VOACaloid*

A "better" "hardware-based" "portable" "solution" for the "real-time" "generation" of "singing"

Aaron Thapa
notsmart98@gmail.com
@yoyo@m.ggrks.moe†

Thomas Jefferson High School for Science and Technology

2023 April 0

Abstract

Singing Voice Synthesis (SVS) technologies have advanced to the point that they can now often be confused for the human singers they are based on. However, software SVS (the most common kind of SVS), has proven difficult to use in live performance. In this article, I present a live, human-controlled, hardware-based SVS device whose construction involves obscure parts obtained in suspicious circumstances, a Raspberry Pi Zero, and apathy on scales never-before-seen, even in this conference.

1. Introduction

Composers of music are often hesitant to include vocals on their work. For singers, singing for extended periods can lead to vocal strain. For non-singers, working with singers for extended periods can cause mental strain. To resolve this issue, various synthesized singing technologies have been developed.

1.1. History of SVS

The first computer singer was an IBM 7904 owned by Bell Labs in 1961 [1]. Becasuse of the hardware, the voice that it produced was limited and not realistic. Later, the Digital Equipment Corporation (DEC) brought several implementations of SVS to a larger market, known collectively as DECtalk [2]. Originally, DECtalk was a hardware product, using ICs to generate the voice, but software implementations and libraries were later produced, allowing programmers to integrate voice synthesis and SVS into their codebases. Notably, DECtalk was featured in NASA's video game, *Moonbase Alpha*, in which players often used the SVS engine in a humorous manner[1] [3]. Most recently, companies such as Google have been developing AI-based text-to-speech engines [4].

*VOAC: Voice synthesizer on a chip; own coinage

†Quite possibly the first Mastodon link in SIGBOVIK History, but I wouldn't be surprised to see more this year.

[1]John Madden John Madden John Madden John Madden John Madden John Madden John Madden John Madden John Madden John Madden

1.2. Vocaloid

In 2000, Hideki Kenmochi created a singing voice synthesizer project called "Vocaloid", from "vocal" and "android" [5]. Vocaloid generated output by concatenating individual recordings of phonemes from a singer, allowing for a much more realistic sound than previous attempts. With support from Yamaha, the first commercial version of Vocaloid was released in 2003 [1].

After the 2007 release of Vocaloid 2, Crypton Future Media created a voice called Hatsune Miku, now better known than Vocaloid itself [6]. Yamaha themselves released two voices, VY1 ("Mizki") and VY2 ("Yūma").

1.3. eVocaloid

In 2012, Yamaha implemented a version of Vocaloid in one of their synthesizer ICs, the YMW820 [7], also known as the NSX-1. The "e" prefix stands for "embedded", and was also added to the name of the voice, VY1, to make eVY1 the name for the voice bank. In 2014, Gakken, a Japanese educational company, released the "Pocket Miku" [8]. Its stylophone-style pitch input is suboptimal for live performance [9]. Additionally, the only way to control the current syllable is by sending it MIDI messages from another device [9].

1.4. VOACaloid

This project aimed to create a "portable" device which uses the capabilities of the YMW820 to allow the user to control it in an "intuitive" manner, while still being able to use the full extent of its features in a live, concert-performance scenario. This could also be described as a "shanzhai[2]" version of the Pocket Miku, making use of the eVY1.

2. Motivation

~~N/A [10]~~ Indeed, software SVS solutions have worked in music production, and any live performance of SVS would be much less practical than just finding a singer. Even Vocaloid concerts which feature live bands simply use a pre-rendered track for the Vocaloid singing [11]. Having a portable and playable Vocaloid is very cool though, thus giving this project some purpose.

2.1. Materials

Name	Price	Qty.
Aides Tech eVY1 Board	¥9000	1
DMK-25 Midi Controller	$65.99	1
DIN Coupler (2 pcs)	$8.49	1
USB Midi Adapter	$6.99	1
Raspberry Pi Zero	$5.00	1
Total:	$155.55	

Most of the parts were bought on Amazon, except for the eVY1 board, which was bought on Yahoo Shopping Japan, and then shipped to the US via a mail forwarding service[3].

