



รายงาน

เรื่อง Mini-Project Game Love Love Jellyfish

จัดทำโดย

นางสาว ชญาดา วนิชกุลพิทักษ์ 6601012620071

นาย นภสินธุ์ เรนเรือง 6601012620119

นางสาว อลีนา แก้วกฤษา 6601012620135

เสนอ

ดร. เว้ด ศิริโภคภิรมย์

รายงานนี้เป็นส่วนหนึ่งของวิชา Introduction To Signal And System

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ภาคเรียนที่ 1 ปีการศึกษา 2567

คำนำ

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา Introduction To Signal And System (010123106)

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ของภาคเรียนที่ 1 ปีการศึกษา 2567 มีจุดประสงค์เพื่อ ศึกษาการใช้ Hardware เช่น บอร์ด FPGA, Microphon และ UART เพื่อเข้ามาร่วมกับ Software โดยรายงานฉบับนี้มีเนื้อหาเกี่ยวกับการใช้ Fast Fourier Transfrom และ Pygame มาประยุกต์ใช้ ขั้นตอนเจ้าหัวว่ารายงานเล่มนี้จะเป็นประโยชน์ต่อผู้อ่าน ถ้าหากรายงานเล่มนี้มีผลลัพธ์ที่ได้ต้องขออภัยและน้อมรับคำติชมเพื่อปรับปรุง

ขอขอบคุณ ดร. เวสต์ ศิริกาภิรมย์ ที่ให้คำแนะนำในการทำงานให้สำเร็จไปด้วยดี

ผู้จัดทำ

สารบัญ

เนื้อหา	หน้า
สารบัญ	3
แนวคิดทฤษฎีที่ใช้ในการทำ MINI – PROJECT	4
MICROPHONE (MAX9814).....	4
ANALOG TO DIGITAL CONVERTER (ADC)	4
FPGA (FIELD PROGRAMMABLE GATE ARRAY)	5
UART (UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER).....	7
FFT (FAST FOURIER TRANSFORM).....	9
SPRITE (COMPUTER GRAPHICS).....	10
DEMO GAME	11
ข้อจำกัดของโครงการ	15
CODE VHDL	16
CODE MAINVHDL/ADC_CONTROLLER.VHD	16
CODE MAINVHDL/UART_TX.VHD	20
CODE MAINVHDL/MAINVHDL.VHD	22
CODE MAINVHDL/MAINVHDL.QSF	23
CODE GAME	25
CODE PYGAME/SRC/CONFIG.PY	25
CODE PYGAME/SRC/INIT.PY	25
CODE PYGAME/SRC/INTROSCREEN.PY	26
CODE PYGAME/SRC/JELLYFISHSPRITE.PY.....	29
CODE PYGAME/SRC/MAIN.PY	31
CODE PYGAME/SRC/OBJECTSPRITE.PY.....	35
CODE PYGAME/SRC/OUTTROSCREEN.PY	36
CODE PYGAME/SRC/READFHZAUDIO.PY.....	37
CODE PYGAME/SRC/READSERIALFREQ.PY	38
บรรณานุกรม	40

แนวคิดทดลองที่ใช้ในการทำ Mini – Project

Microphone (Max9814)

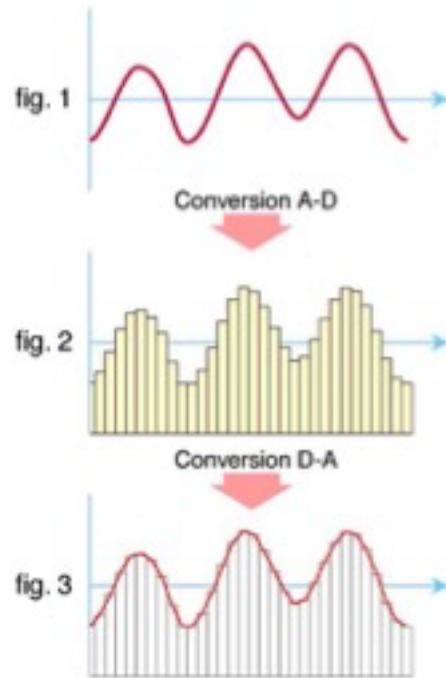


MAX9814 เป็นวงจรขยายสัญญาณไมโครโฟนที่มี AGC (Automatic Gain Control) และแรงดันอ้างอิงไมโครโฟนที่มีสัญญาณรบกวนต่ำ ประกอบด้วยตัวขยายสัญญาณเสียงเบาที่มาจากการรับไมโครโฟน (preamplifier), ตัวขยายสัญญาณแบบปรับได้ (Variable Gain Amplifier – VGA), ตัวขยายสัญญาณออก (Output Amplifier), ตัวสร้างแรงดันอ้างอิงไมโครโฟน และวงจรควบคุม AGC ไมโครโฟนชิป MAX9814 เมมะสำหรับงานที่ต้องการควบคุมระดับเสียงอัตโนมัติและลดสัญญาณรบกวน เพื่อให้การบันทึกเสียงมีคุณภาพดีและชัดเจน

Analog to Digital Converter (ADC)

ADC คือ อุปกรณ์หรือวงจรที่ใช้ในการแปลงสัญญาณ Analog ให้กลายเป็นสัญญาณ Digital โดยมีหน้าที่หลักในการรับสัญญาณที่เป็นคลื่นต่อเนื่อง (ซึ่งเป็นลักษณะของสัญญาณ Analog) เช่น สัญญาณเสียง แล้วแปลงค่าเหล่านั้นให้เป็นตัวเลขดิจิตอลที่สามารถใช้ในระบบคอมพิวเตอร์หรืออุปกรณ์ดิจิตอลอื่นๆ ได้

ADC ถูกใช้ในอุปกรณ์อิเล็กทรอนิกส์ เช่น ไมโครโฟนดิจิตอล เช่นเซอร์วัสดแรงดันไฟฟ้า และอุปกรณ์ตรวจจับอื่นๆ ที่ต้องการแปลงสัญญาณ Analog มาเป็นข้อมูล Digital เพื่อการประมวลผล



รูปนี้แสดงกระบวนการแปลงสัญญาณจาก Analog เป็น Digital

และจาก Digital กลับเป็น Analog

FPGA (Field Programmable Gate Array)

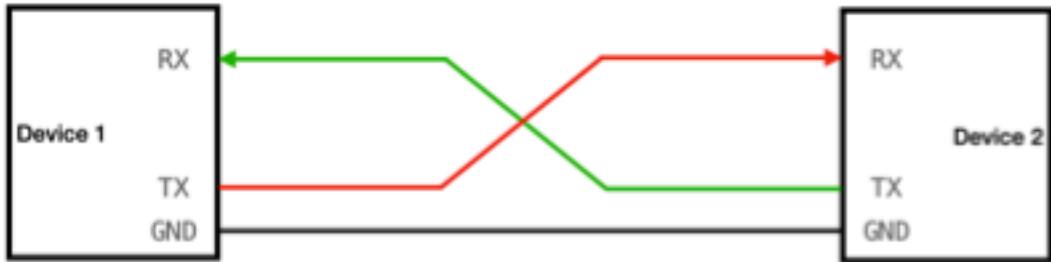


FPGA บอร์ด DE0-Nano ใช้ชิป FPGA รุ่น Cyclone IV จาก Altera
(ปัจจุบันเป็นส่วนหนึ่งของ Intel)

คุณสมบัติหลักๆ ของ DE0-Nano มีดังนี้:

