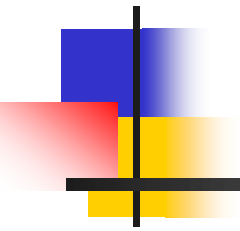
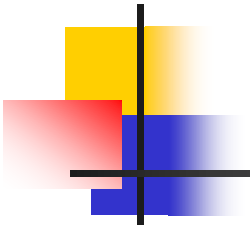


ESP32forth for AI Robot

C. H. Ting

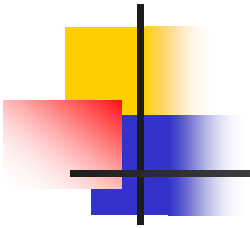
January 26, 2019

SVFIG



Summary

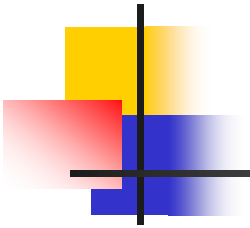
- NodeMCU ESP32S kit
- Esp32forth v5.2
- 8 channel electronic organ
- Organ demo
- Arduino IDE
- Verbase compiler output
- Esp32forth machine code
- Esp32forth demo



ESP32

- Dual 32-bit Xtensa LX106, 240 MHz
- 520 KB SRAM, 4 MB flash
- 28 GPIO pins
- 8 ADC, 2 DAC
- 3 UART, 2 SPI, 3 I2C
- WiFi: IEEE 802.11 b/g/n/e/I
- Bluetooth

