

```
1 //=====
2 //ESP32 WebSocket Server: Pen Plotter Web UI
3 //by: Rajat Joshi
4 //=====
5 #include <WiFi.h>
6 #include <WebServer.h>
7 #include <WebSocketsServer.h>
8 #include <ArduinoJson.h>
9 #include <SPI.h>
10 #include <Adafruit_GFX.h>
11 #include <Adafruit_SSD1306.h>
12
13 //-----NETWORK DETAILS-----
14 const char* ssid = "Bereich 51 24";
15 const char* password = "BB1010e72#16!#6!5bjGu";
16 int clients_connected = 0;
17
18 //----- WEB SERVER AND WEBSOCKET -----
19 WebServer server(80);
20 WebSocketsServer websocket = WebSocketsServer(81);
21
22 //----- Task handle for sensor input -----
23 TaskHandle_t sensor_input;
24
25 //----- DEFINE IO PINS -----
26 #define MOTOR_Y_CW 13
27 #define MOTOR_Y_CCW 12
28 #define MOTOR_Y_PWM 27
29 #define MOTOR_X_CW A5
30 #define MOTOR_X_CCW A1
31 #define MOTOR_X_PWM 18
32 #define Ylim_min 21
33 #define Ylim_max 17
34 #define Xlim_min 16
35 #define Xlim_max 19
36 #define Ymotor_A0 A3
37 #define Ymotor_A1 A2
38 #define Xmotor_A0 32
39 #define Xmotor_A1 33
40 #define E_stop_btn 14
41 #define Z_solenoid 15
42 #define SCREEN_WIDTH 128 // OLED display width, in pixels
43 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
44 #define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)
45 #define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
46 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
47
48 // Setting PWM properties
49 const int freq = 2000;
50 const int pwmChannel_X = 0;
51 const int pwmChannel_Y = 2;
52 const int resolution = 8;
53
54 const char* web_Button = "";
55 int web_Jog_step_X = 20;
56 int web_Jog_step_Y = 20;
57 int web_Jog_speed_XY = 1000;
58 int web_Goto_X = 0;
59 int web_Goto_Y = 0;
60 int web_House_size = 20;
61 int web_Draw_speed = 1000;
62 int web_Start_pos = 1;
63 float web_Cal_X = 0;
64 float web_Cal_Y = 0;
65 float web_Kpx = 1;
66 float web_Kpy = 1;
67 float web_Kix = 0;
68 float web_Kiy = 0;
69 float web_Kdx = 0;
70 float web_Kdy = 0;
71
72
73 int Ymin_limit = 1;
74 int Ymax_limit = 1;
75 int Xmin_limit = 1;
76 int Xmax_limit = 1;
77 int E_stop = 1;
78 int Pen_state = 1;
79 const char* Plotter_status = "Idle";
80 const char* calibrate_axis = "";
81
82 float Plot_area_xmax = 220;
83 float Plot_area_ymax = 245;
84 float X_coordinate = 0;
85 float Y_coordinate = 0;
86 int X_dir = 0;
87 int Y_dir = 0;
88 int X_count = 0;
89 int X_count_dec = 0;
90 int Y_count = 0;
91 int Y_count_dec = 0;
92
```

Pen Plotter Main Code

```

93 float Target_X      = 0;
94 float Target_Y      = 0;
95 float Delta_X       = 0;
96 float Delta_Y       = 0;
97 float Travel_dist_X = 10;
98 float Travel_dist_Y = 10;
99
100 int   X_motor_CW     = 0;
101 int   X_motor_CCW    = 0;
102 int   Y_motor_CW     = 0;
103 int   Y_motor_CCW    = 0;
104 int   X_motor_PWM    = 0;
105 int   Y_motor_PWM    = 0;
106 float d_X = 0.25;
107 float d_Y = 0.25;
108 float Kpx = 70;
109 float Kpy = 70;
110 float Kix = 100;
111 float Kiy = 100;
112 float Kdx = 0.06;
113 float Kdy = 0.06;
114
115 float e_integral_x = 0;
116 float e_integral_y = 0;
117 float e_diff_x     = 0;
118 float e_diff_y     = 0;
119 float Last_Delta_X = 0;
120 float Last_Delta_Y = 0;
121 float ux = 0;
122 float uy = 0;
123
124 float Calibration_sequence[5][2] = { {-220, -245}, {0, 0}, {-10, -10}, {210, 235}, {0, 0} };
125
126 int   position_index      = 0; // position 0-8 in drawing sequence and 0-4 Calibration_sequence
127 int   Draw_sequence_index = 0;
128 int   Draw_sequence[44][9] = {{0, 1, 2, 0, 3, 2, 4, 3, 1}, {0, 2, 3, 0, 1, 2, 4, 3, 1}, {0, 3, 1, 0, 2, 3, 4, 2, 1}, {0
, 1, 3, 2, 0, 3, 4, 2, 1},
129                                     {0, 2, 3, 4, 2, 1, 0, 3, 1}, {0, 3, 2, 0, 1, 2, 4, 3, 1}, {0, 1, 2, 4, 3, 0, 2, 3, 1}, {0,
2, 1, 0, 3, 2, 4, 3, 1},
130                                     {0, 3, 2, 4, 3, 1, 0, 2, 1}, {0, 2, 4, 3, 1, 0, 3, 2, 1}, {0, 3, 4, 2, 1, 0, 2, 3, 1}, {0,
1, 2, 0, 3, 4, 2, 3, 1},
131                                     {0, 2, 3, 0, 1, 3, 4, 2, 1}, {0, 3, 1, 0, 2, 4, 3, 2, 1}, {0, 1, 3, 2, 4, 3, 0, 2, 1}, {0,
2, 3, 4, 2, 1, 3, 0, 1},
132                                     {0, 3, 2, 0, 1, 3, 4, 2, 1}, {0, 1, 2, 4, 3, 2, 0, 3, 1}, {0, 2, 1, 0, 3, 4, 2, 3, 1}, {0,
3, 2, 4, 3, 1, 2, 0, 1},
133                                     {0, 2, 4, 3, 1, 2, 3, 0, 1}, {0, 3, 4, 2, 1, 3, 2, 0, 1}, {0, 1, 2, 3, 0, 2, 4, 3, 1}, {0,
2, 3, 1, 0, 3, 4, 2, 1},
134                                     {0, 3, 1, 2, 3, 4, 2, 0, 1}, {0, 1, 3, 4, 2, 0, 3, 2, 1}, {0, 2, 4, 3, 0, 1, 2, 3, 1}, {0,
3, 2, 1, 0, 2, 4, 3, 1},
135                                     {0, 1, 3, 0, 2, 3, 4, 2, 1}, {0, 2, 1, 3, 2, 4, 3, 0, 1}, {0, 3, 4, 2, 0, 1, 2, 3, 1}, {0,
2, 4, 3, 2, 1, 0, 3, 1},
136                                     {0, 3, 4, 2, 3, 1, 0, 2, 1}, {0, 1, 2, 3, 4, 2, 0, 3, 1}, {0, 2, 3, 1, 2, 4, 3, 0, 1}, {0,
3, 1, 2, 4, 3, 2, 0, 1},
137                                     {0, 1, 3, 4, 2, 3, 0, 2, 1}, {0, 2, 4, 3, 0, 1, 3, 2, 1}, {0, 3, 2, 1, 3, 4, 2, 0, 1}, {0,
1, 3, 0, 2, 4, 3, 2, 1},
138                                     {0, 2, 1, 3, 4, 2, 3, 0, 1}, {0, 3, 4, 2, 0, 1, 3, 2, 1}, {0, 2, 4, 3, 2, 1, 3, 0, 1}, {0,
3, 4, 2, 3, 1, 2, 0, 1}}};
139 float Nikolaus_Haus[5][2]  = { {0, 0}, {20, 0}, {20, 20}, {0, 20}, {10, 30} };
140 int   Inside_draw_area    =1;
141
142 int           Web_update_interval = 200; // send data to the client every 1000ms -> 1s
143 unsigned long Web_Prev_Millis = 0; // we use the "millis()" command for time reference and this will
output an unsigned long
144 int           Sensor_update_interval = 100; // send data to the client every 1000ms -> 1s
145 unsigned long Sensor_Prev_Micros = 0; // we use the "millis()" command for time reference and this will
output an unsigned long
146 unsigned long Serial_update_Micros = 0; // we use the "millis()" command for time reference and this will
output an unsigned long
147
148 int Estop_reset_time = 2e6;
149 unsigned long E_stop_Micros = 0; // we use the "millis()" command for time reference and this will
output an unsigned long
150
151 int           Axis_stable_time = 5e4;
152 unsigned long X_stable_Micros = 0;
153 unsigned long Y_stable_Micros = 0; // we use the "millis()" command for time reference and this will
output an unsigned long
154
155 int           Max_Pen_Down_Time = 44000; // send data to the client every 1000ms -> 1s
156 unsigned long Pen_Down_Time = 0;
157 //----- DEFINE FUNCTION FILES-----
158 #include "WebUI_HTML.h"
159 #include "Display.h"
160 #include "Functions.h"
161 #include "Input_signal_processing.h"
162
163
164
165 //=====
166 void setup()
167 {
168

