



รายงานโครงการ

Embedded Piano

จัดทำโดย

[กลุ่ม Ohno]

6510110086	นาย ชนะ แซ่เอียนบ	SEC-01
6510110192	นาย ธนากร ปานมาส	SEC-02
6510110212	นาย นกฤช กธีไซชนะ	SEC-02

รายวิชา 240-319 Embedded System Developer Module

ภาคการเรียนที่ 2 ปีการศึกษา 2567

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่

สารบัญ

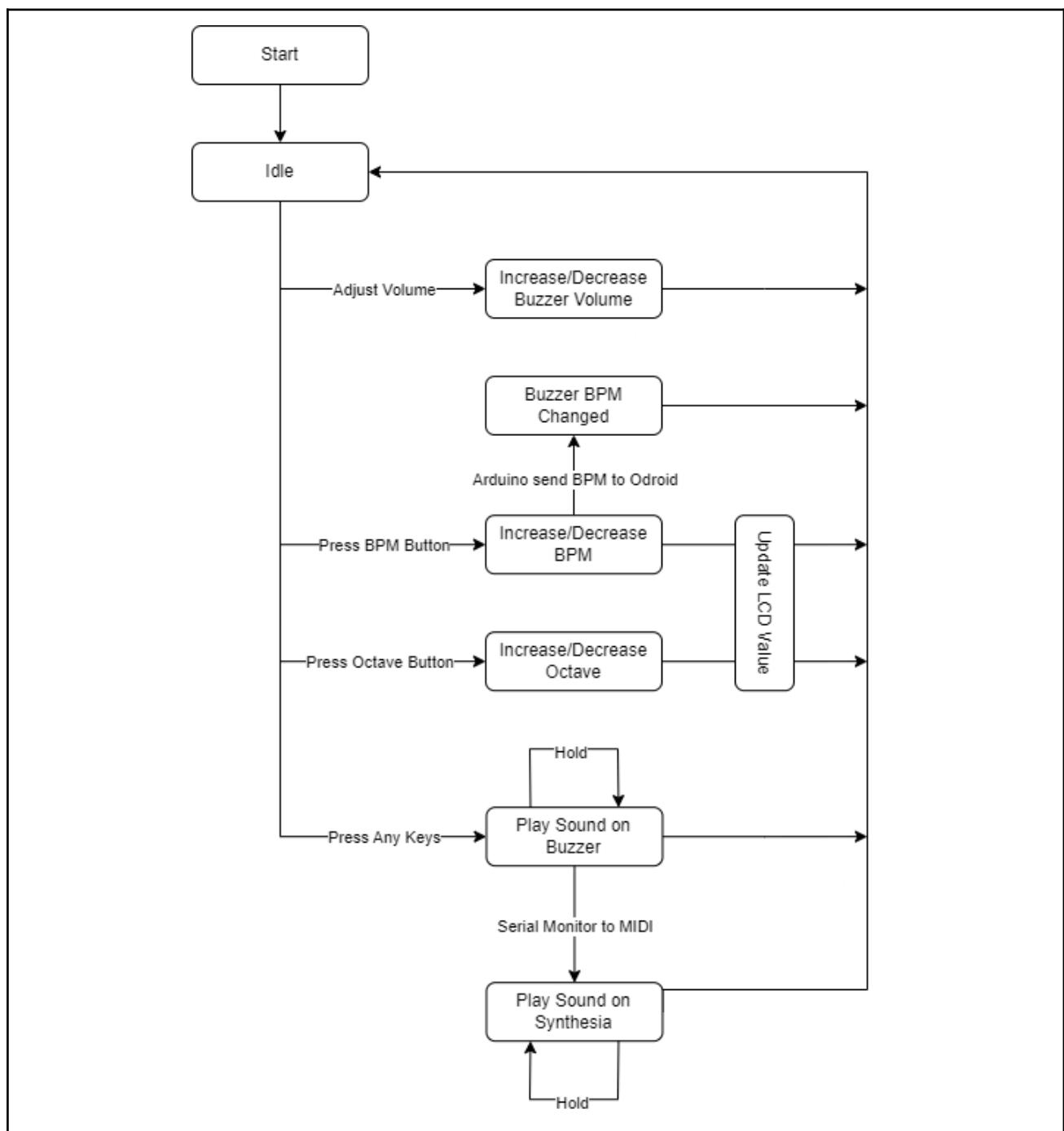
วัตถุประสงค์	2
State Diagram	2
Flowchart Arduino	3
Flowchart Odroid	4
Schematic Design	5
อุปกรณ์ที่ใช้ในการสร้าง	6
Arduino Code	7-16
Odroid Code	17-19
ขั้นตอนการ Setup กับ Synthesia	20-21
ประสีกิจภาพในระบบ	22
การพัฒนาและต่อยอดโปรเจ็ค	22
สรุปผล	22
ภาคผนวก	23-31

วัตถุประสงค์

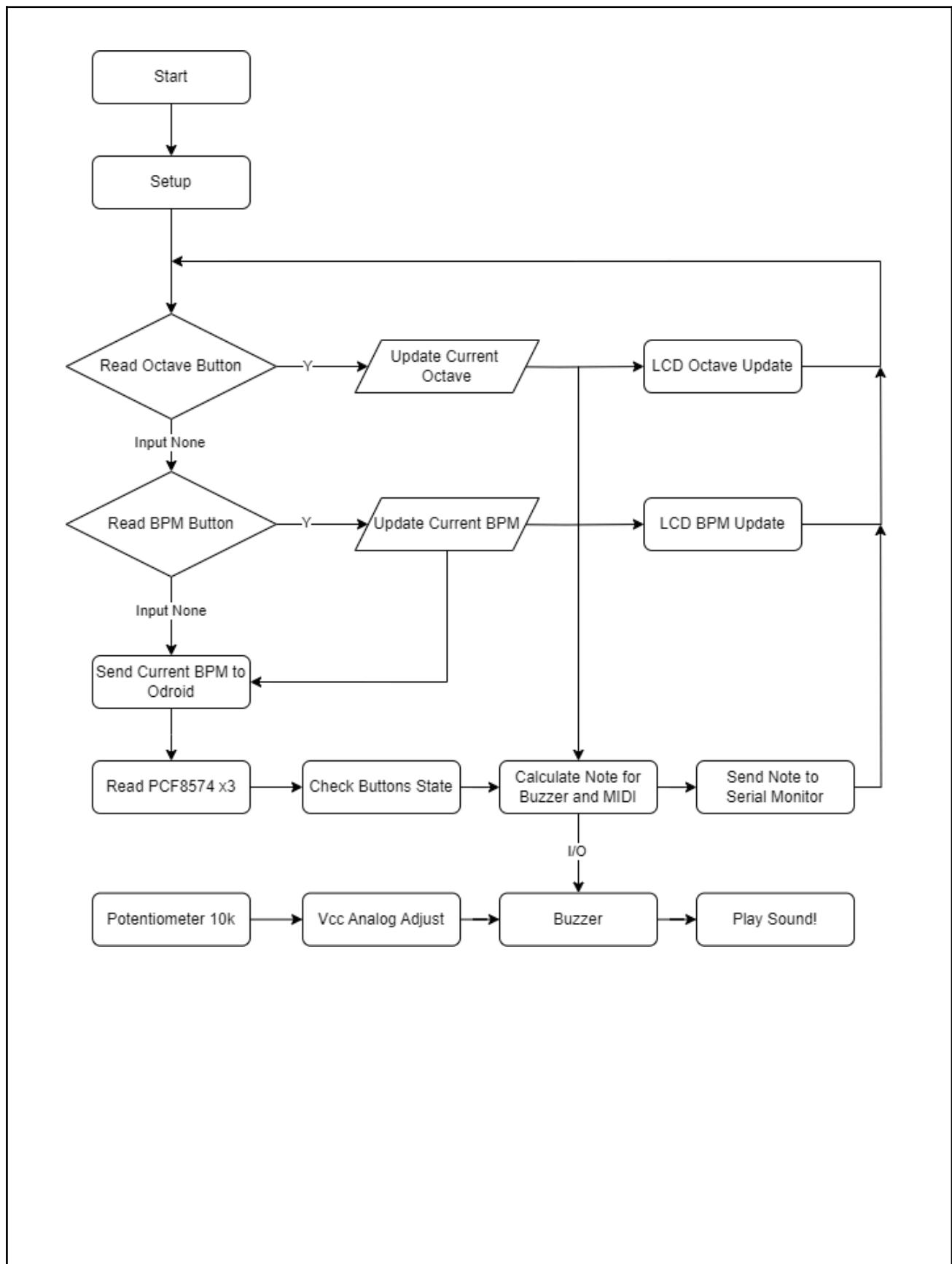
เพื่อกำกการออกแบบอุปกรณ์ทางดนตรี โดยจะมีการใช้จำนวนปุ่มทั้งหมด 24 ปุ่มเพื่อใช้เป็นตัวโน้ต และมีปุ่ม 2 ปุ่มสำหรับเพิ่ม/ลด Octave/BPM และมี LCD สำหรับแสดงข้อมูลต่างๆ

การออกแบบอุปกรณ์ชิ้นนี้ ทำเพื่อให้สามารถเล่นเปียโนได้โดยใช้คอมพิวเตอร์เป็นตัวกลาง ทำให้สามารถเชื่อมต่อกับซอฟแวร์ต่างๆ เช่น Synthesia ซึ่งช่วยฝึกฝนการเล่นเปียโนได้อย่างดี

