Experiment 4: Formant Analysis

In this exercise you need to record your own speech-samples. Before recording is possible the input/output sounddevices have to be set. Use the first cell to display all possible sound-devices and select the wanted devices by assigneng the sounddevice-indices to the variables 'input_device' and 'output_device'.

Then you can use the given record, play and clear buttons to record your own speech-samples.

For this experiment please record 3 different versions of one sentence and analyze the time-domain signal and its formant and f0 structure (as already done in Experiment 2) and plot the results in separate cells. By plotting the results in separate cells it is possible to always use the same record button and audio-array for different recordings.

1.) Recording:

Record a standard sentence and plot the time-domain signal and the spectrogram/formant/f0 plot as you have already done in Experiment 2. For the time-domain signal use the provided function 'get_plot_time_domain_sig()'. The function-arguments are described in the corresponding function header. To analyze the spectrogram/frequency and f0 structure of the recorded sentence you can use the Parselmouth-analysis and the corresponding plot-functions of Experiment 2. Please plot the results in a separate Cell.

> - Are there visible formant contours?
>
> > - Answer: yes there should be visible formants assuming a standard sentence containing vowels was recorded.
>
> - Can you distinguish between vowels, consonants and fricatives?
>
> > - Answer: Yes vowels are distinguishable with a clear formant structure and fricatives don't show a clear formant-structure. For consonants like 'l', 'm' or 'n' there can be visible formant structures because they can also be used in a voiced way.

2.) Recording:

Record a so called Dada-sentence but try to keep the same pitch/pitch course as in the sentence before. The Dada-sentence should not consist of words but rather of sounds like 'da', 'la', 'fa' etc. (or maybe even a combination of meaningless sounds?). After recording please plot the time-domain signal and spectrogram/formant/f0-plot in a separate cell.

> - Is the course of f0 the same as in the Recording before?
>
> > - Answer: Should be the same if it has been possible to keep the same pitch as the sentence before, if not maybe try again.
>
> - How do the formant-structures differ?
>
> > - Answer: If only one sound e.g. 'da' was used the formant-structure should repeat it self independent of the course of f0. Small deviations in the formant-structures are possible due to a certain dependency of the formants towards f0.
>
> - Are there any differences visible in the time-domain?
>
> > - Answer: If a voiced sound like 'da' was used to record the sentence there are only voiced-segments and no "noisy" segments in the time-domain signal indicated by higher amplitudes due to the higher energy content.

3.) Recording:

Record the previous sentence but use a whispery voice and once more please plot the time-domain signal and the spectrogram/formant/f0-plot in a separate cell.

> - How does the formant-structure differ in comparison to the first Recording?
>
> > - Answer: Formant-Tracker results might vary more due to less energy in the speech signal. But some sort of Formant-structure should still be visible.
>
> - How and why does the f0-course differ from the previouse recordings?
>
> > - Answer: F0-Tracking does not work. Whispery voice does not contain a f0 structure due to the fact that the glottis does not close properly when speaking with a whispery voice. The glottis closing impulses determine the f0 we percieve in speech signal. The missing or weakened glottis closing impulses lead to a missing f0 perception.

In [43]:
```python
fs = 44100 #Hz

#show all possible sound-devices
sd.query_devices()
```

Out[43]:
```
> 0 Built-in Microphone, Core Audio (2 in, 0 out)
< 1 Built-in Output, Core Audio (0 in, 2 out)
  2 Sound Siphon In, Core Audio (2 in, 0 out)
  3 Sound Siphon Out, Core Audio (0 in, 2 out)
  4 ZoomAudioDevice, Core Audio (2 in, 2 out)
  5 Hauptgerät, Core Audio (0 in, 2 out)
  6 Zylia + Abhöre, Core Audio (0 in, 2 out)
```