```

```

169 Serial.begin(115200);
170 //-----
171 if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) { // SSD1306_SWITCHCAPVCC = generate display voltage from
3.3V internally
172     Serial.println(F("SSD1306 allocation failed"));
173     for(;;); // Don't proceed, loop forever
174 }
175 Welcome_Screen();
176 //-----
177 WiFi.begin(ssid, password);
178 Serial.println("Establishing connection to WiFi with SSID: " + String(ssid));
179 while(WiFi.status() != WL_CONNECTED){Serial.print("."); delay(500);}
180 WiFi.mode(WIFI_STA);
181 Serial.println();
182 Serial.print("Local IP: ");
183 Serial.println(WiFi.localIP());
184 display.setCursor(4,40);
185 display.println("- Local IP:");
186 display.setCursor(45,50);
187 display.println(WiFi.localIP());
188 display.display();
189 delay(3000);
190 display.clearDisplay();
191 //-----
192 server.on("/", webpage);
193 //-----
194 server.begin(); websocket.begin();
195 websocket.onEvent(websocketEvent);
196 //-----
197 pinMode(Xlim_min, INPUT_PULLUP);
198 pinMode(Xlim_max, INPUT_PULLUP);
199 pinMode(Ylim_min, INPUT_PULLUP);
200 pinMode(Ylim_max, INPUT_PULLUP);
201 pinMode(Xmotor_A0, INPUT_PULLUP);
202 pinMode(Xmotor_A1, INPUT_PULLUP);
203 pinMode(Ymotor_A0, INPUT_PULLUP);
204 pinMode(Ymotor_A1, INPUT_PULLUP);
205 pinMode(E_stop_btn, INPUT_PULLUP);
206 pinMode(Z_solenoid, OUTPUT );
207
208 pinMode(MOTOR_X_CW, OUTPUT );
209 pinMode(MOTOR_X_CCW, OUTPUT );
210 pinMode(MOTOR_Y_CW, OUTPUT );
211 pinMode(MOTOR_Y_CCW, OUTPUT );
212 ledcSetup(pwmChannel_X, freq, resolution);
213 ledcSetup(pwmChannel_Y, freq, resolution);
214 ledcAttachPin(MOTOR_X_PWM, pwmChannel_X);
215 ledcAttachPin(MOTOR_Y_PWM, pwmChannel_Y);
216
217 Xmin_limit = digitalRead(Xlim_min);
218 Xmax_limit = digitalRead(Xlim_max);
219 Ymin_limit = digitalRead(Ylim_min);
220 Ymax_limit = digitalRead(Ylim_max);
221 E_stop = digitalRead(E_stop_btn);
222 digitalWrite(Z_solenoid, LOW);
223 setMotorx();
224 setMotorx();
225 //-----
226 xTaskCreatePinnedToCore(
227     sensor_input_func, /* Function to implement the task */
228     "sensor_input", /* Name of the task */
229     10000, /* Stack size in words */
230     NULL, /* Task input parameter */
231     0, /* Priority of the task */
232     &sensor_input, /* Task handle. */
233     0); /* Core where the task should run */
234 //-----
235 // Target_X = Calibration_sequence[position_index][0];
236 // Target_Y = Calibration_sequence[position_index][1];
237 // Travel_dist_X = abs(Target_X - X_coordinate);
238 // Travel_dist_Y = abs(Target_Y - Y_coordinate);
239 // Plotter_status = "Calibrating";
240
241 }
242
243
244
245
246 //=====
247 void loop(){
248     websocket.loop(); server.handleClient();
249
250     if ((unsigned long)(millis() - Web_Prev_Millis) >= Web_update_interval) { // check if "interval" ms has passed since
last time the clients were updated
251
252         Web_Prev_Millis = millis(); // reset previousMillis
253         String jsonString = ""; // create a JSON string for sending data to the client
254         StaticJsonDocument<200> doc; // create a JSON container
255         JsonObject object = doc.to<JsonObject>(); // create a JSON Object
256         object["X_coor"] = X_coordinate; // write data into the JSON object -> I used "rand1" and "rand2"
here, but you can use anything else
257         object["Y_coor"] = Y_coordinate;

```

```

258 if(digitalRead(Z_solenoid) == 0){ object["Pen_up_down"] = "UP"; } else { object["Pen_up_down"] = "DOWN"; }
259 object["status"] = Plotter_status;
260 if(Xmin_limit == 0){ object["X_Lim_min"] = "OK &#9989;"; } else { object["X_Lim_min"] = "Active &#128680;"; }
261 if(Xmax_limit == 0){ object["X_Lim_max"] = "OK &#9989;"; } else { object["X_Lim_max"] = "Active &#128680;"; }
262 if(Ymin_limit == 0){ object["Y_Lim_min"] = "OK &#9989;"; } else { object["Y_Lim_min"] = "Active &#128680;"; }
263 if(Ymax_limit == 0){ object["Y_Lim_max"] = "OK &#9989;"; } else { object["Y_Lim_max"] = "Active &#128680;"; }
264 if(E_stop == 1){ object["E_Stop_status"] = "OK &#9989;"; } else { object["E_Stop_status"] = "Active &#128680;"; }
265 if(Inside_draw_area == 0){ object["Inside_draw_area"] = "Error"; Inside_draw_area = 1;} else { object[
"Inside_draw_area"] = "ok";}

266
267 serializeJson(doc, jsonString); // convert JSON object to string
268 //Serial.println(jsonString); // print JSON string to console for debug purposes (you can comment this
out)
269 websocket.broadcastTXT(jsonString); // send JSON string to clients
270
271 Update_Display();
272
273
274 if (X_stable_Micros > Axis_stable_time && Y_stable_Micros > Axis_stable_time ){
275
276     if (Plotter_status == "Plotting...") {
277
278         Serial.println("Point reached");
279
280         //point();
281
282         if (web_Start_pos == 1 and position_index != 8){
283             position_index++;
284             Target_X = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][0];
285             Target_Y = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][1];
286             Travel_dist_X = abs (Target_X - X_coordinate);
287             Travel_dist_Y = abs (Target_Y - Y_coordinate);
288
289         } else if (web_Start_pos == 1 and position_index == 8){
290             position_index=0;
291             digitalWrite(Z_solenoid, LOW);
292             if(Draw_sequence_index == 43){Draw_sequence_index=0;} else {Draw_sequence_index++;}
293             Plotter_status = "Idle";
294             // pen up
295         }
296
297         if (web_Start_pos == 2 and position_index != 0){
298             position_index--;
299             Target_X = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][0];
300             Target_Y = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][1];
301             Travel_dist_X = abs (Target_X - X_coordinate);
302             Travel_dist_Y = abs (Target_Y - Y_coordinate);
303
304         } else if (web_Start_pos == 2 and position_index == 0){
305             digitalWrite(Z_solenoid, LOW);
306             if(Draw_sequence_index == 43){Draw_sequence_index=0;} else {Draw_sequence_index++;}
307             Plotter_status = "Idle";
308             // pen up
309         }
310         delay(1000);
311         Serial.print("DRAW (X,Y) = > ");
312         Serial.print(Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][0]);
313         Serial.print(",");
314         Serial.println(Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][1]);
315     }
316
317
318     if (Plotter_status == "Calibrating"){
319
320         Plotter_status == "Hold";
321         Serial.print(Delta_X);
322         Serial.print(" = dx,dy = ");
323         Serial.print(Delta_Y);
324         Serial.print(" index = ");
325         Serial.print(position_index);
326         Update_Display();
327         if (position_index < 4){
328
329             position_index++;
330
331             if (calibrate_axis == "X") {
332                 Target_X = Calibration_sequence[position_index][0];
333             } else if (calibrate_axis == "Y") {
334                 Target_Y = Calibration_sequence[position_index][1];
335             } else {
336                 Target_X = Calibration_sequence[position_index][0];
337                 Target_Y = Calibration_sequence[position_index][1];
338             }
339
340             Travel_dist_X = abs (Target_X - X_coordinate);
341             Travel_dist_Y = abs (Target_Y - Y_coordinate);
342
343             if (position_index > 2) {e_integral_x = 0; e_integral_y = 0;} else {e_integral_x = 260; e_integral_y = 260;}
344
345             Plotter_status = "Calibrating";
346         } else {
347             position_index = 0;

```

```

348         e_integral_x = 0;
349         e_integral_y = 0;
350         Plotter_status = "Idle";
351
352
353     }
354 }
355 }
356 }
357
358 if ((unsigned long)(millis() - Pen_Down_Time) >= Max_Pen_Down_Time){
359     digitalWrite(Z_solenoid, LOW);
360 }
361
362
363 if(Kix * e_integral_x >= 255 || Kiy * e_integral_y >= 255){
364
365     if(abs(Delta_X) > 5 && Plotter_status != "Calibrating") {e_integral_x = 0;}
366     if(abs(Delta_Y) > 5 && Plotter_status != "Calibrating") {e_integral_y = 0;}
367
368     delay(22);
369
370     if(abs(Delta_X) >= 0.4 && Kix * e_integral_x >= 255){
371         X_motor_CW = 0; X_motor_CCW = 0;
372         if (Delta_X > 0 && Xmax_limit == 0) { X_motor_CW = 1; X_motor_CCW = 0; }
373         else if (Delta_X < 0 && Xmin_limit == 0) { X_motor_CW = 0; X_motor_CCW = 1; }
374         if(abs(Delta_X) > 0.4) {X_motor_PWM = 240;}
375         setMotorx();
376     }
377
378     if(abs(Delta_Y) >= 0.4 && Kiy * e_integral_y >= 255){
379         Y_motor_CW = 0; Y_motor_CCW = 0;
380         if (Delta_Y > 0 && Ymax_limit == 0) { Y_motor_CW = 1; Y_motor_CCW = 0; }
381         else if (Delta_Y < 0 && Ymin_limit == 0) { Y_motor_CW = 0; Y_motor_CCW = 1; }
382         if(abs(Delta_Y) > 0.4) {Y_motor_PWM = 255;}
383         setMotory();
384     }
385
386     delay(16);
387
388     if(Kix * e_integral_x >= 255){
389         X_motor_PWM = 0;
390         setMotorx();
391     }
392
393     if(Kiy * e_integral_y >= 255){
394         Y_motor_PWM = 0;
395         setMotory();
396     }
397 }
398 }

