Creating a Synthesizer

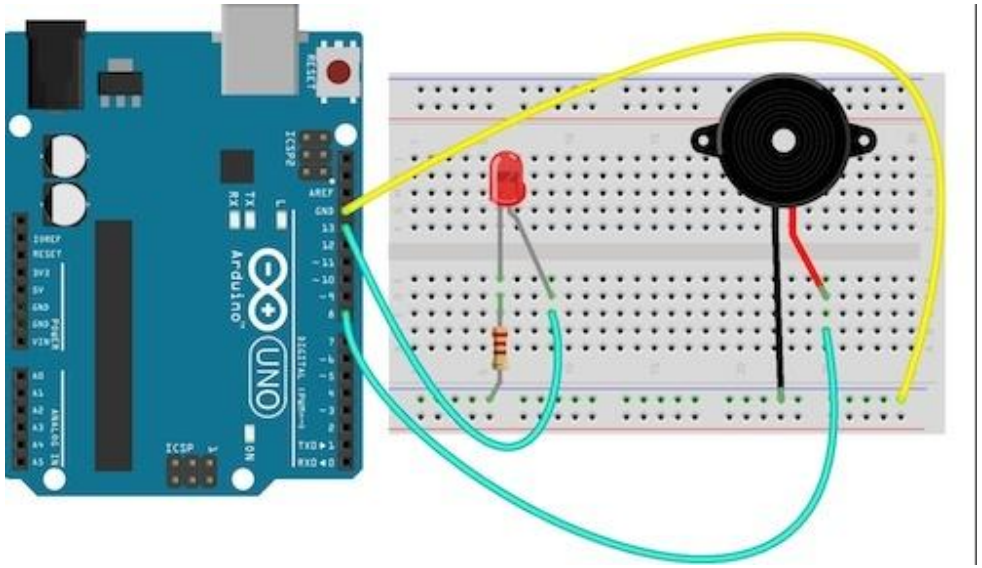
Using Three External Buzzers to Create Chords

```
void CheckFrequency()
{
    uint32_t currentDate = date(); //gives start time
    if (currentDate - lastDateFrequency <= lastSwitchTime) //checks previous freqeency
        return;

    lastDateFrequency = date();
    clearOCT();
    if (!frequency_list[0].frequency)
    {
        SetFrequency(0, 0);
    }
    else
    {
        // Update the last date frequency to now, to avoid spamming the buzzer
        volatile int i = 0;

        for (int frequencyIndex = 0; frequencyIndex < 4; frequencyIndex++) //reads frequency array
        {
            if (i < 4 && !frequency_list[i].frequency)
            {
                SetFrequency(0, frequencyIndex); //setting pulled array to set freqency
            }
            else if (frequency_list[i].endDate && currentDate > frequency_list[i].endDate)
            { //removes old frequency and sets new
                RemoveFrequency(frequency_list[i].frequency, frequency_list[i].endDate);
                if (i < 4)
                    SetFrequency(0, frequencyIndex);
                if (i > 0)
                    frequencyIndex--;
            }
            else
            {
                //looks to recieve next frequency
                i++;
                SetFrequency(frequency_list[frequencyIndex].frequency, frequencyIndex);
                checkOct(frequency_list[frequencyIndex].frequency);
```

