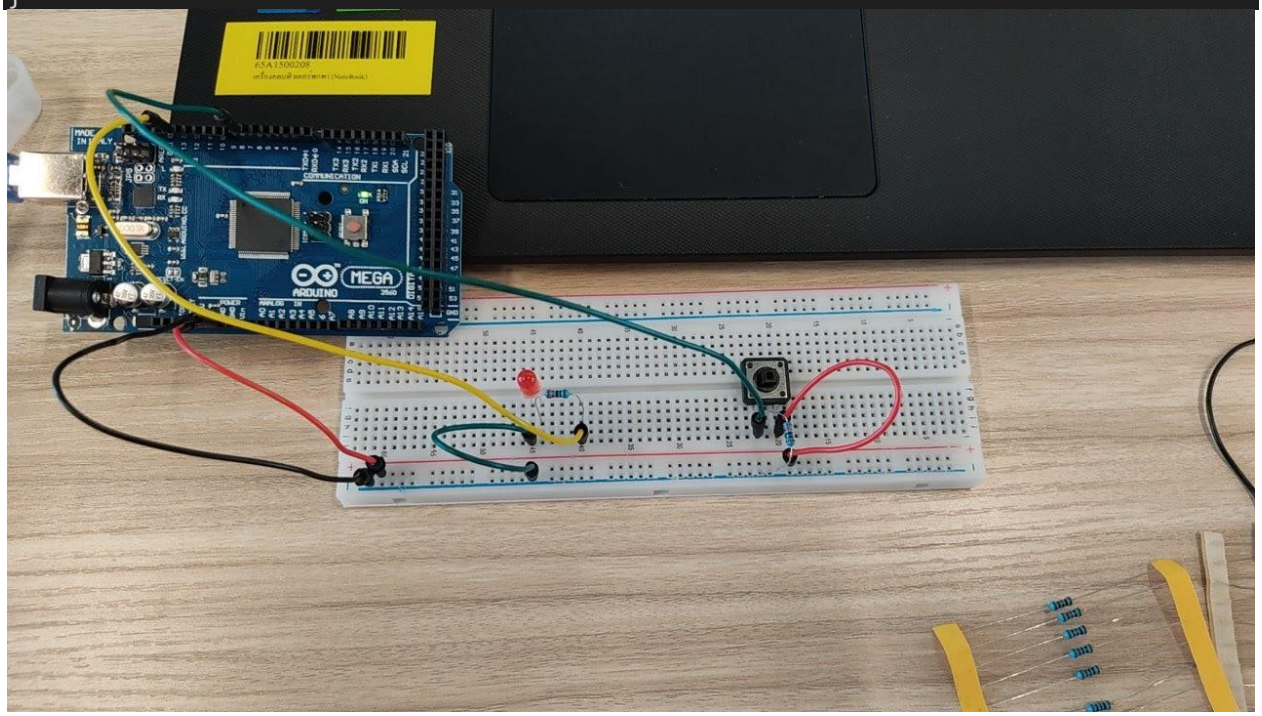
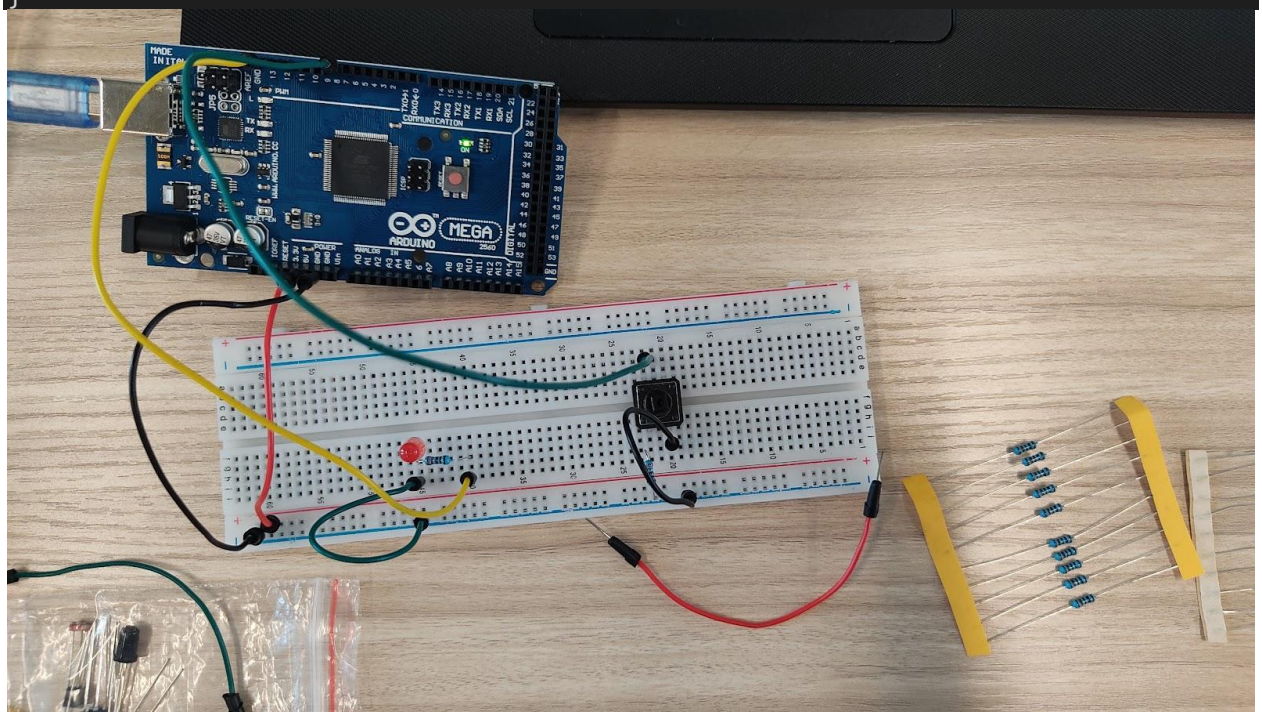
Exercise 1: Pull-down resistor for Digital Input

```
#define BUTTON 7 // the input pin
#define LED 13   // pin for LED
int val = 0;
// used to store the state of the input pin

void setup()
{
    pinMode(BUTTON, INPUT);
    pinMode(LED, OUTPUT);
}

void loop()
{
    val = digitalRead(BUTTON);
    if (val == HIGH)
    {
        digitalWrite(LED, HIGH);
    }
    else
    {
        digitalWrite(LED, LOW);
    }
}
```



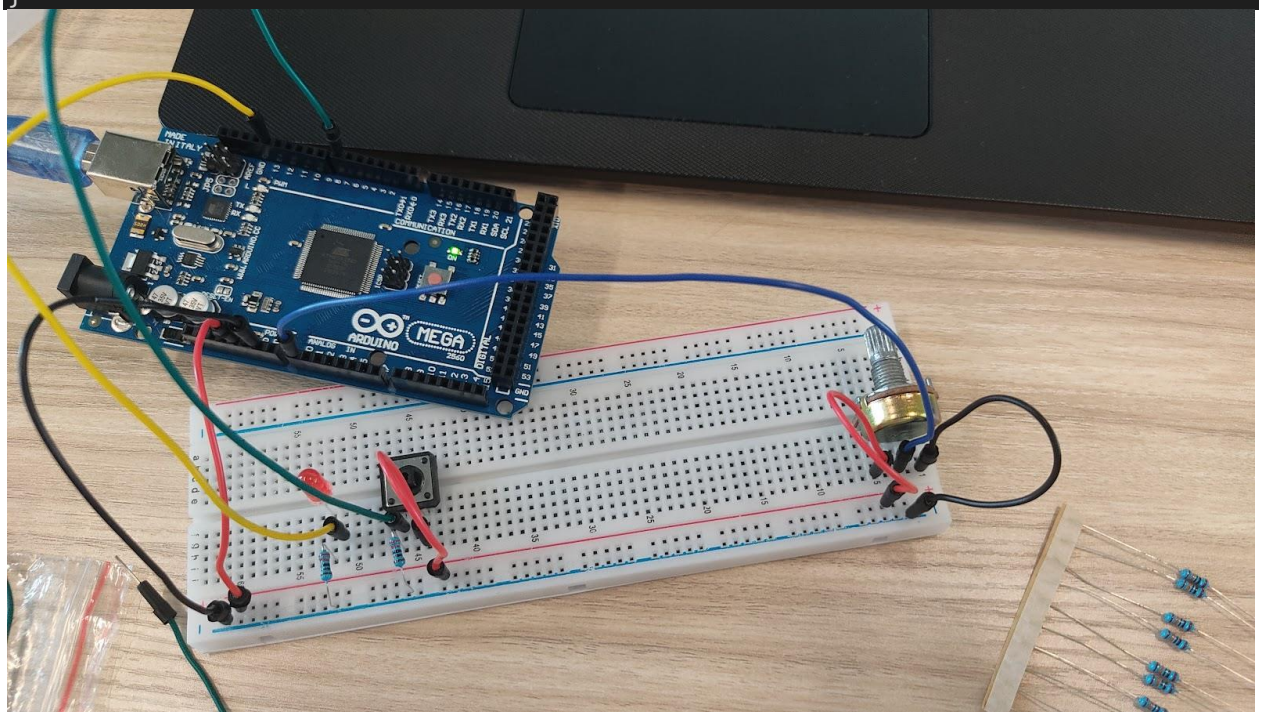
Exercise 2: Pull-up resistor for Digital Input

```
#define BUTTON 7 // the input pin
#define LED 13   // pin for LED
int val = 0;
// used to store the state of the input pin

void setup()
{
    pinMode(BUTTON, INPUT);
    pinMode(LED, OUTPUT);
}

void loop()
{
