'Sing us a song, you're the piano pi' Talking Piano



Team B2 Angela Chang, John Martins, Marco Acea

Use-Case and Requirements

Explore music beyond physical constraints by creating human speech on a piano!

- Record user speech (via UI, that also offers playback features)
 - 200ms end-to-end latency
- Convert input speech into piano notes
 - 80% Frequency extraction accuracy
- Schedule those notes onto a piano
 - <5% of syllables missed (delayed/elongated/sped, not dropped)
- Implement a physical device that can press the keys on a piano
 - 80% Fidelity Rate

ECE Areas: Software Systems, Signals and Signals

Solution Approach



Complete Solution



Notes Scheduler

- Input: a 2-dimensional array of frequencies
 - Each row is a timestamp
 - Each column is a key (69 columns)
- Output: a second 2-dimensional array
 - Volumes for keys at each timestamp
- Differentiate between when to re-press keys and when to keep them held down
 - Decay of keys
 - Separation of phonemes and syllables
- Decay modelling: a logarithmic approximation
 - Volume of the virtual piano is controlled by amplitude scaling
 - This makes the decay even across all volumes
 - Phenomenon in physical pianos more complex



Web Application Interface

- Virtual piano plays corresponding audio file for each piano key as described by note scheduler
- Run Speech-Text libraries on incoming audio files to provide subtitles for better interpretation of output audio from piano





Example of the visualization from 1 note being played

Audio Processing

 $egin{aligned} X_k[n] &= [X_k[n-1] - x[n-N] + x[n]] e^{j2\pirac{k}{N}} \ x[n] &= [X_k e^{-j2\pirac{k}{N}}] - X_k[n-1] - x[n-N] \end{aligned}$

The Sliding Discrete Fourier Transform

Given a window of N samples, if we'd like to know what the frequency is at frequency bin k, we can use the recursive function for X_k . As new samples come in, we can use the previously calculated frequency, $X_k[n\mbox{-}1]$



Change in Freqeuncy Across Time Using SDFT







Risk Factors and Unknowns

- Building the physical interface
 - Might take longer than expected, therefore we'll build a proof of concept build that only uses --5 solenoids to press keys
- Blurring the audio might not extract enough information
- Upload and Download internet speed between remote server and Raspberry
 Pi introduce a bottleneck
- Our piano play rate may be too high, causing keys to be "spammed"

Alternative Design Strategies

- Virtual piano implementation: should our proof of concept for the piano-playing mechanism fail, we will implement a virtual piano solution.
- Near real-time speech-to-piano translation: once MVP is achieved, we hope to allow people to speak and hear their sentence played on the piano once they're done speaking.
- Lower latency backend: once fully committed to the physical interface, we can migrate the backend logic onto a Jetson if speed is a concern.

Testing, Verification, Metrics

- Web-app physical system latency
 - Use Selenium to mimic clicking on UI components
 - Measure the time between pressing a button on our frontend UI and the appropriate reaction of the system
 - Our goal metric is < 200ms
- Fast Fourier transform accuracy
 - Use an input audio recording we create with known frequencies and amplitudes
 - For each window we will compare the frequencies reported by our system to the known frequencies at that time
 - \circ $\,$ Accuracy dependent on chosen time window for FFT $\,$
 - Shorter Window=> Less Accurate
 - Longer Window => Long wait times
 - Adjust our time window for >80% accuracy.



Testing, Verification, Metrics cont.

- Syllable timing
 - Use recordings with labeled start times for each syllable
 - Record the number of syllables whose start time does not match the original recordings labelled start time
- Fidelity of Output Audio
 - Generate a series of prompts and output piano recordings
 - Survey a group of listeners on whether or not they can understand the prompt given the piano audio
 - Collect data on what percentage of listeners were able to make out what the piano was trying to say



Updated Gantt Chart

PROJECT NAME	Talking Piano	UNIVERSITY CLASS	Carnegie Mellon University - 18500 Capstone	
MEMBERS	Marco Acea, Angela Chang,	DATE	9/15/22	

Legend							
Marco							
Angela							
John							
All							

		PHASE ONE - Design and Early Imple				ementation PHASE TWO - Implementation						PHASE THREE - Wrap Up MVP and Testing							PHASE FOUR - Slack and Polishing							
TASK TITLE T	TASK OWNER	10/03/2022	10/10)/2022		10/2	4/2022	10/31/2022			/7/2022	11/	14/2022	11/14/2022		11/21/2022			11/28/2022		2 12/5/2022		12/12/2022		2	
		MTWR	FMT	WRF		MT	WRF	мт	WR	FMT	WR	FMT	WR	F M T	WR	FM	TWR	FM	T W R	F M	TW	RF	MT	WRF		
Audio Processing	MA																									
System Design and Specifications																										
Audio File to data structure with Pydub																										
ourier transform (data structure)	AC																									
ourier transform (optimization)	AC																									
ilter frequencies to piano keys																										
ata struct for velocities and keys																										
ext to piano backend																										
iano Performance Scheduler	AC																									
ystem Design and Specifications																										
eb app and audio processing interfac	ce MA																									
ocket face for web app and hicrocontroller communication	JM																		Than						Proje	
udio stream serializer					Fall Break														ksg							
eb App	JM																		iving							
stem Design and Specifications																			g Br							
eate Boiler-plate Back/Front End	-																		eak							
layback Interaction																										
pload and record																										
ontent Recommendadtion																										
/irtual Piano VIsualization																										
lote Playback Functions																-										
esting and Validation																										
erformance Metric Validation																										
Testing Implementation																										
Slack																										
Slack																						_	_			