```



```

1 //=====
2 //Signal binning: discretization
3 //=====
4 int signal_binning( float value){
5     if (value >1.8){
6         return 2;
7     } else if (value < 1.6 && value > 0.6){
8         return 1;
9     } else if (value < 0.4 && value > -0.4){
10        return 0;
11    } else if (value < -0.6 && value > -1.6){
12        return -1;
13    } else if (value <-1.8){
14        return -2;
15    } else {
16        return 5;
17    }
18 }
19
20 //=====
21 //update function: update position and sensor sttus
22 //=====
23 void sensor_input_func( void * parameter) {
24
25     float Y_A0_raw[]      = {0,0,0};
26     float X_A0_raw[]      = {0,0,0};
27     float Y_A1_raw[]      = {0,0,0};
28     float X_A1_raw[]      = {0,0,0};
29     float Y_A0_filtered[] = {0,0,0};
30     float X_A0_filtered[] = {0,0,0};
31     float Y_A1_filtered[] = {0,0,0};
32     float X_A1_filtered[] = {0,0,0};
33     int Y_A0_binned[]     = {0,0,0};
34     int X_A0_binned[]     = {0,0,0};
35     int Y_A1_binned[]     = {0,0,0};
36     int X_A1_binned[]     = {0,0,0};
37     // float Y_CORCTON    = 1;
38     // float X_CORCTON    = 1;
39
40     // (second order Butterworth coefficients)
41     float b[] = {0.01323107, 0.02646213, 0.01323107};
42     float a[] = {1.64927209, -0.70219636};
43
44     int Print_index=0;
45     float sens[1000][2];
46
47
48
49     for(;;) {
50
51         if ((unsigned long) (micros() - Sensor_Prev_Micros) >= Sensor_update_interval){
52
53             Sensor_Prev_Micros = micros(); // reset previousMicros
54             Print_index++;
55
56             Ymin_limit = digitalRead(Ylim_min);
57             Ymax_limit = digitalRead(Ylim_max);
58             Xmin_limit = digitalRead(Xlim_min);
59             Xmax_limit = digitalRead(Xlim_max);
60             E_stop      = digitalRead(E_stop_btn);
61
62             Y_A0_raw[0] = analogRead(Ymotor_A0);
63             X_A0_raw[0] = analogRead(Xmotor_A0);
64             Y_A1_raw[0] = analogRead(Ymotor_A1);
65             X_A1_raw[0] = analogRead(Xmotor_A1);
66
67             Y_A0_filtered[0] = a[0]*Y_A0_filtered[1] + a[1]*Y_A0_filtered[2] + b[0]*Y_A0_raw[0] + b[1]*Y_A0_raw[1] + b[2]*
Y_A0_raw[2];
68             X_A0_filtered[0] = a[0]*X_A0_filtered[1] + a[1]*X_A0_filtered[2] + b[0]*X_A0_raw[0] + b[1]*X_A0_raw[1] + b[2]*
X_A0_raw[2];
69             Y_A1_filtered[0] = a[0]*Y_A1_filtered[1] + a[1]*Y_A1_filtered[2] + b[0]*Y_A1_raw[0] + b[1]*Y_A1_raw[1] + b[2]*
Y_A1_raw[2];
70             X_A1_filtered[0] = a[0]*X_A1_filtered[1] + a[1]*X_A1_filtered[2] + b[0]*X_A1_raw[0] + b[1]*X_A1_raw[1] + b[2]*
X_A1_raw[2];
71
72             Y_A0_binned[0] = signal_binning((Y_A0_filtered[0]/204.8-9.0 )*1.8);
73             Y_A1_binned[0] = signal_binning((Y_A1_filtered[0]/204.8-9.0 )*1.8);
74             X_A0_binned[0] = signal_binning((X_A0_filtered[0]/204.8-9.0 )*1.8);
75             X_A1_binned[0] = signal_binning((X_A1_filtered[0]/204.8-9.0 )*1.8);
76
77             if (Y_A0_binned[0] == 5) {Y_A0_binned[0] = Y_A0_binned[1];}
78             if (Y_A1_binned[0] == 5) {Y_A1_binned[0] = Y_A1_binned[1];}
79             if (X_A0_binned[0] == 5) {X_A0_binned[0] = X_A0_binned[1];}
80             if (X_A1_binned[0] == 5) {X_A1_binned[0] = X_A1_binned[1];}
81
82
83
84             if ((X_A1_binned[0] == 0 && X_A1_binned[1] != 0) || X_A1_binned[0] * X_A1_binned[1] < 0){
85
86                 if ((X_A1_binned[0] - X_A1_binned[1] > 0 && X_A0_binned[0] < 0) || (X_A1_binned[0] - X_A1_binned[1] < 0 &&
X_A0_binned[0] > 0)){
87                     X_count++;

```

```

88     X_dir = 1;
89     if (X_A1_binned[0] == 0) { X_count_dec = 0;} else {abs(X_A1_binned[0]);}
90
91 } else {
92     X_count--;
93     X_dir = -1;
94     if (X_A1_binned[0] == 0) { X_count_dec = 4;} else {4-abs(X_A1_binned[0]);}
95
96 }
97 } else if (X_A1_binned[0] != X_A1_binned[1] && X_dir == 1 && X_count_dec < 3 ) {
98     X_count_dec++;
99
100 } else if (X_A1_binned[0] != X_A1_binned[1] && X_dir == -1 && X_count_dec > 1 ) {
101     X_count_dec--;
102
103 }
104
105 X_coordinate = (X_count * 4 + X_count_dec) * d_X;
106
107 if ((Y_A0_binned[0] == 0 && Y_A0_binned[1] != 0) || Y_A0_binned[0] * Y_A0_binned[1] < 0){
108
109     if ((Y_A0_binned[0] - Y_A0_binned[1] > 0 && Y_A1_binned[0] > 0) || (Y_A0_binned[0] - Y_A0_binned[1] < 0 &&
110     Y_A1_binned[0] < 0)){
111         if (Y_dir == 1) {Y_count++;}
112         Y_dir = 1;
113         if (Y_A0_binned[0] == 0) { Y_count_dec = 0;} else {abs(Y_A0_binned[0]);}
114
115     } else {
116         if (Y_dir == -1) {Y_count--;}
117         Y_dir = -1;
118         if (Y_A0_binned[0] == 0) { Y_count_dec = 4;} else {4-abs(Y_A0_binned[0]);}
119     }
120 } else if (Y_A0_binned[0] != Y_A0_binned[1] && Y_dir == 1 && Y_count_dec < 3) {
121     Y_count_dec++;
122 }else if (Y_A0_binned[0] != Y_A0_binned[1] && Y_dir == -1 && Y_count_dec > 1) {
123     Y_count_dec--;
124
125 }
126
127 Y_coordinate = (Y_count * 4 + Y_count_dec) * d_Y;
128
129 if(Plotter_status == "Calibrating"){
130     if (Target_X < 0 && Xmin_limit == 1) {
131         X_count = -5;
132         X_count_dec = 0;
133         X_coordinate = (X_count * 4 + X_count_dec) * d_X;
134         Target_X = X_coordinate;
135
136     } else if (Target_X == 210 && Xmax_limit == 1) { Target_X = X_coordinate; Plot_area_xmax = X_coordinate - 10;
137     }//Serial.println("xmax");}
138
139     if (Target_Y < 0 && Ymin_limit == 1) {
140         Y_count = -5;
141         Y_count_dec = 0;
142         Y_coordinate = (Y_count * 4 + Y_count_dec) * d_Y;
143         Target_Y = Y_coordinate;
144
145     } else if (Target_Y == 235 && Ymax_limit == 1) { Target_Y = Y_coordinate; Plot_area_ymax = Y_coordinate - 10;
146     }//Serial.println("ymax");}
147
148 }
149
150 Delta_X = Target_X - X_coordinate;
151 Delta_Y = Target_Y - Y_coordinate;
152
153 if (abs(Delta_X) < 0.4){X_stable_Micros += Sensor_update_interval;} else {X_stable_Micros = 0;}
154 if (abs(Delta_Y) < 0.4){Y_stable_Micros += Sensor_update_interval;} else {Y_stable_Micros = 0;}
155
156 if (Print_index>2 && (X_A1_binned[0] != X_A1_binned[1] || Y_A0_binned[0] != Y_A0_binned[1])){
157     Print_index=0;
158     Serial.print(Target_X);
159     Serial.print(" ");
160     Serial.print(X_coordinate);
161     Serial.print(" ");
162     // Serial.print(Delta_X);
163     // Serial.print(" ");
164     // Serial.print(ux);
165     // Serial.print(" ");
166     // Serial.print(Kix * e_integral_x);
167     // Serial.print(" ");
168     Serial.print(Target_Y);
169     Serial.print(" ");
170     Serial.print(Y_coordinate);
171     // Serial.print(" ");
172     // Serial.print(Delta_Y);
173     // Serial.print(" ");
174     // Serial.println(uy);
175     Serial.print(" ");
176     Serial.print(Y_A0_binned[0]);
177     Serial.print(" ");
178     Serial.println(Y_A1_binned[0]);
179
180 // // while(Print_index > 10){
181 //     // Serial.print(X_A0_raw[1]);
182 //     // Serial.print(" ");

```

```

177 //      // Serial.print(X_A0_filtered[1]);
178 //      // Serial.print(" ,");
179 //      // Serial.println(X_A1_raw[0]);
180 // //      Print_index--;
181 // //      }
182
183 }
184
185 for(int i = 1; i >= 0; i--){
186   Y_A0_raw[i+1]      = Y_A0_raw[i]; // store xi
187   X_A0_raw[i+1]      = X_A0_raw[i]; // store xi
188   Y_A0_filtered[i+1] = Y_A0_filtered[i]; // store yi
189   X_A0_filtered[i+1] = X_A0_filtered[i]; // store yi
190   Y_A0_binned[i+1]   = Y_A0_binned[i]; // store yi
191   X_A0_binned[i+1]   = X_A0_binned[i]; // store yi
192
193   Y_A1_raw[i+1]      = Y_A1_raw[i]; // store xi
194   X_A1_raw[i+1]      = X_A1_raw[i]; // store xi
195   Y_A1_filtered[i+1] = Y_A1_filtered[i]; // store yi
196   X_A1_filtered[i+1] = X_A1_filtered[i]; // store yi
197   Y_A1_binned[i+1]   = Y_A1_binned[i]; // store yi
198   X_A1_binned[i+1]   = X_A1_binned[i]; // store yi
199 }
200
201 if (Ymin_limit + Ymax_limit + Xmin_limit + Xmax_limit != 0 && Plotter_status != "E-Stop !!"){
202
203   if (Plotter_status == "Calibrating" || Plotter_status == "Hold"){
204     // do nothing
205   } else if (Plotter_status == "Plotting..."){
206     Target_X = X_coordinate;
207     Target_Y = Y_coordinate;
208     Plotter_status = "Warning";
209
210   } else if (Plotter_status == "Jogging"){
211     if (Xmax_limit == 1 && Delta_X > 0.4) { Target_X = X_coordinate; }
212     else if (Xmin_limit == 1 && Delta_X < -0.4) {
213       X_count      = -8;
214       X_count_dec  = 2;
215       X_coordinate = (X_count * 4 + X_count_dec) * d_X;
216       Target_X = X_coordinate;
217     }
218     if (Ymax_limit == 1 && Delta_Y > 0.4) { Target_Y = Y_coordinate; }
219     else if (Ymin_limit == 1 && Delta_Y < -0.4) {
220       Y_count      = -8;
221       Y_count_dec  = 2;
222       Y_coordinate = (Y_count * 4 + Y_count_dec) * d_Y;
223       Target_Y = Y_coordinate;
224     }
225     X_motor_PWM = 0;
226     Y_motor_PWM = 0;
227     setMotorx();
228     setMotory();
229     Plotter_status = "Warning";
230   }else{
231     Plotter_status = "Warning";
232   }
233   Delta_X      = Target_X - X_coordinate;
234   Delta_Y      = Target_Y - Y_coordinate;
235 } else if (Plotter_status == "Warning" && Ymin_limit + Ymax_limit + Xmin_limit + Xmax_limit == 0){
236   if (abs(Delta_X) < 0.4 && abs(Delta_Y) < 0.4 ) {Plotter_status = "Idle";}
237   else {Plotter_status = "Jogging";}
238 }
239
240 if (E_stop == 1){ E_stop_Micros = micros(); }
241 if (E_stop == 0 && (micros() - E_stop_Micros) >= Estop_reset_time){
242   Plotter_status = "Idle";
243 } else if (E_stop == 0 && (micros() - E_stop_Micros) >= 100){
244   Plotter_status = "E-Stop !!";
245   Target_X = X_coordinate;
246   Target_Y = Y_coordinate;
247   Delta_X      = Target_X - X_coordinate;
248   Delta_Y      = Target_Y - Y_coordinate;
249   setMotorx();
250   setMotory();
251   digitalWrite(Z_solenoid, LOW);
252 }
253
254 if (Plotter_status != "Idle" && Plotter_status != "Paused" && Plotter_status != "Hold"){
255
256
257
258   if(Kix * e_integral_x < 255){
259
260     if (abs(Delta_X) < 2 && abs(Delta_X) > 0.4 ){e_integral_x = e_integral_x + abs(Delta_X) /100;}
261
262     if (abs(Delta_X) < 5 || abs(Delta_X) < 5 + Travel_dist_X*0.2){ e_diff_x      = (Delta_X - Last_Delta_X) *
263       10000; } else {e_diff_x      = 0;}
264     Last_Delta_X = Delta_X;
265
266     X_motor_CW = 0;  X_motor_CCW = 0;
267
268     if((Travel_dist_X - abs(Delta_X)) < 5 && Travel_dist_X > 10){