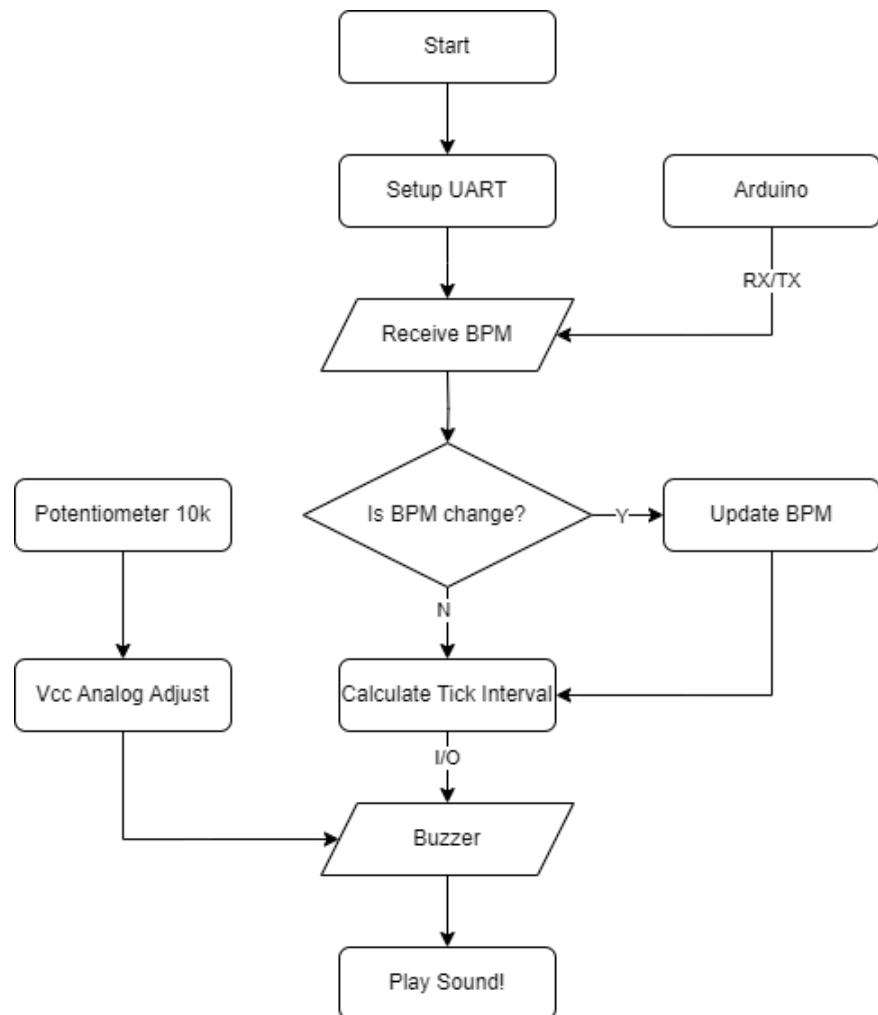
State Diagram



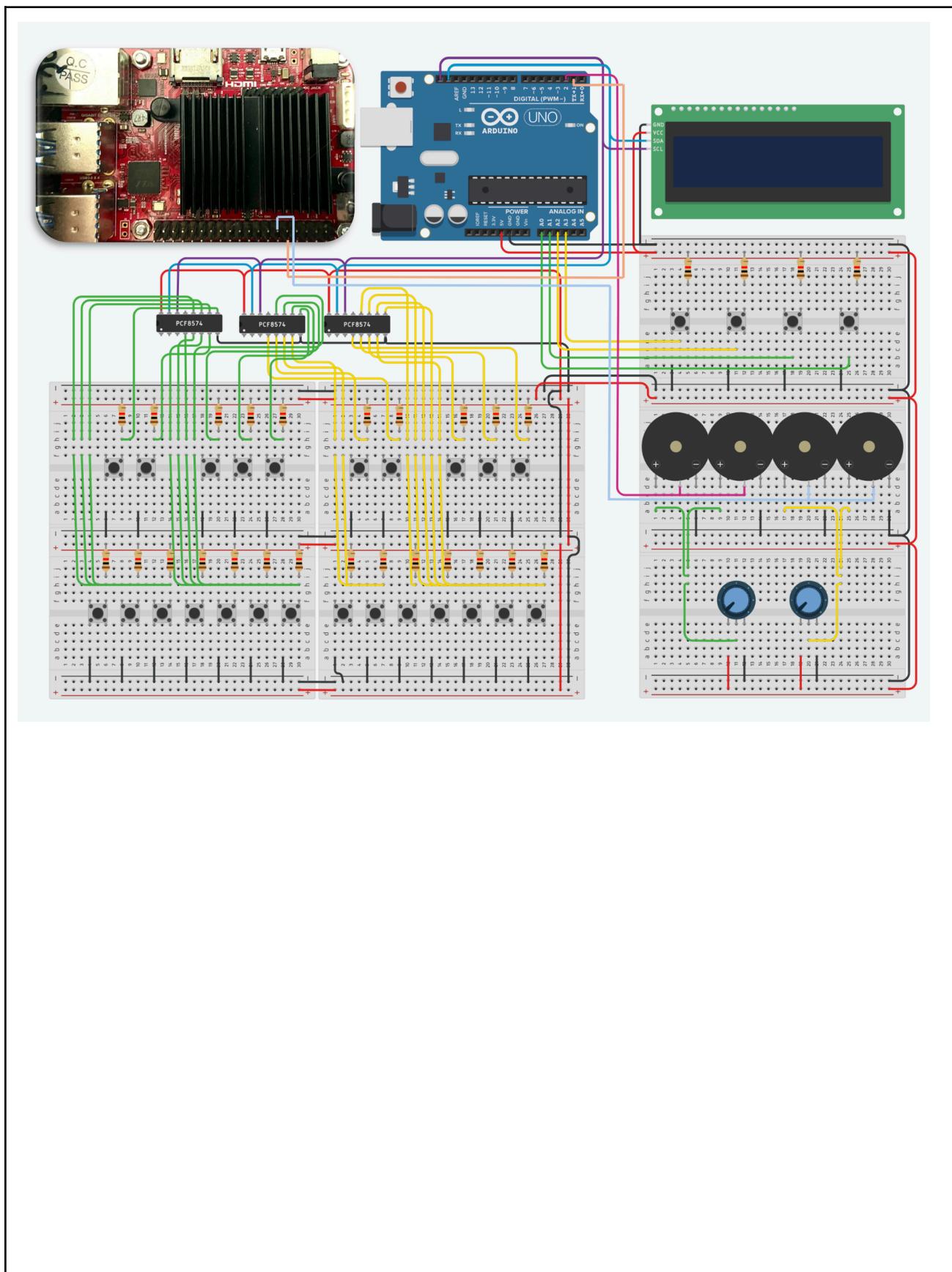
Flowchart Arduino



Flowchart Odroid

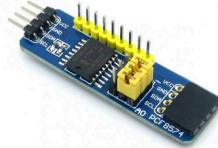


Schematic Design



อุปกรณ์ที่ใช้ในการสร้าง

- Odroid **x1**: ใช้สำหรับประมวลผลค่า BPM ที่รับมาจาก Arduino ผ่าน UART
- Arduino **x1**: ใช้สำหรับอ่านสถานะของปุ่มโน้ต, ปุ่มเพิ่ม/ลด Octave และปุ่มเปลี่ยนเครื่องดนตรี
- PCF8574 **x3**: ใช้สำหรับตรวจจับสถานะของปุ่มโน้ตและส่งข้อมูลไปยัง Arduino
- I2C LCD **x1**: ใช้แสดงข้อมูลการควบคุมและสถานะของอุปกรณ์
- ปุ่มโน้ต **x24**: ใช้สำหรับการกดเพื่อให้ทำการส่งเสียงออกเป็นคีย์ต่างๆ
- ปุ่มเพิ่ม/ลด Octave **x2**: ใช้สำหรับการเพิ่มหรือลด Octave ของปุ่มโน้ต
- Buzzer Module **x4**: ส่งเสียงสำหรับ Piano และ Metronome
- Potentiometer **x2**: ปรับระดับเสียงของ Buzzer Module

Odroid C4	Arduino UNO R3	PCF8574
		
I2C LCD	Button	Passive Buzzer Module
		
Potentiometer 10k	Jumper Wire	Resistors 100 ohms
		

ไฟล์ Arduino ใช้เพื่ออ่านสถานะของปุ่มโน้ตทั้ง 24 ปุ่ม มีปุ่มเพิ่ม/ลด Octave และมีปุ่มเพิ่มลด bpm ของ metronome ซึ่งจะมีการส่ง bpm ไปยัง Odroid โดยจะมีโหมดเป็นดังนี้

ไฟล์ midi-piano.ino ทำหน้าที่เป็นไฟล์หลักสำหรับทำหน้าที่ต่างๆใน Arduino

<pre>#include <PCF8574.h> #include <MIDI.h> #include "buzzerKeys.h" #include "OCTAVE.h" #include "BPM.h" #include "UART.h"</pre>	<p>ทำการใช้ library และ file ที่จำเป็น โดย <PCF8574.h> จะใช้ในการอ่าน input จากปุ่มทั้ง 24 ปุ่ม <MIDI.h> จะใช้ส่งสัญญาณของ MIDI ไปยัง Synthesia “buzzerKeys.h” ใช้ตั้งความถี่เสียงที่ออกจาก buzzer จากการกดโน้ต “OCTAVE.h” ใช้ควบคุมการทำงานของ octave “BPM.h” ใช้ควบคุมการทำงานของ metronome “UART.h” ใช้ควบคุมการส่งข้อมูลจาก Arduino ไปยัง Odroid</p>
<pre>// Pin definitions #define BUZZER_PIN 8</pre>	<p>ทำการตั้งให้ pin สำหรับ buzzer อยู่ที่ pin 8</p>
<pre>// Midi init MIDI_CREATE_DEFAULT_INSTANCE();</pre>	<p>ทำการสร้าง MIDI interface สำหรับใช้ในโปรแกรมโดยจะทำให้สามารถส่งและรับ MIDI messages ได้</p>
<pre>void setup() { PCF1.begin(); PCF2.begin(); PCF3.begin(); setupBuzzer(); setupBPMButtons(); setupOctaveButtons(); setupUART(); setupLCD(); displayOctave(getCurrentOctave()); displayBPM(getCurrentBPM()); MIDI.begin(MIDI_CHANNEL_OMNI); Serial.begin(115200); }</pre>	<p>Function setup จะมีการเริ่มต้นการสื่อสารกับ PCF8574 I/O expanders ทั้ง 3 ตัวเพื่อใช้ในการส่งสัญญาณจากการกดตัวโน้ต มีการเรียกใช้ setupBuzzer, setupBPMButtons, setupOctaveButtons, setupUART และ setupLCD เพื่อกำกการ setup buzzer, ปุ่มสำหรับเพิ่มลด octave, การส่งสัญญาณจาก Arduino ไปยัง Odroid และการแสดงผลบน LCD มีการแสดงผลของ octave และ bpm บน หน้าจอ ทำการเริ่มต้นใช้งาน MIDI เพื่อใช้เชื่อมกับ program Synthesia</p>
<pre>boolean ButtonWasPressed1[] = { false, false, false, false, false, false, false }; boolean ButtonWasPressed2[] = { false, false, false, false, false, false, false }; boolean ButtonWasPressed3[] = { false, false, false, false, false, false, false };</pre>	<p>ทำการตั้ง array สำหรับอ่านสถานะของปุ่มที่เชื่อมอยู่กับ PCF8574 I/O expanders ทำให้สามารถรู้ว่าปุ่มถูกกดอยู่หรือไม่ โดยสถานะเริ่มต้นจะเริ่มที่ false ซึ่งต้องไม่ถูกกด แล้วทำการ initialize PCF8574 ไว้ใน address 0x20, 0x21 และ 0x22 ตามลำดับเพื่อใช้ควบคุม input จากปุ่มที่กด</p>

```

PCF8574 PCF1(0x20);
PCF8574 PCF2(0x21);
PCF8574 PCF3(0x22);

```

```

void checkKey1(int key) {
    unsigned long currentTime = millis();
    const unsigned long DebounceTime = 10;
    static unsigned long ButtonStateChangeTime = 0;
    boolean buttonIsPressed =
        PCF1.readButton(key) == LOW;
    if (buttonIsPressed != ButtonWasPressed1[key] && currentTime - ButtonStateChangeTime > DebounceTime) {
        ButtonWasPressed1[key] = buttonIsPressed;
        ButtonStateChangeTime = currentTime;
        int buzzerNote = 1 + (12 * getCurrentOctave()) + key;
        int midiNote = 24 + (12 * getCurrentOctave()) + key;;
        if (ButtonWasPressed1[key]) {
            // printBuzzerNote(buzzerNote);
            playNoteOnBuzzer(buzzerNote);
            MIDI.sendNoteOn(midiNote, 127, 1);
            delay(200);
        } else {
            stopNoteOnBuzzer();
            MIDI.sendNoteOff(midiNote, 0, 1);
        }
    }
}