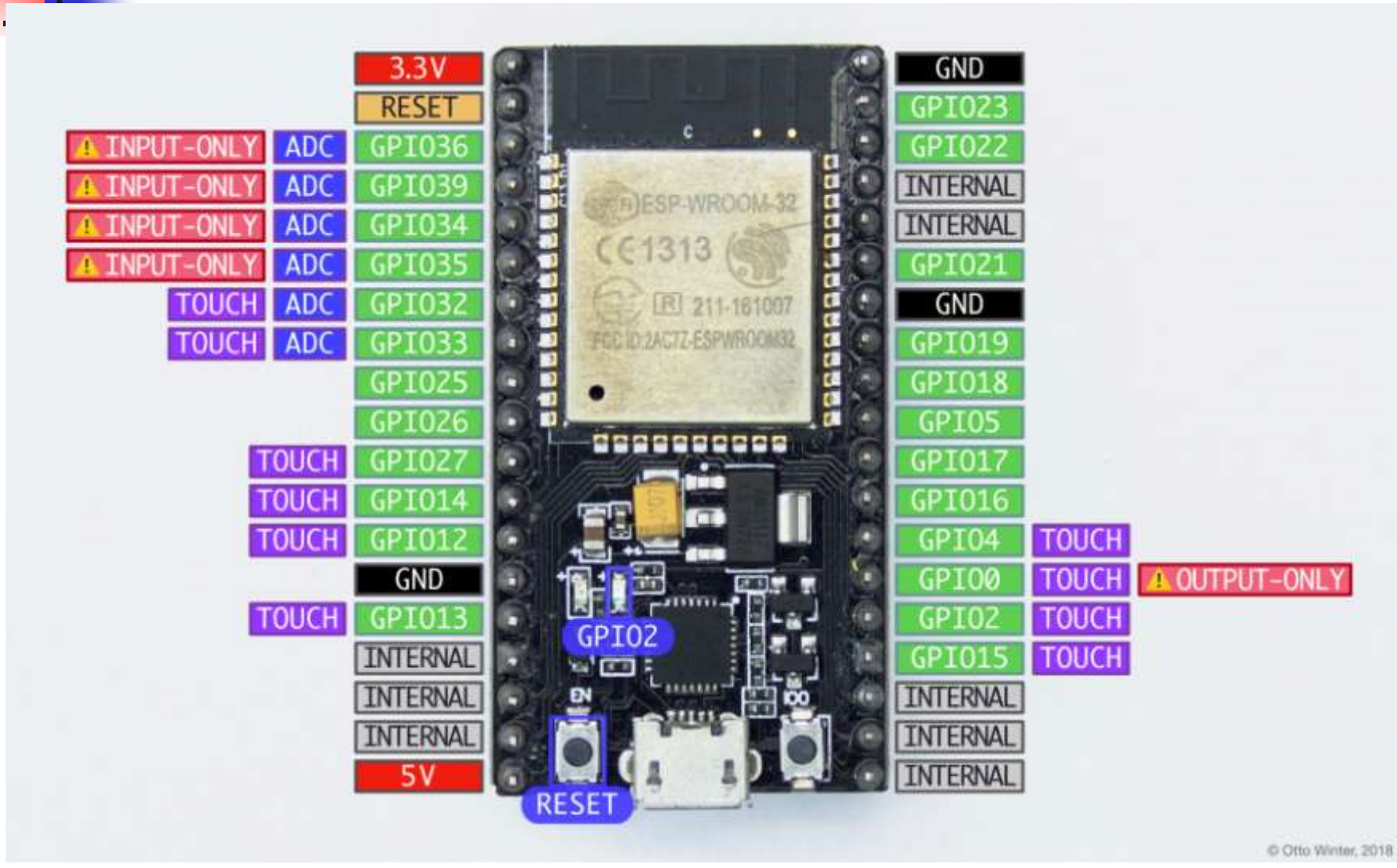
NodeMCU ESP32S

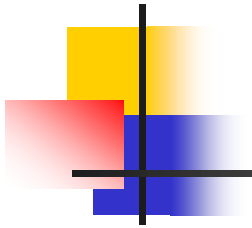


ESP32

- **ESP32 is twice more powerful than ESP8266.**
- **It is ideally suitable for an intelligent robot.**
- **Arduino IDE work well with ESP32.**
- **esp32Forth v5.2 is extended for AIR robot.**

NodeMCU ESP32S

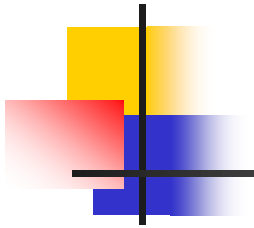




esp32Forth

- esp32Forth emulates eP32, a 32-bit Forth microcontroller.
- esp32Forth is written in C as a sketch on Arduino IDE, extended to run on ESP32 microcontroller.
- esp32Forth Finite State Machine:

```
{primitives[char cData[P++]]();}
```



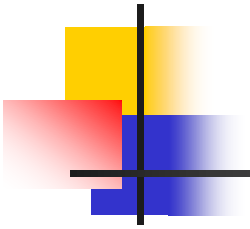
PEEK and POKE

- esp32Forth v5.2 is a Virtual Forth Machine implemented in C.
- It addresses 64 KB of virtual memory.
- PEEK and POKE address absolute address space, and can be used to control IO devices.

Code of PEEK and POKE

```
void poke(void)
{Pointer=(long*)top;*Pointer =
  stack[(unsigned char)S--];pop; }

void peek(void)
{Pointer=(long*)top;top =*Pointer;}
```



Primitive Code

```
void pin(void)
{WP=top;pop;ledcAttachPin(top,WP);pop;}
void duty(void)
{WP=top;pop;ledcAnalogWrite(WP,top,255);
pop;}
void freq(void)
{WP=top;pop;ledcSetup(WP,top,13);pop;}
```

Primitive Code

```
void audio(void)
{WP=top;pop;ledcWriteTone(WP,top);pop
  ;}
void sendPacket(void)
{Udp.endPacket();Udp.beginPacket
 (Udp.remoteIP(),Udp.remotePort());}
void adc(void)
{top=(long)analogRead(top);}
```