```



```

268     ux = sgn(Delta_X)*(255 + 4 * (Travel_dist_X - abs(Delta_X))) ;
269     if (Delta_X > 0 && Xmax_limit == 0)      { X_motor_CW = 1;  X_motor_CCW = 0; }
270     else if (Delta_X < 0 && Xmin_limit == 0)    { X_motor_CW = 0;  X_motor_CCW = 1; }
271
272 } else if (abs(Delta_X) < 5 || abs(Delta_X) < 5 + Travel_dist_X*0.2) {
273     ux = Kpx * Delta_X + Kdx * e_diff_x;
274     if (ux > 0 && Xmax_limit == 0)      { X_motor_CW = 1;  X_motor_CCW = 0; }
275     else if (ux < 0 && Xmin_limit == 0)    { X_motor_CW = 0;  X_motor_CCW = 1; }
276
277 } else {
278     ux = sgn(Delta_X)*255;
279     if (Delta_X > 0 && Xmax_limit == 0)      { X_motor_CW = 1;  X_motor_CCW = 0; }
280     else if (Delta_X < 0 && Xmin_limit == 0)    { X_motor_CW = 0;  X_motor_CCW = 1; }
281 }
282
283 X_motor_PWM = (int) fabs(ux);
284 if (X_motor_PWM > 255) {X_motor_PWM = 255;} else if (X_motor_PWM < 150){X_motor_PWM = 0;}
285 setMotorx();
286
287 }
288
289 if(Kiy * e_integral_y < 255){
290
291     if (abs(Delta_Y) < 2 && abs(Delta_Y) > 0.4 ){e_integral_y = e_integral_y + abs(Delta_Y) /100;}
292
293     if (abs(Delta_Y) < 5 || abs(Delta_Y) < 5 + Travel_dist_Y*0.2){ e_diff_y      = (Delta_Y - Last_Delta_Y) *
10000; } else {e_diff_y      = 0;}
294     Last_Delta_Y = Delta_Y;
295
296     Y_motor_CW = 0;  Y_motor_CCW = 0;
297
298     if ((Travel_dist_Y - abs(Delta_Y)) < 5 && Travel_dist_Y > 10){
299         uy = sgn(Delta_Y)*(255 + 6 * (Travel_dist_Y - abs(Delta_Y))) ;
300         if (Delta_Y > 0 && Ymax_limit == 0)      { Y_motor_CW = 1;  Y_motor_CCW = 0; }
301         else if (Delta_Y < 0 && Ymin_limit == 0)    { Y_motor_CW = 0;  Y_motor_CCW = 1; }
302
303     } else if (abs(Delta_Y) < 5 || abs(Delta_Y) < 5 + Travel_dist_Y*0.2){
304         uy = Kpy * Delta_Y + Kdy * e_diff_y;
305         if (uy > 0 && Ymax_limit == 0)      { Y_motor_CW = 1;  Y_motor_CCW = 0; }
306         else if (uy < 0 && Ymin_limit == 0)    { Y_motor_CW = 0;  Y_motor_CCW = 1; }
307
308     } else {
309         uy = sgn(Delta_Y)*255;
310         if (Delta_Y > 0 && Ymax_limit == 0)      { Y_motor_CW = 1;  Y_motor_CCW = 0; }
311         else if (Delta_Y < 0 && Ymin_limit == 0)    { Y_motor_CW = 0;  Y_motor_CCW = 1; }
312     }
313
314     Y_motor_PWM = (int) fabs(uy);
315     if (Y_motor_PWM > 255) {Y_motor_PWM = 255;} else if (Y_motor_PWM < 150){Y_motor_PWM = 0;}
316     setMotory();
317 }
318
319 }
320
321 if(Plotter_status != "Idle" && X_stable_Micros > Axis_stable_time && Y_stable_Micros > Axis_stable_time){
322     X_motor_CW = 0;  X_motor_CCW = 0; X_motor_PWM = 0;
323     Y_motor_CW = 0;  Y_motor_CCW = 0; Y_motor_PWM = 0;
324     e_integral_x = 0;
325     e_integral_y = 0;
326     setMotorx();
327     setMotory();
328     if (Plotter_status == "Jogging") { Plotter_status = "Idle";}
329 }
330 }
331 }
332 }

```

WebUI Functions.h

```
1 //=====
2 //handle function: send webpage to client
3 //=====
4 void webpage()
5 {
6     server.send(200,"text/html", webpageCode);
7 }
8
9 //=====
10 //Set motors
11 //=====
12 void setMotorx(){
13     if(Plotter_status != "E-Stop !!"){
14         ledcWrite (pwmChannel_X, X_motor_PWM );
15         digitalWrite (MOTOR_X_CW, X_motor_CW );
16         digitalWrite (MOTOR_X_CCW, X_motor_CCW );
17     } else if (Plotter_status == "E-Stop !" || Plotter_status == "Paused" ){
18         ledcWrite (pwmChannel_X, 0 );
19         digitalWrite (MOTOR_X_CW, 0 );
20         digitalWrite (MOTOR_X_CCW, 0 );
21     }
22 }
23
24 void setMotory(){
25     if(Plotter_status != "E-Stop !!"){
26         ledcWrite (pwmChannel_Y, Y_motor_PWM );
27         digitalWrite (MOTOR_Y_CW, Y_motor_CW );
28         digitalWrite (MOTOR_Y_CCW, Y_motor_CCW );
29     } else if (Plotter_status == "E-Stop !" || Plotter_status == "Paused" ){
30         ledcWrite (pwmChannel_Y, 0 );
31         digitalWrite (MOTOR_Y_CW, 0 );
32         digitalWrite (MOTOR_Y_CCW, 0 );
33     }
34 }
35
36
37 //=====
38 //sign of variable
39 //=====
40 int sgn(float val) {
41     if (val > 0) return 1;
42     return -1;
43 }
44
45 //=====
46 //Mark a point
47 //=====
48 void point() {
49     digitalWrite(Z_solenoid, HIGH);
50     delay(3000);
51     digitalWrite(Z_solenoid, LOW);
52 }
53
54 //=====
55 //Calculate haus points 1-5
56 //=====
57 int Calc_nik_points()
58 {
59     if(web_Start_pos == 1) {
60         Nikolaus_Haus[0][0] = X_coordinate; Nikolaus_Haus[0][1] = Y_coordinate;
61         Nikolaus_Haus[1][0] = X_coordinate + web_House_size; Nikolaus_Haus[1][1] = Y_coordinate;
62         Nikolaus_Haus[2][0] = X_coordinate + web_House_size; Nikolaus_Haus[2][1] = Y_coordinate + web_House_size;
63         Nikolaus_Haus[3][0] = X_coordinate; Nikolaus_Haus[3][1] = Y_coordinate + web_House_size;
64         Nikolaus_Haus[4][0] = X_coordinate + web_House_size/2; Nikolaus_Haus[4][1] = Y_coordinate + web_House_size * 1.5;
65     }
66     else if((web_Start_pos == 2)) {
67         Nikolaus_Haus[0][0] = X_coordinate - web_House_size; Nikolaus_Haus[0][1] = Y_coordinate;
68         Nikolaus_Haus[1][0] = X_coordinate; Nikolaus_Haus[1][1] = Y_coordinate;
69         Nikolaus_Haus[2][0] = X_coordinate; Nikolaus_Haus[2][1] = Y_coordinate + web_House_size;
70         Nikolaus_Haus[3][0] = X_coordinate - web_House_size; Nikolaus_Haus[3][1] = Y_coordinate + web_House_size;
71         Nikolaus_Haus[4][0] = X_coordinate - web_House_size/2; Nikolaus_Haus[4][1] = Y_coordinate + web_House_size * 1.5;
72     }
73 }
74
75 Serial.print("(1 ,2 ,Plot_area_xmax ,5 ,Plot_area_ymax) = ( ");
76 Serial.print(Nikolaus_Haus[0][0]);
77 Serial.print(" ,");
78 Serial.print(Nikolaus_Haus[1][0]);
79 Serial.print(" ,");
80 Serial.print(Plot_area_xmax);
81 Serial.print(" ,");
82 Serial.print(Nikolaus_Haus[4][1]);
83 Serial.print(" ,");
84 Serial.print(Plot_area_ymax);
85 Serial.println(" )");
86
87 if (Nikolaus_Haus[0][0]<0 || Nikolaus_Haus[1][0]>Plot_area_xmax || Nikolaus_Haus[4][1]>Plot_area_ymax){return 0;}
88 else {return 1;}
89
```

```

90 }
91
92 //=====
93 //function process event: new data received from client
94 //=====
95 void websocketEvent(byte num, WStype_t type, uint8_t * payload, size_t length) { // the parameters of this
96 // callback function are always the same -> num: id of the client who send the event, type: type of message, payload: actual data
97 // sent and length: length of payload
98     switch (type) { // switch on the type of information sent
99         case WStype_DISCONNECTED: // if a client is disconnected, then type == WStype_DISCONNECTED
100             Serial.println("Client " + String(num) + " disconnected");
101             clients_connected--;
102             break;
103         case WStype_CONNECTED: // if a client is connected, then type == WStype_CONNECTED
104             Serial.println("Client " + String(num) + " connected");
105             clients_connected++;
106             // optionally you can add code here what to do when connected
107             break;
108         case WStype_TEXT: // if a client has sent data, then type == WStype_TEXT
109             // try to decipher the JSON string received
110             StaticJsonDocument<200> doc; // create a JSON container
111             DeserializationError error = deserializeJson(doc, payload);
112             if (error) {
113                 Serial.print(F("deserializeJson() failed: "));
114                 Serial.println(error.f_str());
115                 return;
116             }
117             else {
118                 // JSON string was received correctly, so information can be retrieved:
119                 web_Button = doc["Button"];
120                 web_Jog_step_X = doc["Jog_step_X"];
121                 web_Jog_step_Y = doc["Jog_step_Y"];
122                 web_Jog_speed_XY = doc["Jog_speed_XY"];
123                 web_Goto_X = doc["Goto_X"];
124                 web_Goto_Y = doc["Goto_Y"];
125                 web_House_size = doc["House_size"];
126                 web_Draw_speed = doc["Draw_speed"];
127                 web_Start_pos = doc["Start_pos"];
128                 web_Cal_X = doc["Cal_X"];
129                 web_Cal_Y = doc["Cal_Y"];
130                 web_Kpx = doc["Kp_X"];
131                 web_Kpy = doc["Kp_Y"];
132                 web_Kix = doc["Ki_X"];
133                 web_Kiy = doc["Ki_Y"];
134                 web_Kdx = doc["Kd_X"];
135                 web_Kdy = doc["Kd_Y"];
136
137                 if (String(web_Button) == "Jog_X+" && Xmax_limit == 0 && (Plotter_status == "Idle" || Plotter_status ==
138 "Jogging" || Plotter_status == "Warning")){
139                     Target_X = Target_X + web_Jog_step_X;
140                     Target_Y = Target_Y;
141                     Travel_dist_X = abs (Target_X - X_coordinate);
142                     Travel_dist_Y = abs (Target_Y - Y_coordinate);
143                     Plotter_status = "Jogging";
144
145                 } else if (String(web_Button) == "Jog_X-" && Xmin_limit == 0 && (Plotter_status == "Idle" || Plotter_status
146 == "Jogging" || Plotter_status == "Warning")){
147                     Target_X = Target_X - web_Jog_step_X;
148                     Target_Y = Target_Y;
149                     Travel_dist_X = abs (Target_X - X_coordinate);
150                     Travel_dist_Y = abs (Target_Y - Y_coordinate);
151                     Plotter_status = "Jogging";
152
153                 } else if (String(web_Button) == "Jog_Y+" && Ymax_limit == 0 && (Plotter_status == "Idle" || Plotter_status
154 == "Jogging" || Plotter_status == "Warning")){
155                     Target_X = Target_X;
156                     Target_Y = Target_Y + web_Jog_step_Y;
157                     Travel_dist_X = abs (Target_X - X_coordinate);
158                     Travel_dist_Y = abs (Target_Y - Y_coordinate);
159                     Plotter_status = "Jogging";
160
161                 } else if (String(web_Button) == "Jog_Y-" && Ymin_limit == 0 && (Plotter_status == "Idle" || Plotter_status
162 == "Jogging" || Plotter_status == "Warning")){
163                     Target_X = Target_X;
164                     Target_Y = Target_Y - web_Jog_step_Y;
165                     Travel_dist_X = abs (Target_X - X_coordinate);
166                     Travel_dist_Y = abs (Target_Y - Y_coordinate);
167                     Plotter_status = "Jogging";
168
169                 } else if (String(web_Button) == "Pen_UP"){
170                     digitalWrite(Z_solenoid, LOW);
171
172                 } else if (String(web_Button) == "Pen_Down"){
173                     digitalWrite(Z_solenoid, HIGH);
174                     Pen_Down_Time = millis();
175
176                 } else if (String(web_Button) == "X0" && Xmin_limit == 0 && (Plotter_status == "Idle" || Plotter_status ==
177 "Jogging")){
178                     Target_X = 0;
179                     Target_Y = Y_coordinate;
180                     Travel_dist_X = abs (Target_X - X_coordinate);
181                     Travel_dist_Y = abs (Target_Y - Y_coordinate);