    val = digitalRead(BUTTON);
    if (val == LOW)
    {
        digitalWrite(LED, HIGH);
    }
    else
    {
        digitalWrite(LED, LOW);
    }
}
```



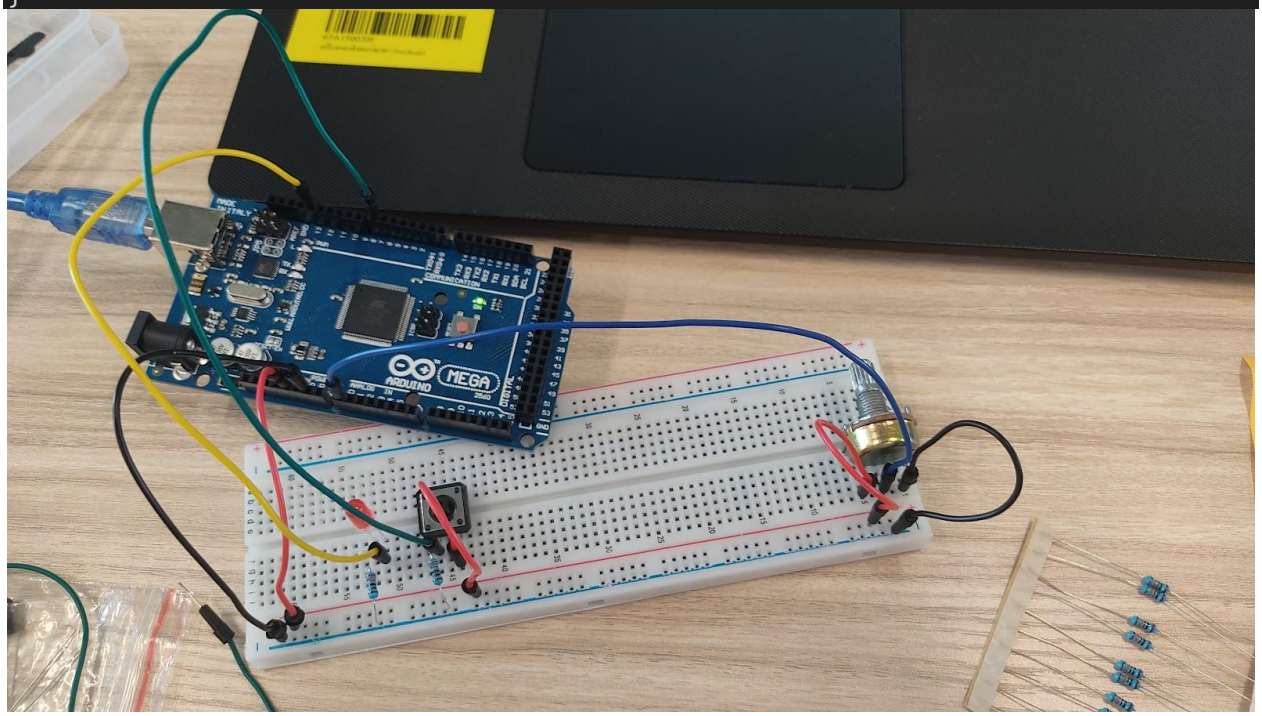
Exercise3: Digital Input and PWM

```
#define LED 13
#define DELAY_TIME 100
void setup()
{
    pinMode(LED, OUTPUT);
}

void loop()
{
    analogWrite(LED, 0);    delay(DELAY_TIME);
    analogWrite(LED, 20);   delay(DELAY_TIME);
    analogWrite(LED, 40);   delay(DELAY_TIME);
    analogWrite(LED, 60);   delay(DELAY_TIME);
    analogWrite(LED, 80);   delay(DELAY_TIME);
    analogWrite(LED, 100);  delay(DELAY_TIME);
    analogWrite(LED, 120);  delay(DELAY_TIME);
    analogWrite(LED, 140);  delay(DELAY_TIME);
    analogWrite(LED, 160);  delay(DELAY_TIME);
    analogWrite(LED, 180);  delay(DELAY_TIME);
