes are working perfectly and allows the user to change the color of waveform/frequency spectrum. ● Record Feature for Measure Activity perfectly working ○ Record feature works for various time intervals ○ Recorded audio is played without any error and displays the corresponding waveform and frequency spectrum on Canvas
July 19 - July 25	● Add Localisation Feature using SugarL10n. ● Add the sample audios (instruments and rhymes) ● Test the Fetch API and play the audio along with the display of its waveform / frequency spectrum on Canvas.
July 26 - Aug 4	● Implement PDF exporter and

	Image exporter ● Test PDF and Image exporter ● Add Sugarizer Storage ● Test the Sugarizer Storage feature ● Using Sugar Presence, add multi-user environment. ● Integrate tutorial ● Add Full Screen Button ● Work on minor things (popups etc.)
Aug 5 - Aug 9	● Test the activity on various platforms and check responsiveness. ● Fix the bugs (if any found)
Aug 10 - Aug 15	● Finish any remaining previous work before final evaluation. ● Review the final documentation ● Prepare for final evaluations.
Aug 16 - Aug 23	**FINAL EVALUATIONS** ● The sample audio files are played

	without any error and display the corresponding waveform and frequency spectrum on Canvas. ● Full functional Measure activity for Sugarizer is available. ● The export features are working perfectly. ● All the general sugarizer features are working perfectly :- ○ Localisation ○ Sugarizer Storage ○ SugarPresence ○ Tutorial ○ Full screen ● The activity is fully responsive and works on any browser (Chrome, Firefox, Safari) and any platform (Android, iOS, Windows, Linux, MacOS) supported by Sugarizer ● Documentation is completed.

My Working Environment and Tools

- **Ubuntu 20.04 LTS Operating System**
- **VS Code**
- **Git Version Control**
- **Chrome Dev Tools**
- **Android Devices - Samsung Galaxy A70, Lava P5, Nokia 6.1**

Open Source Contributions :

- I am currently part of another open source program called Girlscript Summer of Code.
- I am working on the project ML Projectyard. I am looking forward to taking it up as my major project during the later part of the program.
- **This program ends in May and will not be a hindrance to my Google Summer of Code schedule.**
- I decided to participate in this program to gain relevant open source experience before Google Summer of Code.
- My contribution so far - Ramen Ratings

Major Web-Development Projects

IIITH-Buy and Sell Website - Created a project as a part of Hackathon 2020 orgainised by IIIT Hyderabad in my first year of college. Being from the Electronics branch, I had no prior experience with Web Development. In spite of having competition from students of the Computer Science branch, **my team won the competition**. This was my first project in Web Development.

Chat-App - Made a MERN Stack Chat application with Firebase Authentication. Currently used by me and a few of my batchmates. Work on group messaging is going on. We are learning to manage an app with a large number of Users.

XMeme - Worked on a Meme Streaming Web application as part of Crio Winter of Doing. Frontend work was done in React while Backend in Flask. Learned working on a CRUD Application.

Contribution to Sugarlabs

https://github.com/sugarlabs/flappy/pull/13#issue-569011605

https://github.com/sugarlabs/sugar/issues/937#issuecomment-774803849

https://github.com/sugarlabs/flappy-birds-activity/pull/24#issue-573635780

https://github.com/sugarlabs/GSoC/pull/131

Other Questions

How many hours will you spend each week on your project ?

I am planning to spend 40 hours per week on average. It can be extended according to the work requirements.

How will you report progress between evaluations ?

I will communicate with my mentors via email and IRC, seeking their advice and discussing my progress and ideas. I'll also push my work on github on a regular basis so that the mentors can review it.

Will you continue contributing to Sugar Labs after GSOC '21 ends ?

I am very interested in joining Sugar Labs and will continue to contribute regardless of my GSOC '21 selection. I'd like to focus on musicblocks-v4 after GSOC '21.
I also have a strong understanding of Python and have previously resolved a few issues in Sugar activities and will like to maintain the Sugar activities in future.
I will be a complete package to the organisation.