```

- Function checkKey1 จะใช้สำหรับสถานะของปุ่มที่เชื่อมกับ PCF8574 I/O expanders ตัวแรก มีการ debouncing เพื่อให้กดปุ่มได้โดยเพื่อลดการเกิด false triggers
- เมื่อ buzzerNote ซึ่งคำนวนออกมาเพื่อนำไปเทียบกับไฟล์ buzzerKeys.h เพื่อส่งเสียงของโน้ตที่กดออกทาง buzzer
- เมื่อ midiNote ซึ่งคำนวนออกมาเพื่อนำไปเทียบกับไฟล์ midiKeys.h เพื่อส่งข้อมูลของโน้ตที่กดออกทาง MIDI เพื่อนำไปส่งเสียงทาง program Synthesia
- เมื่อทำการกดปุ่มจะทำการเรียก function playNoteOnBuzzer เพื่อส่งเสียงโน้ตออกทาง buzzer และใช้ sendNoteOn ส่งออกทาง MIDI และเมื่อหยุดกดก็จะเรียกใช้ function stopNoteOnBuzzer เพื่อหยุดส่งเสียงออก buzzer และใช้ sendNoteOff เพื่อหยุดส่งข้อมูลไปยัง MIDI
- Function checkKey2 และ checkKey3 ก็จะมีหลักการทำงานคล้ายกันโดยทำบน PCF8574 ตัวที่ 2 และ 3 ตามลำดับ

```

void loop() {
    checkOctaveButtons();
    checkBPMButtons();
    sendCurrentBPM();
    for (int i = 0; i < 8; ++i) {
        checkKey1(i);
        checkKey2(i);
        checkKey3(i);
    }
}

```

ในส่วน main loop นี้ จะมีการเรียก function checkOctaveButtons เพื่อตรวจสอบว่ามีการกดปุ่มเปลี่ยน octave หรือไม่ เรียก checkBPMButtons เพื่อตรวจสอบการกดปุ่มเพิ่มลด bpm ของ metronome เรียก sendCurrentBPM เพื่อส่ง bpm ณ ขณะนั้นจาก Arduino ไปยัง Odroid และมีการตรวจสอบปุ่มทั้ง 8 ที่เชื่อมกับ PCF8574 แต่ละตัวและเรียกใช้ function checkKey ทั้ง 3 เพื่อเช็คปุ่มว่ามีการกดหรือไม่ และตอบสนองการกดตัวโน้ตออกทาง buzzer และ MIDI

ไฟล์ I2C_LCD.h เป็นไฟล์ที่ใช้ควบคุมการแสดงผลบน I2C LCD

<pre>#ifndef I2C_LCD_H #define I2C_LCD_H</pre>	ใช้สำหรับกำกับให้ไฟล์ I2C_LCD.h ถูกใช้งานเมื่อเกิดการ compile เพื่อไม่ให้เกิดปัญหาการนิยามตัวแปรซ้ำ
<pre>#include <Wire.h> #include <LiquidCrystal_I2C.h> #define LCD_ADDRESS 0x27 #define LCD_COLUMNS 16 #define LCD_ROWS 2</pre>	ทำการใช้ library Wire.h เพื่อใช้ในการสื่อสารของ I2C และ LiquidCrystal_I2C.h สำหรับควบคุม interface ของ LCD ทำการตั้ง address ของ LCD เป็น 0x27 และตั้งให้มี 16 columns และ 2 rows
LiquidCrystal_I2C lcd(LCD_ADDRESS, LCD_COLUMNS, LCD_ROWS);	ทำการสร้าง instance ของ LiquidCrystal_I2C class เพื่อใช้ข้อมูลในการ setup LCD
<pre>void setupLCD() { lcd.begin(); lcd.backlight(); }</pre>	ทำการสร้าง function setupLCD เพื่อเริ่มต้นการสื่อสารกับ LCD และเปิด backlight ของ LCD เพื่อให้สามารถมองเห็นข้อความที่ปรากฏบนหน้าจอ
<pre>void displayOctave(int octave) { lcd.setCursor(0, 0); lcd.print("Octave: "); lcd.print(octave+1); lcd.print(", "); lcd.print(octave+2); }</pre>	สร้าง function displayOctave โดยมีการ set ให้ cursor อยู่ที่แรกและหลักแรก จากนั้น print ค่า octave ของโน้ตทั้ง 2 ฝั่ง
<pre>void displayBPM(int bpm) { lcd.setCursor(0, 1); lcd.print("BPM: "); lcd.print(bpm); lcd.print(" "); }</pre>	สร้าง function displayBPM โดย set ให้ cursor อยู่ที่แรก 2 หลักแรก และ print ค่า bpm ของ metronome

ไฟล์ **buzzerKeys.h**

ไฟล์นี้ใช้สำหรับกำหนดค่าความถี่ของโน้ตและมี function สำหรับเล่นเสียง, หยุดเล่น และแสดงข้อมูลของตัวโน้ตที่มีการเล่น

// Struct to hold note information struct BuzzerNote { int frequency; const char* name; };	เริ่มต้นด้วยการสร้าง Structure Note ซึ่งจะมีตัวแปร 2 ตัว คือ frequency ซึ่งเป็นจำนวนเต็ม และ name ซึ่งเป็นตัวอักษร
const int BUZZER_PIN = 2;	กำหนดการตั้งให้ buzzer เชื่อมกับ pin 2 ของ Arduino
void setupBuzzer() { pinMode(BUZZER_PIN, OUTPUT); } void stopNoteOnBuzzer() { noTone(BUZZER_PIN); }	สร้าง function สำหรับ setup buzzer และหยุดการส่งเสียงของ buzzer
BuzzerNote getBuzzerNoteInfo(int midiNote) { BuzzerNote noteInfo; switch (midiNote) { case 1: noteInfo = {33, "C1"}; break; case 2: noteInfo = {35, "C#1"}; break; case 3: noteInfo = {37, "D1"}; break; case 4: noteInfo = {39, "D#1"}; break; case 5: noteInfo = {41, "E1"}; break; case 6: noteInfo = {44, "F1"}; break; case 7: noteInfo = {46, "F#1"}; break; case 8: noteInfo = {49, "G1"}; break; case 9: noteInfo = {52, "G#1"}; break; case 10: noteInfo = {55, "A1"}; break; case 11: noteInfo = {58, "A#1"}; break; case 12: noteInfo = {62, "B1"}; break; ... case 85: noteInfo = {4186, "C8"}; break; default: noteInfo = {0, "Unknown"}; } return noteInfo; }	ทำการตั้ง Function getBuzzerNoteInfo() ซึ่งจะรับ midiNote เพื่อคืนค่าที่ตรงกับ frequency และ name ของ Note structure จากนั้นจะทำการ map midiNote เข้ากับ frequency และ name โดยโน้ตตัวแรกสุดเริ่มที่ C1 ที่ 33Hz ขึ้นไปสูงสุดที่ B8 ที่ 7902Hz ถ้าหากเห็นอ่าจากที่กำหนดไว้จะคืนค่าเป็น "Unknown" และทำการ export เป็น noteInfo ไฟล์นี้จะทำหน้าที่เก็บตัวแปรแต่ละโน้ตไว้เมื่อความถี่ที่เท่าไหร่โดยจะมีทั้งหมด 8 octaves ซึ่งโน้ตตัวแรกจะเป็น C1 และตัวสุดท้ายจะเป็น C8

```

void playNoteOnBuzzer(int buzzerNote) {
    BuzzerNote noteInfo =
getBuzzerNoteInfo(buzzerNote);
    if (noteInfo.frequency > 0) {
        tone(BUZZER_PIN, noteInfo.frequency);
    } else {
        noTone(BUZZER_PIN);
    }
}

void stopNoteOnBuzzer() {
    noTone(BUZZER_PIN);
}

```

กำหนด function playNoteOnBuzzer จะทำการเล่นเสียงโน้ตที่สอดคล้องกับความถี่ โดยใช้ข้อมูลจาก buzzerNote และเมื่อ function stopNoteOnBuzzer สำหรับหยุดใช้งาน buzzer

```

void printBuzzerNote(int buzzerNote) {
    BuzzerNote noteInfo =
getBuzzerNoteInfo(buzzerNote);
    if (noteInfo.frequency > 0) {
        Serial.print("Note: ");
        Serial.print(noteInfo.name);
        Serial.print(", Frequency: ");
        Serial.println(noteInfo.frequency);
    } else {
        Serial.println("Invalid note");
    }
}

```

Function printBuzzerNote ใช้สำหรับ print ข้อมูลของตัวโน้ตที่กด ณ ขณะนั้น โดยจะมีชื่อโน้ตและความถี่ที่ถูก print ออกมา

ไฟล์ UART.h

ใช้ในการ setup และจัดการการทำงานของการสื่อสาร UART โดยจะทำการเริ่มต้นการทำงานของการเชื่อมต่อของ UART และทำการส่งค่า bpm ผ่าน UART interface

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include "BPM.h"
```

ทำการใช้ library Wire.h เพื่อใช้ในการสื่อสารของ I2C ใช้ SoftwareSerial.h เพื่อใช้ในการสื่อสารผ่าน serial port และใช้ file BPM.h ซึ่งมี currentBPM สำหรับทำการ serial print

```
const int rxPin = 2;
const int txPin = 11;
SoftwareSerial mySerial(rxPin, txPin);
```

กำหนดให้ receive pin อยู่ที่ pin 2 และ transmit pin อยู่ที่ pin 11 ของ Arduino เพื่อใช้ในการรับส่งข้อมูลระหว่าง Arduino และ Odroid โดยทำการสร้าง SoftwareSerial object ซึ่งว่า mySerial ในการควบคุม serial communication

```
void setupUART() {
    mySerial.begin(115200);
}
```

ทำการสร้าง function สำหรับ setup UART โดยจะมีการสื่อสารที่ baud rate = 115200

```
void sendCurrentBPM() {
    mySerial.println(currentBPM);
    // Serial.print(currentBPM);
}
```

สร้าง function สำหรับส่งค่า bpm ณ ขณะนั้นผ่าน UART connection

ไฟล์ BPM.h

เป็นไฟล์ที่ใช้ควบคุม bpm ของ metronome โดยกำหนดการกดปุ่มเพิ่มลด bpm

<pre>#include <Arduino.h> #include "I2C_LCD.h"</pre>	มีการใช้ library Arduino เพื่อใช้ function เช่น pinMode, , digitalRead และ millis ใช้ไฟล์ I2C_LCD.h เพื่อให้สามารถใช้งาน function displayBPM ได้
--	---

<pre>const int bpmIncPin = 14; const int bpmDecPin = 15;</pre>	กำหนด pin สำหรับเพิ่มและลด bpm เป็น pin 14 และ 15 ตามลำดับ
--	--

<pre>const int minBPM = 10; const int maxBPM = 300; int currentBPM = 120; unsigned long lastBpmIncTime = 0; unsigned long lastBpmDecTime = 0; const unsigned long debounceBPM = 100;</pre>	กำหนด bpm ต่ำสุดและสูงสุดเป็น 10 และ 300 ตามลำดับ กำหนดให้ bpm เริ่มต้นเป็น 120 มีการกำหนดตัวแปรสำหรับ debouncing ซึ่ง lastBpmIncTime และ lastBpmDecTime ใช้สำหรับ track ว่าการกดครั้งสุดท้ายเกิดขึ้นเมื่อไหร่ และมี debounceBPM สำหรับ debounce การกดปุ่ม
--	---

<pre>void setupBPMButtons() { pinMode(bpmIncPin, INPUT_PULLUP); pinMode(bpmDecPin, INPUT_PULLUP); }</pre>	สร้าง function setupBPMButtons เพื่อกำหนดค่าของปุ่มเพิ่มลด bpm
---	--

<pre>int getCurrentBPM() { return currentBPM; }</pre>	สร้าง function getCurrentBPM สำหรับคืนค่า bpm กลับมา
---	--

<pre>void increaseBPM() { if (currentBPM < maxBPM) { currentBPM += 5; } else { currentBPM = minBPM; } displayBPM(getCurrentBPM()); } void decreaseBPM() { if (currentBPM > minBPM) { currentBPM -= 5; } else {</pre>	สร้าง function สำหรับเพิ่มลด bpm โดยจะทำการเพิ่มลดทีละ 5 เมื่อกดปุ่มและจะทำการย้อนกลับเป็นค่าต่ำสุดของ bpm เมื่อเพิ่ม bpm จนเกินค่าสูงสุด และจะเปลี่ยนค่าเป็นค่า bpm สูงสุดถ้าลด bpm จนต่ำกว่าค่าต่ำสุด จากนั้นจะแสดงค่าบน LCD
---	--

