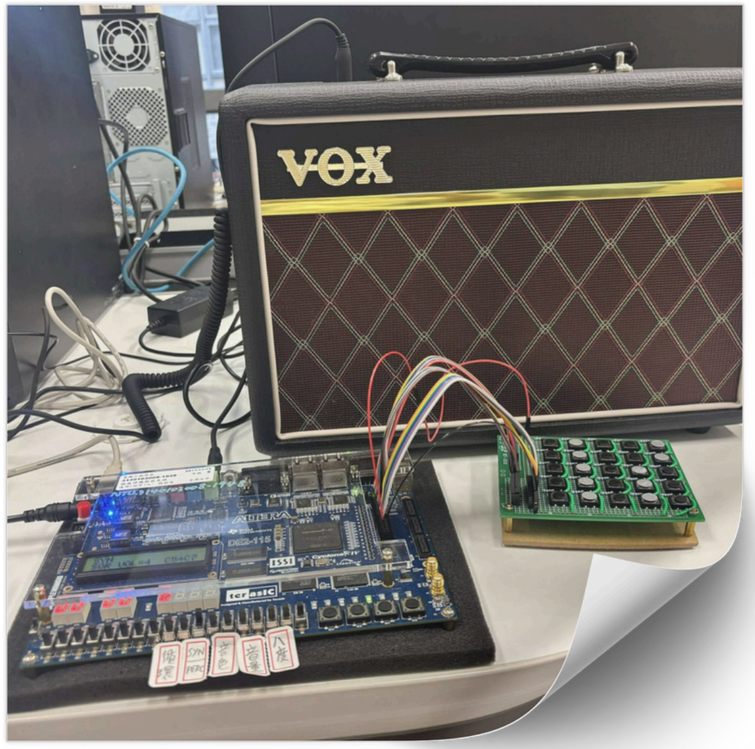
Music Mixer Pad

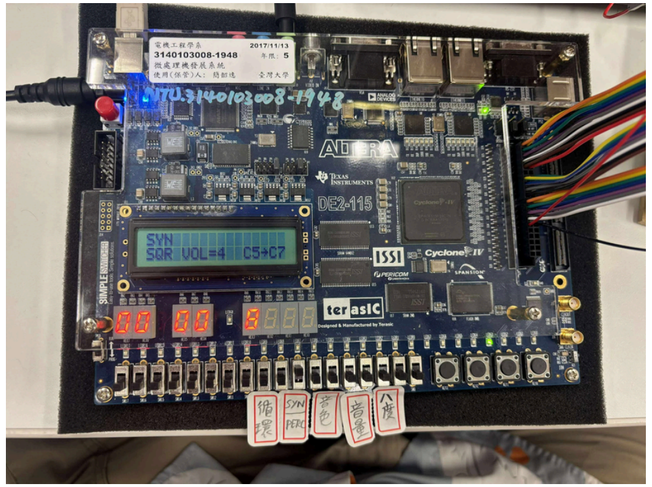


Presentated by
b10901010 劉又豪
b10901023 蔡仁揚
b10901062 方陳慶

Outline

- How to use it
- Features
- Architecture
- Data Flow & Storage
- Calculation
- **DEMO**

Slide Switches & Buttons



Button	3	2	0
	Reset	Play / Pause	Undo

Slide Switch	9	8	7	6 ~ 5	4 ~ 3	2 ~ 1
	Repetition	Preview	Perc / Syn	Wave Type	Volume	Octave
	0: Off 1: On	0: Off 1: On	0: Percussion 1: Synthesizer	00: Square 01: Sawtooth 10: Triangle 11: Sine	00: Vol = 4 01: Vol = 3 10: Vol = 2 11: Vol = 1	00: C5 ~ C7 01: C4 ~ C6 10: C3 ~ C5 11: C2 ~ C4