- ชิป FPGA บอร์ดนี้ใช้ Cyclone IV FPGA ซึ่งมี logic elements (LE) จำนวน 22,320 LE รองรับการประมวลผลที่ซับซ้อน
- หน่วยความจำ มี SDRAM ขนาด 32 MB ใช้สำหรับเก็บข้อมูลที่ใช้ในการประมวลผล รวมถึงมี EEPROM ขนาด 2 Kb สำหรับการจัดเก็บค่าต่างๆ อย่างถาวร
- อินพุตและเอาต์พุต (I/O) มี LED สีเขียว 8 ดวง, ปุ่มกดแบบเด้งกลับ (debounced pushbuttons) 2 ปุ่ม และ DIP switch 4 ตำแหน่ง ซึ่งสามารถใช้เป็นสวิตซ์เพื่อควบคุมการทำงานต่างๆ
- การเชื่อมต่อแบบขยาย (Expansion Headers) มีพอร์ต GPIO ขนาด 40 พิน 2 ชุด ให้ความยืดหยุ่นในการเชื่อมต่อกับอุปกรณ์เพิ่มเติม เช่น Sensor Motor และอุปกรณ์ขยายอื่นๆ
- เซ็นเซอร์วัดการเคลื่อนไหว (Accelerometer) มีเซ็นเซอร์ ADXL345 สำหรับการวัดการเคลื่อนไหวแบบ 3 แกน สามารถใช้สำหรับแอปพลิเคชันที่ต้องการตรวจจับการเคลื่อนไหว
- A/D Converter มี A/D Converter แบบ 12 บิต 8 ช่องจาก National Semiconductor ใช้สำหรับอ่านค่าจากอุปกรณ์ Analog เช่น Sensor
- วงจรนาฬิกา (Clock) มีอสซิลเลเตอร์ความถี่ 50 MHz บันบอร์ด ใช้เป็นแหล่งสัญญาณนาฬิกาสำหรับ FPGA บอร์ดนี้ยังมาพร้อมกับ USB ที่ใช้ในโปรแกรม FPGA ผ่านซอฟต์แวร์ Quartus II ของ Altera ซึ่งสามารถออกแบบวงจร Digital ในรูปแบบของ HDL (Hardware Description Language) เช่น Verilog หรือ VHDL และดาวน์โหลดลงในบอร์ดเพื่อทดลองใช้งานได้

UART (Universal Asynchronous Receiver/Transmitter)



UART เป็นไปริโตร็อกเกลส์อาร์แบบอนุกรมที่ใช้เพื่อส่งข้อมูลระหว่างอุปกรณ์สองตัว ซึ่งอุปกรณ์นี้จะเปลี่ยนข้อมูลระหว่างรูปแบบ Analog และ Digital ทำงานโดยมีการส่งข้อมูลที่ลับบิดฝ่านสายสัญญาณแบบอนุกรม

การทำงานของ UART

4.1) การส่งข้อมูล : ข้อมูลจะถูกส่งเป็นบิต โดยจะถูกจัดเป็นแพ็กเกจในรูปแบบที่มีส่วนต่าง ๆ คือ บิตเริ่มต้น (Start Bit), บิตข้อมูล (Data Bits), บิตหยุด (Stop Bit), และอาจมีบิตตรวจสอบความถูกต้อง (Parity Bit) เพื่อช่วยตรวจสอบความถูกต้องของข้อมูล

4.2) Start Bit และ Stop Bit :

Start Bit : เป็นบิตที่ใช้เพื่อบอกว่าเริ่มต้นการส่งข้อมูล โดยจะเปลี่ยนสัญญาณจากสถานะสูง (High) ไปยังสถานะต่ำ (Low)

Stop Bit : บิตที่ใช้เพื่อบอกจุดสิ้นสุดของข้อมูล

4.3) Data Bits : บิตข้อมูลมักมีขนาด 7-8 บิต โดยข้อมูลจริงที่ต้องการส่งจะถูกแปลงให้อยู่ในบิตเหล่านี้

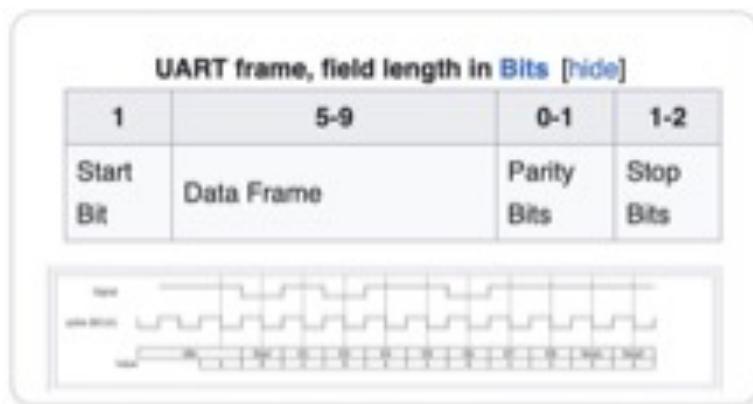
4.4) Parity Bit (อาจมีหรือไม่มี): ใช้ในการตรวจสอบความถูกต้องของข้อมูล โดยตรวจสอบว่าจำนวนบิต “1” มีจำนวนคู่หรือคี่ตามที่ตั้งค่าไว้ ถ้ามีความผิดพลาดในระหว่างส่งก็จะตรวจพบจาก Parity Bit นี้

4.5) Baud Rate : ความเร็วในการส่งข้อมูลซึ่งระบุเป็นจำนวนบิตต่อวินาที เช่น 9600, 115200 เป็นต้น โดยอุปกรณ์ทั้งสองฝ่ายต้องตั้งค่า Baud Rate เหมือนกัน

4.6) การรับข้อมูล: UART ในฝ่ายรับจะตรวจจับ Start Bit เพื่อตรวจสอบจุดเริ่มต้นของข้อมูล และทำการรับบิตข้อมูลที่ลับบิด เมื่อถึงบิตสุดท้ายหรือ Stop Bit จะทราบว่าข้อมูลนั้นสิ้นสุดแล้ว

องค์ประกอบของกรอบข้อมูล (frame) สำหรับ UART โดยมี 5 ส่วนประกอบหลักดังนี้ :

- Idle เป็นสถานะที่ไม่มีการส่งข้อมูล โดยสัญญาณจะอยู่ในระดับสูง (logic high หรือ 1)
- Start Bit ใช้เป็นจุดเริ่มต้นของการส่งข้อมูล โดยสัญญาณจะเปลี่ยนเป็นระดับต่ำ (logic low หรือ 0)
- Data Bits ส่วนนี้ประกอบด้วยบิตข้อมูลที่ต้องการส่ง
- Parity Bit บิตที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง (อาจมีหรือไม่มีก็ได้)
- Stop เป็นสัญญาณที่บ่งบอกว่าข้อมูลสิ้นสุดแล้ว โดยสัญญาณจะอยู่ในระดับสูง (logic high หรือ 1)

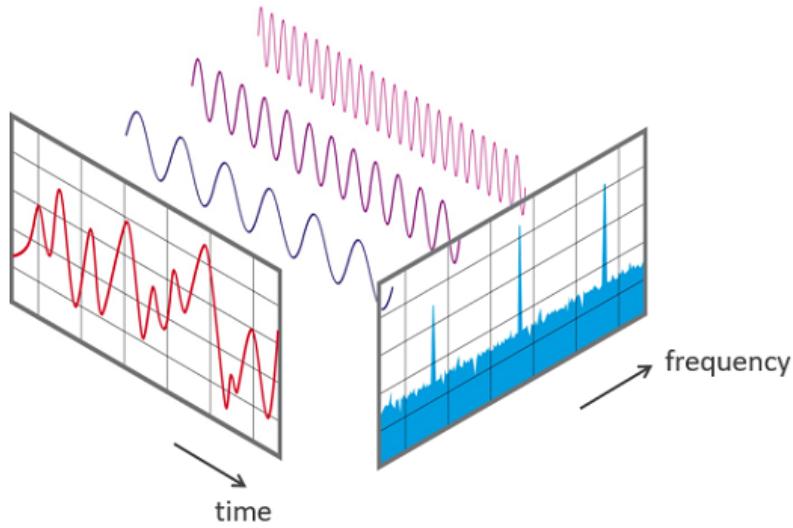


รูปนี้แสดงโครงสร้างของกรอบข้อมูล (frame) สำหรับ UART โดยมีรายละเอียดดังนี้ :

- Start Bit : มีค่า 1 บิต ใช้เป็นสัญญาณเริ่มต้นเพื่อบอกว่ากรอบข้อมูลใหม่กำลังถูกส่งมา
- Data Frame : มีความยาวระหว่าง 5 ถึง 9 บิต ขึ้นอยู่กับการตั้งค่า ใช้สำหรับบรรจุข้อมูลที่ต้องการส่ง
- Parity Bit (0-1 บิต) : เป็นบิตที่ใช้ตรวจสอบความถูกต้องของข้อมูลที่ส่งไป โดยสามารถเลือกใส่หรือไม่ใส่ก็ได้
- Stop Bit (1-2 บิต) : ใช้บอกจุดสิ้นสุดของกรอบข้อมูล