```

        currentBPM = maxBPM;
    }
    displayBPM(getCurrentBPM());
}

```

```

void checkBPMButtons() {
    unsigned long currentMillis = millis();
    if (digitalRead(bpmIncPin) == LOW) {
        if (currentMillis - lastBpmIncTime > debounceBPM) {
            increaseBPM();
            lastBpmIncTime = currentMillis;
        }
    }
    if (digitalRead(bpmDecPin) == LOW) {
        if (currentMillis - lastBpmDecTime > debounceBPM) {
            decreaseBPM();
            lastBpmDecTime = currentMillis;
        }
    }
}

```

สร้าง function checkBPMButtons เพื่อทำการเช็คการเพิ่มลด bpm โดยมีการ debouncing เพื่อให้มีการ register หลังจาก กดปุ่ม 100 ms และมีการตรวจสอบการกดซ้ำจะทำการ update เวลาล่าสุดที่ทำการกดปุ่มเพิ่มลด bpm

ไฟล์ OCTAVE.h

<pre>#include <Arduino.h> #include "I2C_LCD.h"</pre>	ทำการใส่ library <Arduino.h> สำหรับใช้ function pinMode[], digitalRead[], and millis[]
<pre>// Define pins for buttons const int buttonIncreasePin = 16; const int buttonDecreasePin = 17;</pre>	เป็นการกำหนด pin สำหรับเพิ่มและลด octave โดยกำหนดให้ pin 16 ทำการเพิ่ม octave และ pin 17 ทำการลด octave ซึ่งตั้งตัวแปรเป็น buttonIncreasePin และ buttonDecreasePin ตามลำดับ
<pre>// Define octave const int minOctave = 0; const int maxOctave = 5; int currentOctave = 3; unsigned long lastIncreaseTime = 0; unsigned long lastDecreaseTime = 0; const unsigned long debounceOctave = 200;</pre>	เป็นการกำหนดให้ octave ที่ปรับได้สูงสุดคือ 5 และปรับได้ต่ำสุดคือ 0 มีการตั้งให้ปัจจุบันเป็น octave ที่ 3 มีการตั้งให้ตัวแปรที่เก็บค่าล่าสุดที่ปุ่มเพิ่ม octave ถูกกดเป็น lastIncreaseTime และตัวแปรที่เก็บค่าล่าสุดที่ปุ่มลด octave ถูกกดเป็น lastDecreaseTime มีการตั้งให้ debounce time เป็น 200 ms
<pre>void setupOctaveButtons() { pinMode(buttonIncreasePin, INPUT_PULLUP); pinMode(buttonDecreasePin, INPUT_PULLUP); } int getCurrentOctave() { return currentOctave; } void increaseOctave() { if (currentOctave < maxOctave) { currentOctave++; displayOctave(getCurrentOctave()); } } void decreaseOctave() { if (currentOctave > minOctave) { currentOctave--; } else { currentOctave = minOctave; } displayOctave(getCurrentOctave()); }</pre>	มีการกำหนด function setupOctaveButtons เพื่อเรียกใช้งาน ปุ่มเพิ่มลด octave และกำหนด function getCurrentOctave เพื่อคืนค่า octave ณ ขณะนั้นกลับศีบมา จากนั้นกำหนด function สำหรับเพิ่มลด octave คือ increaseOctave และ decreaseOctave เพื่อใช้สำหรับเพิ่มและลด octave ทีละ 1 โดยจะนำไปใช้มีการเพิ่มลด octave เกินค่าสูงสุดและต่ำสุดที่กำหนดไว้

```

void checkOctaveButtons() {
    unsigned long currentMillis = millis();
    if (digitalRead(buttonIncreasePin) == LOW)
    {
        if (currentMillis - lastIncreaseTime > debounceOctave)
        {
            increaseOctave();
            lastIncreaseTime = currentMillis;
        }
    }
    if (digitalRead(buttonDecreasePin) == LOW)
    {
        if (currentMillis - lastDecreaseTime > debounceOctave)
        {
            decreaseOctave();
            lastDecreaseTime = currentMillis;
        }
    }
}

```

ทำการสร้าง function checkOctaveButtons เพื่อตรวจสอบว่าปุ่มใดถูกกด ถ้าหาก buttonIncreasePin ถูกกดก็จะทำการเรียก function increaseOctave เพื่อเพิ่ม octave ถ้า buttonDecreasePin ถูกกดก็จะเรียก function decreaseOctave เพื่อลด octave ซึ่งจะมีการคำนวณระยะเวลาในการกดปุ่มโดยใช้เวลา ณ ขณะนั้นลบกับเวลาล่าสุดที่กดปุ่มเพิ่มหรือลด ถ้าหากเวลาที่คำนวณมากกว่าเวลาสำหรับ debounce จะถือว่ากดปุ่มสำเร็จ ซึ่งจะช่วยป้องกันปัญหา input noise

ไฟล์ Odroid จะทำการรับข้อมูล bpm จาก Arduino และทำการส่งเสียงออกทาง buzzer เพื่อกำการจำลอง metronome ซึ่งการปรับ bpm ของ metronome จะสามารถทำได้ด้วยการกดปุ่มที่เชื่อมกับ Arduino ซึ่งมีการตั้งให้ส่งข้อมูลไปยัง Odroid และจะแสดงค่า bpm บน LCD

ไฟล์ main.py

ใช้เป็นไฟล์หลักในการทำงานของ Odroid โดยจะเรียกใช้งาน function ของไฟล์อื่นๆ มาใช้งาน

<pre>from uart import setup_uart, read_bpm from metronome import setup_buzzer, run_metronome</pre>	จะมีการ import function จากไฟล์ uart.py ต่อ setup_uart และ read_bpm สำหรับการสื่อสารของ UART มีการ import function จากไฟล์ metronome.py ต่อ setup_buzzer และ run_metronome ซึ่งใช้สำหรับควบคุมและจัดการ buzzer
--	---

<pre>def main(): buzzer_pin = 0 prev_bpm = None uart = setup_uart() setup_buzzer(buzzer_pin) while True: bpm = read_bpm(uart) if bpm != prev_bpm: print(f"BPM changed from {prev_bpm} to {bpm}. Restarting metronome.") prev_bpm = bpm run_metronome(bpm, buzzer_pin) else: run_metronome(bpm, buzzer_pin)</pre>	สร้าง function main ซึ่งจะมีการตั้งตัวแปรต่างๆดังนี้ -buzzer pin อยู่ที่ pin 0 ของ Odroid -prev_bpm เพื่อเก็บค่า bpm โดยเริ่มที่ค่า None เพราะยังไม่มีการรับค่า bpm ในตอนแรก -uart จะเรียกใช้ function สำหรับ setup UART connection ระหว่าง Arduino และ Odroid จากนั้นใน loop while true จะทำการอ่าน bpm โดยใช้ function read_bpm ผ่าน UART connection โดยจะเก็บไว้ในตัวแปร “bpm” ถ้าค่า bpm ที่รับมา valid ถ้าหากไม่ valid จะทำการคืนกลับเป็น None จะมีการเช็คค่า bpm ว่ามีการเปลี่ยนแปลงหรือไม่ ถ้าหากมีการเปลี่ยนแปลงจะแสดงผลว่ามีการเปลี่ยน bpm เป็นเท่าไหร่ หากไม่มีการเปลี่ยนแปลงจะทำงานต่อด้วยค่า bpm เดิม และส่ง bpm ออกไปยัง buzzer เพื่อส่งเสียง
---	---

ไฟล์ metronome.py

ไฟล์นี้จะทำหน้าที่ส่งสัญญาณไปยังลำโพงเพื่อส่งเสียงในรูปแบบของ metronome ตามจังหวะที่ตั้งไว้

```
import odroid_wiringpi as wiringpi  
import time
```

มีการ import odroid_wiringpi เพื่อใช้งาน function สำหรับควบคุม GPIO pin และ import time เพื่อควบคุม timing ของ buzzer

```
def setup_buzzer(pin=0):  
    wiringpi.wiringPiSetup()  
    wiringpi.pinMode(pin, 1)  
    wiringpi.softToneCreate(pin)
```

ทำการสร้าง function สำหรับ setup buzzer โดยเริ่มการใช้งาน wiringPi library และตั้งให้ pin 0 เป็น output เพื่อส่งสัญญาณออก จากนั้นจึง generate tone จาก pin ที่ตั้งไว้

```
def run_metronome(bpm, buzzer_pin=0):  
  
    if bpm is None:  
        return  
    elif bpm < 10:  
        bpm = 300  
  
    print(f"Buzzer ON with {bpm} BPM")  
    interval = 60.0 / bpm  
  
    wiringpi.softToneWrite(buzzer_pin, 1000)  
    time.sleep(0.1)  
    wiringpi.softToneWrite(buzzer_pin, 0)  
    time.sleep(interval - 0.1)  
  
