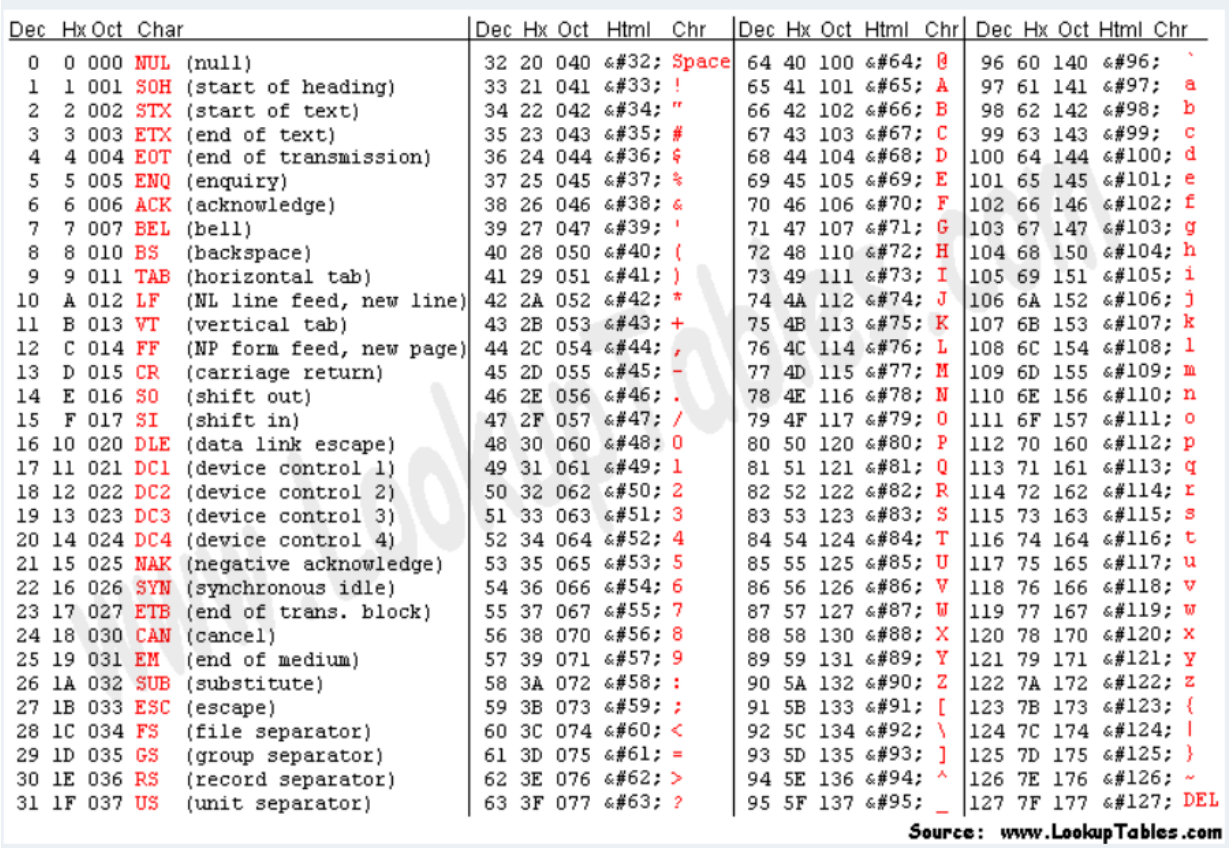
CODE SHARING DOCUMENT

GPT C CODE FOR DSP CHIP: TEST SERIAL READ

```
#include <stdio.h>

#define MAX_STRING_LENGTH 100

int main(void) {
    char serialBuffer[MAX_STRING_LENGTH];

    // Open the serial port for reading
    FILE *serialPort = fopen("COM1", "r");
    if (serialPort == NULL) {
        printf("Error opening serial port.\n");
        return 1;
    }

    while (1) {
        // Read a string from the serial port
        if (fgets(serialBuffer, MAX_STRING_LENGTH, serialPort) != NULL) {
            // Process the received string
            printf("Received string: %s", serialBuffer);
        }
    }

    // Close the serial port
    fclose(serialPort);

    return 0;
}
```