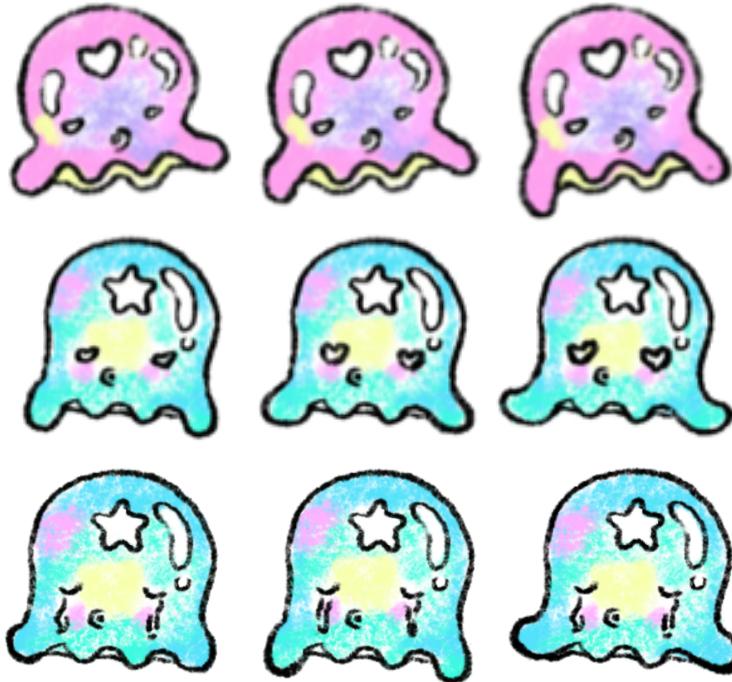
ด้านล่างของรูปเป็นแผนภาพเวลา (timing diagram) ที่แสดงลำดับการส่งสัญญาณในแต่ละบิต เริ่มจากสถานะ Idle (ไม่มีการส่งข้อมูล) จากนั้นส่งบิต Start ต่อด้วยข้อมูล D1 ถึง D8 และสิ้นสุดด้วย Stop Bits

FFT (Fast Fourier Transform)



เป็นอัลกอริทึมที่ใช้ในการคำนวณการแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform – DFT) อย่างรวดเร็ว โดยหลักการของ DFT คือการเปลี่ยนข้อมูลจากโดเมนเวลา (Time Domain) ไปเป็นโดเมนความถี่ (Frequency Domain) ซึ่งมีประโยชน์ในหลาย ๆ ด้าน เช่น การประมวลผลสัญญาณ การบีบอัดข้อมูล และการวิเคราะห์เสียง

Sprite (Computer Graphics)



ในบริบทของการพัฒนาเกมและกราฟิกคอมพิวเตอร์ Sprite (สไปรต์) คือภาพกราฟิกหรือวัตถุที่ใช้แทนตัวละคร, ไอเทม, หรือวัตถุอื่น ๆ ภายในเกม โดยทั่วไป Sprite เป็นภาพกราฟิกที่เคลื่อนไหวได้บนหน้าจอ มักจะใช้ในเกมสองมิติ (2D) เพื่อแสดงตัวละคร, ศัตรู, และไอเทมต่าง ๆ

Sprite ถูกจัดเก็บในรูปแบบของ bitmap หรือ texture ที่เป็นภาพตารางสี่เหลี่ยมซึ่งประกอบด้วยกราฟิกหลายภาพเพื่อใช้ในการแสดงผลลัพธ์การเคลื่อนไหวหรือเปลี่ยนแปลงของตัวละครนั้น ๆ การจัดเก็บและการเคลื่อนไหวของ Sprite สามารถทำได้หลายวิธี เช่น การเปลี่ยนเฟรมหรือการใช้เทคนิค sprite sheet ที่ประกอบไปด้วยเฟรมหลายเฟรมในไฟล์เดียว

การใช้งาน Sprite ใน การพัฒนาเกม ช่วยให้การจัดการกราฟิกง่ายขึ้น เนื่องจากสามารถนำภาพ Sprite มาแสดงบนหน้าจอ เปลี่ยนตำแหน่ง และปรับขนาดได้โดยไม่ต้องโหลดกราฟิกใหม่ทั้งหมด

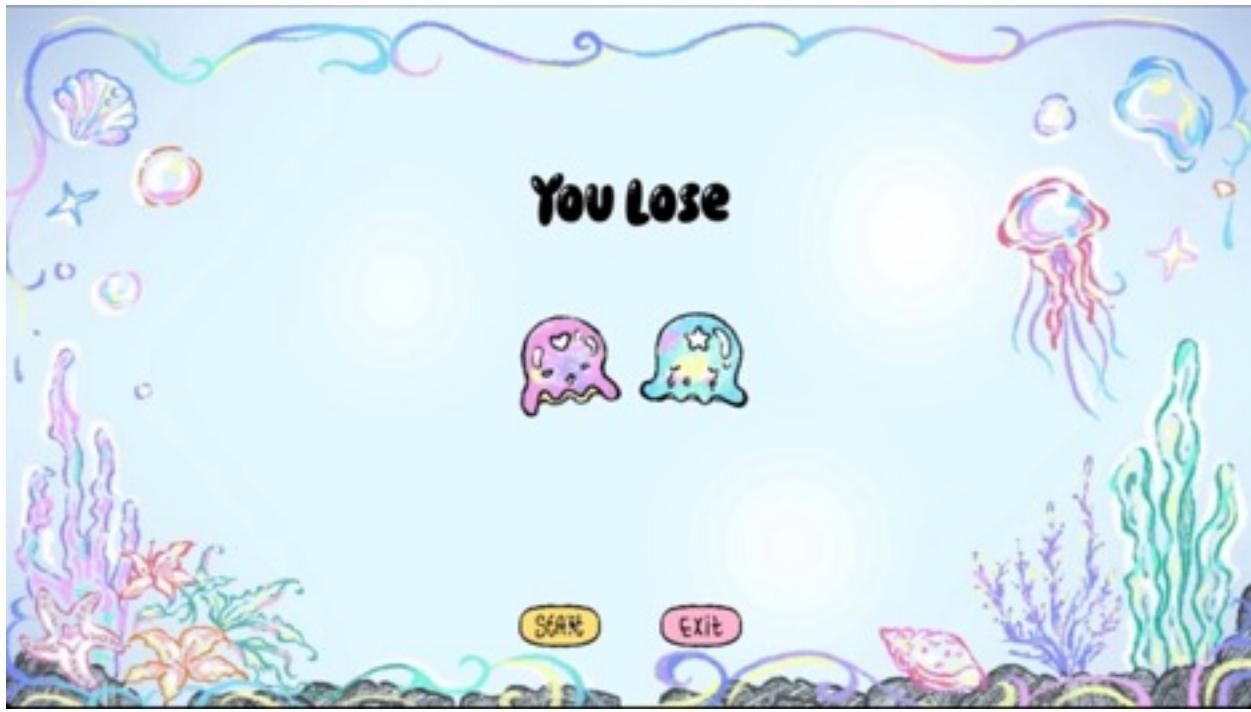
DEMO GAME



หน้าเริ่มต้นของเกม – มีคำอธิบายวิธีเล่นของเกม



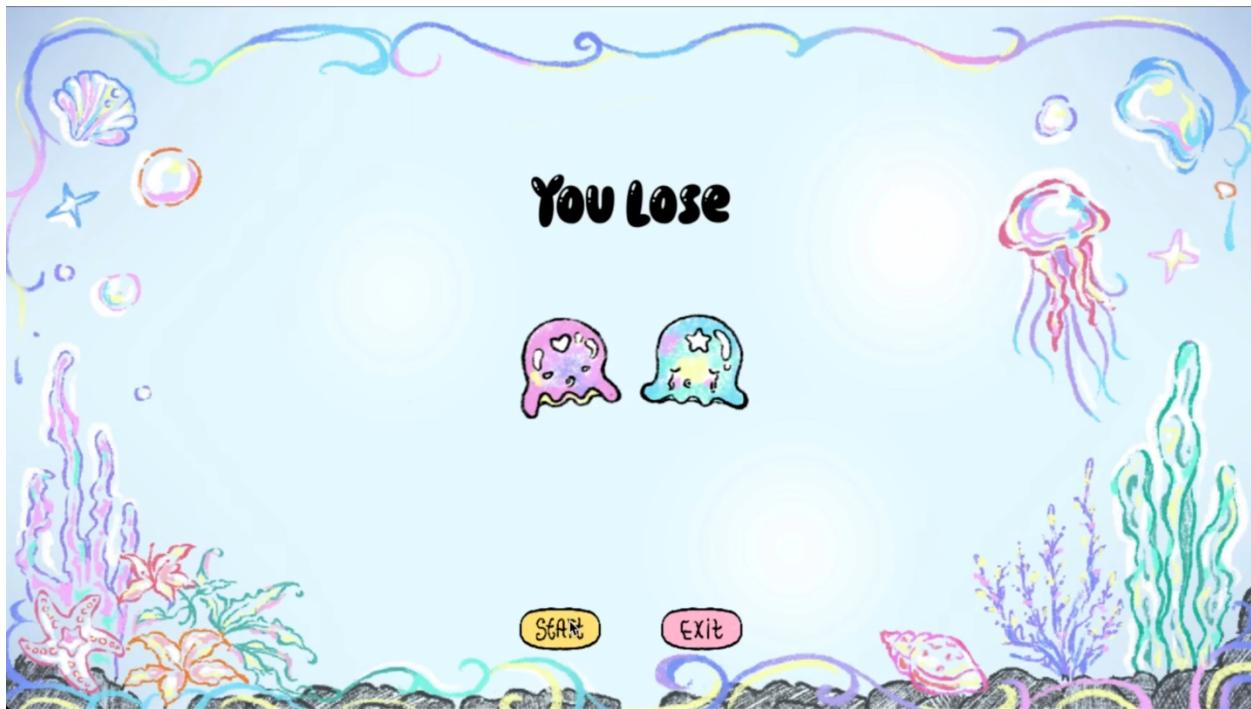
ภายในเกม - มีสิ่งกีดขวางต่างๆ โดยใช้เสียงเป็นตัวควบคุมแมงกะพรุน