    print(f"interval: {interval:.2f} sec")
```

สร้าง function สำหรับการทำงานของ metronome โดยจะตั้งให้เริ่มทำงานที่ bpm กับ pin ที่ตั้งไว้ โดยถ้าค่า bpm ที่ได้รับมาเป็น invalid จะทำการ return ค่าเปล่าซึ่งจะทำให้ไม่เกิดอะไรมีขึ้น เมื่อค่า bpm น้อยกว่า 10 จะทำการเปลี่ยนเป็น 300 ซึ่งจะใช้แสดงว่าเกิดปัญหาหรือไม่หาก bpm เกิดความผิดพลาดกล้ายเป็นค่า = 0 และทำการคำนวณ interval เพื่อหาระยะห่างแต่ละ tick จากนั้นทำการส่ง tone ที่จะส่งเสียงออกมากาง buzzer ที่ 1000 Hz และหยุดการทำงานของ program เป็นเวลา 100 ms ขณะที่ buzzer กำลังส่งเสียงเพื่อควบคุมเวลาที่ buzzer จะส่งเสียง และเมื่อจังหวะที่ 0 Hz เพื่อให้ buzzer หยุดส่งเสียง เมื่อ buzzer ส่งเสียงเป็นเวลา 100 ms และจะให้ program หยุดทำงานเป็นเวลา = interval - 0.1 เพื่อให้ interval ในระหว่าง tick และ print ค่า interval ออกมา

ไฟล์ uart.py

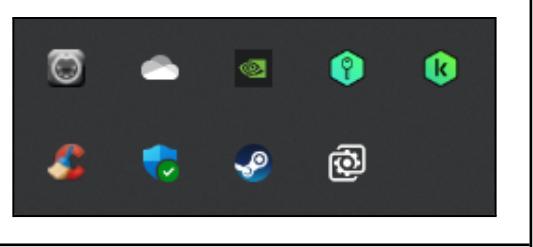
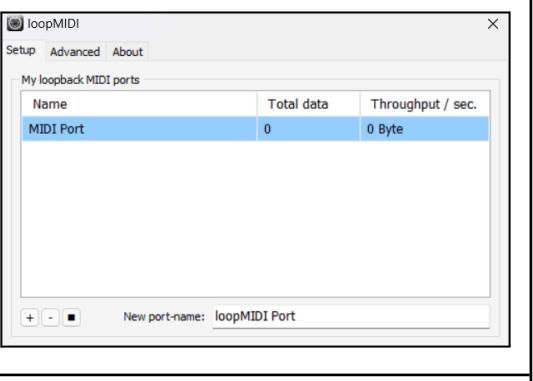
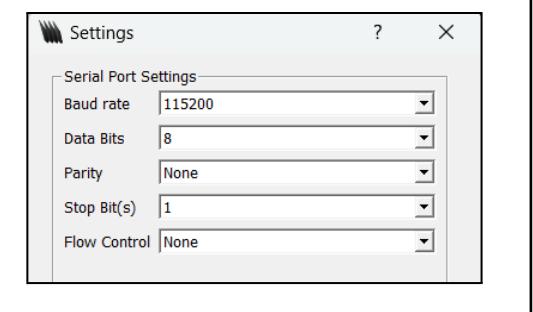
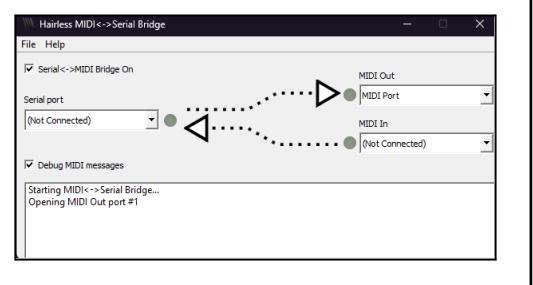
ไฟล์นี้ใช้สำหรับควบคุม UART communication สำหรับการอ่านค่า bpm ที่ได้รับมาจาก Arduino

<pre>import serial</pre>	จะมีการ import serial ซึ่งเป็นส่วนหนึ่งของ pySerial library ซึ่งจะมีการจัดหา interface สำหรับ serial communication ใน python
--------------------------	--

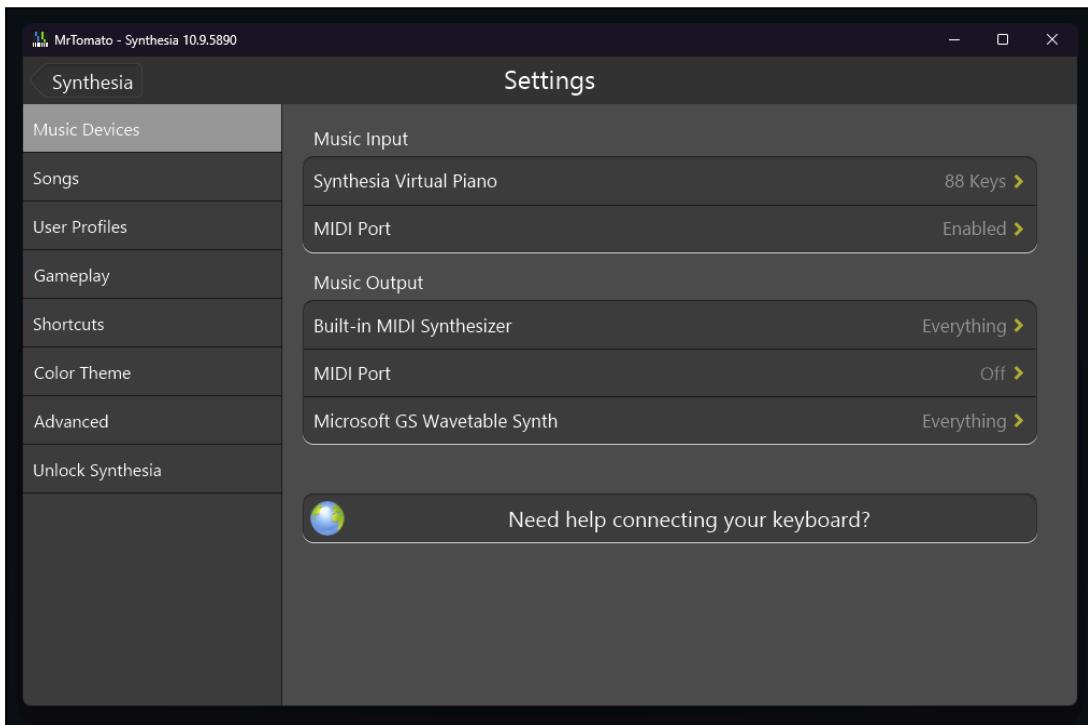
<pre>def setup_uart(port='/dev/ttyS1', baudrate=115200): uart = serial.Serial(port, baudrate, timeout=1) uart.reset_input_buffer() return uart</pre>	ทำการสร้าง function สำหรับ setup UART connection โดยใช้ port ttyS1 และมี baud rate ที่ 115200 โดยถ้าหากไม่ได้รับ data ในเวลา 1 วินาที จะทำการคืนค่า None และทำการ clear data เก่าใน buffer เพื่อให้มั่นใจว่าจะมีเพียง data ใหม่ที่ได้รับการประมวลผลหลังจากทำการเชื่อมต่อ จากนั้นจะคืนค่า uart เพื่อใช้สำหรับอ่านข้อมูลที่ได้จาก serial connection
--	---