```

```

175     Plotter_status = "Jogging";
176
177 } else if (String(web_Button) == "Y0" && Ymin_limit == 0 && (Plotter_status == "Idle" || Plotter_status ==
178 "Jogging")){
179     Target_X      = X_coordinate;
180     Target_Y      = 0;
181     Travel_dist_X  = abs (Target_X - X_coordinate);
182     Travel_dist_Y  = abs (Target_Y - Y_coordinate);
183     Plotter_status = "Jogging";
184
185 } else if (String(web_Button) == "Home" && Xmin_limit == 0 && Ymin_limit == 0 && (Plotter_status == "Idle" ||
186 Plotter_status == "Jogging")){
187     Target_X      = 0;
188     Target_Y      = 0;
189     Travel_dist_X  = abs (Target_X - X_coordinate);
190     Travel_dist_Y  = abs (Target_Y - Y_coordinate);
191     Plotter_status = "Jogging";
192
193 } else if (String(web_Button) == "Go_To_XY" && (Plotter_status == "Idle" || Plotter_status == "Jogging")){
194     if (web_Goto_X >= 0 && web_Goto_X <= Plot_area_xmax && web_Goto_Y >= 0 && web_Goto_Y <= Plot_area_ymax){
195         Target_X      = web_Goto_X;
196         Target_Y      = web_Goto_Y;
197         Travel_dist_X  = abs (Target_X - X_coordinate);
198         Travel_dist_Y  = abs (Target_Y - Y_coordinate);
199         Plotter_status = "Jogging";
200     } else {
201         Inside_draw_area = 0;
202     }
203
204 } else if (String(web_Button) == "Draw" && Plotter_status == "Idle"){
205     if(web_Start_pos == 1){position_index=0;} else {position_index=8;}
206     Inside_draw_area = Calc_nik_points();
207     if(Inside_draw_area==1){
208         digitalWrite(Z_solenoid, HIGH);
209         Pen_Down_Time = millis();
210         delay(50);
211         Target_X = X_coordinate;
212         Target_Y = Y_coordinate;
213         Travel_dist_X  = abs (Target_X - X_coordinate);
214         Travel_dist_Y  = abs (Target_Y - Y_coordinate);
215         Plotter_status = "Plotting...";
216     }
217
218 } else if (String(web_Button) == "Pause" && Plotter_status == "Plotting..."){
219     Plotter_status = "Paused";
220     digitalWrite(Z_solenoid, LOW);
221
222 }else if (String(web_Button) == "Resume" && Plotter_status == "Paused"){
223     digitalWrite(Z_solenoid, HIGH);
224     Pen_Down_Time = millis();
225     delay(50);
226     Plotter_status = "Plotting...";
227
228 } else if (String(web_Button) == "Stop" && Plotter_status != "E-Stop !!"){
229     Plotter_status = "Idle";
230     Target_X = X_coordinate;
231     Target_Y = Y_coordinate;
232     Travel_dist_X  = abs (Target_X - X_coordinate);
233     Travel_dist_Y  = abs (Target_Y - Y_coordinate);
234     e_integral_x = 0;
235     e_integral_y = 0;
236     digitalWrite(Z_solenoid, LOW);
237
238 } else if (String(web_Button) == "E_Stop_btn"){
239     Plotter_status = "E-Stop !!";
240     digitalWrite(Z_solenoid, LOW);
241
242 } else if (String(web_Button) == "Cal_X_btn" && Plotter_status != "E-Stop !" && (Plotter_status == "Idle" ||
243 Plotter_status == "Jogging")){
244     position_index = 0;
245     calibrate_axis = "X";
246     Target_X      = Calibration_sequence[position_index][0];
247     Target_Y      = Y_coordinate;
248     Travel_dist_X  = abs (Target_X - X_coordinate);
249     Travel_dist_Y  = abs (Target_Y - Y_coordinate);
250     Plotter_status = "Calibrating";
251     digitalWrite(Z_solenoid, LOW);
252
253 } else if (String(web_Button) == "Cal_Y_btn" && Plotter_status != "E-Stop !" && (Plotter_status == "Idle" ||
254 Plotter_status == "Jogging")){
255     position_index = 0;
256     calibrate_axis = "Y";
257     Target_X      = X_coordinate;
258     Target_Y      = Calibration_sequence[position_index][1];
259     Travel_dist_X  = abs (Target_X - X_coordinate);
260     Travel_dist_Y  = abs (Target_Y - Y_coordinate);
261     Plotter_status = "Calibrating";
262     digitalWrite(Z_solenoid, LOW);
263
264 } else if (String(web_Button) == "Cal_XY_btn" && Plotter_status != "E-Stop !" && (Plotter_status == "Idle" ||
265 Plotter_status == "Jogging")){
266     position_index = 0;

```

```
262     calibrate_axis = "";
263     Target_X       = Calibration_sequence[position_index][0];
264     Target_Y       = Calibration_sequence[position_index][1];
265     Travel_dist_X  = abs (Target_X - X_coordinate);
266     Travel_dist_Y  = abs (Target_Y - Y_coordinate);
267     Plotter_status = "Calibrating";
268     digitalWrite(Z_solenoid, LOW);
269
270     } else if (String(web_Button) == "Update_pid"){
271         Kpx = web_Kpx;
272         Kpy = web_Kpy;
273         Kix = web_Kix;
274         Kiy = web_Kiy;
275         Kdx = web_Kdx;
276         Kdy = web_Kdy;
277         Serial.print("(Kp X ,Kp Y ,Ki X ,Ki Y) = ( ");
278         Serial.print(Kpx);
279         Serial.print(" ,");
280         Serial.print(Kpy);
281         Serial.print(" ,");
282         Serial.print(Kix);
283         Serial.print(" ,");
284         Serial.print(Kiy);
285         Serial.print(" ,");
286         Serial.print(Kdx);
287         Serial.print(" ,");
288         Serial.print(Kdy);
289         Serial.println(" )");
290     }
291 }
292 //Serial.println("");
293 break;
294 }
295 }
296
297
298
299
```



```
1 //=====
2 //HTML code for webpage
3 //=====
4
5 const char webpageCode[] PROGMEM = R"=====(
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10     <meta charset="UTF-8">
11     <title>Pen Plotter Web UI</title>
12     <style>
13
14       *{
15         margin: 0;
16         padding: 0;
17         box-sizing: border-box;
18         font-family: Arial, Helvetica, sans-serif;
19       }
20       body{
21         background-color: rgb(206, 231, 255);
22         display: flex;
23         flex-direction: column;
24         justify-content: flex-start;
25         align-items: center ;
26         height: 100px;
27       }
28
29       h1{
30         color: rgb(255, 255, 255);
31         background-color: rgb(37, 78, 150);
32         box-shadow: inset -5px 0px 15px rgba(0, 0, 0, 0.623);
33         border-radius: 0px 10px 10px 0px;
34         font-size: 30px;
35         padding: 20px 85px;
36         margin: 0px;
37         text-align: center;
38       }
39
40       .container{
41         width: 800px;
42         margin-top: 100px;
43         background-color: white;
44         padding: 0px;
45         box-shadow: 0 2px 16px rgb(0,0,0,0.3);
46         border-radius: 0px 20px 0px 20px;
47         animation-duration: 0.5s;
48       }
49
50       .H_container{
51         margin: 10px;
52         display: flex;
53         flex-direction: row;
54         background-color: white;
55         border: 0.8px solid rgb(170, 170, 170);
56         padding: 0px;
57         box-shadow: 2px 2px 5px rgba(4, 74, 179, 0.5);
58         border-radius: 10px;
59       }
60
61       .V_container{
62         margin: 10px;
63         display: flex;
64         flex-direction: column;
65         flex-wrap: wrap;
66         justify-content: flex-start;
67         background-color: white;
68         border: 0.8px solid rgb(170, 170, 170);
69         padding: 0px;
70         box-shadow: 2px 2px 5px rgba(4, 74, 179, 0.5);
71         border-radius: 10px;
72       }
73
74       .Status_container{
75         color: #000000;
76         font-size: 20px;
77         font-weight: 600;
78         text-align:center;
79         justify-content: center;
80         padding: 15px 15px 10px 20px;
81         margin: 10px;
82         justify-content: flex-start;
83         background-color: rgb(133, 253, 129);
84         border: 0.8px solid rgb(73, 73, 73);
85         box-shadow: inset 2px 2px 5px rgba(105, 105, 105, 0.5);
86         border-radius: 10px;
87       }
88
89       .Status_container.idle{
90         color: #0045db;
91         background-color: rgb(255, 255, 255);
92       }
```