เมื่อแมงกะพรุนชนกับสิ่งกีดขวางจะถือว่าแพ้ทันที



เมื่อผ่านสิ่งกีดขวางจนหมด จะเจอกับคู่รักของแมงกะพรุน



หากทั้งคู่ไม่ชนกัน จะถือว่าแพ้ทันที



หากทั้งคู่ชูนกันจะถือว่าชนะ

ข้อจำกัดของโครงการ

หากเราแบ่งการทำงานออกเป็นสองส่วน เราจะได้ส่วนย่อย ๆ คือ ส่วนของเกมและส่วนของการรับค่าอินพุต (ไมโครโฟน) ดังนี้:

1.) เกม

- ใช้ pygame

2.) อินพุต

- Pyserial (ใช้รับค่าอินพุตจากไมโครโฟนในโน๊ตบุ๊ก)
- Pyaudio (ใช้รับค่าอินพุตจากไมโครโฟนภายนอกที่เชื่อมต่อกับ FPGA)

ได้ดีสำหรับรับอินพุตผ่าน Pyserial อยู่ที่ pygame/src/readSerialfreq.py ซึ่งเมื่อทดสอบแล้วสามารถรับอินพุตและแปลงค่าเป็นความถี่ได้อย่างถูกต้อง

ได้ดีสำหรับรับอินพุตผ่าน Pyaudio อยู่ที่ pygame/src/readfHzaudio.py ซึ่งเมื่อทดสอบแล้วสามารถรับอินพุตและแปลงค่าเป็นความถี่ได้อย่างถูกต้อง

เมื่อวันนี้ได้ Pyaudio ร่วมกับ Pygame ผ่าน pygame/src/main.py พบร้าสามารถทำงานได้ถูกต้อง

เมื่อวันนี้ได้ Pyserial ร่วมกับ Pygame ผ่าน pygame/src/main.py พบร้าสามารถทำงานได้ แต่การอ่านค่าช้ากว่าปกติ

Code VHDL

Code mainVHDL/ADC_Controller.vhd

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY ADC_Controller IS
  PORT (
    CLK : IN STD_LOGIC; -- 50 MHz system clock
    NRST : IN STD_LOGIC; -- Active-low asynchronous reset
    ADC_CSN : OUT STD_LOGIC; -- ADC SPI chip-select
    ADC_SCLK : OUT STD_LOGIC; -- ADC SPI SCLK
    ADC_MOSI : OUT STD_LOGIC; -- ADC SPI MOSI
    ADC_MISO : IN STD_LOGIC; -- ADC SPI MISO
    uart_send_trigger : OUT STD_LOGIC;
    LEDS : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) -- 8-bit ADC OUTPUT
  );
END ADC_Controller;

ARCHITECTURE behavioral OF ADC_Controller IS

  CONSTANT SPI_CLK_DIV : INTEGER := 156;
  CONSTANT DATA_WIDTH : INTEGER := 16; -- for ADC128S022

  TYPE state_type IS (ST_IDLE, ST_START, ST_SCK_H, ST_SCK_L, ST_WAIT);
  SIGNAL state : state_type := ST_IDLE;

  SIGNAL cs_n : STD_LOGIC := '1';
  SIGNAL sclk : STD_LOGIC := '0';
  SIGNAL mosi : STD_LOGIC := '0';

  SIGNAL bit_index : INTEGER RANGE 0 TO DATA_WIDTH - 1 := 0;
  SIGNAL adc_data : STD_LOGIC_VECTOR(11 DOWNTO 0) := (OTHERS => '0');
  SIGNAL channel : STD_LOGIC_VECTOR(2 DOWNTO 0) := "000";

  SIGNAL shift_en : STD_LOGIC := '0';
  SIGNAL shift_reg : STD_LOGIC_VECTOR(DATA_WIDTH - 1 DOWNTO 0);

  CONSTANT WAIT_CNT_MAX : INTEGER := 31;
  SIGNAL wait_cnt : INTEGER := 0;
```

```

BEGIN

adc_csn <= cs_n;
adc_sclk <= sclk;
adc_mosi <= mosi;

-- LEDS <= adc_data(11 downto 4); -- show 8-bit ADC value directly to LEDs

PROCESS (adc_data)
  VARIABLE leds_on : INTEGER RANGE 0 TO 7;
  VARIABLE value : INTEGER RANGE 0 TO 255;
BEGIN

  -- value := to_integer(unsigned(adc_data(11 DOWNTO 4)));
  -- value := value / 32;
  -- leds_on := value;
  LEDS <= (OTHERS => '0'); -- Default to all LEDs OFF
  LEDS(7 DOWNTO 0) <= (adc_data(11 DOWNTO 4)); -- Turn ON the corresponding LEDs
END PROCESS;

PROCESS (CLK, NRST)
  VARIABLE count : INTEGER RANGE 0 TO SPI_CLK_DIV - 1 := 0;
BEGIN
  IF NRST = '0' THEN
    count := 0;
    shift_en <= '0';
  ELSIF rising_edge(CLK) THEN
    IF count = SPI_CLK_DIV - 1 THEN
      count := 0;
      shift_en <= '1';
    ELSE
      count := count + 1;
      shift_en <= '0';
    END IF;
  END IF;
END PROCESS;

PROCESS (CLK, NRST)
BEGIN
  IF NRST = '0' THEN
    cs_n <= '1';
    mosi <= '0';
    sclk <= '0';
    adc_data <= (OTHERS => '0');
    channel <= (OTHERS => '0');
    bit_index <= 0;
    wait_cnt <= 0;
    state <= ST_IDLE;
    uart_send_trigger <= '0';
  END IF;
END PROCESS;

```

```

ELSIF rising_edge(CLK) THEN

CASE state IS
    WHEN ST_IDLE =>
        bit_index <= 0;
        channel <= "001"; -- Select channel ADC_IN1
        cs_n <= '1';
        sclk <= '1';
        state <= ST_START;
        uart_send_trigger <= '0';

    WHEN ST_START =>
        shift_reg <= (OTHERS => '0');
        shift_reg(13 DOWNTO 11) <= channel; -- for ADC128S022
        cs_n <= '0';
        state <= ST_SCK_L;
        uart_send_trigger <= '0';

    WHEN ST_SCK_L =>
        IF shift_en = '1' THEN
            sclk <= '0';
            mosi <= shift_reg(shift_reg'left);
            state <= ST_SCK_H;
        END IF;
        uart_send_trigger <= '0';

    WHEN ST_SCK_H =>
        IF shift_en = '1' THEN
            sclk <= '1';
            shift_reg <= shift_reg(shift_reg'left - 1 DOWNTO 0) & adc_miso;
            IF bit_index = DATA_WIDTH - 1 THEN
                cs_n <= '1';
                wait_cnt <= WAIT_CNT_MAX;
                state <= ST_WAIT;
            ELSE
                bit_index <= bit_index + 1;
                state <= ST_SCK_L;
            END IF;
        END IF;
        uart_send_trigger <= '0';

```

```
WHEN ST_WAIT =>
    adc_data <= shift_reg(11 DOWNTO 0);