<pre>def read_bpm(uart): if uart.in_waiting > 0: try: bpm = int(uart.readline().decode('utf-8', errors='ignore').strip()) uart.reset_input_buffer() return bpm except ValueError: print("Invalid BPM received.") uart.reset_input_buffer() return None return None</pre>	จะมี function สำหรับอ่าน bpm ที่ได้จากการอ่าน data ผ่าน uart connection โดยถ้ามี data รออยู่ใน serial input buffer จะทำงานต่อ แต่ถ้าไม่มีจะ return None เมื่อได้รับ data มา จะทำการอ่าน data จาก serial input ซึ่งจะมีการ decode byte data เป็น UTF-8 string โดยจะทำการ ignore error ที่เกิดขึ้นระหว่างการ decode และทำการลบ whitespace ออกทั้งหมด และเปลี่ยน string ที่ได้เป็น int โดยถ้า data เป็น invalid integer จะทำการ raise ValueError ถ้า ValueError เกิดขึ้น จะ print error message และ clear input buffer เพื่อกำจัดข้อมูลที่เกิดการ error และ return ว่า None เพื่อแสดงว่าได้รับ invalid bpm
---	---

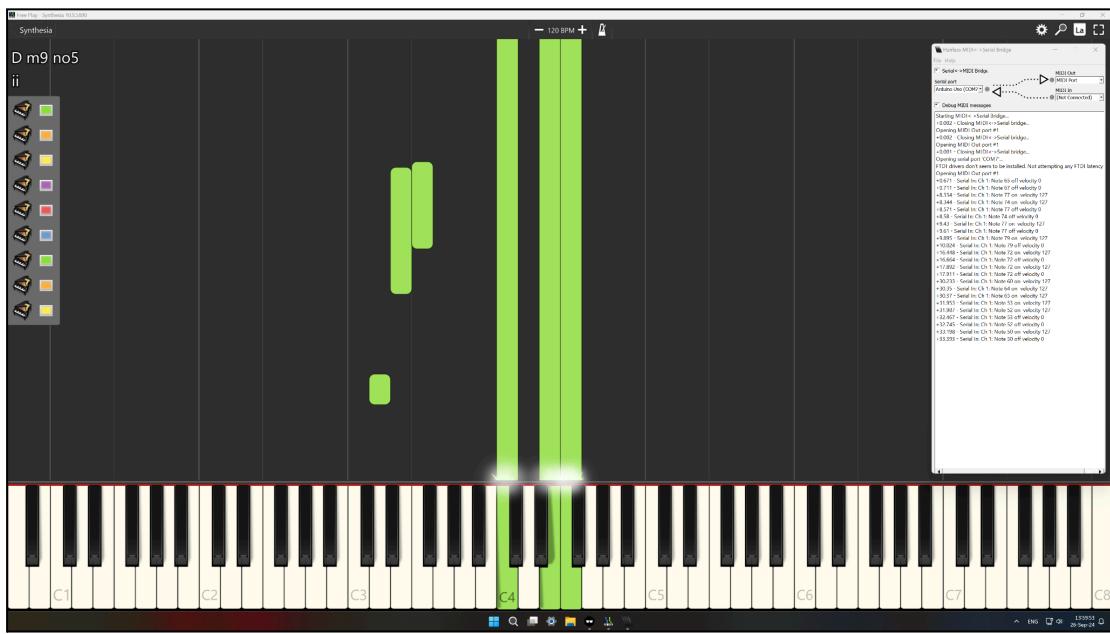
ขั้นตอนการ Setup เพื่อใช้งานกับโปรแกรมที่ชัพพอร์ต MIDI [Synthesia]

<p>1] ติดตั้ง loopMIDI เพื่อสร้าง Virtual MIDI Port สำหรับการสื่อสาร Arduino กับ Synthesia เมื่อติดตั้งเสร็จแล้วจะจะขึ้นใน icons ดังนี้</p>	
<p>2] ทำการ setup loopMIDI โดยตั้งชื่อตามที่ต้องการ เช่น MIDI Port เพื่อใช้เป็น Port เชื่อมไปยัง Synthesia</p>	
<p>3] จากนั้นเข้าไปยัง hairless-midiserial โดยทำการ setup ใน preference</p>	
<p>4) ถ้าหาก loopMIDI มีการ setup ถูกต้องจะขึ้นใน hairless-midiserial ดังนี้</p>	
<p>5) ทำการ setup ในโค้ดให้ serial begin เท่ากับ baud rate ที่ตั้งไว้ใน hairless-midiserial</p>	<pre>void setup() { setupPinMode(); setupOctaveButtons(); MIDI.begin(MIDI_CHANNEL_OMNI); Serial.begin(115200);</pre>

6] ทำการ setup ใน Synthesia โดยเลือกให้ MIDI Port เปิดการใช้งาน



7] สามารถเล่น Piano ได้ทั้ง 2 Octave โดยสามารถดูสัญญาณ Input ได้จากหน้าต่าง hairless-midiserial ในการใช้ Debug ได้



ประสิทธิภาพในระบบ

ผลการทดสอบ	จำนวนกด	เวลาเฉลี่ยการตอบสนอง	เสียงต่ำ	Throughput
Buzzer Response	84	~184 μs	67.86%	-
Synthesia Response	84	-	99.71%	3 Byte/s
Ghosting Keys	<12	-	-	-

การพัฒนาและต่อยอดโปรเจ็ค