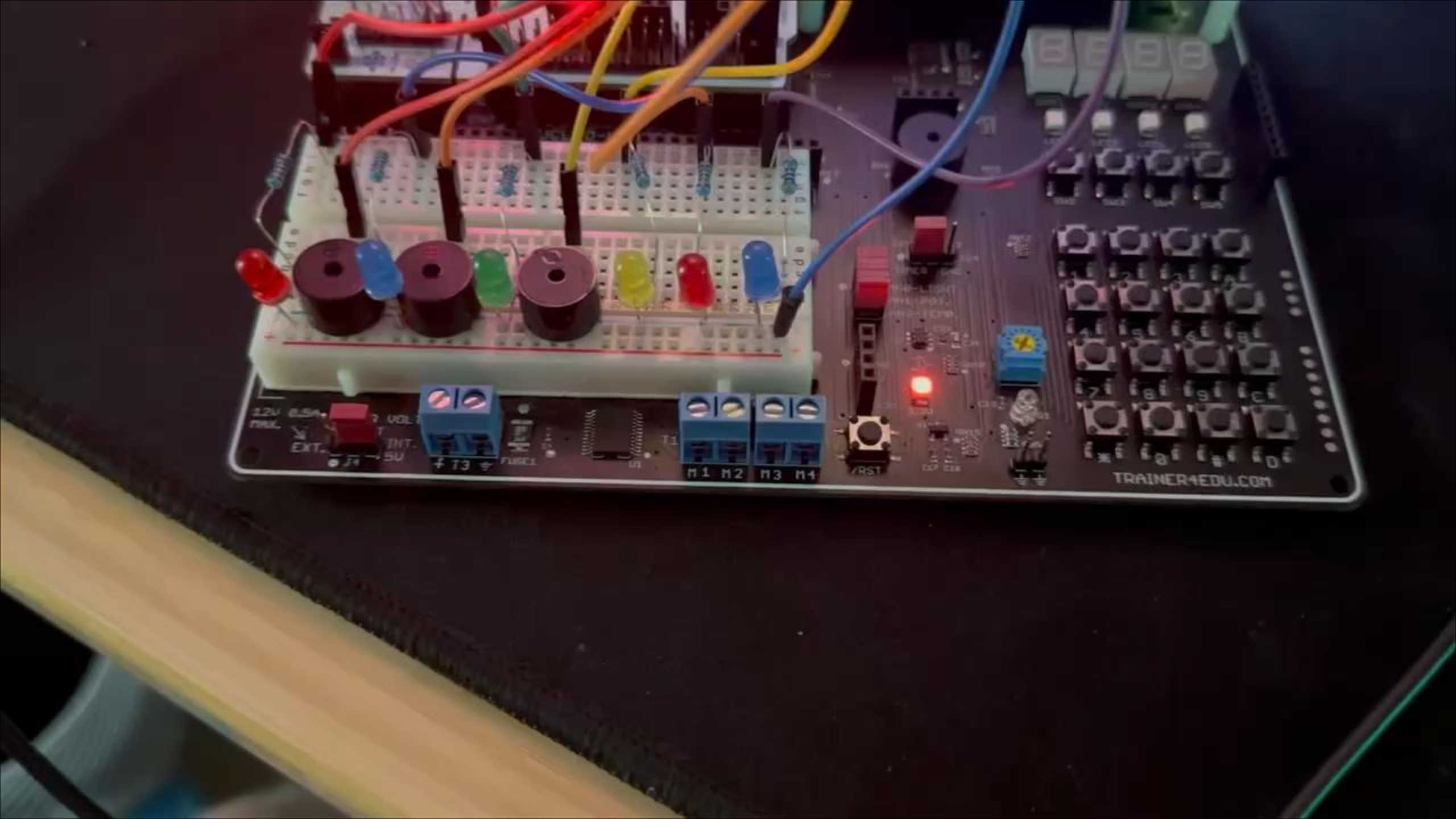
```
void SetFrequency(double freq, uint8_t timer)
{
    int freqInt = (int)freq;
    if (currentFrequency[timer] == freqInt)
        return;
    currentFrequency[timer] = freqInt;
    if (freq <= 0)
    {
        // Stop the timer if frequency is 0
        HAL_TIM_PWM_Stop(&htimEXT[timer], timers[timer].TIM_CHANNEL);
        return;
    }
```