Forth Words

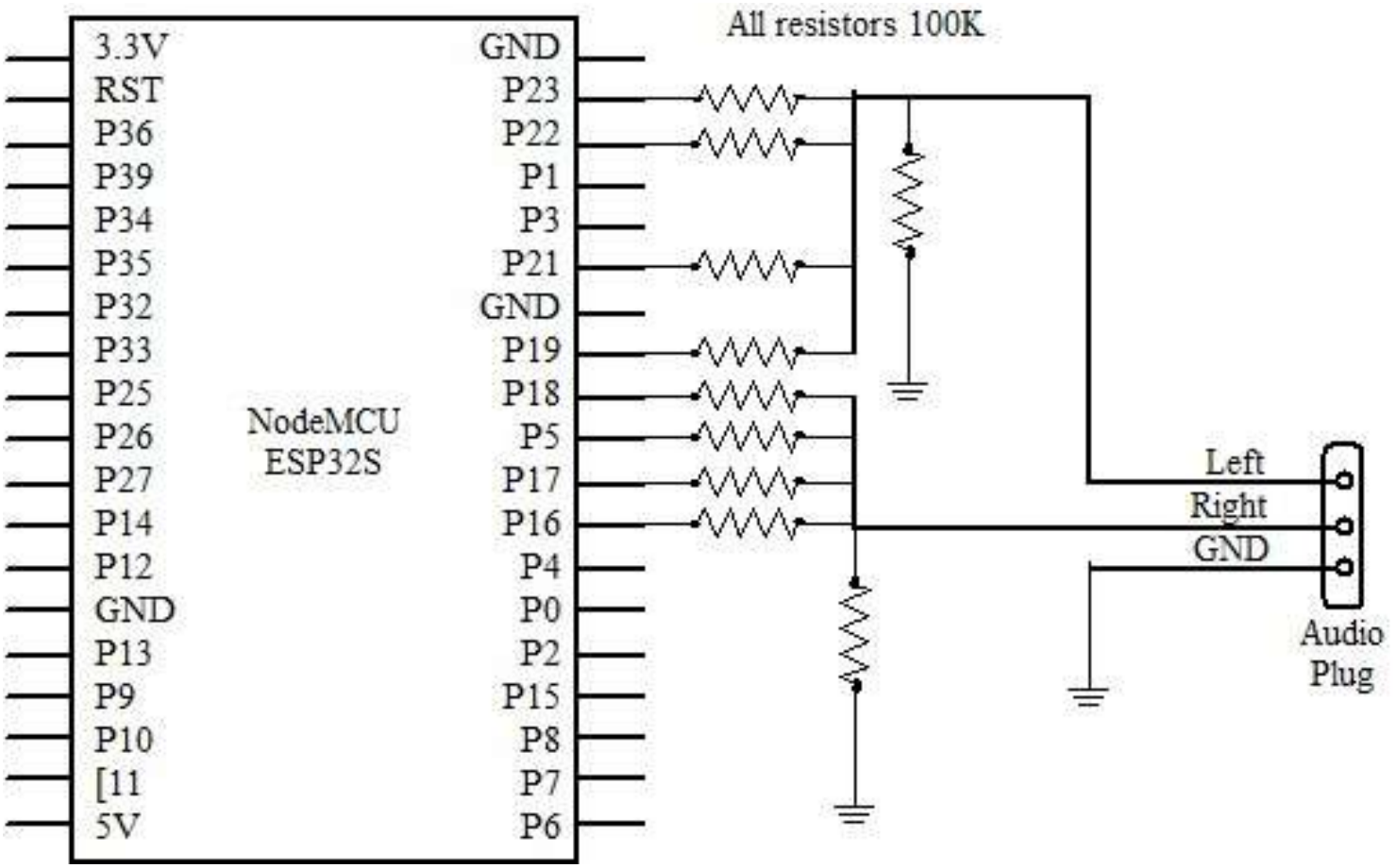
```
CODE sendPacket sendPacket, next,
CODE POKE poke, next,
CODE PEEK peek, next,
CODE ADC adc, next,
CODE PIN pin, next,
CODE TONE tone, next,
CODE DUTY duty, next,
CODE FREQ freq, next,
```