1st character: type effect state = 1, go to state 2
2nd character: lpf multipler
3rd character:

5h character



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Pin value conversion table

ASCII (C Code)	Actual Value (Median)
Space	6
!	18
“	30
#	42
$	
%	
&	
‘	
(	
)	
*	

+	
,	
/	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
:	
;	
<	
=	
>	
?	
@	
A	
B	
C	
D	
E	
F	
G	

H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
[	
\	
]	
^	
_	
`	
a	
b	

c	
d	
e	
f	
g	
h	
i	
j	
k	
l	
m	
n	
o	
p	
q	
r	
s	
t	
u	
v	
w	
x	
y	
z	

Range
0 - 11
11 - 22
22 - 33
33 - 44
44 - 55
55 - 66
66 - 77
77 - 88
88 - 99
99 - 110
110 - 121
121 - 132
132 - 143
143 - 154
154 - 165
165 - 176
176 - 187
187 - 198
198 - 209
209 - 220
220 - 231
231 - 242
242 - 253
253 - 264
264 - 275
275 - 286
286 - 297
297 - 308
308 - 319
319 - 330
330 - 341
341 - 352
352 - 363
363 - 374
374 - 385
385 - 396
396 - 407
407 - 418
418 - 429
429 - 440
440 - 451
451 - 462

| 462 - 473 |
| 473 - 484 |
| 484 - 495 |
| 495 - 506 |
| 506 - 517 |
| 517 - 528 |
| 528 - 539 |
| 539 - 550 |
| 550 - 561 |
| 561 - 572 |
| 572 - 583 |
| 583 - 594 |
| 594 - 605 |
| 605 - 616 |
| 616 - 627 |
| 627 - 638 |
| 638 - 649 |
| 649 - 660 |
| 660 - 671 |
| 671 - 682 |
| 682 - 693 |
| 693 - 704 |
| 704 - 715 |
| 715 - 726 |
| 726 - 737 |
| 737 - 748 |
| 748 - 759 |
| 759 - 770 |
| 770 - 781 |
| 781 - 792 |
| 792 - 803 |
| 803 - 814 |
| 814 - 825 |
| 825 - 836 |
| 836 - 847 |
| 847 - 858 |
| 858 - 869 |
| 869 - 880 |
| 880 - 891 |
| 891 - 902 |
| 902 - 913 |
| 913 - 924 |
| 924 - 935 |
| 935 - 946 |
| 946 - 957 |
| 957 - 968 |
| 968 - 979 |

```
| 979 - 990 |
| 990 - 1001|
| 1001 - 1012|
| 1012 - 1023|
```

Sensorvalue = input to function, character = output
If (0 <=sensorvalue < 11) then
Character = chr(33);
end

serial.print(char_2)

Effect value conversion table(print at the front of everything)

Character value	UART value (Decimal)
1	49
2	50
3	51
4	52
5	53

If (effectname = 1){
char_1 = '!' (or char(n) for any dec number n)
} …

serial.pring(char_1);