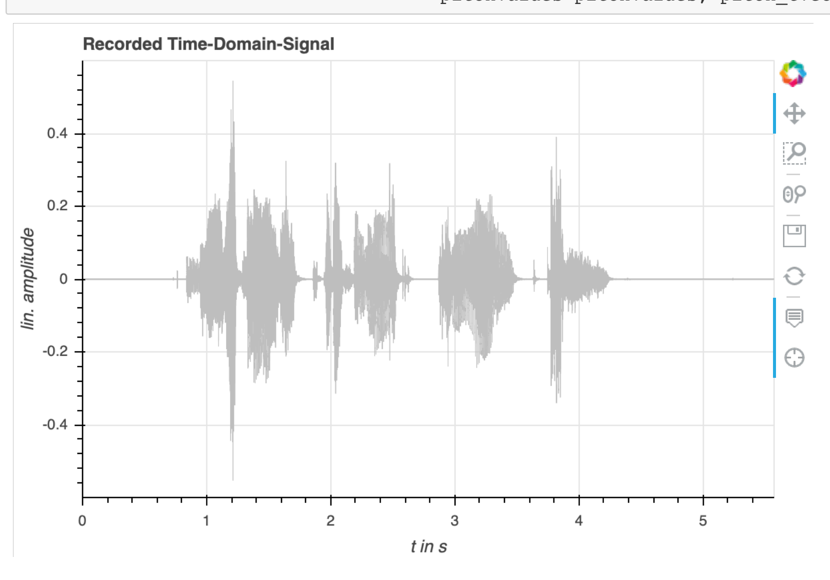
In [35]:
```python
#set default fs and number of channels
sd.default.samplerate = fs
num_channels = 2

# select sound-device by choosing ID of above shown list.
input_device = 0
output_device = 1

fs_target = 8000

sd.default.device = [input_device,output_device]
```

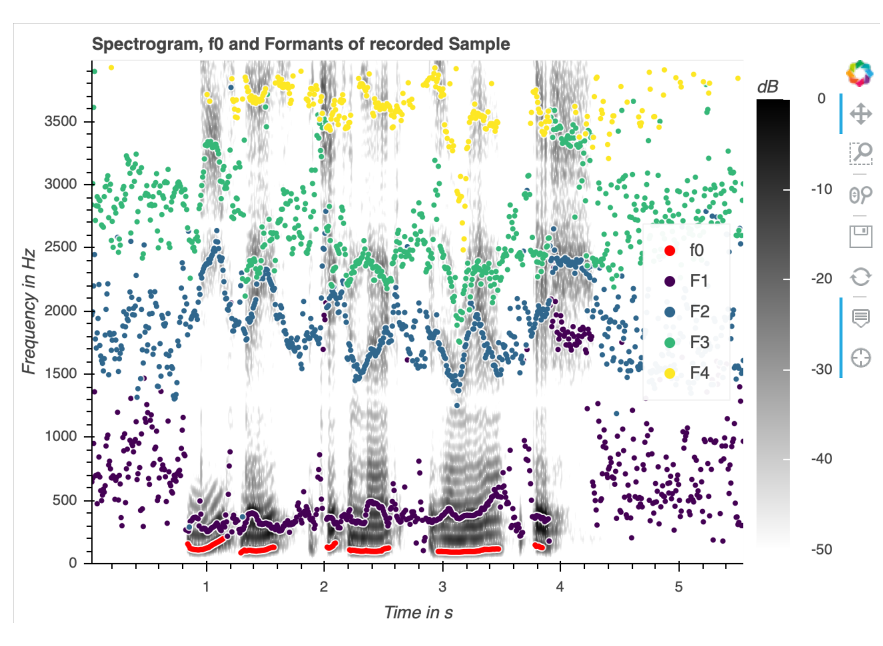
In [36]:
```python
def on_click_Rec(button):
#callback function for Rec-Toggle button. Determines what happens if Rec-Button is switched On/Off
    value = button.new
    if (button.new == True):
        button.owner.button_style='danger'
        with out:
            print('Recording started!')
        sd.rec(out=indata,samplerate=fs)

    else:
        button.owner.button_style=''
        sd.stop()
        with out:
            print('Recording stopped!')

def deleteNan(indata):
    # helper function to delete Nans in record-array (indata). Trims data-vector to all values which are NOT Nan.
    indata_no_Nan = np.zeros([np.sum(~np.isnan(indata[:,0])),indata.shape[1]])
    indata_no_Nan[:,0]=indata[~np.isnan(indata[:,0]),0]
    indata_no_Nan[:,1]=indata[~np.isnan(indata[:,0]),1]
    return indata_no_Nan



def on_click_Play(button):
    #callback function for Play-Toggle button. Determines what happens if Play-Button is switched On/Off
    if (button.new == True):
        button.owner.button_style='Success'
        with out:
            print('Playback started!')
        indata_no_Nan = deleteNan(indata)
        sd.play(indata_no_Nan)
        button.owner.description = 'Stop'
        button.owner.icon = 'stop-circle'
    if (button.new == False):
        sd.stop()
        with out:
            print('Playback stoped!')
        button.owner.button_style = ''
        button.owner.description = 'Play'
        button.owner.icon = 'play-circle'

def on_click_Clear(button):
    indata[:] = np.NaN
    with out:
        ipd.clear_output()
#        print('Audio-Array has been cleared!')
    return indata

def get_plot_time_domain_sig(snd, dt_snd, plottitle,showPlot):
    #
    #get_plot_time_domain_sig(): plots the time-domain signal of a given 1-D audio-file
    #
    #Input: snd        ... 1D array containing audio-file which is to be plotted