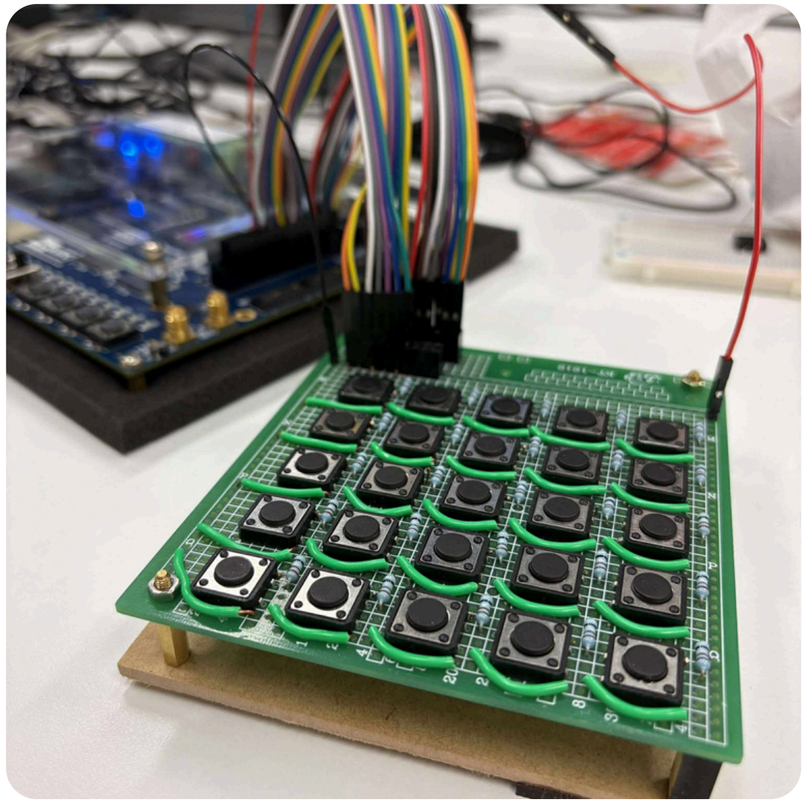
GPIO Buttons Array Module

- Pull-up Circuit
- Need debounce
- 3.3V and 9.09k ohm resistors
- 25 button fanout

$$\frac{1}{R_{\text{eq}}} = \sum_{i=1}^{n} \frac{1}{R} = \frac{n}{R} = \frac{25}{9090}$$