0,0,0,0
1,0,0,0
5,0,0,0
20,0,0,0

```
if value >= 0 and value < 11:
    print(" ")
elif value >= 11 and value < 22:
    print("!")
elif value >= 22 and value < 33:
    print("")
elif value >= 33 and value < 44:
    print("#")
elif value >= 44 and value < 55:
    print("$")
```

```
elif value >= 55 and value < 66:
    print("%")
elif value >= 66 and value < 77:
    print("&")
elif value >= 77 and value < 88:
    print("'")
elif value >= 88 and value < 99:
    print("(")
elif value >= 99 and value < 110:
    print(")")
elif value >= 110 and value < 121:
    print("*")
elif value >= 121 and value < 132:
    print("+")
elif value >= 132 and value < 143:
    print(",")
elif value >= 143 and value < 154:
    print("-")
elif value >= 154 and value < 165:
    print(".")
elif value >= 165 and value < 176:
    print("/")
elif value >= 176 and value < 187:
    print("0")
elif value >= 187 and value < 198:
    print("1")
elif value >= 198 and value < 209:
    print("2")
elif value >= 209 and value < 220:
    print("3")
elif value >= 220 and value < 231:
    print("4")
elif value >= 231 and value < 242:
    print("5")
elif value >= 242 and value < 253:
    print("6")
elif value >= 253 and value < 264:
    print("7")
elif value >= 264 and value < 275:
    print("8")
elif value >= 275 and value < 286:
    print("9")
elif value >= 286 and value < 297:
    print(":")
```

```
elif value >= 297 and value < 308:
    print(";")
elif value >= 308 and value < 319:
    print("<")
elif value >= 319 and value < 330:
    print("=")
elif value >= 330 and value < 341:
    print(">")
elif value >= 341 and value < 352:
    print("?")
elif value >= 352 and value < 363:
    print("@")
elif value >= 363 and value < 374:
    print("A")
elif value >= 374 and value < 385:
    print("B")
elif value >= 385 and value < 396:
    print("C")
elif value >= 396 and value < 407:
    print("D")
elif value >= 407 and value < 418:
    print("E")
elif value >= 418 and value < 429:
    print("F")
elif value >= 429 and value < 440:
    print("G")
elif value >= 440 and value < 451:
    print("H")
elif value >= 451 and value < 462:
    print("I")
elif value >= 462 and value < 473:
    print("J")
elif value >= 473 and value < 484:
    print("K")
elif value >= 484 and value < 495:
    print("L")
elif value >= 495 and value < 506:
    print("M")
elif value >= 506 and value < 517:
    print("N")
elif value >= 517 and value < 528:
    print("O")
elif value >= 528 and value < 539:
    print("P")
```

```
elif value >= 539 and value < 550:
    print("Q")
elif value >= 550 and value < 561:
    print("R")
elif value >= 561 and value < 572:
    print("S")
elif value >= 572 and value < 583:
    print("T")
elif value >= 583 and value < 594:
    print("U")
elif value >= 594 and value < 605:
    print("V")
elif value >= 605 and value < 616:
    print("W")
elif value >= 616 and value < 627:
    print("X")
elif value >= 627 and value < 638:
    print("Y")
elif value >= 638 and value <= 1023:
    print("Z")
```

4/22 char map coordination:
I'm going to delegate "~" as the mute identifier for serial comms. It will print that individual character on a new line when the mute button is selected. Note that "}" isn't being used currently.

Note: print ~ x5

Here's my code:

```
#parallelv8
#Updated UI! Vertical Large sliders, custom labels
import tkinter as tk
from tkinter import ttk
import threading
import serial
import time
import re

char_map = {
    range(0, 11): " ",
    range(11, 22): "!",
    range(22, 33): "",
    range(33, 44): "#",
    range(44, 55): "$",
    range(55, 66): "%",
    range(66, 77): "&",
    range(77, 88): "",
    range(88, 99): "(",
    range(99, 110): ")",
    range(110, 121): "*",
    range(121, 132): "+",
    range(132, 143): ",",
    range(143, 154): "-",
    range(154, 165): ".",
    range(165, 176): "/",
    range(176, 187): "0",
    range(187, 198): "1",
    range(198, 209): "2",
    range(209, 220): "3",
    range(220, 231): "4",
    range(231, 242): "5",
    range(242, 253): "6",
    range(253, 264): "7",
    range(264, 275): "8",
```