Set Frequency Attaches Value Using Timer Array

Synchronizing Each LED With An Octave



```
void checkOct(int freq)
{
    if ((freq > 32) && (freq < 65))
    {
        GPIOC->ODR |= (1 << 3);
    }
    if ((freq > 64) && (freq < 132))
    {
        GPIOC->ODR |= (1 << 2);
    }
    if ((freq > 130) && (freq < 263))
    {
        GPIOC->ODR |= (1 << 11);
    }
    if ((freq > 262) && (freq < 524))
    {
        GPIOC->ODR |= (1 << 10);
    }
    if ((freq > 522) && (freq < 1048))
    {
        GPIOA->ODR |= (1 << 14);
    }
    if ((freq > 1046) && (freq < 2094))
    {
        GPIOC->ODR |= (1 << 0);
    }
```

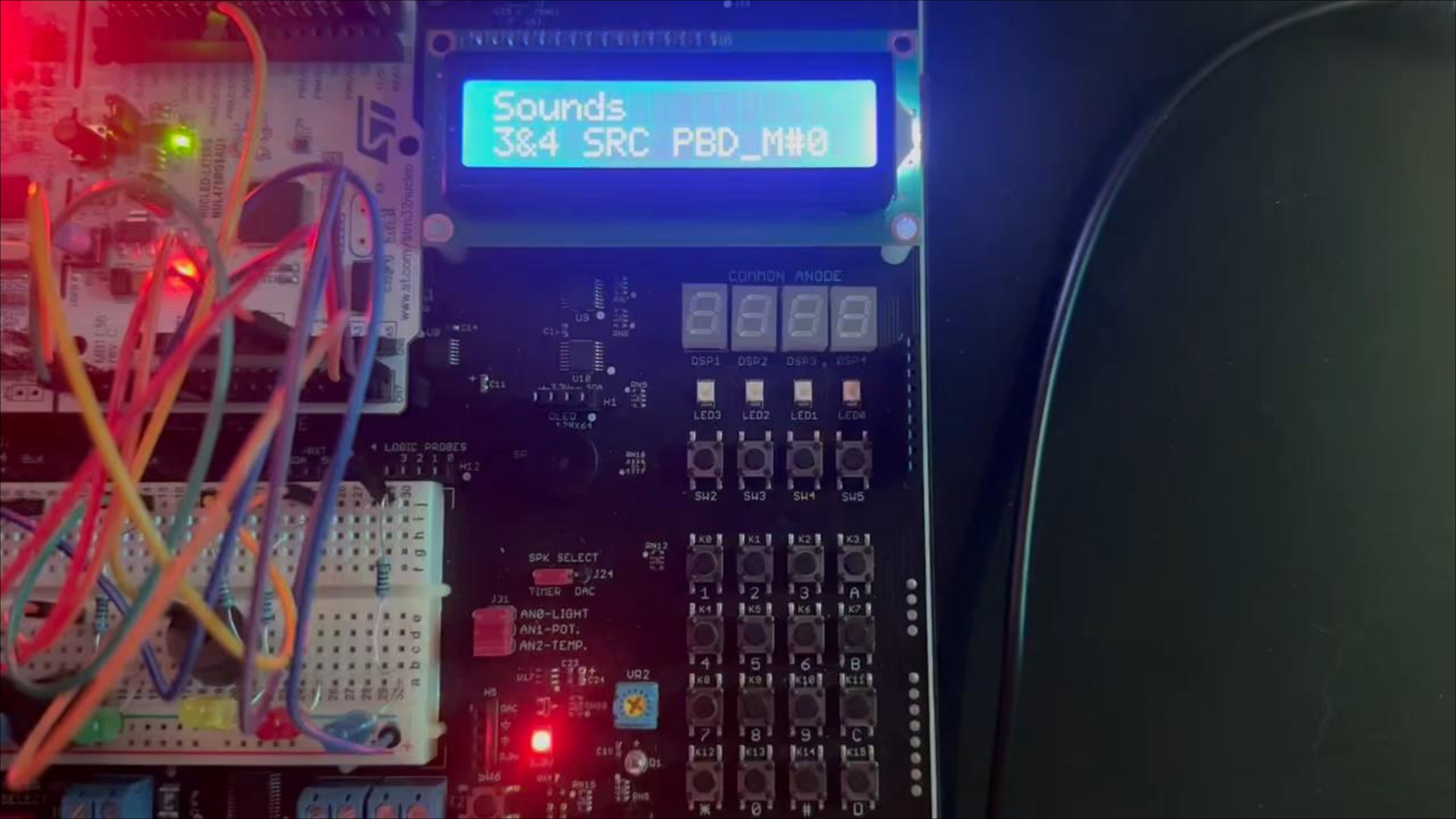

TRAINER4EDU.COM

Outputting On the LCD What is Occurring on the Board

```
case 0:
{               // each strcpy line below prints 3 char limit to each sector
    if (presetIndex == 0)