Web Page HTML Code

```
93
94     .Status_container.plotting{
95         color: #000000;
96         background-color: rgb(133, 253, 129);
97     }
98
99     .Status_container.warning{
100         color: #000000;
101         background-color: rgb(255, 235, 59);
102     }
103
104     .Status_container.estop{
105         color: #ffffff;
106         background-color: rgb(255, 63, 63);
107     }
108
109     .TH_container{
110         margin: 0px;
111         display: flex;
112         flex-direction: row;
113         align-items: center;
114         justify-content: center;
115         background-color: transparent;
116         padding: 0px;
117         border-radius: 10px;
118     }
119
120     .TV_container{
121         margin: 00px;
122         display: flex;
123         flex-direction: column;
124         justify-content: center;
125         background-color: transparent;
126         padding: 5px;
127         border-radius: 10px;
128     }
129
130     .footer_des{
131         width: 800px;
132         font-size: 14px;
133         text-align: end;
134         margin-top: 10px;
135         margin-right: 40px;
136         margin-bottom: 0px;
137         background-color: none;
138         padding: 0px;
139     }
140
141     .tab_box{
142         display: flex;
143         margin-left: 30px;
144         margin-right: 30px;
145         justify-content: space-around;
146         align-items: center;
147         border-bottom: 2px solid rgb(172, 172, 172);
148         position: relative;
149     }
150
151     .tab_box .tab_btn{
152         font-size: 20px;
153         font-weight: 600;
154         color: darkslategray;
155         background: none;
156         border: none;
157         padding: 18px;
158         cursor: pointer;
159         transition: all 0.2s;
160     }
161
162     .tab_box .tab_btn.active{
163         color: rgb(255, 48, 82);
164         font-weight: 600;
165         transition: all 0.2s;
166     }
167
168     .content_box {
169         padding: 20px;
170         margin: 0;
171         border-radius: 0px 20px 0px 20px;
172         transition: all 0.2s;
173     }
174
175     .content_box.estop {
176         background-color: rgb(255, 98, 98);
177         transition: all 0.2s;
178     }
179
180     .content_box.warning {
181         background-color: rgb(255, 235, 59);
182         transition: all 0.2s;
183     }
184
```

```
.content_box .content {
  display: none;
  animation: moving 0.5s ease;
}
@keyframes moving{
  from{transform: translate(50px); opacity: 0;}
  to{transform: translate(0px); opacity: 1;}
}

.content_box .content.active {
  display: flex;
  flex-direction: row;
}

.line{
  position: absolute;
  top: 57px;
  left: 36px;
  width: 92px;
  height: 6px;
  background-color: rgb(255, 69, 100);
  box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.6);
  border-radius: 10px;
  transition: all 0.3s;
}

h2{
  font-size: 25px;
  color: dimgrey;
  margin: 15px 15px 10px 20px;
}

h3{
  font-size: 20px;
  color: dimgrey;
  margin: 15px 15px 10px 20px;
}

p{
  line-height: 25px;
  margin: 0px 20px 5px 20px;
  font-size: 14px;
}

.p1{
  line-height: 28px;
  margin: 0px 20px 15px 20px;
  font-size: 16px;
}

.myTable{
  width: 350px;
  border-collapse: collapse;
  margin: 5px 20px 0px 20px;
}

.myTable td{
  border: solid 1px #747474;
  padding: 10px;
  font-size: 15px;
  font-weight: 600;
  text-align: center;
}

.myTable tr{
  background-color: #FFFFFF;
  color: #535353;
}

.myTable_active{
  background-color: #df5757;
  color: #ffffff;
}

.myTable_ok{
  background-color: #93ff7d;
  color: #000000;
}

.myTable th{
  border: solid 1px #005947;
  padding: 10px;
  color: #FFFFFF;
  background-color: #009879;
  text-align: center;
  font-size: 16px;
}

label {
  color: #007032;
  font-size: 16px;
  font-weight: 600;
```

```
277     }
278
279     input {
280         height: 28px;
281         margin: 0;
282         padding-right: 32px;
283         text-align: center;
284     }
285
286     .blue_button {
287         color: #FFFFFF;
288         width: 22.5%;
289         font-size: 16px;
290         border-radius: 5px;
291         border: solid 1px #3866a3;
292         padding: 10px 10px;
293         margin: 5px;
294         text-shadow: 1.2px 1.2px 1.5px #172e44;
295         box-shadow: inset 1px 1px 0px 0px #BBD AF7;
296         text-decoration: none;
297         cursor: pointer;
298         position: relative;
299         overflow: hidden;
300         font-family: Arial;
301         background: linear-gradient(180deg, #63B8EE 10%, #468CCF 100%);
302         display: inline-block;
303         vertical-align: middle;
304     }
305     .blue_button:active {
306         background: linear-gradient(180deg, #468CCF 10%, #63B8EE 100%);
307     }
308     .blue_button-text {
309         position: relative;
310         display: inline-block;
311     }
312
313     .yellow_button {
314         color: #333333;
315         width: 22.5%;
316         font-size: 16px;
317         border-radius: 5px;
318         border: solid 1px #533300;
319         padding: 10px 10px;
320         margin: 5px;
321         text-shadow: 1px 1px 0px #ffed66;
322         box-shadow: inset 1px 1px 0px 0px #fff6af;
323         text-decoration: none;
324         cursor: pointer;
325         position: relative;
326         overflow: hidden;
327         font-family: Arial;
328         background: linear-gradient(180deg, #ffed64 10%, #ffab23 100%);
329         display: inline-block;
330         vertical-align: middle;
331     }
332     .yellow_button:active {
333         background: linear-gradient(180deg, #ffab23 10%, #ffed64 100%);
334     }
335     .yellow_button-text {
336         position: relative;
337         display: inline-block;
338     }
339
340     .red_button {
341         color: #ffffff;
342         font-weight: 600;
343         width: 80%;
344         height: 65px;
345         font-size: 22px;
346         border-radius: 5px;
347         border: solid 1px #530000;
348         padding: 10px 10px;
349         margin: 5px;
350         margin-top: 30px;
351         text-shadow: 1px 1px 0px #ff6666;
352         box-shadow: inset 1px 1px 0px 0px #ffafaf;
353         text-decoration: none;
354         cursor: pointer;
355         position: relative;
356         overflow: hidden;
357         font-family: Arial;
358         background: linear-gradient(180deg, #ff6464 10%, #ff2323 100%);
359         display: inline-block;
360         vertical-align: middle;
361     }
362     .red_button:active {
363         background: linear-gradient(180deg, #ff2323 10%, #ff6464 100%);
364     }
365     .red_button-text {
366         position: relative;
367         display: inline-block;
368     }
```

```
369
370     .switch-field {
371         display: flex;
372         margin: 10px;
373         overflow: hidden;
374     }
375
376     .switch-field input {
377         position: absolute !important;
378         clip: rect(0, 0, 0, 0);
379         height: 1px;
380         width: 1px;
381         border: 0;
382         overflow: hidden;
383     }
384
385     .switch-field label {
386         background-color: #e7e7e7;
387         color: rgba(0, 0, 0, 0.8);
388         font-size: 14px;
389         line-height: 1;
390         text-align: center;
391         padding: 8px 16px;
392         margin-right: 1px;
393         border: 1px solid rgba(0, 0, 0, 0.2);
394         box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255, 0.1);
395         transition: all 0.1s ease-in-out;
396     }
397
398     .switch-field label:hover {
399         cursor: pointer;
400     }
401
402     .switch-field input:checked + label {
403         background-color: #9bdf73;
404         box-shadow: none;
405     }
406
407     .switch-field label:first-of-type {
408         border-radius: 4px 0 0 4px;
409     }
410
411     .switch-field label:last-of-type {
412         border-radius: 0 4px 4px 0;
413     }
414 </style>
415 </head>
416 <body>
417     <div class = "container" style="">
418         <div class="TH_container" style="justify-content: space-between;">
419             <h1>#14 Pen Plotter Web UI</h1>
420             
422         </div>
423         <div class = "tab_box">
424             <button class="tab_btn active">Home</button>
425             <button class="tab_btn">Calibrate</button>
426             <button class="tab_btn">About</button>
427             <button class="tab_btn">Contact Us</button>
428             <div class="line"></div>
429         </div>
430         <div class="content_box" id="UI_content">
431             <div class="content active">
432                 <div class="V_container" style="width: 395px;">
433                     <h3>Pen Position</h3>
434                     <table class="myTable" id="PositionTable">
435                         <colgroup>
436                             <col style="width:30%">
437                             <col style="width:30%">
438                             <col style="width:40%">
439                         </colgroup>
440                         <thead>
441                             <tr>
442                                 <th>X</th>
443                                 <th>Y</th>
444                                 <th>Pen UP/Down</th>
445                             </tr>
446                         </thead>
447                         <tbody>
448                             <tr>
449                                 <td><span id='X_coor'>--</span></td>
450                                 <td><span id='Y_coor'>--</span></td>
451                                 <td><span id='Pen_up_down'>--</span></td>
452                             </tr>
453                         </tbody>
454                     </table>
455                     <h3>Jog Control</h3>
456                     <div class="TH_container" style="justify-content: space-evenly; padding-right: 8px;" >
457                         <div class="TH_container" style="align-items: center;">
458                             <label for="Jog_step_X">Jog X: &#160;</label>
459                             <div class="TH_container" style="margin-top: 6px;">
```



```
460         <input style="width: 80px;" type="number" id="Jog_step_X" name="Jog_step_X" min="0" max="100" step="10"
461         value="20" placeholder="Step size"><span style="margin-left:-32px;">mm</span>
462     </div>
463 </div>
464 <div class="TH_container" style="align-items: center;">
465     <label for="Jog_step_Y">Jog Y: &#160;</label>
466     <div class="TH_container" style="margin-top: 6px;">
467         <input style="width: 80px;" type="number" id="Jog_step_Y" name="Jog_step_Y" min="0" max="100" step="10"
468         value="20" placeholder="Step size"><span style="margin-left:-32px;">mm</span>
469     </div>
470 </div>
471 <div class="TH_container" style="flex-wrap: wrap; margin-top: 10px;">
472     <button type="button" id='Jog_X_+' class="blue_button" style="padding: 5px;">
473         <span class="blue_button-text" ><font size="+2">&#10236;</font> X +</span>
474     </button>
475     <button type="button" id='Jog_Y_+' class="blue_button" style="padding: 5px;">
476         <span class="blue_button-text"><font size="+2">&#8613;</font> Y +</span>
477     </button>
478     <button type="button" id='Pen_UP' class="blue_button" style="width: 30%; padding: 5px;">
479         <span class="blue_button-text">Pen UP <font size="+2">&#8613;</font></span>
480     </button>
481     <button type="button" id='Jog_X_-' class="blue_button" style="padding: 5px;">
482         <span class="blue_button-text"><font size="+2">&#10235;</font> X -</span>
483     </button>
484     <button type="button" id='Jog_Y_-' class="blue_button" style="padding: 5px;">
485         <span class="blue_button-text"><font size="+2">&#8615;</font> Y -</span>
486     </button>
487     <button type="button" id='Pen_Down' class="blue_button" style="width: 30%; padding: 5px;">
488         <span class="blue_button-text">Pen Down <font size="+2">&#10515;</font></span>
489     </button>
490     <button type="button" id='X0' class="yellow_button">
491         <span class="yellow_button-text"> X0 <font size="+1.5"><b>&#171;</b></font> &#127968;</span>
492     </button>
493     <button type="button" id='Y0' class="yellow_button">
494         <span class="yellow_button-text"> Y0 <font size="+1.5"><b>&#171;</b></font> &#127968;</span>
495     </button>
496     <button type="button" id='Home' class="yellow_button" style="width: 30%;">
497         <span class="yellow_button-text">&#127968; Home XY</span>
498     </button>
499 </div>
500 <div class="TH_container" style="margin-bottom: 20px;">
501     <div class="TV_container" style="margin-right: 10px;">
502         <div class="TH_container" >
503             <label for="Goto_X">Goto X: &#160;</label>
504             <input style="width: 120px;" type="number" id="Goto_X" name="Goto_X" min="0" max="300" step="10"
505             placeholder="X position"><span style="margin-left:-32px;">mm</span>
506         </div>
507         <div class="TH_container" style="margin-top: 6px;">
508             <label for="Goto_Y">Goto Y: &#160;</label>
509             <input style="width: 120px;" type="number" id="Goto_Y" name="Goto_Y" min="0" max="300" step="10"
510             placeholder="Y position"><span style="margin-left:-32px;">mm</span>
511         </div>
512     </div>
513     <button type="button" id="Go_To_XY" class="yellow_button" style="width: 25%;">
514         <span class="yellow_button-text">Go To XY ¶ &nbsp; &#128663;...</span>
515     </button>
516 </div>
517 <div class="V_container" style="width: 395px;">
518     <h3 style="margin-top: 20px; margin-bottom: 20px;">Plotter Status / Warnings</h3>
519     <div id="Status_container" class="Status_container idle" style="margin: 0px 30px 0px 30px; align-items: center;">
520         <span id="status">
521             Initialising...
522         </span>
523     </div>
524     <h3 style="margin-top: 30px; margin-bottom: 12px;">Draw Controls</h3>
525     <div class="TH_container" style="align-items: baseline;">
526         <label for="House_size">House Width</label>
527         <div class="TH_container" style="margin: 6px; justify-content: flex-start;">
528             <input style="width: 98px; padding-right: 40px;" type="number" id="House_size" name="House_size" min="10"
529             max="60" step="5" value="20" ><span style="margin-left:-35px;">mm</span>
530         </div>
531     </div>
532     <div class="TH_container" style="margin: 0px 30px;">
533         <p style="color: #007032; font-size: 16px; font-weight: 600; margin: 0;">Start Position : </p>
534         <div class="switch-field">
535             <input type="radio" id="Start_pos_1st" name="Start_pos" value="1st" checked/>
536             <label for="Start_pos_1st">1st</label>
537             <input type="radio" id="Start_pos_2nd" name="Start_pos" value="2nd" />
538             <label for="Start_pos_2nd">2nd</label>
539         </div>
540     </div>
541     <div class="TH_container" style="flex-wrap: wrap;">
542         <button type="button" id="Draw" class="blue_button">
543             <span class="blue_button-text">Draw <br> &#9998;</span>
544         </button>
545         <button type="button" id="Pause_Resume" class="blue_button" style="width: 90px;">