range(275, 286): "9",
range(286, 297): ":",
range(297, 308): ";",
range(308, 319): "<",
range(319, 330): "=",
range(330, 341): ">",
range(341, 352): "?",
range(352, 363): "@",
range(363, 374): "A",
range(374, 385): "B",
range(385, 396): "C",
range(396, 407): "D",
range(407, 418): "E",
range(418, 429): "F",
range(429, 440): "G",
range(440, 451): "H",
range(451, 462): "I",
range(462, 473): "J",
range(473, 484): "K",
range(484, 495): "L",
range(495, 506): "M",
range(506, 517): "N",
range(517, 528): "O",
range(528, 539): "P",
range(539, 550): "Q",
range(550, 561): "R",
range(561, 572): "S",
range(572, 583): "T",
range(583, 594): "U",
range(594, 605): "V",
range(605, 616): "W",
range(616, 627): "X",
range(627, 638): "Y",
range(638, 649): "Z",
range(649, 660): "[",
range(660, 671): "\\",
range(671, 682): "]",
range(682, 693): "^",
range(693, 704): "_",
range(704, 715): "`",
range(715, 726): "a",
range(726, 737): "b",
range(737, 748): "c",
range(748, 759): "d",

```
    range(759, 770): "e",
    range(770, 781): "f",
    range(781, 792): "g",
    range(792, 803): "h",
    range(803, 814): "i",
    range(814, 825): "j",
    range(825, 836): "k",
    range(836, 847): "l",
    range(847, 858): "m",
    range(858, 869): "n",
    range(869, 880): "o",
    range(880, 891): "p",
    range(891, 902): "q",
    range(902, 913): "r",
    range(913, 924): "s",
    range(924, 935): "t",
    range(935, 946): "u",
    range(946, 957): "v",
    range(957, 968): "w",
    range(968, 979): "x",
    range(979, 990): "y",
    range(990, 1001): "z",
    range(1001, 1012): "{",
    range(1012, 1024): "|"
    #range(638, 649): "}",      //may replace with space, it's a little tricky? currently turns MUTE
OFF
    #range(638, 649): "~"       //used for turning MUTE ON
}

def update_value(sensor_index, value):
    global serial_thread_running
    global sensor_values
    global selected_tab_index

    if serial_thread_running:
        sensor_values[sensor_index] = int(value)
        ser.write(f"Slider Value {sensor_index + 1}:
{sensor_values[sensor_index]}\n".encode('utf-8'))
    else:
        sensor_values[sensor_index] = int(value)
        char_values = [next(char for range_, char in char_map.items() if value in range_) for value
in sensor_values]

        print(selected_tab_index, end="")
```

```
        print("".join(char_values))

def serial_reader():
    global serial_thread_running
    global sensor_values
    global notebook
    global selected_tab_index

    ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=1.0)

    while True:
        time.sleep(0.03)
        if serial_thread_running:
            ser.write("READ_SENSOR\n".encode('utf-8'))
            if ser.in_waiting > 0:
                response = ser.readline().decode('utf-8').rstrip()
                print("NANO:", response)
                pattern = r"\[(\d+),\s*(\d+),\s*(\d+),\s*(\d+)\]"
                match = re.search(pattern, response)

                if match:
                    sensor_values = [int(match.group(i)) for i in range(1, 5)]
                    selected_tab_index = notebook.index(notebook.select())
                    scales = slider_sets[selected_tab_index]
                    print(selected_tab_index, end="")
                    for i, value in enumerate(sensor_values):
                        scales[i].set(value)
                        for range_, char in char_map.items():
                            if value in range_:
                                print(char, end="")
                                break

def toggle_serial_thread():
    global serial_thread_running
    global ser

    serial_thread_running = not serial_thread_running
    if serial_thread_running:
        ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=1.0)
        start_button.config(text="Stop External Control")
    else:
        ser.close()
        start_button.config(text="Start External Control")
```

```
def create_sliders(tab, tab_index):
    scales = []
    frame = ttk.Frame(tab)
    frame.pack(fill='both', expand=True)
    for i in range(4):
        if tab_index == 1:  # For tab 2 (EQ)
            if i == 0:
                label_text = "Low"
            elif i == 1 or i == 2:
                label_text = "Mid"
            else:
                label_text = "High"
        else:
            label_text = f"Slider {i+1}"

        label = tk.Label(frame, text=label_text)
        label.grid(row=0, column=i, pady=5)  # Place label above the slider
        scale = tk.Scale(frame, from_=1023, to=0, orient=tk.VERTICAL, length=250,width=106,
command=lambda value, idx=i: update_value(idx, value))
        scale.grid(row=1, column=i, padx=20)  # Place scale below the label
        scales.append(scale)
    return scales




def on_tab_selected(event):
    global selected_tab_index

    selected_tab_index = notebook.index(notebook.select())
    print(selected_tab_index, end="")
    print("    ")

def muteToggle():
    global toggle_state
    if toggle_state:
        mute_button.config(text="Mute Off")
        print("\n}}}}}")
    else:
        mute_button.config(text="Mute On")
        print("\n~~~~~")
    toggle_state = not toggle_state

toggle_state = False
```

```
serial_thread_running = False
sensor_values = [0, 0, 0, 0]

root = tk.Tk()
root.title("Starter Gig Effects Unit")

notebook = ttk.Notebook(root)
notebook.pack(fill='both', expand=True)
notebook.bind("<<NotebookTabChanged>>", on_tab_selected)

for i, tab_name in enumerate(["Distortion", "EQ", "Tremolo"]):
    tab = ttk.Frame(notebook)
    notebook.add(tab, text=tab_name)

start_button = tk.Button(root, text="Start External Control", command=toggle_serial_thread)
start_button.pack(pady=10)

serial_thread = threading.Thread(target=serial_reader, daemon=True)
serial_thread.start()

mute_button = tk.Button(root, text="Mute Off", command=muteToggle)
mute_button.pack(pady=10)

slider_sets = {}
for i in range(3):
    slider_sets[i] = create_sliders(notebook.winfo_children()[i], i)

root.mainloop()
```