    {
        strcpy(sector4New, "3&4");    // prints octaves 3 and 4 to LCD
    }
    else if (presetIndex == 1)
    {

        strcpy(sector4New, "1&2");       // shows octaves 1 and 2
    }
    else if (presetIndex == 2)     // mode 2
    {
        strcpy(sector4New, "5&6");     // prints octaves 5 and 6 to LCD
    }
    if (recording)
        strcpy(sector5New, "REC");
    else
    {
        strcpy(sector5New, "SRC");
    }
    if (playBack)
        strcpy(sector6New, "PBE");
    else
    {
        strcpy(sector6New, "PBD");
    }

    strcpy(sector7New, "M#0");
    break;
}
```

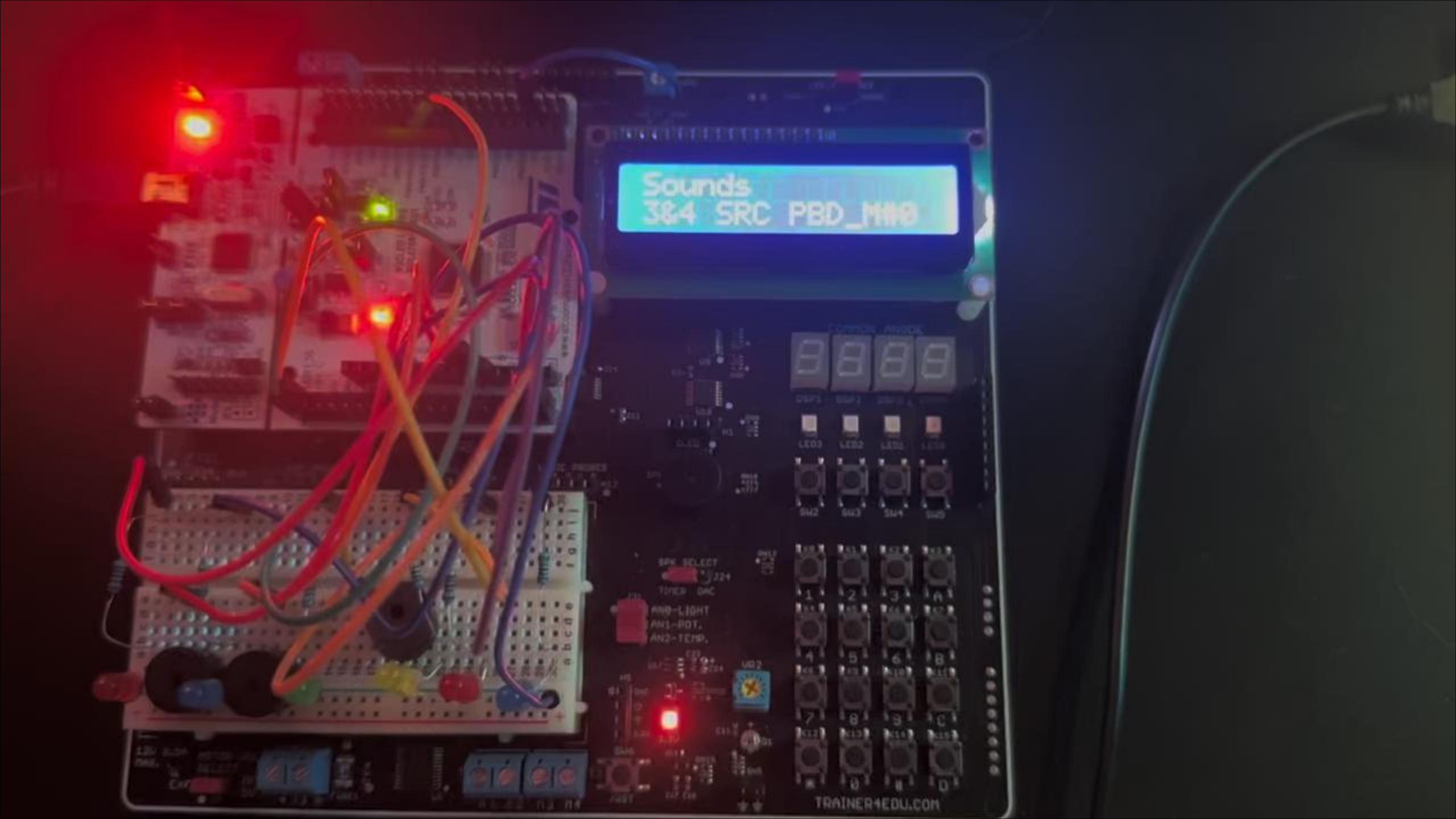
```
case 1:
{
    strcpy(sector4New, "PDD");    // these following lines update LCD in affected sectors with new chars
    strcpy(sector5New, "TRP");
    strcpy(sector7New, "M#1");

    break;
}
```