546             <span class="blue_button-text" id="Pause_Resume_text">Pause <br> ¶</span>
547         </button>
548         <button type="button" id="Stop" class="blue_button">
549             <span class="blue_button-text">Stop <br> &#9724;</span>
```

```
547         </button>
548         <button type="button" id="E_Stop_btn" class="red_button">
549             <span class="red_button-text">&#9940; Emergency Stop !!</span>
550         </button>
551     </div>
552 </div>
553 </div>
554
555 <div class="content">
556     <div class="TV_container" style="width: 760px;">
557         <div class="TH_container">
558             <div class="V_container" style="width: 760px; height: 518.5px; padding-bottom: 10px;" >
559                 <h3>Calibration Instructions</h3>
560                 <p>
561                     <b>Make sure there is paper in the plotting area before beginning the calibration process!!</b><br><br>
562                     <b>Step 1A :</b> To calibrate the X-Axis, Press the Calibrate X Button.<br>
563                     <b>Step 2A :</b> Wait for the machine to finishes the calibration sequence for the X-axis.<br><br>
564                     <b>Step 1B :</b> To calibrate the Y-Axis, Press the Calibrate Y Button.<br>
565                     <b>Step 2B :</b> Wait for the machine to finishes the calibration sequence for the Y-axis.<br><br>
566                     <b>Step 1C :</b> To calibrate both the X and Y-Axis, press the Calibrate XY Button and wait for the machine
                    to complete the calibration sequence.
                    </p>
567                 </div>
568             <div class="V_container" style="width: 100%; height: 96%;">
569                 <h3>Sensor Status</h3>
570                 <table class="myTable" id="SensorTable">
571                     <colgroup>
572                         <col style="width:40%">
573                         <col style="width:30%">
574                         <col style="width:30%">
575                     </colgroup>
576                     <thead>
577                         <tr>
578                             <th>Sensor</th>
579                             <th>Min</th>
580                             <th>Max</th>
581                         </tr>
582                     </thead>
583                     <tbody>
584                         <tr>
585                             <td>Limit Switch <br>X-Axis</td>
586                             <td><span id='X_Lim_min'>OK &#9989;</span></td>
587                             <td><span id='X_Lim_max'>--</span></td>
588                         </tr>
589                         <tr>
590                             <td>Limit Switch <br>Y-Axis</td>
591                             <td><span id='Y_Lim_min'>Active &#128680;</span></td>
592                             <td><span id='Y_Lim_max'>--</span></td>
593                         </tr>
594                         <tr>
595                             <td>Emergency Switch</td>
596                             <td colspan="2"><span id='E_Stop_status' style="font-size: 20px;">&#128721; ACTIVE! &#128219;</span></td>
597                         </tr>
598                     </tbody>
599                 </table>
600                 <h3 style="margin-top: 45px; margin-bottom: 0px;">Calibration Controls</h3>
601                 <div class="TH_container" style="flex-wrap: wrap;">
602                     <button type="button" id="Cal_X_btn" class="blue_button" style="width: 35%; margin-left: 0;">
603                         <span class="blue_button-text">&#128295; Calibrate X</span>
604                     </button>
605                     <button type="button" id="Cal_Y_btn" class="blue_button" style="width: 35%; margin-right: 0;">
606                         <span class="blue_button-text">&#128296; Calibrate Y</span>
607                     </button>
608                 </div>
609                 <button type="button" id="Cal_XY_btn" class="yellow_button" style="width: 290px; height: 60px; margin: 0px
                50px 20px 50px;">
610                     <span class="yellow_button-text" style="font-size: 20px;">&#128295; Calibrate X-Y &#128296;</span>
611                 </button>
612             </div>
613         </div>
614     </div>
615     <div class="H_container" >
616         <h3 style="margin: 15px 10px 15px 20px;">PID <br>Controll</h3>
617         <div class="TH_container" style="justify-content: space-between;">
618             <div class="TH_container" style="width: 400px; margin-bottom: 15px; margin-top: 15px; flex-wrap: wrap">
619                 <label for="Kp_X">Kp X: </label>
620                 <input style="width: 60px; height: 30px; padding-right: 0px; margin: 0px 20px 5px 4px; text-align: right;"
                type="number" id="Kp_X" name="Kp_X" step="0.2" min="0" value="70">
621                 <label for="Ki_X">Ki X: </label>
622                 <input style="width: 60px; height: 30px; padding-right: 0px; margin: 0px 20px 5px 4px; text-align: right;"
                type="number" id="Ki_X" name="Ki_X" step="0.2" min="0" value="100">
623                 <label for="Kd_X">Kd X: </label>
624                 <input style="width: 60px; height: 30px; padding-right: 0px; margin: 0px 20px 5px 4px; text-align: right;"
                type="number" id="Kd_X" name="Kd_X" step="0.2" min="0" value="0.06">
625                 <label for="Kp_Y">Kp Y: </label>
626                 <input style="width: 60px; height: 30px; padding-right: 0px; margin: 0px 20px 0px 5px; text-align: right;"
                type="number" id="Kp_Y" name="Kp_Y" step="0.2" min="0" value="70">
627                 <label for="Ki_Y">Ki Y: </label>
628                 <input style="width: 60px; height: 30px; padding-right: 0px; margin: 0px 20px 0px 4px; text-align: right;"
                type="number" id="Ki_Y" name="Ki_Y" step="0.2" min="0" value="100">
629                 <label for="Kd_Y">Kd Y: </label>
630                 <input style="width: 60px; height: 30px; padding-right: 0px; margin: 0px 20px 0px 4px; text-align: right;"
                type="number" id="Kd_Y" name="Kd_Y" step="0.2" min="0" value="0.06">
```

```
631         </div>
632         <button type="button" id="Update_pid" class="yellow_button" style="width: 30%; height: 50px; margin: 20px;">
633             <span class="yellow_button-text" style="font-size: 20px;">Update PID</span>
634         </button>
635     </div>
636 </div>
637 </div>
638 </div>
639
640 <div class="content">
641     <div class="V_container">
642         <h3>About</h3>
643         <p class="p1">
644             <b>Welcome to PenPlotter WebUI</b>, a project born out of a mechatronics workshop led by university students.
645         </p>
646         <p class="p1">
647             PenPlotter WebUI is a user-friendly web-based interface designed specifically for pen plotters which
648             are remarkable electromechanical device used to create precise and high-quality line drawings of the iconic
649             Nikolaus house.
650         </p>
651         <p class="p1">
652             Tailored for university students between the ages of 20 and 30, PenPlotter WebUI is very simple and
653             straightforward to
654             use with little to no prior knowledge of pen plotters, 3D printers or programming required. We've designed our
655             interface
656             with simplicity and efficiency in mind, allowing you to start drawing straight away.
657         </p>
658         <p class="p1">
659             The PenPlotter provides an accessible and engaging platform for indoor use. Whether
660             it's in an office or an educational institution, our platform encourages you to explore the potential of pen
661             plotting technology.
662         </p>
663         <p class="p1">
664             We highly value your input, as it shapes the future of PenPlotter WebUI. Our team aims to continuously
665             improve and develop the platform based on user feedback. We encourage you to contribute your ideas, report any
666             issues you encounter,
667             or even become part of our open-source community.
668         </p>
669         <p class="p1">
670             Experience the joy of pen plotting with PenPlotter WebUI, created by passionate university students with a focus
671             on mechatronics
672             and creativity. Explore the precise art of line drawings, and witness your Nikolaus house take shape.
673         </p>
674     </div>
675 </div>
676
677 <div class="content">
678     <div class="V_container">
679         <h3>Contact Us</h3>
680         <p class="p1">
681             Thank you for your interest in PenPlotter WebUI. We value your feedback and are here to assist you with
682             any inquiries or concerns you may have.
683         </p>
684         <p class="p1">
685             For general inquiries or information about PenPlotter WebUI, please feel free to reach out to us via email
686             at <b>r.joshi@stud.fh-sm.de</b> We will do our best to respond to your message promptly and provide you with the
687             information you need.
688         </p>
689         <p class="p1">
690             If you require technical support or have specific questions regarding the usage of PenPlotter WebUI, our
691             team is ready to assist you. Please send your support-related queries to <b>r.joshi@stud.fh-sm.de</b> We will
692             prioritize
693             your request and ensure that you receive the necessary assistance to make the most out of the platform.
694         </p>
695         <p class="p1">
696             We greatly appreciate your feedback and suggestions for improving PenPlotter WebUI. If you have ideas for new
697             features, enhancements, or any other ways we can enhance your experience, please don't hesitate to let us know.
698             Your input plays a crucial role in shaping the future of our project.
699         </p>
700         <p class="p1">
701             PenPlotter WebUI is a project fueled by your passion and creativity. We are excited to have you as part of our
702             community and look forward to hearing from you. Your support and involvement are vital to our continued success.
703         </p>
704         <p class="p1">
705             <b>Thank you for choosing PenPlotter WebUI.</b>
706         </p>
707     </div>
708 </div>
709 </div>
710
711 <p class="footer_des">Developed by Team #14 MERO 22/23 HS Schmalkalden</p>
712 <p class="footer_des" style="margin-top: 0px;">Web UI V1.5 (HTML, CSS, JS)</p>
713 <p class="footer_des" style="margin-top: 0px;">ESP32 Web Server</p>
714 <script>
715     const tabs= document.querySelectorAll('.tab_btn');
716     const all_content= document.querySelectorAll('.content');
717
718     tabs.forEach((tab, index)=>{
719         tab.addEventListener('click', (e)=>{
720             tabs.forEach(tab=>{tab.classList.remove('active')});
721             tab.classList.add('active');
722         });
723     });
724 </script>
```

```

715     var line=document.querySelector('.line');
716     line.style.width = e.target.offsetWidth + "px";
717     line.style.left = e.target.offsetLeft + "px";
718
719     all_content.forEach(content=>{content.classList.remove('active')});
720     all_content[index].classList.add('active');
721 })
722 })
723
724 var Socket;
725 let btns = document.querySelectorAll('button');
726
727 btns.forEach(function (i) {
728     i.addEventListener('click', function() {
729         if(i.id != "" && document.getElementById('status').innerHTML != "E-Stop !!"){
730             if(i.innerText == "Pause\n■"){
731                 var Button_ID = "Pause";
732             } else if (i.innerText == "Resume\n□") {
733                 var Button_ID = "Resume";
734             } else {
735                 var Button_ID = i.id;
736             };
737
738             var msg = { Button: Button_ID, };
739
740             if(Button_ID == "Pause" || Button_ID == "Stop" || Button_ID == "E_Stop_btn" ){
741                 msg = { Button: Button_ID, };
742             } else if (Button_ID == "Draw" || Button_ID == "Resume" ){
743                 msg = {
744                     Button: Button_ID,
745                     Start_pos:    Start_pos_1st.checked*1 + Start_pos_2nd.checked*2,
746                     House_size:  House_size.valueAsNumber,
747                 };
748
749             } else if (Button_ID == "Update_pid"){
750                 msg = {
751                     Button: Button_ID,
752                     Kp_X :      Kp_X.valueAsNumber,
753                     Ki_X :      Ki_X.valueAsNumber,
754                     Kd_X :      Kd_X.valueAsNumber,
755                     Ki_Y :      Ki_Y.valueAsNumber,
756                     Kp_Y :      Kp_Y.valueAsNumber,
757                     Kd_Y :      Kd_Y.valueAsNumber,
758                 }
759             } else {
760                 msg = {
761                     Button: Button_ID,
762                     Jog_step_X:  Jog_step_X.valueAsNumber,
763                     Jog_step_Y:  Jog_step_Y.valueAsNumber,
764                     Goto_X:      Goto_X.valueAsNumber,
765                     Goto_Y:      Goto_Y.valueAsNumber,
766                 }
767             };
768
769             console.log(msg);
770             Socket.send(JSON.stringify(msg));
771         };
772     });
773 });
774
775 function init() {
776     Socket = new WebSocket('ws://' + window.location.hostname + ':81/');
777     Socket.onmessage = function(event) {
778         processCommand(event);
779     };
780 }
781
782 function processCommand(event) {
783     var obj = JSON.parse(event.data);
784     document.getElementById('X_coor').innerHTML = Math.round((obj.X_coor + Number.EPSILON) * 100) / 100 + " mm";
785     document.getElementById('Y_coor').innerHTML = Math.round((obj.Y_coor + Number.EPSILON) * 100) / 100 + " mm";
786     document.getElementById('Pen_up_down').innerHTML = obj.Pen_up_down;
787     document.getElementById('status').innerHTML = obj.status;
788     switch(obj.status) {
789         case "Plotting...":
790             document.getElementById('UI_content').className = "content_box";
791             document.getElementById('Status_container').className = "Status_container plotting";
792             break;
793         case "Warning":
794             document.getElementById('UI_content').className = "content_box warning";
795             document.getElementById('Status_container').className = "Status_container warning";
796             break;
797         case "E-Stop !!":
798             document.getElementById('UI_content').className = "content_box estop";
799             document.getElementById('Status_container').className = "Status_container estop";
800             break;
801         default:
802             document.getElementById('UI_content').className = "content_box";
803             document.getElementById('Status_container').className = "Status_container idle";
804     };
805     if(obj.status == "Paused"){
806         document.getElementById('Pause_Resume_text').innerHTML = "Resume <br> □";