    #       dt_snd     ... 1D array containing time-vector of given audio-file
    #       plottitile ... String containing tile of Plot
    #       showPlot   ... Bool in order to surpress plot display.
    #                      If set to 'true' plot is displayed.

    TOOLS="hover,crosshair,pan,wheel_zoom,box_zoom,save,reset"
    TOOLTIPS = [
        ("index", "$index"),
        ("(x,y)", "($x, $y)"),
    ]
    p = figure(title=plottitle,plot_width=600, plot_height=400, x_range=(dt_snd[0], dt_snd[-1]), y_range=(np.floor(np.
min(snd)*10)/10, np.ceil(np.max(snd)*10)/10),tools=TOOLS,tooltips=TOOLTIPS)
    p.line(dt_snd, snd, line_width = 0.5, color = '#BEBEBE')
    p.xaxis.axis_label = 't in s'
    p.yaxis.axis_label = 'lin. amplitude'
    if showPlot:
        show(p, notebook_handle=False)
        pass
    else:
        return p


#maximum-duration for recorded samples ==> recorded sample shouldn't be so long (nothing is recorded after max_duratio
n)
max_duration = 30 # seconds
indata = np.empty([max_duration*fs,num_channels])
indata[:] = np.NaN


toggleRec =widgets.ToggleButton(
    value=False,
    description='Recording',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Record_Button',
    icon='circle' # (FontAwesome names without the `fa-` prefix)
)

togglePlay = widgets.ToggleButton(
    value=False,
    description='Play',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Play_Button',
    icon='play-circle' # (FontAwesome names without the `fa-` prefix)
)

clearButton = widgets.Button(
    description='Clear Audio-Array',
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Clear_Button',
    icon='trash' # (FontAwesome names without the `fa-` prefix)
)

toggleRec.observe(on_click_Rec, 'value')

togglePlay.observe(on_click_Play, 'value')

clearButton.on_click(on_click_Clear)

out = widgets.Output()
#display(out)

# object_methods = [method_name for method_name in dir(clearButton)
#                   if callable(getattr(clearButton, method_name))]
# print(object_methods)


box_layout = widgets.Layout(display='flex',
                flex_flow='column',
                align_items='center',
                width='100%')

widgets.HBox([toggleRec,togglePlay,clearButton,out],layout=box_layout)
```