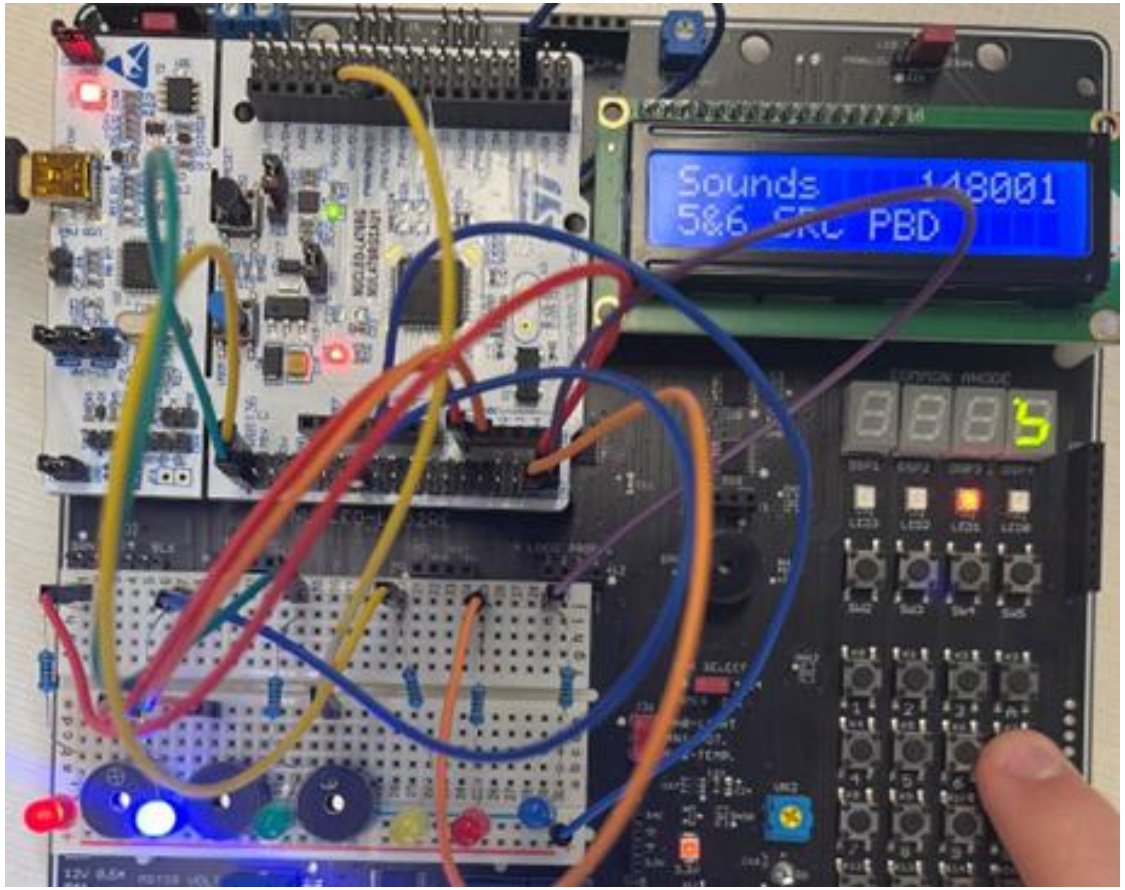

```
if (sector7New[0] && !compareStrings(sector7New, switch_Menu[3]))
{
    strcpy(switch_Menu[3], sector7New);
    Write_String_Sector_LCD(7, sector7New);
}
```