การพัฒนาต่อเน้นสามารถทำได้ เช่น การเพิ่มจำนวนตัวโน้ต จะสามารถทำการเพิ่ม PCF8574 เข้าไปตามจำนวนตัวโน้ตที่ต้องการจะใช้งาน สามารถเปลี่ยนจากการใช้ buzzer ไปใช้ลำโพงแทนเพื่อคุณภาพเสียงที่ดีขึ้น สามารถทำการเปลี่ยนกล่องที่ใส่เปียโนให้เป็นวัสดุอื่นที่มีคุณภาพกว่า future board สามารถเปลี่ยนไปใช้ปุ่มกดของตัวโน้ตที่กดได้ง่ายเพื่อความสะดวกสบายของผู้ใช้งาน จากการที่ metronome นั้นมีเสียงเบาเป็นเพราะ กระและไฟไม่พร้อม การพัฒนาต่อจึงสามารถทำได้โดยเพิ่มแหล่งจ่ายไฟ

สรุปผล

การทำ MIDI Piano มีกั้งการใช้งานของ Arduino และ Odroid ทำให้เราได้ใช้งานอุปกรณ์นี้ในรูปแบบของ piano โดยการใช้งานปุ่มทั้ง 4 ปุ่มจะแบ่งให้ 2 ปุ่มทำการเพิ่มลด octave ของตัวโน้ต และอีก 2 ปุ่มสำหรับเพิ่มลด BPM ของ metronome และยังมี potentiometer 2 ตัวสำหรับเพิ่มลดระดับเสียงของ buzzer

เมื่อใช้งาน จะมีการใช้ loop MIDI เพื่อสร้าง virtual port ซึ่งจะรับข้อมูลผ่าน USB port แล้วเปิด hairless-midiserial เพื่อใช้ส่งสัญญาณจากการกดตัวโน้ตเข้าไปยัง program Synthesia ที่เชื่อมกับ loop MIDI เมื่อกดตัวโน้ตจะทำให้เสียงมีเสียงกับว่ากดตัวโน้ตใน Synthesia และยังสามารถส่งเสียงออกทาง buzzer ได้อีกด้วย

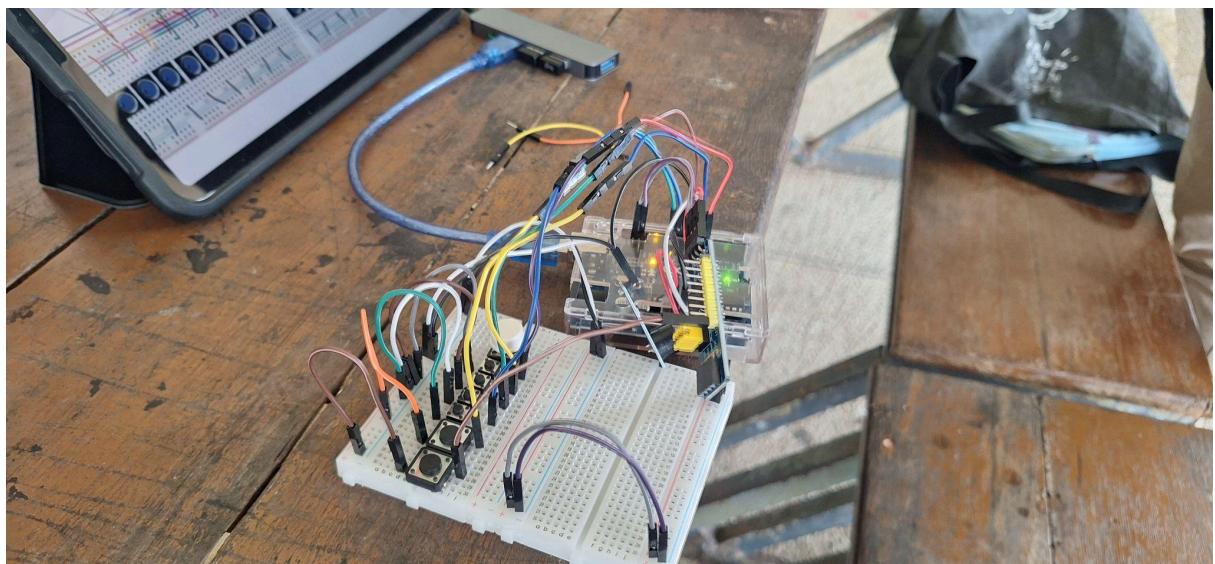
ในส่วน Arduino จะทำการควบคุมการทำงานของตัวโน้ตทั้ง 24 ตัวเพื่อส่งสัญญาณเสียงออกกั้ง program Synthesia และ buzzer ให้ส่งเสียงของตัวโน้ต และยังควบคุมปุ่มทั้ง 4 ปุ่ม และการแสดงผลบน LCD นอกจากนี้ยังมีการส่งข้อมูลของ BPM ไปยัง Odroid

ในส่วน Odroid จะมีการควบคุมการทำงานของ buzzer สำหรับส่งเสียงของ metronome โดยรับข้อมูลของ BPM มาใช้เป็นข้อมูลสำหรับจังหวะเสียงของ metronome ซึ่งข้อมูล BPM นั้นมีการรับมาจาก Arduino

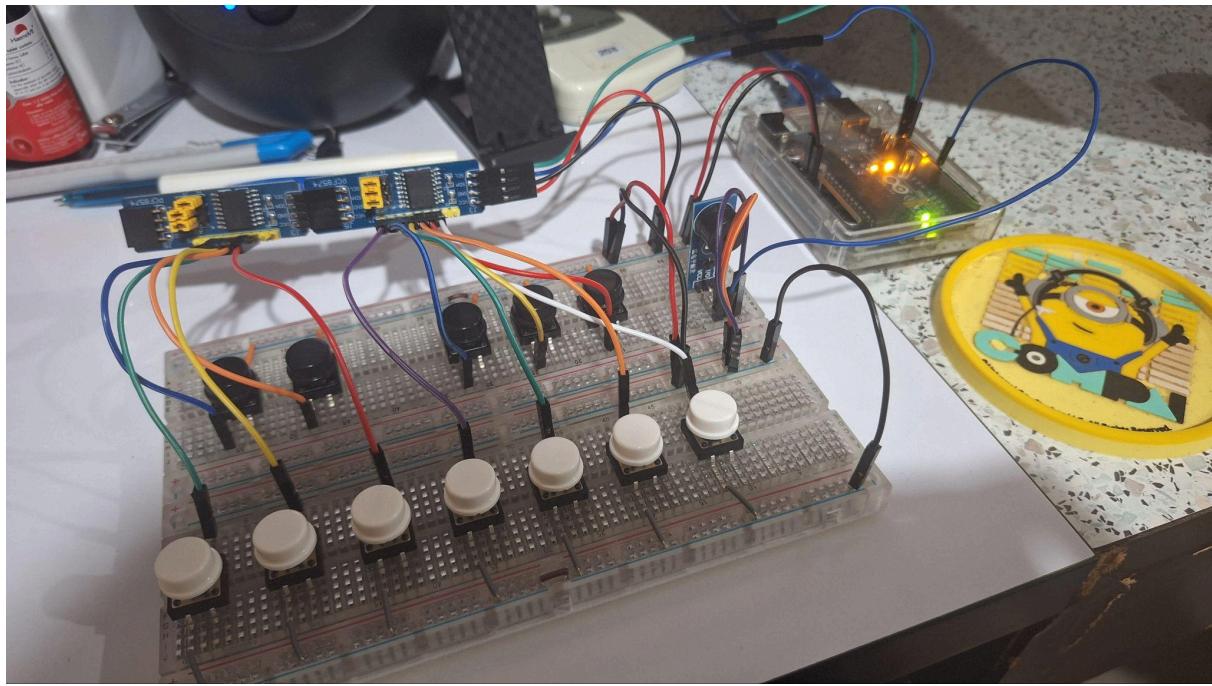
- [Source code & working environment](#)

การแผนฯ

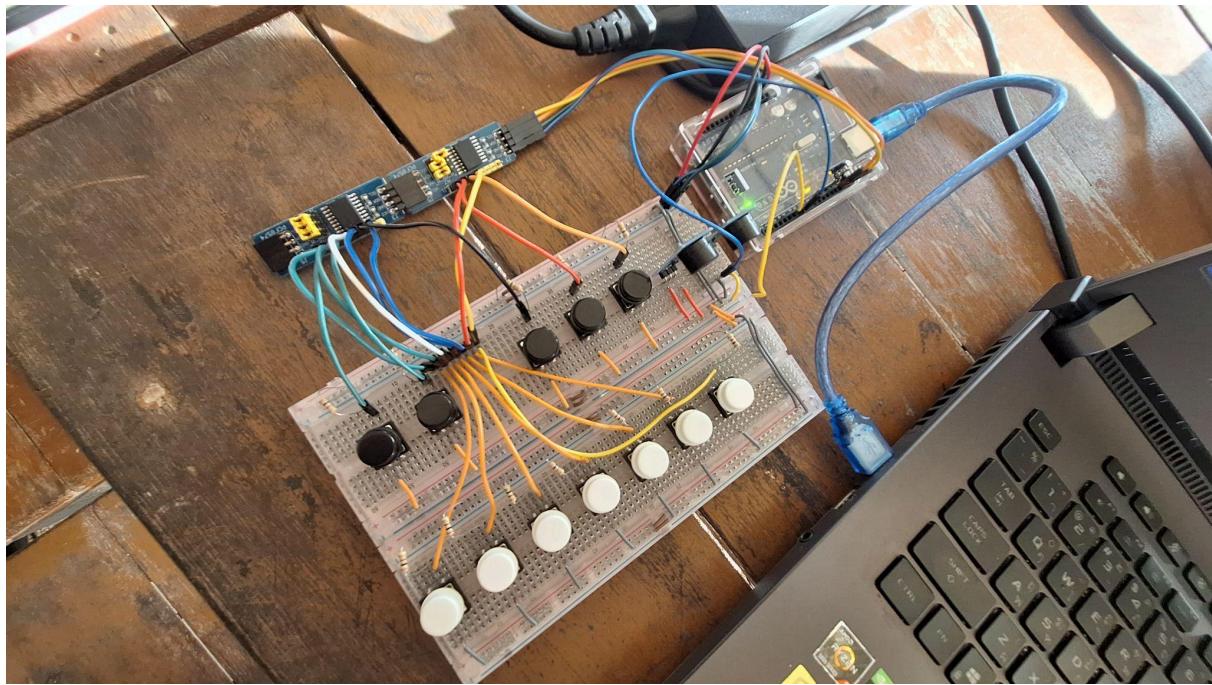
31/08/2567 - ทดลองการใช้ PCF8574 กับปุ่มและหาวีร์ที่ชั้นบัน Buzzer



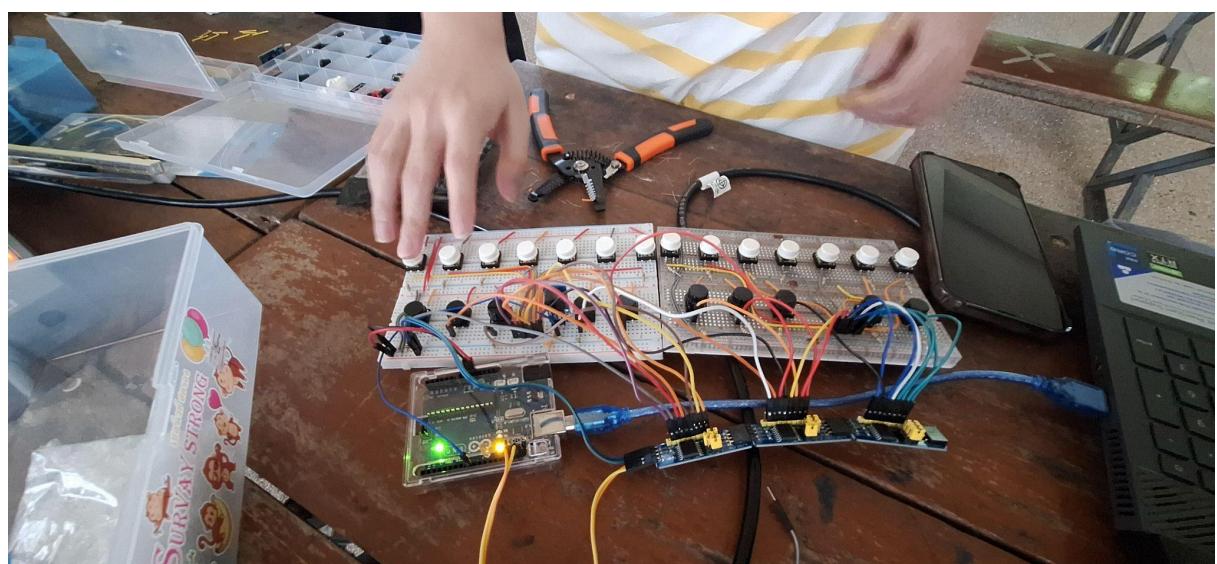
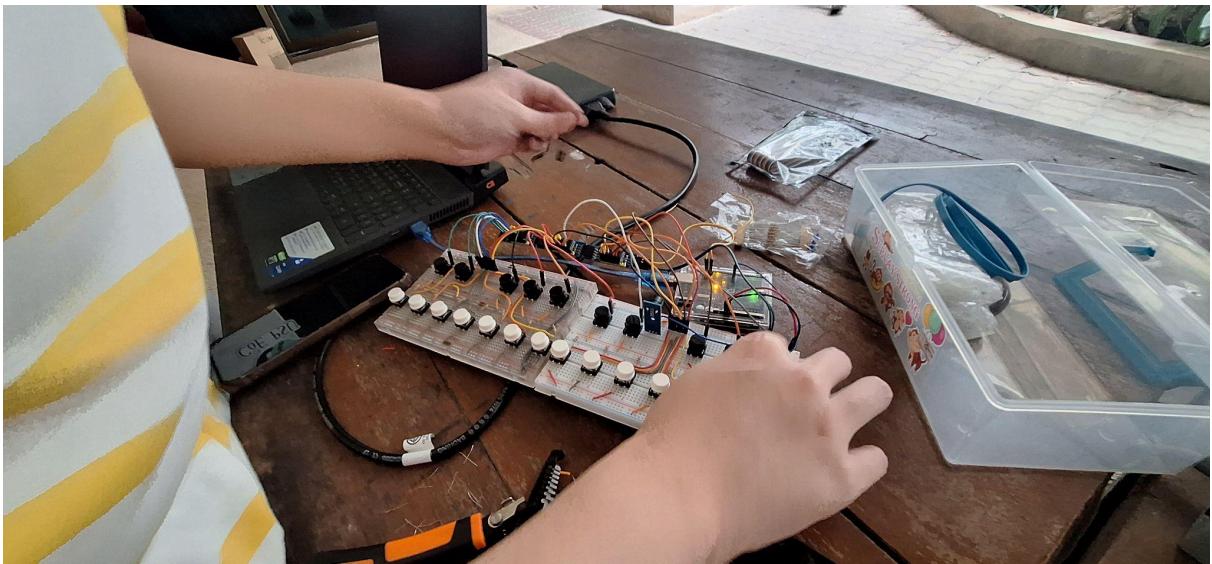
03/09/2567 - วางแผนรูปแบบการจัดวงปุ่มให้เหมาะสมกับเปียโน



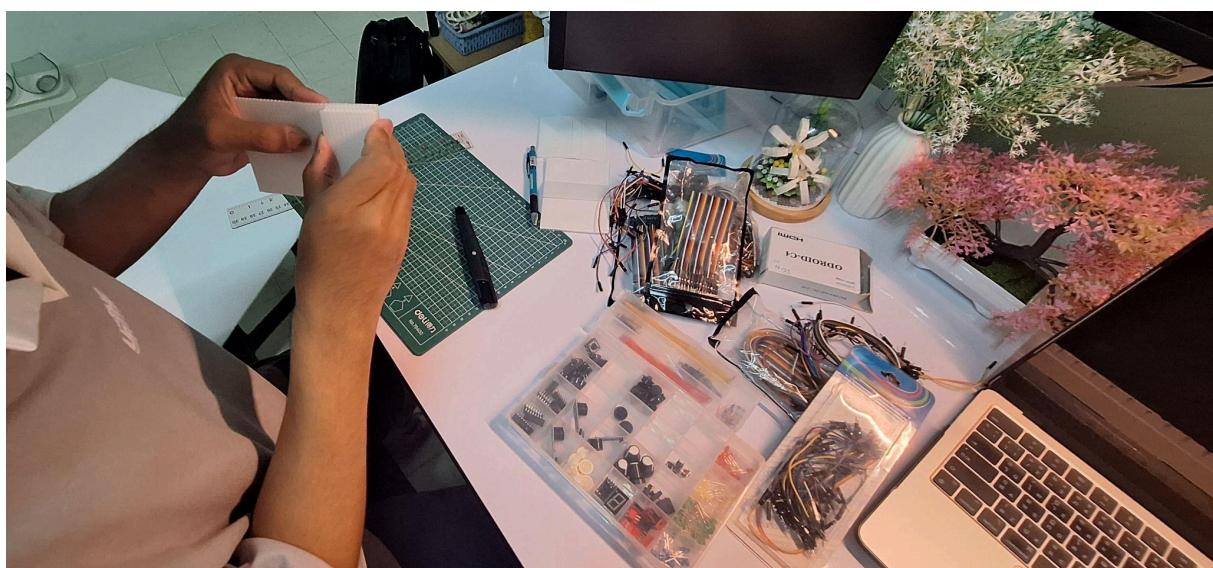
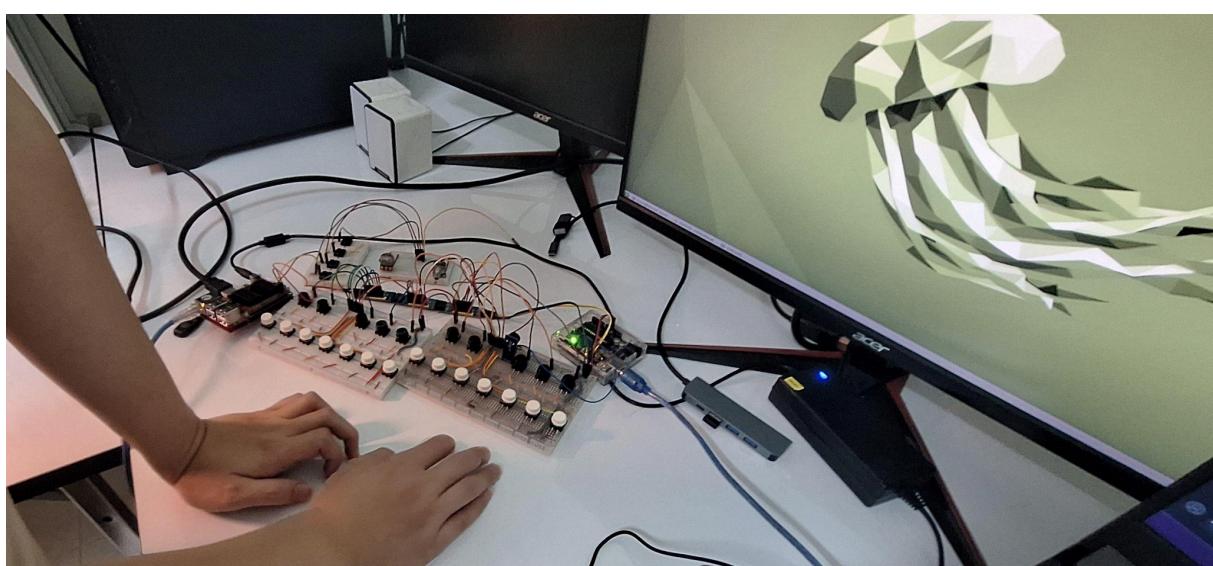