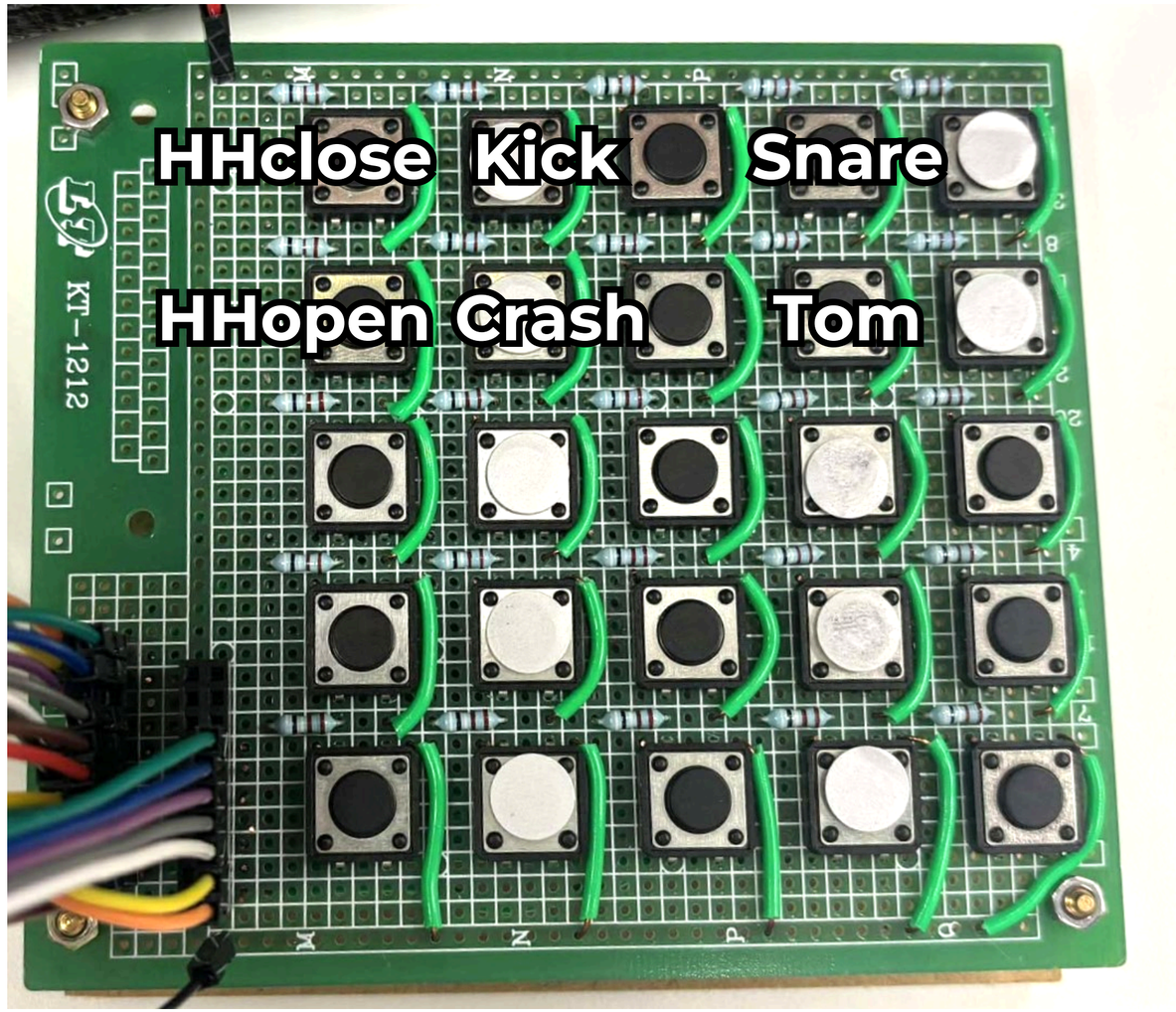
$$R_{\text{eq}} = \frac{9090}{25} = 363.6\,\Omega$$

$$I = \frac{V}{R_{\text{eq}}} = \frac{3.3}{363.6} \approx 0.00908\,\text{A} = \boxed{9.08\,\text{mA}}$$