2.2. Software

The Raspberry Pi Zero runs a version of Raspbian with the desktop environment disabled. The program which routes MIDI messages between the USB-MIDI adapter, the MIDI controller, and the eVY1 board is written in Python. When plugged in to power, the Raspberry Pi boots up and runs `detector.py` in a headless session.

3. Development Process

3.1. Device

Once I received the eVY1, I spent a few days learning how to format the MIDI messages that I would need to send it from a Jupyter Notebook (see `evy1.ipynb` for implementation). Standard Japanese Romanization is ambiguous, so the eVY1 shield uses its own phonetic transcription system, somewhat related to that used in software versions of VOCALOID. I used the NSX-1 datasheet [7] to write `japanese_to_phoneme()`

[2] Chinese word for Chinese copy

[3] My uncle

which would take a string of hiragana and convert it to the phoneme text. Using the new text, `phoneme_to_midi_message()` creats a SysEx message to send to the eVY1.

For pitch input, the `mido` Python module had most of the tools that were necessary built-in. I wrote `router.py` to take incoming MIDI messages from the MIDI controller and then modify them to be suitable for the eVY1. At this point, I still had not decided what I wanted to do with the project, so I wrote `player.py` to convert and play MIDI files.

Then came the Summer of 2022. I stopped working on the project for about 6 months for several reasons but mostly because I was out of the country and was focused on other work.

Once I started having time in my school schedule, I resumed the project, and came up with the live performance idea. Optimizing for this proved to be difficult. In `detector.py` I implemented what would later become the most important part of the project, the phoneme input system. Crucially, when reading the button combinations as pressed by the user, it only updates when a new button is pressed, and ignores the event generated by the button being released. With this feature working I decided that I did not want to work anymore.

3.2. Various Side-Projects

There were many points during and after this project in which I grew bored with developing the main features. I began working on various semi-related endeavours which I will write about here for the purposes of space-filling.

One of the first things I had done which was unrelated to the project was that I used the eVY1 as a General MIDI-compatible module for composing music. Using it in this manner did mean that I was unable to use the vocal aspect in any of the music, alongside MIDI Channel 1 (Channel 0) because it is exclusively used for the voice. In FL Stuido, I used it as the target for the "MIDI Out" plugin. The music I produced with the module was used for a LOVE Jam 2021 Submission, "Scraper Escaper"[4].

Continuing on the MIDI file idea from earlier, one of the examples that Aides Tech had released for using the eVY1 shield was an Arduino sketch which took the binary data of a MIDI file and sent it to the eVY1 for playback [12]. Based on that code, I wrote a Python script which could take a MIDI file and output an Arduino sketch which could then be uploaded[5].

After converting a few songs, I noticed that the code did not give a correct tempo for certain input files. Instead of finding the root cause, I opted to manually change one of the constants in the file until it sounded about right. Not doing this resulted in several humorous files, such as `s2chemical3.ino` from Sonic 2, which played much too fast.

`Router.py` was an early attempt at what would eventually become the main functionality. Its purpose was to route midi msesages from the selected device to the eVY1, but I also left in some aspects such as choosing the MIDI channel which note events would be sent to, so that other instrument sounds could be used bseides the voice.

4. Usage

Using the device is quite simple. Once the Pi has power, one can simply press the pads on the midi controller in accordance with the table in Appendix A and press the melodic keys to achieve the desired pitch. Understanding how to get the software into a usable state is left as an exercise for the reader, as is learning how to play any real songs.

5. Demonstration

I understand that giving some kind of output sample or a demonstration at all is pretty important. However, if you are reading this paragraph, forces beyond my control have conspired

[4]Game can be played at: `https://bluesheep7.itch.io/scraper-escaper`, music can be found at: `https://youtu.be/X0FZDJCPhIU`

[5]Yes, I understand the irony of using a Python script to write C code. I also think it's very funny.

such that I am unable to access this device until beyond the deadline. However, any skilled performance would sound no different from one being controlled by a MIDI file. Thus, I will point to the demonstrations from Aides Tech, available on their page for the eVY1 as sufficient examples of eVY1 output.

6. Discussion

This device is completely impractical, and it does not satisfy a live performer's needs. The output is far too noisy, and it is hard to power cleanly.