    IF wait_cnt = 0 THEN
        state <= ST_IDLE;
        uart_send_trigger <= '1';
    ELSE
        wait_cnt <= wait_cnt - 1;
        uart_send_trigger <= '0';
    END IF;

    WHEN OTHERS =>
        state <= ST_IDLE;
        uart_send_trigger <= '0';
    END CASE;
END IF;
END PROCESS;

END behavioral;
```

Code mainVHDL/UART_TX.vhd

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY UART_TX IS
    PORT (
        clk : IN STD_LOGIC; -- Assuming 50MHz input clock
        reset_n : IN STD_LOGIC;
        uart_send_trigger : IN STD_LOGIC;
        adcData : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        tx : OUT STD_LOGIC
    );
END ENTITY;

ARCHITECTURE rtl OF UART_TX IS
    -- UART constants (for 2M baud with 50MHz clock)
    CONSTANT CLKS_PER_BIT : INTEGER := 25;
    -- CONSTANT CLKS_PER_BIT : INTEGER := 200; -- 250_000

    -- UART signals
    TYPE uart_state_type IS (IDLE, START_BIT, DATA_BITS, STOP_BIT);
    SIGNAL uart_state : uart_state_type := IDLE;
    SIGNAL clk_count : INTEGER RANGE 0 TO CLKS_PER_BIT - 1 := 0;
    SIGNAL bit_index : INTEGER RANGE 0 TO 7 := 0;
    SIGNAL tx_data : STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0');
    -- SIGNAL uart_send_trigger : STD_LOGIC := '0';
    SIGNAL byte_select : INTEGER RANGE 0 TO 1 := 0;

BEGIN
    -- UART transmission process (remains largely the same)
    PROCESS (clk)
    BEGIN
        IF rising_edge(clk) THEN
            CASE uart_state IS
                WHEN IDLE =>
                    tx <= '1';
                IF uart_send_trigger = '1' THEN
                    -- IF byte_select = 0 THEN
                    --     tx_data <= adcData;
                    -- ELSE
                    --     tx_data <= "00000000";
                    -- END IF;
                    -- tx_data <= adcData;
                    uart_state <= START_BIT;
                    clk_count <= 0;
            END IF;
        END IF;
    END PROCESS;

```

```

WHEN START_BIT =>
    tx <= '0';
    IF clk_count = CLKS_PER_BIT - 1 THEN
        clk_count <= 0;

        uart_state <= DATA_BITS;
        bit_index <= 0;
    ELSE
        clk_count <= clk_count + 1;
    END IF;

WHEN DATA_BITS =>
    tx <= tx_data(bit_index);
    IF clk_count = CLKS_PER_BIT - 1 THEN
        clk_count <= 0;
        IF bit_index = 7 THEN
            uart_state <= STOP_BIT;
        ELSE
            bit_index <= bit_index + 1;
        END IF;
    ELSE
        clk_count <= clk_count + 1;
    END IF;

WHEN STOP_BIT =>
    tx <= '1';
    IF clk_count = CLKS_PER_BIT - 1 THEN
        clk_count <= 0;
        uart_state <= IDLE;
        -- IF byte_select = 0 THEN
        --     byte_select <= 1;
        -- ELSE
        --     byte_select <= 0;
        -- END IF;
    ELSE
        clk_count <= clk_count + 1;
    END IF;
END CASE;
END IF;
END PROCESS;
END ARCHITECTURE;

```

Code mainVHDL/mainVHDL.vhd

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY mainVHDL IS
  PORT (
    CLK : IN STD_LOGIC; -- 50 MHz system clock
    NRST : IN STD_LOGIC; -- Active-low asynchronous reset
    ADC_CSN : OUT STD_LOGIC; -- ADC SPI chip-select
    ADC_SCLK : OUT STD_LOGIC; -- ADC SPI SCLK
    ADC_MOSI : OUT STD_LOGIC; -- ADC SPI MOSI
    ADC_MISO : IN STD_LOGIC; -- ADC SPI MISO
    LEDS : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); -- 8-bit ADC OUTPUT
    tx : OUT STD_LOGIC -- UART
  );
END mainVHDL;

ARCHITECTURE behavioral OF mainVHDL IS
  SIGNAL adc_output : STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0');
  SIGNAL uart_send_trigger : STD_LOGIC := '0';
BEGIN

  ADC_Controller : entity work.ADC_Controller
    port map(
      CLK => CLK,
      NRST => NRST,
      ADC_CSN => ADC_CSN,
      ADC_SCLK => ADC_SCLK,
      ADC_MOSI => ADC_MOSI,
      ADC_MISO => ADC_MISO,
      uart_send_trigger => uart_send_trigger,
      LEDS => adc_output
    );

  ADC_Controller : entity work.ADC_Controller
    port map(
      CLK => CLK,
      NRST => NRST,
      ADC_CSN => ADC_CSN,
      ADC_SCLK => ADC_SCLK,
      ADC_MOSI => ADC_MOSI,
      ADC_MISO => ADC_MISO,
      uart_send_trigger => uart_send_trigger,
      LEDS => adc_output
    );

```

```

UART_TX : entity work.UART_TX
port map(
    clk => CLK,
    reset_n => NRST,
    adcData => adc_output,
    uart_send_trigger => uart_send_trigger,
    tx => tx
);
LEDS <= adc_output;

-- PROCESS (CLK, NRST)
-- END PROCESS;

END behavioral;

```

Code mainVHDL/mainVHDL.qsf

```

# -----
#
# Copyright (C) 2020 Intel Corporation. All rights reserved.
# Your use of Intel Corporation's design tools, logic functions
# and other software and tools, and any partner logic
# functions, and any output files from any of the foregoing
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Intel Program License
# Subscription Agreement, the Intel Quartus Prime License Agreement,
# the Intel FPGA IP License Agreement, or other applicable license
# agreement, including, without limitation, that your use is for
# the sole purpose of programming logic devices manufactured by
# Intel and sold by Intel or its authorized distributors. Please
# refer to the applicable agreement for further details, at
# https://fpgasoftware.intel.com/eula.
#
# -----
#
# Quartus Prime
# Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition
# Date created = 16:42:51 November 01, 2024
#
# -----
#
# Notes:
#
# 1) The default values for assignments are stored in the file:
#     mainVHDL_assignment_defaults.qdf
#     If this file doesn't exist, see file:
#     assignment_defaults.qdf
#
# 2) Altera recommends that you do not modify this file. This
#     file is updated automatically by the Quartus Prime software
#     and any changes you make may be lost or overwritten.
#
# -----

```

```

set_global_assignment -name FAMILY "Cyclone IV E"
set_global_assignment -name DEVICE EP4CE22F17C6
set_global_assignment -name TOP_LEVEL_ENTITY mainVHDL
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 20.1.1
set_global_assignment -name PROJECT_CREATION_TIME_DATE "16:42:51 NOVEMBER 01, 2024"
set_global_assignment -name LAST_QUARTUS_VERSION "20.1.1 Lite Edition"
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
set_global_assignment -name NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V
set_global_assignment -name VHDL_FILE mainVHDL.vhd
set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -
section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top

set_location_assignment PIN_J15 -to NRST
set_location_assignment PIN_R8 -to CLK
set_location_assignment PIN_A10 -to ADC_CS_N
set_location_assignment PIN_A9 -to ADC_MISO
set_location_assignment PIN_B10 -to ADC_MOSI
set_location_assignment PIN_B14 -to ADC_SCLK
set_location_assignment PIN_A15 -to LEDS[0]
set_location_assignment PIN_A13 -to LEDS[1]
set_location_assignment PIN_B13 -to LEDS[2]
set_location_assignment PIN_A11 -to LEDS[3]
set_location_assignment PIN_D1 -to LEDS[4]
set_location_assignment PIN_F3 -to LEDS[5]
set_location_assignment PIN_B1 -to LEDS[6]
set_location_assignment PIN_L3 -to LEDS[7]
set_location_assignment PIN_F9 -to tx

set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to NRST
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to CLK
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ADC_CS_N
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ADC_MISO
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ADC_MOSI
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to ADC_SCLK
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[0]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[1]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[2]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[3]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[4]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[5]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[6]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to LEDS[7]
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to tx
set_global_assignment -name VHDL_FILE ADC_Controller.vhd
set_global_assignment -name VHDL_FILE UART_TX.vhd
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top

```

Code Game

Code pygame/src/config.py

```
imagesPath = r'./asset/images/'  
fontsPath = r'./asset/fonts/'  
soundsPath = r'./asset/sounds/'
```

Code pygame/src/init.py