Forth GPIO Words

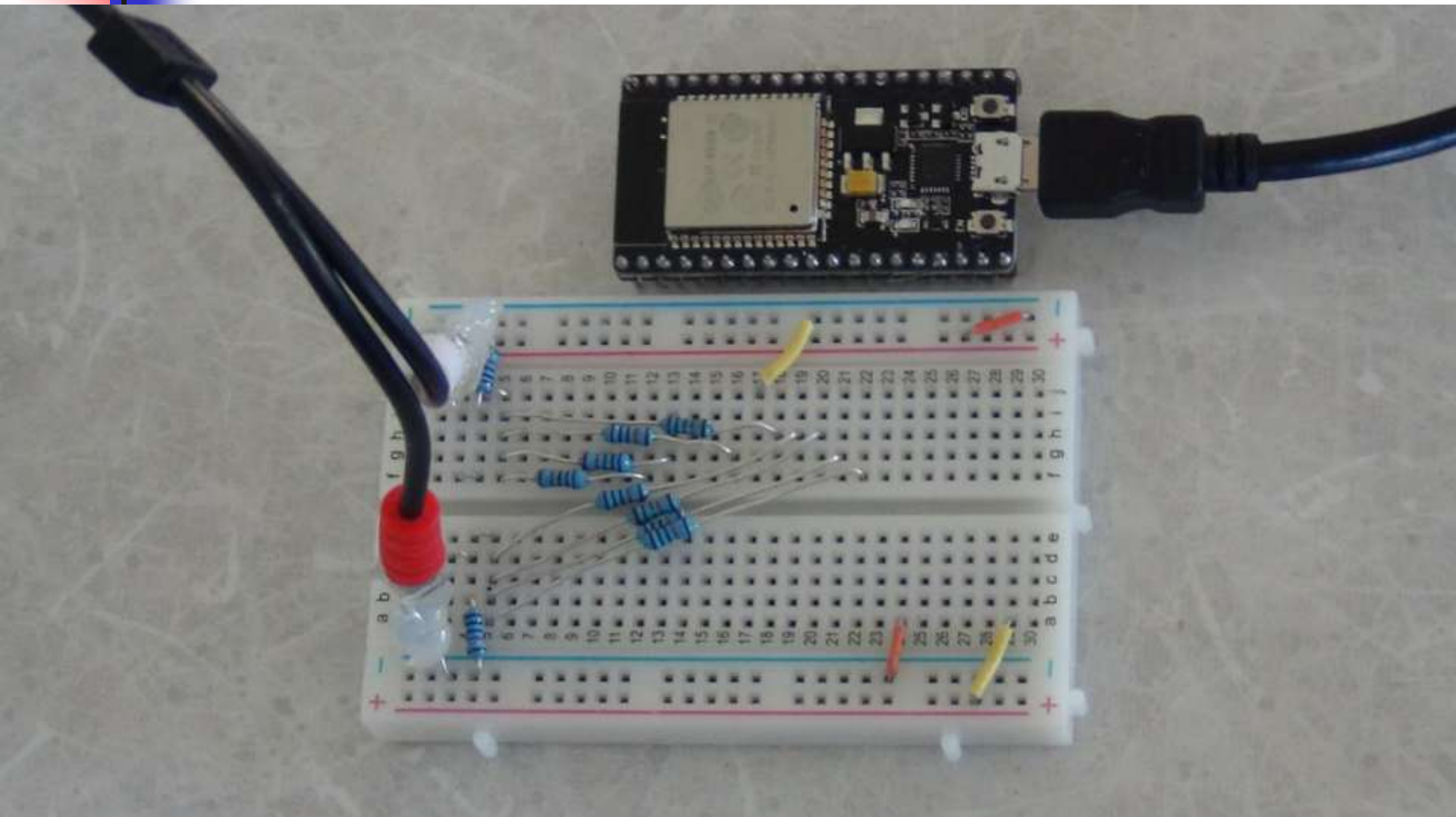
```
:: P0 $3FF44004 LIT POKE ;;
:: P0S $3FF44008 LIT POKE ;;
:: P0C $3FF4400C LIT POKE ;;
:: P1 $3FF44010 LIT POKE ;;
:: P1S $3FF44014 LIT POKE ;;
:: P1C $3FF44018 LIT POKE ;;
:: P0EN  $3FF44020 LIT POKE ;;
:: P0ENS  $3FF44024 LIT POKE ;;
:: P0ENC  $3FF44028 LIT POKE ;;
:: P1EN $3FF4402C LIT POKE ;;
:: P1ENS $3FF44030 LIT POKE ;;
:: P1ENC $3FF44034 LIT POKE ;;
:: P0IN  $3FF4403C LIT PEEK . ;;
:: P1IN $3FF44040 LIT PEEK . ;;
```

Electronic Organ

Electronic Organ



Electronic Organ



Electronic Organ

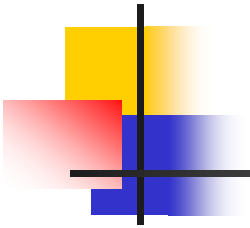
```
:: notes ( notes mask -- mask )
 3 LIT FOR DUP 1 LIT AND ( notes mask ? )
 IF SWAP DUP ( note mask n1 n1 )
 IF channel @ TONE ( note mask )
 ELSE DROP 128 LIT channel @ DUTY ( note mask )
 THEN
 THEN 2/ 2 LIT channel +!
 NEXT ;;
:: PLAY ( notes mask delay -- )
 >R Z channel ! notes 16 LIT / notes
 DROP R> FOR AFT THEN NEXT ;;
```