    analogWrite(LED, 200);  delay(DELAY_TIME);
    analogWrite(LED, 220);  delay(DELAY_TIME);
    analogWrite(LED, 240);  delay(DELAY_TIME);
    analogWrite(LED, 255);  delay(DELAY_TIME);
}
```



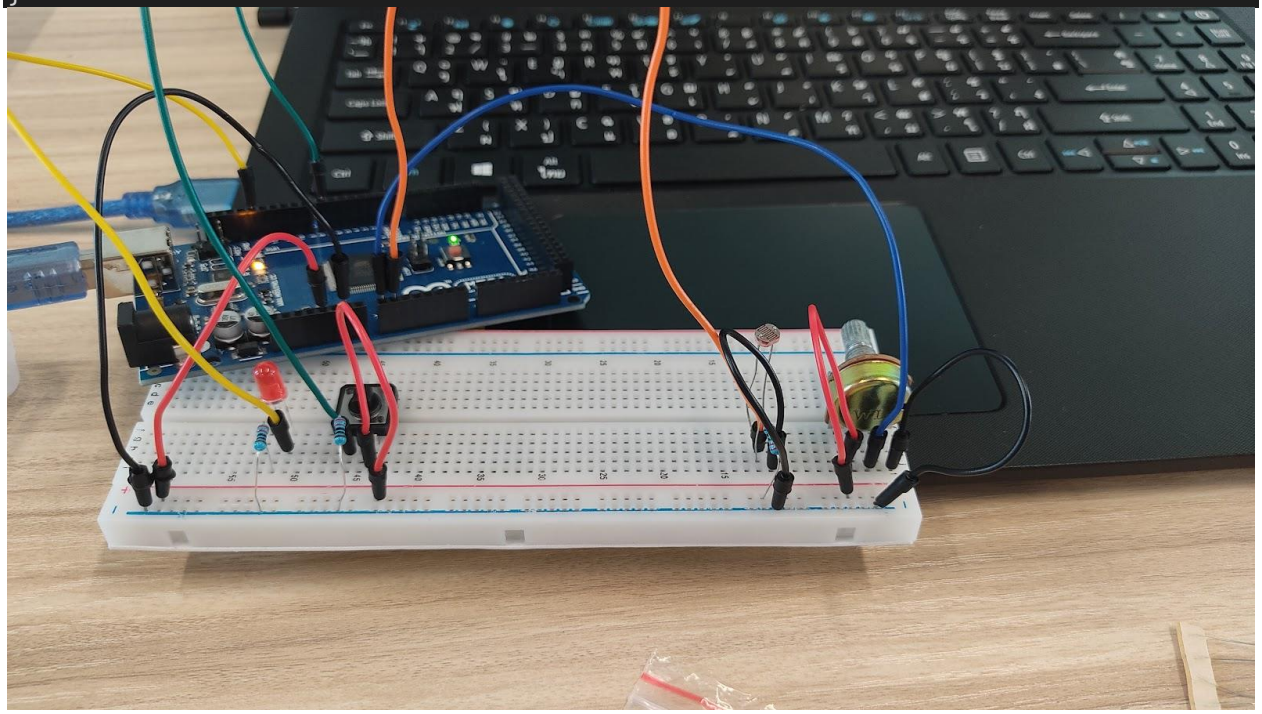
Exercise 4: Analog Input and PWM

```
#define LED 13
#define POTENTIOMETER A0
#define DELAY_TIME 100

void setup()
{
    pinMode(LED, OUTPUT);
    pinMode(POTENTIOMETER, INPUT);
}

void loop()
{
    int val = analogRead(POTENTIOMETER);
    val = map(val, 0, 1023, 0, 255);
    analogWrite(LED, val);
}
```



Exercise 5: Light Sensor

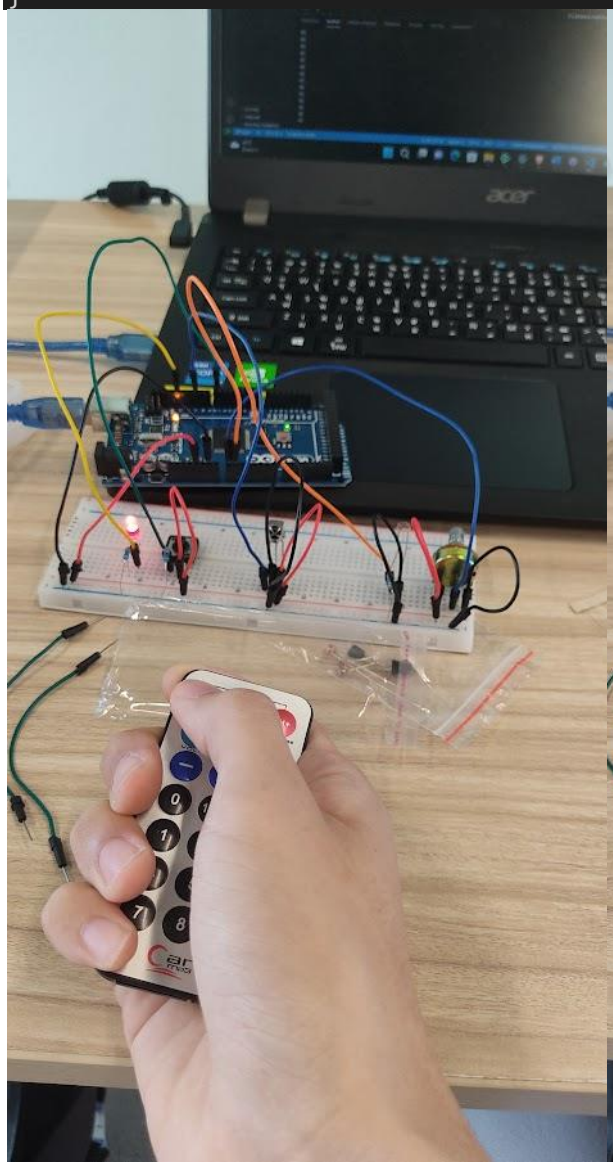
```
#define LED 13
#define LDR A1
#define DELAY_TIME 100

void setup()
{
    Serial.begin(115200);
    pinMode(LED, OUTPUT);
    pinMode(LDR, INPUT);
}

void loop()
{
    int val = analogRead(LDR);
    Serial.println(val);