```
import pygame  
from config import *  
  
SCREEN_W = 1280  
SCREEN_H = 720  
# SCREEN_W = 1920  
# SCREEN_H = 1080  
BLACK = (0, 0, 0)  
WHITE = (255, 255, 255)  
BLUESKY = (200, 220, 255)  
BLUE = (0, 0, 255)  
# DARK_ORCHID = (252,130,191)  
PINK = (252,130,191)  
DEEP_SKY_BLUE = (0,191,255)  
VIOLET = (30,144,255)  
FPS = 60  
  
# ----- Images and Sounds -----  
bg = pygame.transform.scale(pygame.image.load(fr'{imagesPath}Background.jpg'),  
(SCREEN_W, SCREEN_H))  
bg.blur = pygame.transform.scale(pygame.image.load(fr'{imagesPath}BGblurr.jpg'),  
(SCREEN_W, SCREEN_H))  
  
# -----  
  
pygame.init()  
pygame.display.set_caption('Introduction to Signals and Systems MiniProject')  
screen = pygame.display.set_mode((SCREEN_W, SCREEN_H))  
screen_rect = screen.get_rect()  
  
clock = pygame.time.Clock()  
  
# -----
```

Code pygame/src/IntroScreen.py

```
import sys
import pygame
from pygame.locals import *
from config import *
from init import *
from JellyfishSprite import *

# ----- Text & Intro Screen -----
jellyfish_man_intro = Jellyfish(
    jellyfish_img=jellyfish_man_img,
    jellyfish_num_sub_imgs=3,
    x_position=int(SCREEN_W // 2 - SCREEN_W * 0.3),
    y_position=int(SCREEN_H // 2)
)
jellyfish_girl_cry_intro = Jellyfish(
    jellyfish_img=jellyfish_girl_cry_img,
    jellyfish_num_sub_imgs=3,
    x_position=int(SCREEN_W // 2 + SCREEN_W * 0.3),
    y_position=int(SCREEN_H // 2)
)

game_description = {
    'game_description': [
        "Game Description:",
        "In Love Love Jellyfish, players step into the translucent world",
        "of a determined jellyfish on a mission to find and rescue",
        "his beloved. The jellyfish's boyfriend is trapped in the",
        "deep sea, and our hero must navigate through an underwater",
        "maze of obstacles to reach her. The ocean is filled with",
        "floating debris, dangerous sea creatures, and mysterious",
        "objects that, if hit, will cause him to lose his energy",
        "and restart his journey."
    ],
    'gameplay_mechanics': [
        "Gameplay Mechanics:",
        "Frequency Control: Players use their own voice or",
        "a frequency controller to adjust the jellyfish's movement",
        "and frequency, avoiding obstacles. Humming or adjusting",
        "pitch lets the jellyfish dodge, dive, or rise as needed."
    ]
}
```

```

# Define ratios for font sizes and margins
TITLE_FONT_SIZE_RATIO = 128 / 1080 # Original title size was 128
HEADER_FONT_SIZE_RATIO = 46 / 1080 # Original header size was 50
TEXT_FONT_SIZE_RATIO = 40 / 1080    # Original text size was 42
MARGIN_RATIO = 5 / 1080           # Original margin was 40

def draw_text(text, size, color, x, y, fontFile=None):
    if fontFile == None:

        font = pygame.font.SysFont(None, size)
    else:

        font = pygame.font.Font(fontFile, size)

    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect(midtop=(x, y))
    screen.blit(text_surface, text_rect)

def render_text_responsive(screen_width, screen_height, text_data):
    cx = screen_rect.centerx

    # Calculate responsive title font size
    title_font_size = int(screen_height * TITLE_FONT_SIZE_RATIO)

    # Calculate y position for title
    # Position for title at the top
    title_y_position = int(screen_height * 0.07)

    # Render the title
    draw_text('Love Love Jellyfish', title_font_size, PINK,
              cx, title_y_position, fr'{fontsPath}SmothBubble.com.otf')

    # Calculate responsive sizes
    header_font_size = int(screen_height * HEADER_FONT_SIZE_RATIO)
    text_font_size = int(screen_height * TEXT_FONT_SIZE_RATIO)
    margin = int(screen_height * MARGIN_RATIO)

    # Calculate initial y position for game description
    # y_position = int(screen_height * 0.23) # 250px as a ratio of 1080
    y_position = int(screen_height * 0.2) # 250px as a ratio of 1080

    # Render game description
    for line in text_data['game_description']:
        draw_text(line, text_font_size if line != "Game Description:" else
                  header_font_size,
                  DEEP_SKY_BLUE if line != "Game Description:" else VIOLET, cx,
                  y_position, fr'{fontsPath}Stanberry.ttf')
        y_position += text_font_size + margin

```

```

# Add some space before the next section
y_position += margin * 2

# Render gameplay mechanics
for line in text_data['gameplay_mechanics']:
    draw_text(line, text_font_size if line != "Gameplay Mechanics:" else
header_font_size,
              DEEP_SKY_BLUE if line != "Gameplay Mechanics:" else VIOLET, cx,
y_position, fr'{fontsPath}Stanberry.ttf')
    y_position += text_font_size + margin

def intro_screen():

    waiting = True
    while waiting:
        screen.fill(BLUE)
        screen.blit(bg, bg.get_rect())
        cx = screen_rect.centerx

        render_text_responsive(SCREEN_W, SCREEN_H, game_description)

        btn_exit_image =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}btn-exit.png'),(SCREEN_W // 15, SCREEN_H // 15))
        btn_exit_rect = btn_exit_image.get_rect(
            left=cx+30, top=int(SCREEN_H * 0.85))

        btn_start_image =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}btn-start.png'),(SCREEN_W // 15, SCREEN_H // 15))
        btn_start_rect = btn_start_image.get_rect(
            right=cx-30, top=int(SCREEN_H * 0.85))

        jellyfish_man_intro.update()
        jellyfish_girl_cry_intro.update()

        screen.blit(jellyfish_man_intro.image, jellyfish_man_intro.rect)
        screen.blit(jellyfish_girl_cry_intro.image,
                    jellyfish_girl_cry_intro.rect)

        screen.blit(btn_start_image, btn_start_rect)
        screen.blit(btn_exit_image, btn_exit_rect)

```

```

pygame.display.flip()
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        sys.exit()
    elif event.type == MOUSEBUTTONDOWN: # type: ignore
        if btn_start_rect.collidepoint(pygame.mouse.get_pos()):
            waiting = False
    elif btn_exit_rect.collidepoint(pygame.mouse.get_pos()):
        pygame.quit()
        sys.exit()

```

Code pygame/src/JellyfishSprite.py

```

import pygame
from pygame.locals import *
from config import *
import random
from init import *

# ----- Jellyfish Sprite -----
jellyfish_man_img =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}JellyfishM.png'),
(SCREEN_W//3.5, SCREEN_H//3.5))
jellyfish_girl_love_img =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}JellyfishGL.png'),
(SCREEN_W//3.5, SCREEN_H//3.5))
jellyfish_girl_cry_img =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}JellyfishGC.png'),
(SCREEN_W//3.5, SCREEN_H//3.5))

class Jellyfish(pygame.sprite.Sprite):
    def __init__(self, jellyfish_img, jellyfish_num_sub_imgs, x_position,
y_position):
        super(Jellyfish, self).__init__()

        # jellyfish_num_sub_imgs = 3
        jellyfish_sub_img_w = jellyfish_img.get_width() // jellyfish_num_sub_imgs
        jellyfish_sub_img_h = jellyfish_img.get_height()
        self.jellyfish_sub_imgs = []

        for i in range(jellyfish_num_sub_imgs):
            x = i * jellyfish_sub_img_w
            f = jellyfish_img.subsurface(x, 0, jellyfish_sub_img_w,
jellyfish_sub_img_h)
            self.jellyfish_sub_imgs.append(f)

        self.jellyfish_repeat = FPS // jellyfish_num_sub_imgs
        self.jellyfish_last_frame = (jellyfish_num_sub_imgs * self.jellyfish_repeat)

```