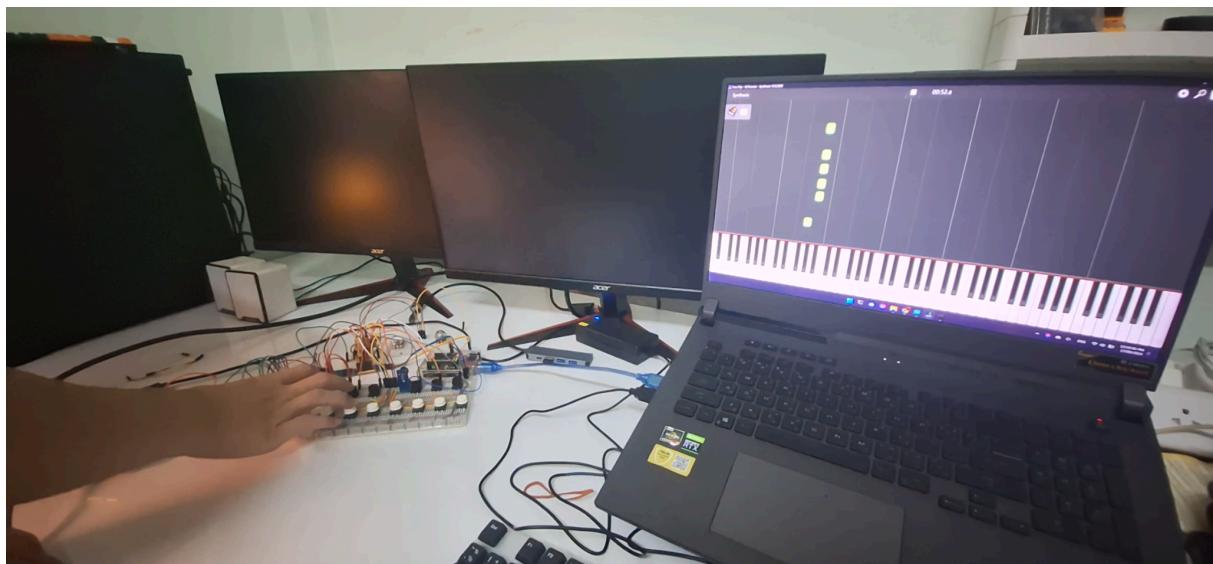
10/09/2567 - เริ่มการจัดสายไฟและทดสอบแก้ไขปัญหาเสียง Keys และ Metronome บน Buzzer



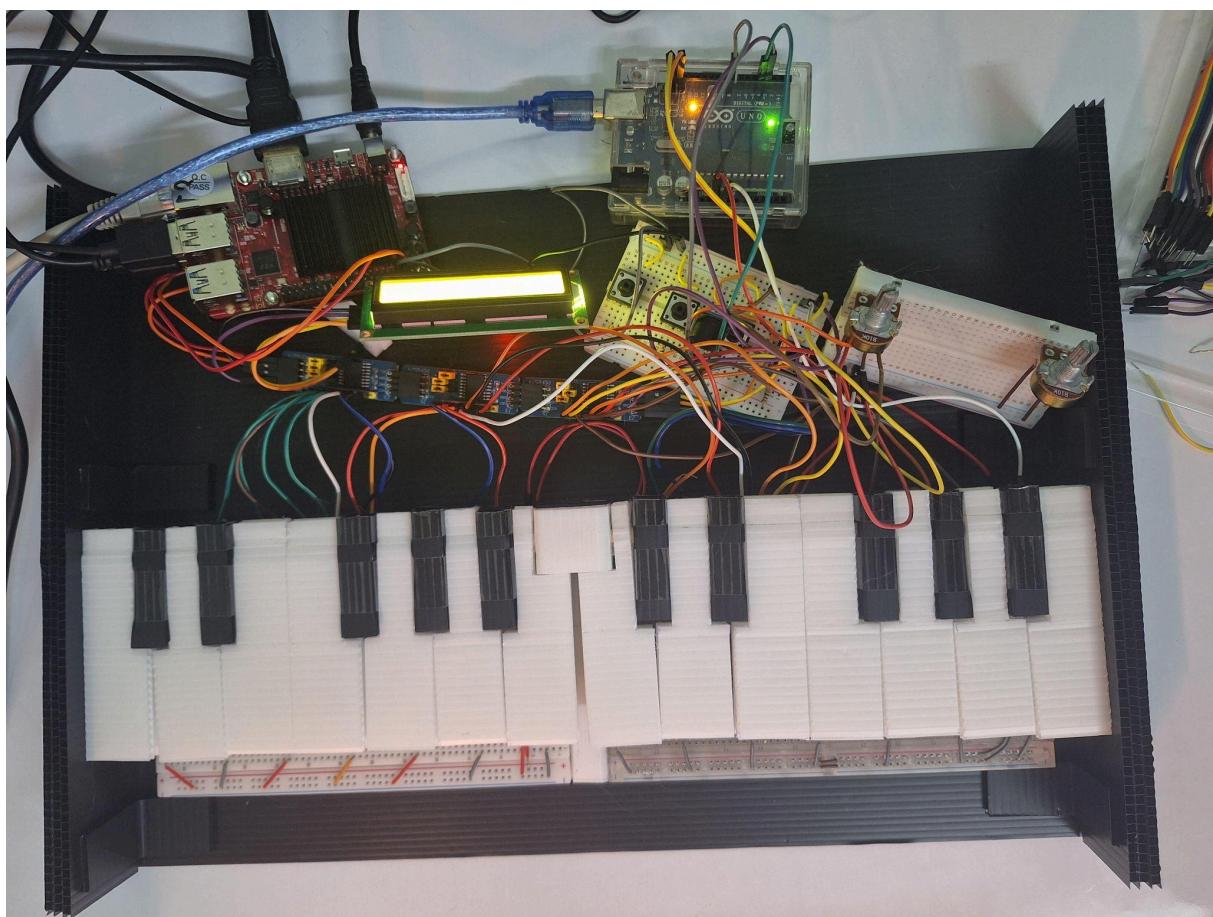
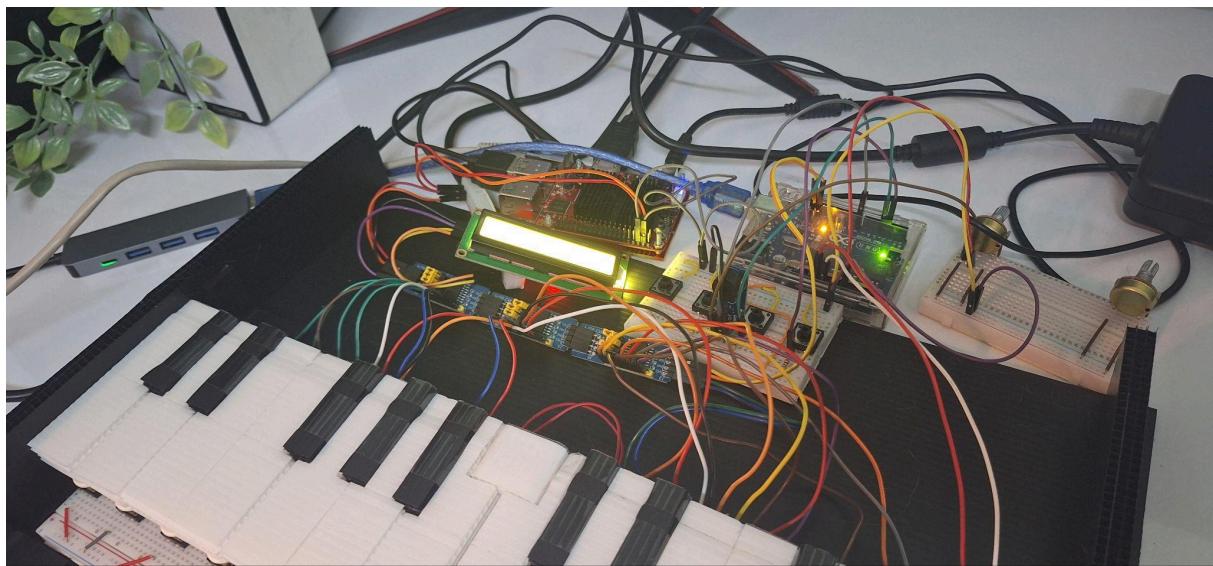
14/09/2567 - ขยายขนาด Layout ให้เป็นสอง Octave โดยใช้ปุ่มไป 24 ปุ่ม



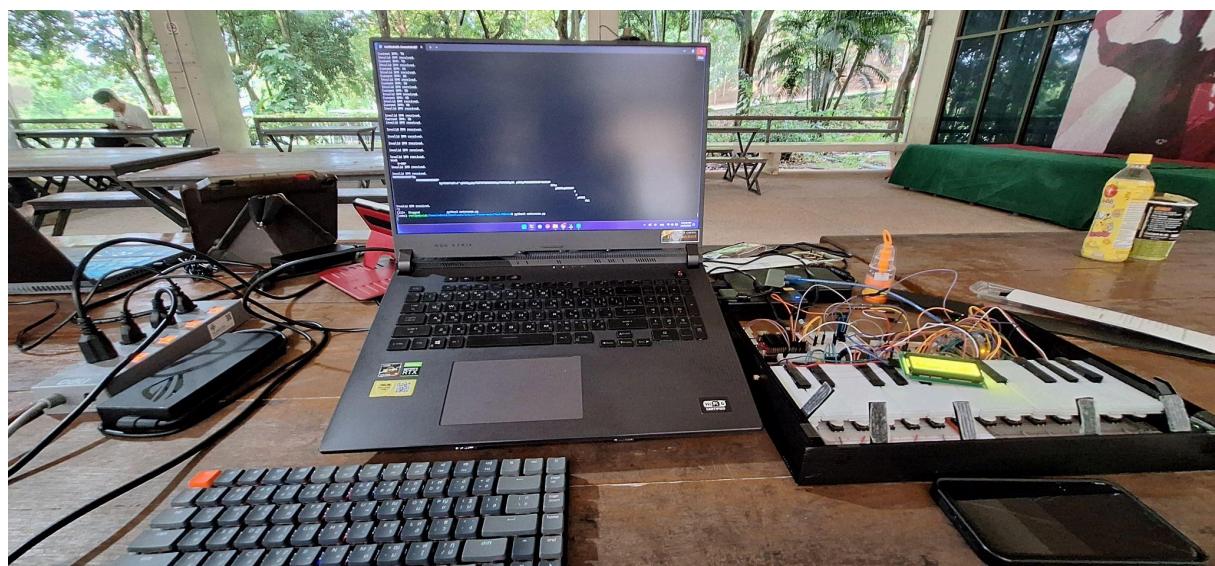
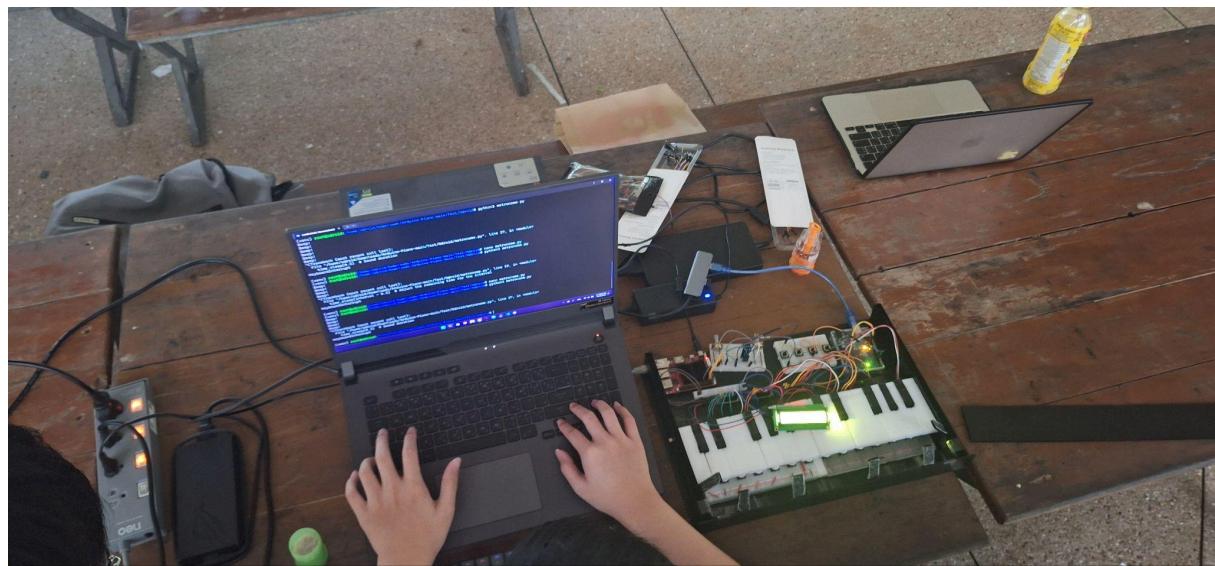
17/09/2567 - ทดลองการส่งและแปลงค่าสัญญาณ Serial Monitor เป็น MIDI และทดลอง Odroid



22/09/2567 - ประกอบ Model โดยใช้ Future board และทดลองการส่งค่าไปให้จอ LCD



24/09/2567 - ย้ายระบบ Metronome มาไว้ใน Oroid โดยโปรแกรมผ่าน SSH ด้วยสาย LAN



29/09/2567 - เก็บรายละเอียดและบันทึกผลงานรายงานให้ครับ



ตารางสำหรับการอ้างอิงความแม่นยำของโน้ตที่ได้

Note Frequency Chart

This is the note frequency chart that I find most useful.

Note	Oct -1	Oct 0	Oct 1	Oct 2	Oct 3	Oct 4	Oct 5	Oct 6	Oct 7	Oct 8	Oct 9	Oct 10
C	8.18	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01	8372.02	16744.04
C#/Db	8.66	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92	8869.84	17739.69
D	9.18	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64	9397.27	18794.54
D#/Eb	9.72	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03	9956.06	19912.13
E	10.30	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04	10548.08	21096.16
F	10.91	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65	11175.30	22350.61
F#/Gb	11.56	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91	11839.82	23679.64
G	12.25	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93	12543.85	25087.71
G#/Ab	12.98	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88	13289.75	26579.50
A	13.75	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00	14080.00	28160.00
A#/Bb	14.57	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62	14917.24	29834.48
B	15.43	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13	15804.27	31608.53