    if (val >= 925) digitalWrite(LED, HIGH);
    else digitalWrite(LED, LOW);
}
```

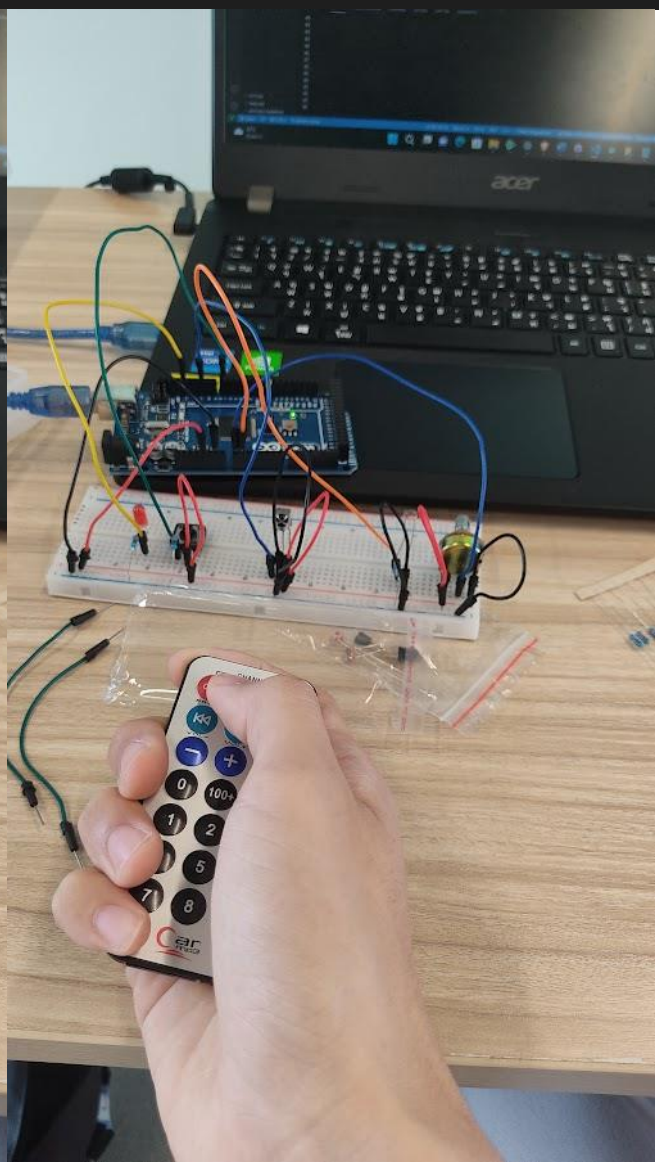


Exercise 6: IR receiver Circuit

```
/*
remote value that come with kit
CHANNEL
45 46 47
44 40 43
07 15  9
16 19  D
 C 18 5E
 8 1C 5A
42 52 4A
LOGO
*/

#include <IRremote.h>
#define IR_PIN 10
#define LED 13
#define DELAY_TIME 500

#define POWER 0x45
#define STOP 0x46

IRrecv IR; // Declare the name to use IR receiver
void setup()
{
    Serial.begin(115200);
    IR.begin(IR_PIN); // Begin to use IR receiver
    pinMode(LED, OUTPUT);
}
void loop()
{
    if (IR.decode())
    { // check if the button of IR remote pushed or not
        unsigned int val = IR.decodedIRData.command;
        Serial.println(val, HEX); // check the value when you push each button.