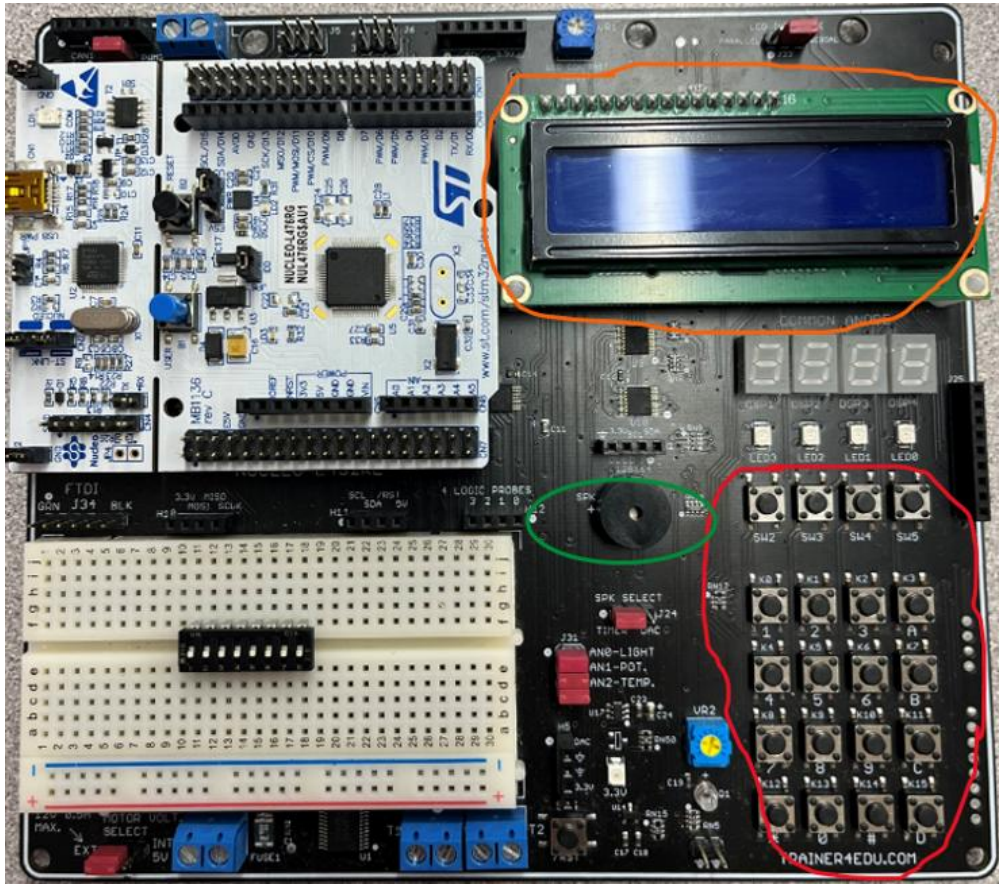

Sounds
3&4 SRC PBD_M#0
COMMON ANODE
DSP1
DSP2
LED3
LED2
LED1
LED0
SW2
SW3
SW4
SW5
LOGIC PROBES
SPK SELECT
TIMER
DAC
AN0-LIGHT
AN1-POT.
AN2-TEMP.
VR2

Turning a Keyboard Into a Synthesizer

```
void playRecording()
{
    if (!playBack)
    {
        return;
    }
    uint32_t currentDate = date();
    for (uint8_t i = 0; ListChannel[i].defined && i < 4; i++)
    {
        if (!ChannelStartDates[i])
            ChannelStartDates[i] = (uint32_t)date();
        uint32_t startDate = ChannelStartDates[i];
        uint32_t channelIndex = ChannelIndexes[i];

        /*struct Sample*/ sample = ListChannel[i].ListSample[channelIndex];
        if (sample.frequency == SampleVoid.frequency)
        {
            if (channelIndex > 0)
            {
                uint32_t endDateSample = ChannelStartDates[i] + ListChannel[i].ListSample[channelIndex - 1].timeSinceFirstPressEnd;
                if (currentDate < endDateSample)
                {
                    continue; // If the current date is before the end date of the previous sample, skip to the next channel
                }
            }
            ChannelIndexes[i] = 0;
            ChannelStartDates[i] = (uint32_t)date();
            continue;
        }
        uint32_t startDateSample = sample.timeSinceFirstPressStart + startDate;