6.1. Comparison with Human Singer

Comparison	VOACaloid	Human Singer
Wages	✓	
Energy Consumption	✓	
Polyphony	✓	
Vocal Range	✓	
Percussive Ability	✓	
Total:	5	0

As this impartial and objective comparison shows, the VOACaloid is unequivocally the best singing device ever created. After all, just about anything can beat working with a human.

7. Future Developments

7.1. Of this project

This section is inherently difficult to write because anything I write in here could theoretically be implemented before the deadline and thus would need to be removed from this section. Thus, I will not be including any possible future developments, and instead be implementing them and describing them elsewhere in this paper. Of course, I could have waited until I was truly "finished" with the project, but that is no fun. I cannot predict the future, so I do not know what I will implement. Any ideas for future developments can be sent to my email, as listed at the top of this paper.

Since I have written the last paragraph, the SIGBOVIK deadline has been announced, and I had long decided to stop working. I ended up not using the MIDI in and out connections which I had bought. One idea I had was to be able to connect to the shell using a terminal program over the MIDI connectors. Obviously it would have been terrible to use, but I could probably implement it in the future if I cared to continue working on this (which I don't).

7.2. Of SVS in general

Just who do you think I am? We've proven that I'm not smart enough to make anything useful, let alone predict the future of a fast-moving scientific field. We've all seen what ChatGPT has done to writing and what stable diffusion has done to art, who's to say that something similar can't come of CeVio or Neutrino. CeVio is already generally indistinguishable from a human voice with heavy processing.

8. Acknowledgments

I would first like to thank those at Yamaha responsible for the development of Vocaloid. I would then like to thank my lab director and supervisor, Kuprenas, who managed to keep me mostly on track during the creation of this project, and gave me the funds to buy the MIDI controller, and Paul Drongowski, for maintaining his blog and archiving the datasheets and web tools for the NSX-1. Thanks to the developers of TEX and LATEX for making writing papers enjoyable, and thanks to the contributors to XETEX for adding Unicode support to make writing this specific paper possible. ~~Thanks to Donner for not writing any useful documentation for their MIDI controller.~~ I would like to thank Thomas Chick for the previous joke [13], as well as Tom Wildenhain for introducing me to SIGBOVIK, and Tom Murphy VII for reminding me it exists every year[6].

[6]Wow, I didn't notice that they were all Toms; that's pretty cool.

References

[1] Gakken, *Pocket Miku: The singing keyboard*, 2014. [Online]. Available: https://cdn-shop.adafruit.com/pdfs/pocket_miku.pdf.

[2] A. Pollack, "Technology; audiotex: Data by telephone," *The New York Times*, Jan. 5, 1984.

[3] VocaDB, *Moonbase Alpha text-to-speech*, 2023. [Online]. Available: https://vocadb.net/Ar/85226.

[4] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016. arXiv: 1609.03499. [Online]. Available: http://arxiv.org/abs/1609.03499.

[5] H. Kenmochi, *Vocaloid2*、初音ミク、ユーザ、*UGM* サイト、権利者 *[VOCALOID2, Hatsune Miku, users, UGM sites and rightful claimants - the report]*, 2008. [Online]. Available: http://www.dcaj.or.jp/project/report/pdf/2007/dc08_03.pdf.

[6] B. Roseboro, "The Vocaloid phenomenon: A glimpse into the future of songwriting, community-created content, art, and humanity," DePauw University, 2019. [Online]. Available: https://scholarship.depauw.edu/cgi/viewcontent.cgi?article=1125&context=studentresearch.

[7] Yamaha, *YMW820(NSX-1) sound generator datasheet*, 2012. [Online]. Available: http://sandsoftwaresound.net/wp-content/uploads/2017/07/YMW820_data_sheet_en.pdf.

[8] Gakken, ポケット・ミク カスタマイズガイド *[Pocket Miku customize guide]*, 2008. [Online]. Available: https://otonanokagaku.net/nsx39/data/nsx39midiguide.pdf.

[9] D. Battino, "Hacking Pocket Miku, the singing stylophone," *Electronic Musician*, vol. 31, Jan. 2018.

[10] T. Wildenhain, "On the turing completeness of MS PowerPoint," in *SIGBOVIK*, 2021.