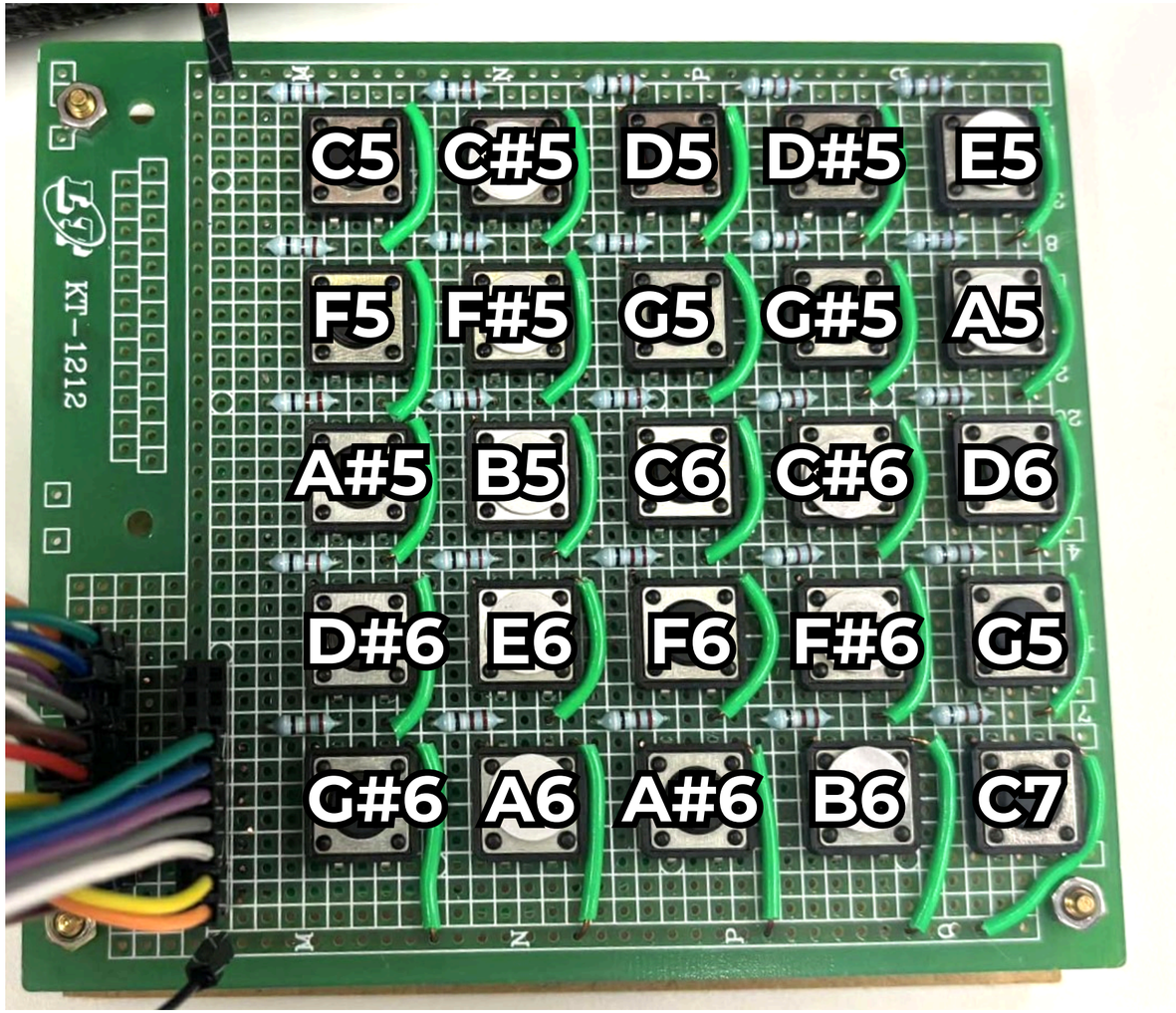
GPIO Buttons * 25

- Percussion



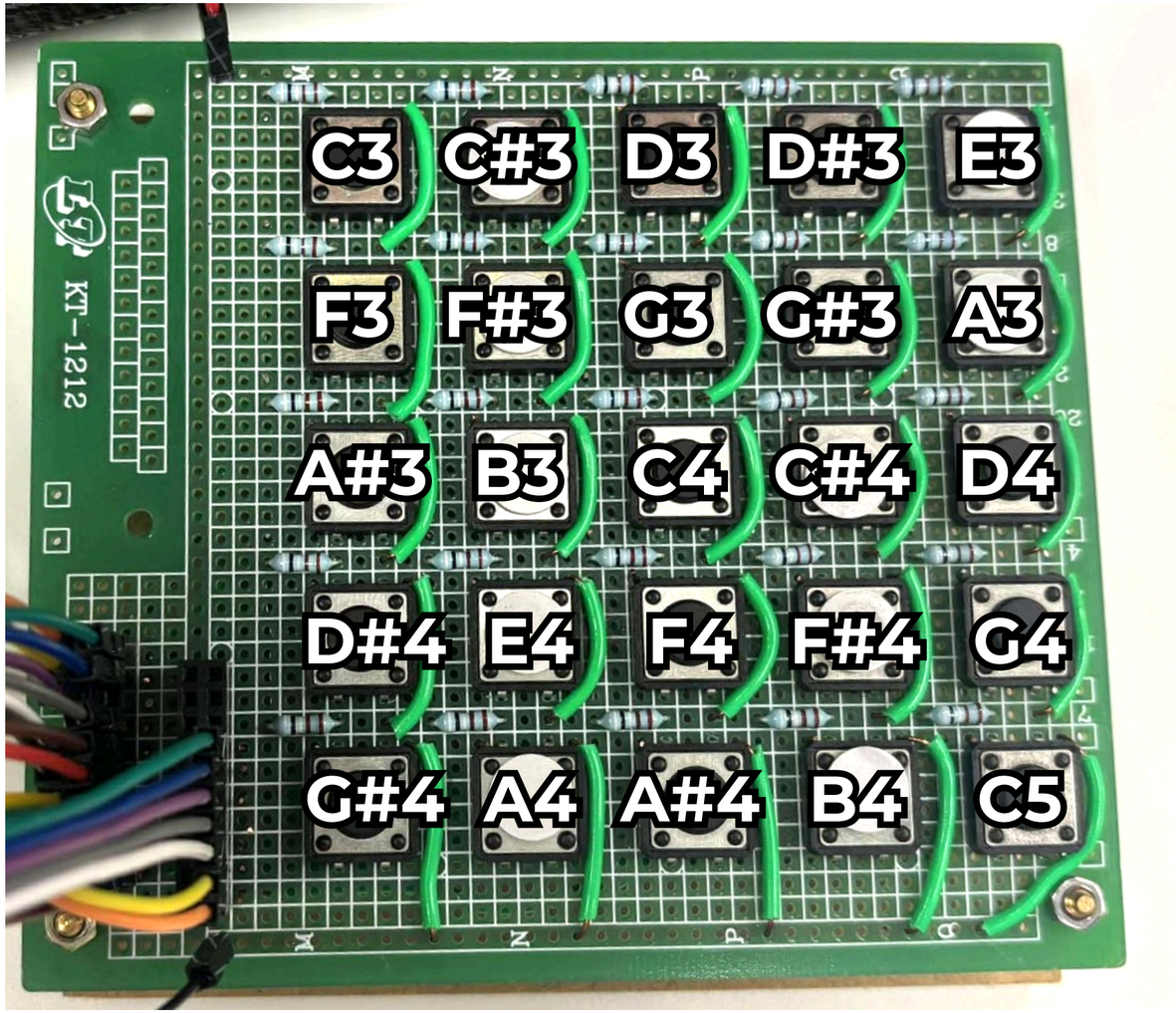
GPIO Buttons * 25

- **Synthesizer**
  - Octave
    - C5 ~ C7



GPIO Buttons * 25

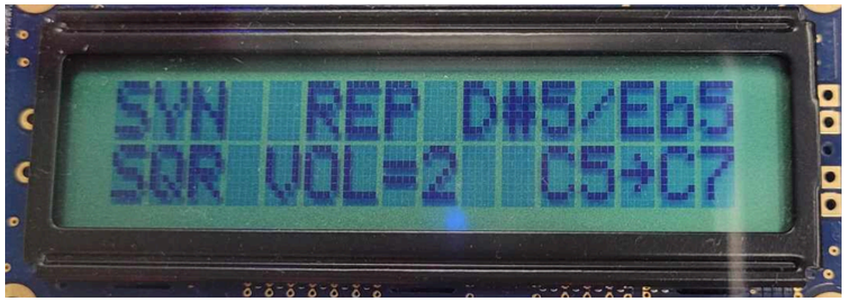
- **Synthesizer**
  - Octave
    - C3 ~ C5



LCD Monitor (Percussion)