1. Plot: Recording of Standard Sentence

> -time-domain signal plot
>
> -Spectrogram/Formant/f0 plot

In [44]:
```python
#prepare recordings for plots
indata_no_Nan = deleteNan(indata)
#convert to mono
recData = (indata_no_Nan[:,0]+indata_no_Nan[:,1])/2
rec_time = np.linspace(0,recData.shape[0]/fs,recData.shape[0])



#plot time-domain signal

get_plot_time_domain_sig(recData,rec_time,'Recorded Time-Domain-Signal',showPlot=True)


# Analyze Formants and f0 with parselmouth:
pitchLo = 75 #Hz
pitchHi = 400 #Hz
pitchTimeStep = 30 #ms

snd = parselmouth.Sound(recData)

PM_pitch = snd.to_pitch(pitch_floor = pitchLo, pitch_ceiling=pitchHi)

pitch_tVec = PM_pitch.ts()
pitchValues = np.zeros_like(pitch_tVec)


for timeIdx, time in enumerate(pitch_tVec):
    pitchValues[timeIdx] = PM_pitch.get_value_at_time(time=time)

speech_spectro, speech_spectro_t, speech_spectr_f = PM_get_spectrogram(snd, 30, maximumFrequency=4000)


windowLength = 30 # ms
maxNumberFormants = 4
maxFormantFreq = 4000

PM_formants = snd.to_formant_burg(maximum_formant=maxFormantFreq, window_length=windowLength/1000,
                                  max_number_of_formants=maxNumberFormants)

formant_tVec = PM_formants.ts()
formantValues = np.zeros((maxNumberFormants,formant_tVec.size))


for timeIdx, time in enumerate(formant_tVec):
    for formantIdx in range(maxNumberFormants):
        formantValues[formantIdx,timeIdx] = PM_formants.get_value_at_time(formant_number=formantIdx+1, time=time)

plottitle = 'Spectrogram, f0 and Formants of recorded Sample'
plot_spectrogram_with_f0_and_formants(speech_spectro, speech_spectro_t, speech_spectr_f, formantValues, formant_tVec,p
lottitle,
                                      pitchValues=pitchValues, pitch_tVec=pitch_tVec)
```