_____SAVE OF parallelv8 1:09pm 4/26/24

```
import tkinter as tk
from tkinter import ttk
import threading
import serial
import time
import re

char_map = {
    range(0, 11): " ",
    range(11, 22): "!",
    range(22, 33): "",
```

range(33, 44): "#",
range(44, 55): "$",
range(55, 66): "%",
range(66, 77): "&",
range(77, 88): "'",
range(88, 99): "(",
range(99, 110): ")",
range(110, 121): "*",
range(121, 132): "+",
range(132, 143): ",",
range(143, 154): "-",
range(154, 165): ".",
range(165, 176): "/",
range(176, 187): "0",
range(187, 198): "1",
range(198, 209): "2",
range(209, 220): "3",
range(220, 231): "4",
range(231, 242): "5",
range(242, 253): "6",
range(253, 264): "7",
range(264, 275): "8",
range(275, 286): "9",
range(286, 297): ":",
range(297, 308): ";",
range(308, 319): "<",
range(319, 330): "=",
range(330, 341): ">",
range(341, 352): "?",
range(352, 363): "@",
range(363, 374): "A",
range(374, 385): "B",
range(385, 396): "C",
range(396, 407): "D",
range(407, 418): "E",
range(418, 429): "F",
range(429, 440): "G",
range(440, 451): "H",
range(451, 462): "I",