Electronic Organ

```
:: INIT ( freq -- )
 DUP 2DUP 2DUP 2DUP $F0F LIT 100 LIT PLAY ;;
:: HUSH 0 INIT ;;
1047 CONSTANT C6 988 CONSTANT B5 932 CONSTANT A5#
932 CONSTANT B5b 880 CONSTANT A5 831 CONSTANT G5#
831 CONSTANT A5b 784 CONSTANT G5 740 CONSTANT F5#
740 CONSTANT G5b 698 CONSTANT F5 659 CONSTANT E5
622 CONSTANT D5# 622 CONSTANT E5b 587 CONSTANT D5
554 CONSTANT C5# 554 CONSTANT D5b 523 CONSTANT C5
```

Electronic Organ

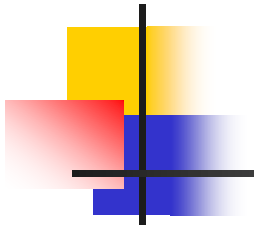
```
:: ppqn@ ppqn @ ;;
:: 1/4 ppqn@ PLAY ;;
:: 1/2 ppqn@ 2* PLAY ;;
:: 1/1 ppqn@ CELLS PLAY ;;
:: 1/8 ppqn@ 2/ PLAY ;;
:: 1/16 ppqn@ CELL/ PLAY ;;
:: 1/32 ppqn@ 8 LIT / PLAY ;;
:: 1/64 ppqn@ 16 LIT / PLAY ;; CRR
```



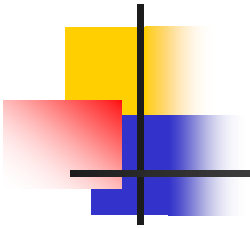
Electronic Organ

```
:: P1 G3 B3 D4 7 1/4 A3 E4 3 1/8 C4 1 1/8
 B3 D4 3 1/4 C4 G4 3 1/8 A4 1 1/8
 D4 B4 3 1/4 C4 A4 3 1/8 F4# 1 1/8
 B3 G4 3 1/4 C4 D4 3 1/4
 B3 G4 3 1/4 A3 A4 3 1/8 F4# 1 1/8
 B3 G4 3 1/8 A4 1 1/8 C4 B4 3 1/8 C5 1 1/8
 B3 D4 3 1/8 B4 1 1/8 A3 C5 3 1/8 A4 1 1/8
 G3 G3 B4 7 1/2 ;;
...
:: MUSETTE $100000 LIT ppqn !
 P1 P1 P2 P3 P4 P2 P3 P4 ;;
```

Organ Demo

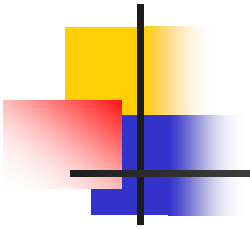


- 8 Channel electronic organ
- 8 Digital outputs are summed to a left voice and a right voice through an array of 100KΩ resistors.
- Left and right voices are amplified by a speaker.
- A musette dance and a fugue.



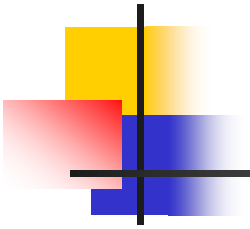
Arduino IDE

- Arduino seems to be the universal IDE for all microcontrollers.
- Sketches are C++ like programs very easy to write even for beginners.
- Its tool chains are hidden from casual users.



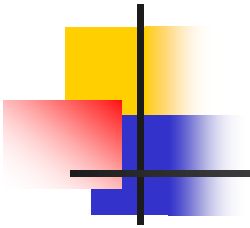
Arduino IDE

- Setting verbose output will show you the compile and upload command sequences.
- Verbose output can be saved to a batch (.bat) file. When the batch file is executed, Windows opens a cmd console to repeat the build.



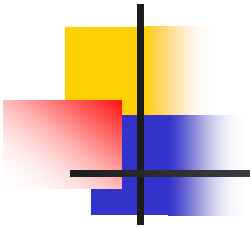
Arduino IDE

- Individual command copied to a batch file can be executed and tested.
- I used –Wa,-a option to force gcc to spill out assembly code for my espForth_51.ino sketch.