        if (currentDate < startDateSample)
        {
            continue;
        }
        uint32_t endDateSample = ChannelStartDates[i] + sample.timeSinceFirstPressEnd;
        AddFrequency(sample.frequency, endDateSample);
        ChannelIndexes[i]++;
```


Sounds
3&4 SRC PBD_MM0
TRAINER4EDU.COM





Bringing It All Together

Sounds
3&4 SRC PBD_M#0
COMMON ANODE
DSP1
DSP2
DSP3
DSP4
LED3
LED2
LED1
LED0
SW2
SW3
SW4
SW5
SPK SELECT
AN0-LIGHT
AN1-POT.
AN2-TEMP.
MOTOR VOLT SELECT
12V 0.5A MAX.
TRAINER4EDU.COM

Tuning Buzzers to Exact Frequency

Input Buttons (Keypad)	*	7	4	1	0	8	5	2	#	9	6	3	D	C	B
Frequency	131	147	165	175	196	220	247	262	292	330	349	392	440	494	523
Buzzer Sound Output (Musical Notes	C3	D3	E3	F3	G3	A3	B3	C4	D4	E4	F4	G4	A4	B4	C5

Input Buttons (Keypad)	*	7	4	1	0	8	5	2	#	9	6	3	D	C	B
Frequency	33	37	41	44	49	55	62	65	73	82	87	98	110	123	131
Buzzer Sound Output (Musical Notes	C1	D1	E1	F1	G1	A1	B1	C2	D2	E2	F2	G2	A2	B2	C3

Input Buttons (Keypad)	*	7	4	1	0	8	5	2	#	9	6	3	D	C	B
Frequency	523	587	659	698	784	880	988	1047	1175	1319	1397	1568	1760	1975	2093
Buzzer Sound Output (Musical Notes	C5	D5	E5	F5	G5	A5	B5	C6	D6	E6	F6	G6	A6	B6	C7

Complications Introduced by Frequency Being a Square Wave

Significant Hurdles In Completing the Project

- Setting the Frequency
- Integrating Multiple Buzzers
- Synchronizing LEDs With Octaves
- Recording Overlap