                                  // LED control code is here later.
        switch (val)
        {
        case POWER:
            digitalWrite(LED, HIGH);
            break;

        case STOP:
            digitalWrite(LED, LOW);
```

```
            break;

        default:
            break;
        }

        IR.resume(); // Prepare to receive the next command. Use this with
decode.
    }
    delay(DELAY_TIME);
}
```





Advanced Exercise 2: Buzzer

```
#define BUZZER 2
#include "pitches.h"

// notes in the melody:
int melody[] = {
    NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
    4, 8, 8, 4, 4, 4, 4, 4};

void setup()
{
    Serial.begin(115200);
    // iterate over the notes of the melody:
    for (int thisNote = 0; thisNote < sizeof(melody) / 2; thisNote++)
    {

        // to calculate the note duration, take one second divided by the note
type.

        // e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

        int noteDuration = 1000 / noteDurations[thisNote];

        tone(BUZZER, melody[thisNote], noteDuration);

        // to distinguish the notes, set a minimum time between them.

        // the note's duration + 30% seems to work well:

        int pauseBetweenNotes = noteDuration * 1.30;

        delay(pauseBetweenNotes);

        // stop the tone playing:

        noTone(BUZZER);
    }
}

void loop() {}
```

```
//! Pitches.h
/*********************************

 * Public Constants

 *********************************/

#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
```

```
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
```

```
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```