ROW1	**15 ~ 12**	**10 ~ 8**	**6 ~ 0**
	PERC	Repetition On/Off	Percussion Name
ROW2	**15 ~ 13**	**11 ~ 7**	**4 ~ 0**
		Volume	

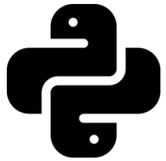
LCD Monitor (Synthesizer)



ROW1	**15 ~ 12**	**10 ~ 8**	**6 ~ 0**
	SYN	Repetition On/Off	Pitch Name
ROW2	**15 ~ 13**	**11 ~ 7**	**4 ~ 0**
	SQR/SAW/TRI/SIN	Volume	Pitch Range

Writting Perc. Sound in SRAM

Original sound files are **16-bit signed, 32kHz WAV format**.



Use Python to:

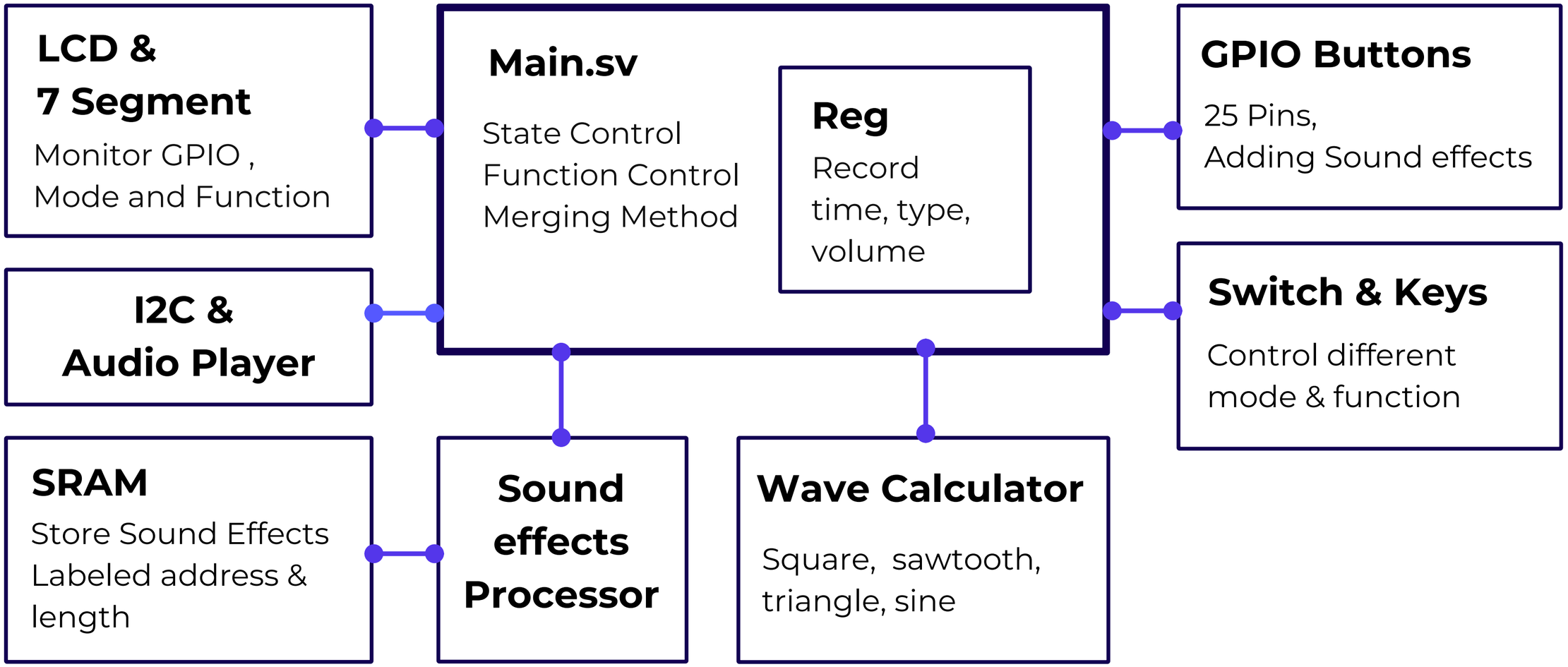
- Combine all sound effects into a single **binary file**
- Generate a CSV file containing the **start address and length** of each sound effect

We use a **Control Panel** to load binary files.