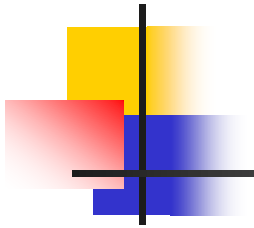
Xtensa Machine Code

- Xtensa is a CPU architecture originally designed by Cadence and then implemented by Tensilica.
- It has a RSIC-like 32-bit core with 24 bit instructions, many of them have 16-bit reduced instructions to save memory.



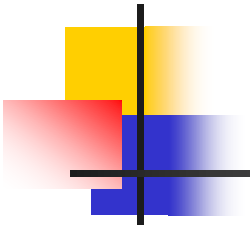
Xtensa Machine Code

- It has 16 registers. Instructions has 4 bit fields, allowing up to 3 registers to be reference in a single instruction.
- Machine code are very difficult to read because instructions are of variable lengths, though data are always 32-bits aligned to cells.



Primitive Code: next

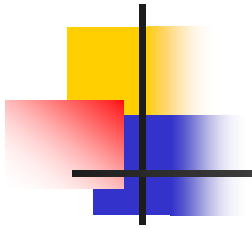
```
17 0000 364100          entry  sp, 32
20 0003 A10000          l32r   a10, .LC0
21 0006 B10000          l32r   a11, .LC1
22 0009 980A            l32i.n a9, a10, 0
23 000b 908221          srai   a8, a9, 2
24 000e B088A0          addx4  a8, a8, a11
25 0011 8808            l32i.n a8, a8, 0
27 0013 4B99            addi.n a9, a9, 4
29 0015 B10000          l32r   a11, .LC2
31 0018 990A            s32i.n a9, a10, 0
33 001a 910000          l32r   a9, .LC3
35 001d 890B            s32i.n a8, a11, 0
37 001f 4B88            addi.n a8, a8, 4
38 0021 8909            s32i.n a8, a9, 0
39 0023 1DF0            retw.n
```



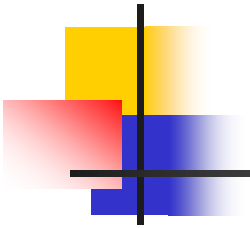
Primitive Code: dolist

```
134 0000 364100          entry  sp, 32
137 0003 910000          l32r   a9, .LC14
138 0006 820900          l8ui   a8, a9, 0
139 0009 1B88            addi.n a8, a8, 1
140 000b 808074          extui  a8, a8, 0, 8
141 000e 824900          s8i    a8, a9, 0
142 0011 910000          l32r   a9, .LC15
143 0014 9088A0          addx4  a8, a8, a9
144 0017 910000          l32r   a9, .LC16
145 001a A809            l32i.n a10, a9, 0
146 001c A908            s32i.n a10, a8, 0
148 001e 810000          l32r   a8, .LC17
149 0021 8808            l32i.n a8, a8, 0
150 0023 826900          s32i.n a8, a9, 0
152 0026 810000E0        call8  _Z4nextv
152      0800
154 002c 1DF0            retw.n
```

Primitive Code: exit

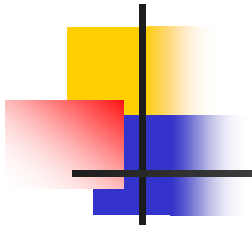


```
168 0000 364100          entry  sp, 32
171 0003 910000          l32r   a9, .LC18
172 0006 820900          l8ui   a8, a9, 0
173 0009 0BA8            addi.n a10, a8, -1
174 000b A24900          s8i    a10, a9, 0
175 000e 910000          l32r   a9, .LC20
176 0011 9088A0          addx4  a8, a8, a9
177 0014 9808            l32i.n a9, a8, 0
178 0016 810000          l32r   a8, .LC19
179 0019 9908            s32i.n a9, a8, 0
181 001b 810000E0        call8  _Z4nextv
181      0800
183 0021 1DF0            retw.n
```

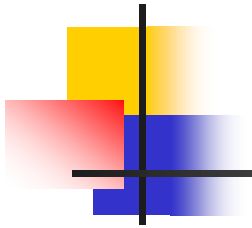


Esp32forth Demo

- Arduino IDE
- Sketch esp32forth_52
- Verbose compiler output
- Compile-Upload
- Gcc command sequence
- Serial Monitor
- Esp32forth



Questions?



Thank you.