```

        self.image = self.jellyfish_sub_imgs[0]
        self.rect = self.image.get_rect(center=(x_position, y_position))
        self.index = 0
        self.speedx = 5
        self.distance = 10

    def update_animation(self):
        if self.index >= self.jellyfish_last_frame:
            self.index = 0

        i = self.index // self.jellyfish_repeat
        self.image = self.jellyfish_sub_imgs[i]
        self.index += 1

    def update_movement(self, keys = None, dominant_freq = None):

        if keys is None and dominant_freq is None: return

        if dominant_freq > 500: # type: ignore
            self.rect.move_ip(0, -self.distance)
            if self.rect.top <= 0:
                self.rect.top = 0
        elif keys[K_UP]: # type: ignore
            self.rect.move_ip(0, -self.distance)
            if self.rect.top <= 0:
                self.rect.top = 0
        else:
            self.rect.move_ip(0, self.distance)
            if self.rect.bottom >= SCREEN_H:
                self.rect.bottom = SCREEN_H

    def update_auto_movement(self):
        self.rect.move_ip(-self.distance, 0)
        if self.rect.right < 0:
            self.kill()

    def update(self, keys = None, dominant_freq = None, is_auto = None):
        self.update_animation()
        self.update_movement(keys, dominant_freq)
        if not is_auto: return
        self.update_auto_movement()

```

Code pygame/src/main.py

```
import sys
import pygame
import random
from pygame.locals import *
from config import *
from init import *
from datetime import datetime
import threading

from ObjectSprite import *
from JellyfishSprite import *
from IntroScreen import *

from readfHzaudio import *
from OuttroScreen import *
import time
from readSerialfreq import *
# -----
# Variable to control thread activity
# run_threads = True
run_threads = False
dominant_freq = 0
# Thread function
def get_dominant_freq():
    global dominant_freq
    while run_threads:
        dominant_freq_temp = read_uart_data() / 2
        # dominant_freq_temp = read_uart_data()
        if dominant_freq_temp == 0:
            continue
        dominant_freq = dominant_freq_temp
        print(f" dominant_freq:",dominant_freq)

# Start the thread
thread = threading.Thread(target=get_dominant_freq)
thread.start()

playing = False
running = True
```

```

try:
    while running:
        if not playing:
            intro_screen()
            playing = True
            # ----- init value ----- #
            img_top = random.choice(objects_top)
            img_button = random.choice(objects_bottom)

            group_jellyfish_man = pygame.sprite.Group()
            group_jellyfish_man.add(
                Jellyfish(
                    jellyfish_img=jellyfish_man_img,
                    jellyfish_num_sub_imgs=3,
                    x_position=int(SCREEN_W * 0.07),
                    y_position=int(SCREEN_H * 0.9)
                )
            )

            group_jellyfish_girl = pygame.sprite.Group()

            count_obj = 0
            OFFSET_NUMBER_OF_OBJ = 2
            is_empty_obj = False
            do_empty_obj = False
            is_win = False
            is_lose = False

            group_object_top = pygame.sprite.Group()
            group_object_top.add(ObjectTop(img_top))

            group_object_bottom = pygame.sprite.Group()
            group_object_bottom.add(ObjectBottom(img_button))

            ADD_object_TOP = pygame.USEREVENT + 1
            pygame.time.set_timer(ADD_object_TOP, 7000)
            ADD_object_BOTTOM = pygame.USEREVENT + 2
            pygame.time.set_timer(ADD_object_BOTTOM, 7000)

        for event in pygame.event.get():
            if event.type == QUIT: # type: ignore
                running = False
                pygame.quit()
                sys.exit()

```

```

    if event.type == ADD_object_TOP and count_obj <= OFFSET_NUMBER_OF_OBJ:
        count_obj += 1
        img_top = random.choice(
            objects_top) if img_button != object_bottom3 else object_top1
        if len(group_object_top) < 3:
            group_object_top.add(ObjectTop(img_top))
    if event.type == ADD_object_BOTTOM and count_obj <=
OFFSET_NUMBER_OF_OBJ:
        count_obj += 1
        img_button = random.choice(
            objects_bottom) if img_top != object_top3 else object_bottom1
        if len(group_object_bottom) < 3:
            group_object_bottom.add(ObjectBottom(img_button))
    if count_obj >= OFFSET_NUMBER_OF_OBJ and len(group_object_top) == 0 and
len(group_object_bottom) == 0:
        is_empty_obj = True
        pass
    if not run_threads:
        dominant_freq = readfHz()
        screen.fill(BLUESKY)
        screen.blit(bg_blur, bg_blur.get_rect())

        keys = pygame.key.get_pressed()
        # group_jellyfish_man.update()
        group_jellyfish_man.update(keys, dominant_freq)
        group_jellyfish_girl.update(is_auto = True)

        group_object_top.update()
        group_object_bottom.update()

        group_object_top.draw(screen)
        group_object_bottom.draw(screen)
        group_jellyfish_man.draw(screen)
        group_jellyfish_girl.draw(screen)

        hits_top = pygame.sprite.groupcollide(
            group_jellyfish_man, group_object_top, False, False,
pygame.sprite.collide_mask
        )
        hits_bottom = pygame.sprite.groupcollide(
            group_jellyfish_man, group_object_bottom, False, False,
pygame.sprite.collide_mask
        )
        if len(hits_top) > 0 or len(hits_bottom) > 0:
            if (len(hits_top) > 0):
                first_hit = list(hits_top.values())[0][0]
            elif (len(hits_bottom) > 0):
                first_hit = list(hits_bottom.values())[0][0]
            center = first_hit.rect.center
            is_lose = True

```

```

        if is_empty_obj and not do_empty_obj:
            do_empty_obj = True
            group_jellyfish_girl.add(
                Jellyfish(
                    jellyfish_img=jellyfish_girl_cry_img,
                    jellyfish_num_sub_imgs=3,
                    x_position=SCREEN_W + random.randint(0, 500),
                    y_position=int(random.randint(SCREEN_H * 0.2, SCREEN_H * 0.8))
                )
            )

        hits_jellyfish_girl_cry = pygame.sprite.groupcollide(
            group_jellyfish_man, group_jellyfish_girl, True, True,
            pygame.sprite.collide_mask
        )
        if len(hits_jellyfish_girl_cry) > 0:
            if (len(hits_jellyfish_girl_cry) > 0):
                first_hit = list(hits_jellyfish_girl_cry.values())[0][0]
                center = first_hit.rect.center
                is_win = True
            if is_win:
                outro_screen(text="You Win", jellyfish_girl_outtro=jellyfish_girl_love_outtro)
                playing = False

            elif (not is_win and do_empty_obj and len(group_jellyfish_girl) == 0) or is_lose:
                outro_screen(text="You Lose", jellyfish_girl_outtro=jellyfish_girl_cry_outtro)
                playing = False

            pygame.display.flip()
            clock.tick(FPS)

    except KeyboardInterrupt:
        # Stop and close the stream
        stream.stop_stream()
        stream.close()
        p.terminate()
        print("\nStopped recording.")

    except Exception as e:
        # Handle any unexpected errors
        print(f"An error occurred: {e}")

    finally:
        # Stop the background task
        run_threads = False
        thread.join()
        # Ensure the stream is closed upon exit
        stream.stop_stream()
        stream.close()
        p.terminate()

```

Code pygame/src/ObjectSprite.py

```
import pygame
from config import *
import random
from init import *

# ----- Object Sprite -----
object_bottom1 =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}Objects/ObjectBottom1.png'),
(SCREEN_H//2.5, SCREEN_W//2.5))
object_bottom2 =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}Objects/ObjectBottom2.png'),
(SCREEN_H//2.5, SCREEN_W//2.5))
object_bottom3 =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}Objects/ObjectBottom3.png'),
(SCREEN_H//2.5, SCREEN_W//2.5))

object_top1 =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}Objects/ObjectTop1.png'),
(SCREEN_H//2.5, SCREEN_W//2.5))
object_top2 =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}Objects/ObjectTop2.png'),
(SCREEN_H//2.5, SCREEN_W//2.5))
object_top3 =
pygame.transform.scale(pygame.image.load(fr'{imagesPath}Objects/ObjectTop3.png'),
(SCREEN_H//2.5, SCREEN_W//2.5))

objects_bottom = [object_bottom1, object_bottom2, object_bottom3]
objects_top = [object_top1, object_top2, object_top3]

class ObjectBottom(pygame.sprite.Sprite):
    def __init__(self, image = None):
        super(ObjectBottom, self).__init__()
        self.image = random.choice(objects_bottom) if image == None else image
        # self.image = objectImage
        h = self.image.get_height()
        # start_left = SCREEN_W + random.randint(0, 500)
        start_left = SCREEN_W + random.choice([random.randint(0, 200),
random.randint(700, 900)])
        # start_left = SCREEN_W + 700
        start_top = SCREEN_H - h
        self.rect = self.image.get_rect(topleft=(start_left, start_top))
        self.speed = 5

    def update(self):
        self.rect.move_ip(-self.speed, 0)
        if self.rect.right < 0:
            self.kill()

class ObjectTop(pygame.sprite.Sprite):
    def __init__(self, image = None):
        super(ObjectTop, self).__init__()
```