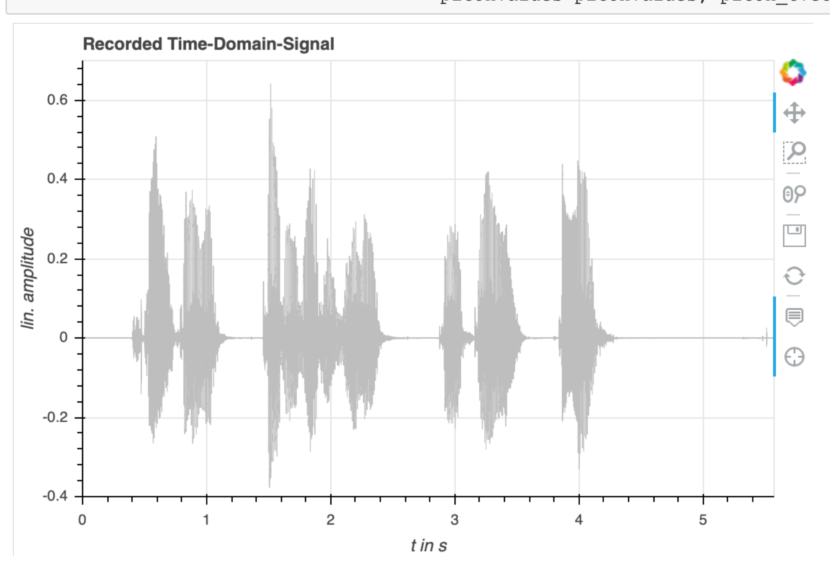


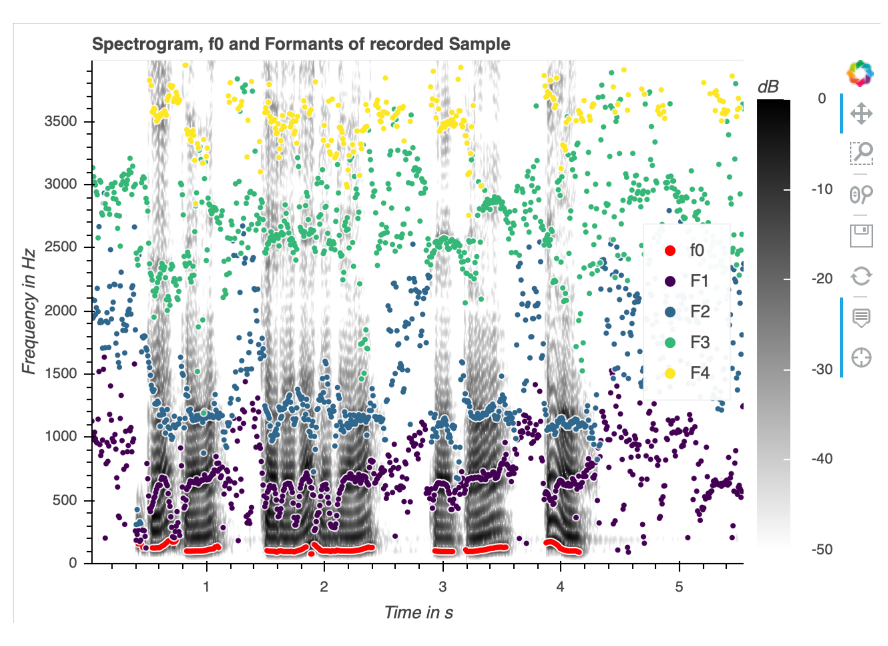


2. Plot: Recording of Dada-Sentence

> -time-domain signal plot
>
> -Spectrogram/Formant/f0 plot

In [45]:
```python
#prepare recordings for plots
indata_no_Nan = deleteNan(indata)
#convert to mono
recData = (indata_no_Nan[:,0]+indata_no_Nan[:,1])/2

rec_time = np.linspace(0,recData.shape[0]/fs,recData.shape[0])



#plot time-domain signal

get_plot_time_domain_sig(recData,rec_time,'Recorded Time-Domain-Signal',showPlot=True)


# Analyze Formants and f0 with parselmouth:
pitchLo = 75 #Hz
pitchHi = 400 #Hz
pitchTimeStep = 30 #ms

snd = parselmouth.Sound(recData)

PM_pitch = snd.to_pitch(pitch_floor = pitchLo, pitch_ceiling=pitchHi)

pitch_tVec = PM_pitch.ts()
pitchValues = np.zeros_like(pitch_tVec)


for timeIdx, time in enumerate(pitch_tVec):
    pitchValues[timeIdx] = PM_pitch.get_value_at_time(time=time)

speech_spectro, speech_spectro_t, speech_spectr_f = PM_get_spectrogram(snd, 30, maximumFrequency=4000)


windowLength = 30 # ms
maxNumberFormants = 4
maxFormantFreq = 4000

PM_formants = snd.to_formant_burg(maximum_formant=maxFormantFreq, window_length=windowLength/1000,
                                  max_number_of_formants=maxNumberFormants)

formant_tVec = PM_formants.ts()
formantValues = np.zeros((maxNumberFormants,formant_tVec.size))


for timeIdx, time in enumerate(formant_tVec):
    for formantIdx in range(maxNumberFormants):
        formantValues[formantIdx,timeIdx] = PM_formants.get_value_at_time(formant_number=formantIdx+1, time=time)

plottitle = 'Spectrogram, f0 and Formants of recorded Sample'
plot_spectrogram_with_f0_and_formants(speech_spectro, speech_spectro_t, speech_spectr_f, formantValues, formant_tVec,p
lottitle,
                                      pitchValues=pitchValues, pitch_tVec=pitch_tVec)
```





3. Plot: Recording of whispered Sentence

> -time-domain signal plot
>
> -Spectrogram/Formant/f0 plot

In [46]:
```python
#prepare recordings for plots
indata_no_Nan = deleteNan(indata)
#convert to mono
recData = (indata_no_Nan[:,0]+indata_no_Nan[:,1])/2

rec_time = np.linspace(0,recData.shape[0]/fs,recData.shape[0])



#plot time-domain signal

get_plot_time_domain_sig(recData,rec_time,'Recorded Time-Domain-Signal',showPlot=True)


# Analyze Formants and f0 with parselmouth:
pitchLo = 75 #Hz
pitchHi = 400 #Hz
pitchTimeStep = 30 #ms

snd = parselmouth.Sound(recData)

PM_pitch = snd.to_pitch(pitch_floor = pitchLo, pitch_ceiling=pitchHi)

pitch_tVec = PM_pitch.ts()
pitchValues = np.zeros_like(pitch_tVec)


for timeIdx, time in enumerate(pitch_tVec):
    pitchValues[timeIdx] = PM_pitch.get_value_at_time(time=time)

speech_spectro, speech_spectro_t, speech_spectr_f = PM_get_spectrogram(snd, 30, maximumFrequency=4000)


windowLength = 30 # ms
maxNumberFormants = 4
maxFormantFreq = 4000

PM_formants = snd.to_formant_burg(maximum_formant=maxFormantFreq, window_length=windowLength/1000,
                                  max_number_of_formants=maxNumberFormants)

formant_tVec = PM_formants.ts()
formantValues = np.zeros((maxNumberFormants,formant_tVec.size))


for timeIdx, time in enumerate(formant_tVec):
    for formantIdx in range(maxNumberFormants):
        formantValues[formantIdx,timeIdx] = PM_formants.get_value_at_time(formant_number=formantIdx+1, time=time)

plottitle = 'Spectrogram, f0 and Formants of recorded Sample'
plot_spectrogram_with_f0_and_formants(speech_spectro, speech_spectro_t, speech_spectr_f, formantValues, formant_tVec,p
lottitle,
                                      pitchValues=pitchValues, pitch_tVec=pitch_tVec)
```