[11] H. Miku, 【*Hatsune Miku*】*World is Mine / ryo*(*supercell*)【初音ミク】, 2013. [Online]. Available: https://www.youtube.com/watch?v=jhl5afLEKdo.

[12] Switchscience, *eVY1 shield example*, 2014. [Online]. Available: https://github.com/SWITCHSCIENCE/eVY1_Shield.

[13] T. Chick, "'The SIGBOVIK paper to end all SIGBOVIK papers' will not be appearing at this conference," in *SIGBOVIK*, 2021.

A. Phoneme Table

Kana	Romaji	Phoneme Text	Buttons
あ	a	a	(0)
い	i	i	(1)
う	u	M	(2)
え	e	e	(3)
お	o	o	(4)
か	ka	k a	(5)
き	ki	k' i	(6)
く	ku	k M	(7)
け	ke	k e	(0, 1)
こ	ko	k o	(0, 2)
さ	sa	s a	(0, 3)
し	shi	S i	(0, 4)
す	su	s M	(0, 5)
せ	se	s e	(0, 6)
そ	so	s o	(0, 7)
た	ta	t a	(1, 2)
ち	chi	tS i	(1, 3)
つ	tsu	ts M	(1, 4)
て	te	t e	(1, 5)
と	to	t o	(1, 6)
な	na	n a	(1, 7)
に	ni	J i	(2, 3)
ぬ	nu	n M	(2, 4)
ね	ne	n e	(2, 5)
の	no	n o	(2, 6)
は	ha	h a	(2, 7)
ひ	hi	C i	(3, 4)
ふ	hu/fu	p\M	(3, 5)
へ	he	h e	(3, 6)
ほ	ho	h o	(3, 7)
ま	ma	m a	(4, 5)
み	mi	m' i	(4, 6)
む	mu	m M	(4, 7)
め	me	m e	(5, 6)
も	mo	m o	(5, 7)
ら	ra	4 a	(6, 7)
り	ri	4' i	(0, 1, 2)
る	ru	4 M	(0, 1, 3)
れ	re	4 e	(0, 1, 4)
ろ	ro	4 o	(0, 1, 5)
が	ga	g a	(0, 1, 6)
ぎ	gi	g' i	(0, 1, 7)
ぐ	gu	g M	(0, 2, 3)
げ	ge	g e	(0, 2, 4)

ご	go	g o	(0, 2, 5)
ざ	za	dz a	(0, 2, 6)
じ	zi/ji	dZ i	(0, 2, 7)
ず	zu	dz M	(0, 3, 4)
ぜ	ze	dz e	(0, 3, 5)
ぞ	zo	dz o	(0, 3, 6)
だ	da	d a	(0, 3, 7)
ぢ	di/zi	dZ i	(0, 4, 5)
づ	du/dzu	dz M	(0, 4, 6)
で	de	d e	(0, 4, 7)
ど	do	d o	(0, 5, 6)
ば	ba	b a	(0, 5, 7)
び	bi	b' i	(0, 6, 7)
ぶ	bu	b M	(1, 2, 3)
べ	be	b e	(1, 2, 4)
ぼ	bo	b o	(1, 2, 5)
ぱ	pa	p a	(1, 2, 6)
ぴ	pi	p i	(1, 2, 7)
ぷ	pu	p M	(1, 3, 4)
ぺ	pe	p e	(1, 3, 5)
ぽ	po	p o	(1, 3, 6)
ゃ	ya	j a	(1, 3, 7)
ゅ	yu	j M	(1, 4, 5)
ょ	yo	j o	(1, 4, 6)
わ	wa	w a	(1, 4, 7)
ゐ	wi	w i	(1, 5, 6)
ゑ	we	w e	(1, 5, 7)
を	wo/o	o	(1, 6, 7)
ふぁ	fa	p\a	(2, 3, 4)
つぁ	tsa	ts a	(2, 3, 5)
うぃ	wi	w i	(2, 3, 6)
すぃ	si	s i	(2, 3, 7)
ずぃ	zi	dz i	(2, 4, 5)
つぃ	tsi	ts i	(2, 4, 6)
てぃ	ti	t' i	(2, 4, 7)
でぃ	di	d' i	(2, 5, 6)
ふぃ	fi	p\' i	(2, 5, 7)
とぅ	tu	t M	(2, 6, 7)
どぅ	du	d M	(3, 4, 5)
いぇ	ye	j e	(3, 4, 6)
うぇ	we	w e	(3, 4, 7)
きぇ	kye	k' e	(3, 5, 6)
しぇ	she	S e	(3, 5, 7)
ちぇ	che	tS e	(3, 6, 7)
つぇ	tse	ts e	(4, 5, 6)
てぇ	tee	t' e	(4, 5, 7)
にぇ	nye	J e	(4, 6, 7)
ひぇ	hye	C e	(5, 6, 7)