```

        self.image = random.choice(objects_top) if image == None else image
        h = self.image.get_height()
        # start_left = SCREEN_W + random.randint(0, 500)
        start_left = SCREEN_W + random.randint(400, 500)
        # start_left = SCREEN_W + 100
        start_top = 0
        self.rect = self.image.get_rect(topleft=(start_left, start_top))
        self.speed = 5

def update(self):
    self.rect.move_ip(-self.speed, 0)
    if self.rect.right < 0:
        self.kill()

```

Code pygame/src/OuttroScreen.py

```

import sys
import pygame
from pygame.locals import *
from config import *
from init import *
from JellyfishSprite import *
# ----- Text & Intro Screen -----
jellyfish_man_outtro = Jellyfish(
    jellyfish_img=jellyfish_man_img,
    jellyfish_num_sub_imgs=3,
    x_position=int(SCREEN_W // 2 - SCREEN_W * 0.05),
    y_position=int(SCREEN_H // 2)
)
jellyfish_girl_love_outtro = Jellyfish(
    jellyfish_img=jellyfish_girl_love_img,
    jellyfish_num_sub_imgs=3,
    x_position=int(SCREEN_W // 2 + SCREEN_W * 0.05),
    y_position=int(SCREEN_H // 2)
)
jellyfish_girl_cry_outtro = Jellyfish(
    jellyfish_img=jellyfish_girl_cry_img,
    jellyfish_num_sub_imgs=3,
    x_position=int(SCREEN_W // 2 + SCREEN_W * 0.05),
    y_position=int(SCREEN_H // 2)
)

def draw_text(text, size, color, x, y, fontFile=None):
    if fontFile == None:
        font = pygame.font.SysFont(None, size)
    else:
        font = pygame.font.Font(fontFile, size)

    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect(midtop=(x, y))
    screen.blit(text_surface, text_rect)

```

```

def outtro_screen(text, jellyfish_girl_outtro):

    waiting = True
    while waiting:
        screen.fill(BLUE)
        screen.blit(bg, bg.get_rect())
        cx = screen_rect.centerx

        btn_exit_image = pygame.transform.scale(pygame.image.load(fr'{imagesPath}btn-
exit.png'),(SCREEN_W // 15, SCREEN_H // 15))
        btn_exit_rect = btn_exit_image.get_rect(
            left=cx+30, top=int(SCREEN_H * 0.85))

        btn_start_image =
        pygame.transform.scale(pygame.image.load(fr'{imagesPath}btn-start.png'),(SCREEN_W //
15, SCREEN_H // 15))
        btn_start_rect = btn_start_image.get_rect(
            right=cx-30, top=int(SCREEN_H * 0.85))
        draw_text(text, 64, BLACK,
                  screen_rect.centerx, screen_rect.centery - 200,
fr'{fontsPath}SmothyBubble.com.otf')

        jellyfish_man_outtro.update()
        jellyfish_girl_outtro.update()

        screen.blit(jellyfish_man_outtro.image, jellyfish_man_outtro.rect)
        screen.blit(jellyfish_girl_outtro.image,
                    jellyfish_girl_outtro.rect)

        screen.blit(btn_start_image, btn_start_rect)
        screen.blit(btn_exit_image, btn_exit_rect)

        pygame.display.flip()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == MOUSEBUTTONDOWN: # type: ignore
                if btn_start_rect.collidepoint(pygame.mouse.get_pos()):
                    waiting = False
            elif btn_exit_rect.collidepoint(pygame.mouse.get_pos()):
                pygame.quit()
                sys.exit()

```

Code pygame/src/readfHzaudio.py

```

dominant_freq = abs(freqs[np.argmax(magnitudes[:CHUNK // 2])])

# Print the dominant frequency in real-time
print(f"Real-time Dominant Frequency: {dominant_freq:.2f} Hz")

```

Code pygame/src/readSerialfreq.py

```
import serial
import numpy as np
import time
from collections import deque
import pyautogui

# Serial configuration
SERIAL_PORT = '/dev/tty.usbserial-0001'      # Adjust as needed
BAUD_RATE = 2_000_000      # Set to match each 8-bit transmission's baud rate
# BAUD_RATE = 500000      # Set to match each 8-bit transmission's baud rate
TIMEOUT = 1      # Timeout for serial read in seconds

# FFT configuration
SAMPLE_SIZE = 1024      # Number of samples for FFT (adjustable)
# SAMPLING_RATE = 250000      # Effective data rate for complete 12-bit samples (Hz)
SAMPLING_RATE = 2000      # Effective data rate for complete 12-bit samples (Hz)
# SAMPLING_RATE = 40000      # Effective data rate for complete 12-bit samples (Hz)
start_time = time.time()
# Buffer for storing 12-bit ADC samples
sample_buffer = deque(maxlen=SAMPLE_SIZE)

try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=TIMEOUT)
except:
    print("Error Serial")
def read_uart_data():
    global start_time
    # Open the serial connection
    try:
        # while True:
        if ser.in_waiting >= 1:
            msb = ser.read(1)
            msb = int.from_bytes(msb, byteorder='big')
            sample_buffer.append(msb)
            if len(sample_buffer) == SAMPLE_SIZE:
                start_time = time.time()
                temp = perform_fft_analysis(list(sample_buffer))
                end_time = time.time()
                elapsed_time_ms = (end_time - start_time) * 1000
                # print(f"Runtime: {elapsed_time_ms:.3f} milliseconds")
                return temp
    except KeyboardInterrupt:
        ser.close()
    return 0
```

```

def perform_fft_analysis(samples):
    """
    Perform FFT analysis on the provided samples and print the dominant frequency.
    """

    # Convert samples to numpy array and apply FFT
    data = np.array(samples) - np.mean(samples)
    fft_result = np.fft.fft(data)

    freqs = np.fft.fftfreq(len(fft_result), 1 / SAMPLING_RATE)

    # Use only the positive half of the FFT result (real frequencies)
    positive_freqs = freqs[:len(freqs) // 2]
    positive_magnitudes = np.abs(fft_result[:len(fft_result) // 4])

    # Identify the frequency with the maximum magnitude
    dominant_freq = positive_freqs[np.argmax(positive_magnitudes)]
    # index = np.argmax(positive_magnitudes)

    # dominant_freq /= 100
    # print(f"Dominant Frequency: {dominant_freq :.2f} Hz")
    return dominant_freq
    # return index

if __name__ == '__main__':
    try:
        print(f"Starting to read data from {SERIAL_PORT} at {BAUD_RATE} baud...")
        while True:
            data = read_uart_data() /2
            # data = read_uart_data()
            if data == 0: continue
            # if data > 10_000:
            #     continue
            # if data > 2_000:
            #     continue
            print(data)
            if data >500:
                # pyautogui.press('up')
                pass
    except KeyboardInterrupt:
        print("\nProgram interrupted by user.")
    except serial.SerialException as e:
        print(f"Serial error: {e}")
    except Exception as e:
        print(f"Error: {e}")

```

บรรณานุกรม

บัญชา ประสีลະเตสส์ง. (2565). เขียนโปรแกรมด้วยภาษา Python ฉบับเพิ่มเติมสำหรับ PyQt และ

Pygame. สีบคัน 1 พฤศจิกายน 2567, จาก <https://www.developerthai.com>

อาจารย์ เรวัต ศิริโภคภิรมย์. (2021). ตัวอย่างการเขียนโค้ด VHDL สำหรับ ADC128S022. สีบคัน

1 พฤศจิกายน 2567, จาก https://iot-kmutnb.github.io/blogs/fpga/fpga_adc_spi/

Geeksforgeeks. (2024). Multithreading in Python. สีบคัน 1 พฤศจิกายน 2567, จาก

<https://www.geeksforgeeks.org/multithreading-python-set-1/>

Pygame news. (2014). Document built in function pygame. สีบคัน 1 พฤศจิกายน 2567, จาก

<https://www.pygame.org/news>

Russell Merrick. (2014). UART, Serial Port, RS-232 Interface. สีบคัน 1 พฤศจิกายน 2567,

จาก <https://nandland.com/uart-serial-port-module/>

Chat gpt

พีรภาส นำนุช