range(462, 473): "J",
range(473, 484): "K",
range(484, 495): "L",
range(495, 506): "M",
range(506, 517): "N",
range(517, 528): "O",
range(528, 539): "P",
range(539, 550): "Q",
range(550, 561): "R",
range(561, 572): "S",
range(572, 583): "T",
range(583, 594): "U",
range(594, 605): "V",
range(605, 616): "W",
range(616, 627): "X",
range(627, 638): "Y",
range(638, 649): "Z",
range(649, 660): "[",
range(660, 671): "\\",
range(671, 682): "]",
range(682, 693): "^",
range(693, 704): "_",
range(704, 715): "`",
range(715, 726): "a",
range(726, 737): "b",
range(737, 748): "c",
range(748, 759): "d",
range(759, 770): "e",
range(770, 781): "f",
range(781, 792): "g",
range(792, 803): "h",
range(803, 814): "i",
range(814, 825): "j",
range(825, 836): "k",
range(836, 847): "l",
range(847, 858): "m",
range(858, 869): "n",
range(869, 880): "o",
range(880, 891): "p",

```
    range(891, 902): "q",
    range(902, 913): "r",
    range(913, 924): "s",
    range(924, 935): "t",
    range(935, 946): "u",
    range(946, 957): "v",
    range(957, 968): "w",
    range(968, 979): "x",
    range(979, 990): "y",
    range(990, 1001): "z",
    range(1001, 1012): "{",
    range(1012, 1024): "|"
    #range(638, 649): "}",      //may replace with space, it's a little tricky? currently turns
MUTE OFF
    #range(638, 649): "~"       //used for turning MUTE ON
}

def update_value(sensor_index, value):
    global serial_thread_running
    global sensor_values
    global selected_tab_index

    if serial_thread_running:
        sensor_values[sensor_index] = int(value)
        ser.write(f"Slider Value {sensor_index + 1}:
{sensor_values[sensor_index]}\n".encode('utf-8'))
    else:
        sensor_values[sensor_index] = int(value)
        char_values = [next(char for range_, char in char_map.items() if value in range_) for
value in sensor_values]

        print(selected_tab_index, end="")
        print("".join(char_values))

def serial_reader():
    global serial_thread_running
    global sensor_values
    global notebook
    global selected_tab_index
```

```
    ser = serial.Serial('/dev/ttyUSB0', 57600, timeout=1.0)        #previously 115200

    while True:
        time.sleep(0.03)
        if serial_thread_running:
            ser.write("READ_SENSOR\n".encode('utf-8'))
            if ser.in_waiting > 0:
                response = ser.readline().decode('utf-8').rstrip()
                print("NANO:", response)
                pattern = r"\[(\d+),\s*(\d+),\s*(\d+),\s*(\d+)\]"
                match = re.search(pattern, response)

                if match:
                    sensor_values = [int(match.group(i)) for i in range(1, 5)]
                    selected_tab_index = notebook.index(notebook.select())
                    scales = slider_sets[selected_tab_index]
                    print(selected_tab_index, end="")
                    for i, value in enumerate(sensor_values):
                        scales[i].set(value)
                        for range_, char in char_map.items():
                            if value in range_:
                                print(char, end="")
                                break

def toggle_serial_thread():
    global serial_thread_running
    global ser

    serial_thread_running = not serial_thread_running
    if serial_thread_running:
        ser = serial.Serial('/dev/ttyUSB0', 57600, timeout=1.0)
        start_button.config(text="Stop External Control")
    else:
        ser.close()
        start_button.config(text="Start External Control")

def create_sliders(tab, tab_index):
    scales = []
```

```
    frame = ttk.Frame(tab)
    frame.pack(fill='both', expand=True)
    for i in range(4):
        if tab_index == 1:  # For tab 2 (EQ)
            if i == 0:
                label_text = "Low"
            elif i == 1 or i == 2:
                label_text = "Mid"
            else:
                label_text = "High"
        else:
            label_text = f"Slider {i+1}"

        label = tk.Label(frame, text=label_text)
        label.grid(row=0, column=i, pady=5)  # Place label above the slider
        scale = tk.Scale(frame, from_=1023, to=0, orient=tk.VERTICAL,
length=100,width=106, command=lambda value, idx=i: update_value(idx, value))
        scale.grid(row=1, column=i, padx=20)  # Place scale below the label
        scales.append(scale)
    return scales




def on_tab_selected(event):
    global selected_tab_index

    selected_tab_index = notebook.index(notebook.select())
    print(selected_tab_index, end="")
    print("    ")

def muteToggle():
    global toggle_state
    if toggle_state:
        mute_button.config(text="Mute Off")
        print("\n}}}}}")
    else:
        mute_button.config(text="Mute On")
        print("\n~~~~~")
```

```
    toggle_state = not toggle_state

def reset_sliders():
    for scales in slider_sets.values():
        for scale in scales:
            scale.set(0)

toggle_state = False

serial_thread_running = False
sensor_values = [0, 0, 0, 0]

root = tk.Tk()
root.title("Starter Gig Effects Unit")

notebook = ttk.Notebook(root)
notebook.pack(fill='both', expand=True)
notebook.bind("<<NotebookTabChanged>>", on_tab_selected)

for i, tab_name in enumerate(["Distortion", "EQ", "Tremolo"]):
    tab = ttk.Frame(notebook)
    notebook.add(tab, text=tab_name)

start_button = tk.Button(root, text="Start External Control",
command=toggle_serial_thread)
start_button.pack(pady=10)

serial_thread = threading.Thread(target=serial_reader, daemon=True)
serial_thread.start()

mute_button = tk.Button(root, text="Mute Off", command=muteToggle)
mute_button.pack(pady=10)

reset_button = tk.Button(root, text="Reset Sliders", command=reset_sliders)
reset_button.pack(pady=10)

slider_sets = {}
for i in range(3):
    slider_sets[i] = create_sliders(notebook.winfo_children()[i], i)
```

```
root.mainloop()
```