みぇ	mye	m' e	(0, 1, 2, 3)
りぇ	rye	4' e	(0, 1, 2, 4)
ぎぇ	gye	g' e	(0, 1, 2, 5)
じぇ	jye	dZ e	(0, 1, 2, 6)
でぇ	dee	d' e	(0, 1, 2, 7)
びぇ	bye	b' e	(0, 1, 3, 4)
ぴぇ	pye	p' e	(0, 1, 3, 5)
ふぇ	fe	p\e	(0, 1, 3, 6)
うぉ	wo	w o	(0, 1, 3, 7)
つぉ	tso	ts o	(0, 1, 4, 5)
ふぉ	fo	p\o	(0, 1, 4, 6)
きゃ	kya	k' a	(0, 1, 4, 7)
しゃ	sha	S a	(0, 1, 5, 6)
ちゃ	cha	tS a	(0, 1, 5, 7)
てゃ	tya	t' a	(0, 1, 6, 7)
にゃ	nya	J a	(0, 2, 3, 4)
ひゃ	hya	C a	(0, 2, 3, 5)
みゃ	mya	m' a	(0, 2, 3, 6)
りゃ	rya	4' a	(0, 2, 3, 7)
ぎゃ	gya	N' a	(0, 2, 4, 5)
じゃ	ja/jya	dZ a	(0, 2, 4, 6)
でゃ	dya	d' a	(0, 2, 4, 7)
びゃ	bya	b' a	(0, 2, 5, 6)
ぴゃ	pya	p' a	(0, 2, 5, 7)
ふゃ	fya	p\' a	(0, 2, 6, 7)
きゅ	kya	k' M	(0, 3, 4, 5)
しゅ	shu	S M	(0, 3, 4, 6)
ちゅ	chu	tS M	(0, 3, 4, 7)
てゅ	tyu	t' M	(0, 3, 5, 6)
にゅ	nyu	J M	(0, 3, 5, 7)
ひゅ	hyu	C M	(0, 3, 6, 7)
みゅ	myu	m' M	(0, 4, 5, 6)
りゅ	ryu	4' M	(0, 4, 5, 7)
ぎゅ	gyu	g' M	(0, 4, 6, 7)
じゅ	jyu	dZ M	(0, 5, 6, 7)
でゅ	dyu	d' M	(1, 2, 3, 4)
びゅ	byu	b' M	(1, 2, 3, 5)
ぴゅ	pyu	p' M	(1, 2, 3, 6)
ふゅ	fyu	p\' M	(1, 2, 3, 7)
きょ	kyo	k' o	(1, 2, 4, 5)
しょ	sho	S o	(1, 2, 4, 6)
ちょ	cho	tS o	(1, 2, 4, 7)
てょ	tyo	t' o	(1, 2, 5, 6)
にょ	nyo	J o	(1, 2, 5, 7)
ひょ	hyo	C o	(1, 2, 6, 7)
みょ	myo	m' o	(1, 3, 4, 5)
りょ	ryo	4' o	(1, 3, 4, 6)
ぎょ	gyo	N' o	(1, 3, 4, 7)

じょ	jo	dZ o	(1, 3, 5, 6)
でょ	dyo	d’ o	(1, 3, 5, 7)
びょ	byo	b’ o	(1, 3, 6, 7)
ぴょ	pyo	p’ o	(1, 4, 5, 6)

This table is based on data from the source code for this project, available at `https://github.com/yoyoyonono/evy1py`. A good amount of the romaji in this table is complete guesswork.

B. Images

Here you can see the various parts which comprise the VOACaloid. In the top-left corner there is the Raspberry Pi Zero, to the right of it is the eVY1 Shield, and under it is the MIDI controller.