Features

- **25 sound effects** (2MB in on-chip SRAM)
- **4 waveform types**, pitch range **C2–C7**
- **Volume control** per sound
- Stores up to **1024 entries**
- **Preview** mode (play without saving)
- **Undo** support (Ctrl+Z)
- **LCD monitor** for feedback
- Built-in **metronome** for timing

Architecture



Data Flow

- Button triggered (GPIO)
- Writing to Regs (Main.sv)
- Read the Regs during DAC_LRCK = 1
- Find the time triggered equal to current time
- Calculate and merge the sound during DAC_LRCK = 1

Data Storage

- Storing pressed data using registers
- 1024 x 33 bits
- On/off bit is for wave enable/disable

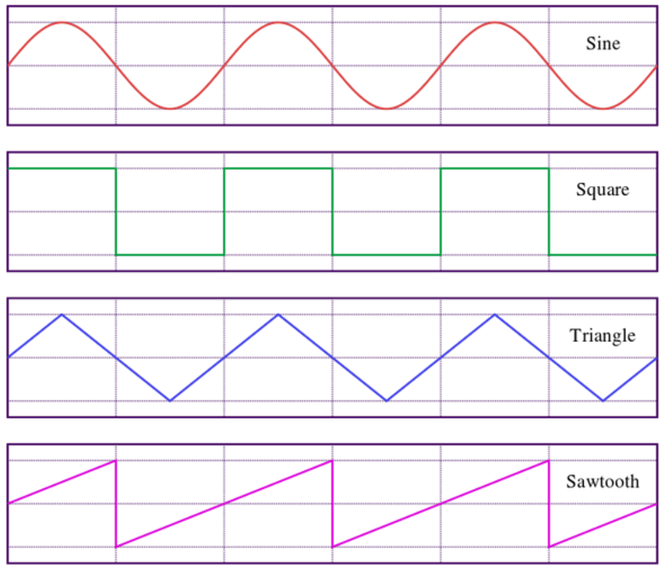
bits	**32**	**31**	**30 : 29**	**28 : 27**	**26 : 25**	**24 : 20**	**19 : 0**
	Perc/ Syn	on/ off	wave type	volume (12.5~100%)	octave (C2~C7)	button ID	triggered time

Data Storage

- A **Counter** to record the **number of datas** in the Regs
- Traverse all data in Regs during DAC_LRCK = 1
- Advantage:
  - Edit sound track easily.
- Preview Mode:
  - store a checkpoint for counter.
  - When off preview mode, let the counter = checkpoint
- Undo: erase the last data in the Regs.

Wave Calculation

- Sine wave: mini - LUT
- Square wave: calculate the triggered time.
- Triangle wave: interpolation
- Sawtooth wave: interpolation



Wave Calculation

- Construct a **adjusted frequency list**.
- Calculate the corresponding amplitude with the corresponding time.
- Store enable / disable state to evaluate the duration.

$$
\begin{aligned}
\theta &= \left[\frac{\text{count}}{32000} \Big/ \frac{1}{f} \right] \\
&= \left[\frac{\text{count} \cdot f \cdot \frac{32768}{32000}}{32768} \right] \\
&= \text{the last 15 bits of count} \cdot \left(f \cdot \frac{32768}{32000} \right)
\end{aligned}
$$

$$
\begin{aligned}
f_{\text{A4}} &= 440 \\
f_{\text{A4, adjusted}} &= 440 \cdot \left(\frac{32768}{32000} \right) \\
&\approx 451
\end{aligned}
$$

DEMO

By 靈魂樂手 方吉