```

```
807     } else if (obj.status == "Plotting...") {
808         document.getElementById('Pause_Resume_text').innerHTML = "Pause <br> ⏸";
809     };
810     document.getElementById('X_Lim_min').innerHTML = obj.X_Lim_min;
811     document.getElementById('Y_Lim_min').innerHTML = obj.Y_Lim_min;
812     document.getElementById('X_Lim_max').innerHTML = obj.X_Lim_max;
813     document.getElementById('Y_Lim_max').innerHTML = obj.Y_Lim_max;
814     document.getElementById('E_Stop_status').innerHTML = obj.E_Stop_status;
815     if(obj.Inside_draw_area == "Error"){
816         alert(" ERROR ⚠ : Drawing/Move not possible. 💔 \n\n Figure/point is out of bounds🚫. Change starting position or
            house size and try again later.");
817     };
818     console.log(event.data);
819 }
820 window.onload = function(event) {
821     init();
822 }
823 </script>
824 </body>
825 </html>
826
827 )=====";
828
829
```


OLED Display Code

```

1  #define LOGO_HEIGHT    40
2  #define LOGO_WIDTH    40
3  static const unsigned char PROGMEM logo_bmp[] ={
4      0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x00, 0x00,
5      0x00, 0x00, 0x07, 0xff, 0x80, 0x00, 0x00, 0x3f, 0xf9, 0x80, 0x00, 0x01, 0xff,
6      0x0f, 0xfe, 0x01, 0x80, 0x00, 0x3f, 0xf0, 0x01, 0x80, 0x00, 0xff, 0x80, 0x01,
7      0x00, 0x07, 0x80, 0x0f, 0xf8, 0x00, 0x3f, 0x80, 0x1f, 0xc0, 0x03, 0xff, 0x80, 0x7f, 0x80, 0x1f,
8      0xfd, 0x80, 0xfe, 0x00, 0xff, 0xc1, 0x80, 0xfc, 0x07, 0xfe, 0x01, 0x81, 0xf8, 0x1f, 0xf0, 0x01,
9      0x83, 0xf0, 0x7f, 0x80, 0x01, 0x83, 0xe1, 0xfe, 0x00, 0x07, 0x83, 0xc7, 0xf0, 0x00, 0x3f, 0x81,
10     0xdf, 0xc0, 0x03, 0xff, 0x81, 0xff, 0x80, 0x1f, 0xfd, 0x80, 0xfe, 0x00, 0xff, 0xc1, 0x80, 0xfc,
11     0x03, 0xfe, 0x01, 0x81, 0xf8, 0x1f, 0xf0, 0x01, 0x83, 0xf0, 0x7f, 0x80, 0x01, 0x83, 0xe1, 0xfe,
12     0x00, 0x01, 0x83, 0xc7, 0xf0, 0x00, 0x01, 0x81, 0xdf, 0xc0, 0x00, 0x01, 0x81, 0xff, 0x80, 0x00,
13     0x01, 0x80, 0xfe, 0x00, 0x00, 0x01, 0x80, 0xfc, 0x00, 0x00, 0x01, 0x81, 0xf0, 0x00, 0x00, 0x01,
14     0x83, 0xf0, 0x00, 0x00, 0x01, 0x83, 0xe0, 0x00, 0x00, 0x01, 0x83, 0xc0, 0x00, 0x00, 0x01, 0x81,
15     0xc0, 0x00, 0x00, 0x01, 0x81, 0xc0, 0x00, 0x00, 0x01, 0x80, 0xc0, 0x00, 0x00, 0x01, 0x80, 0x40,
16     0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff
17 };
18
19 const unsigned char PROGMEM logo_estop[] = {
20     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00,
21     0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00,
22     0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x0c, 0x18,
23     0x30, 0x00, 0x00, 0x06, 0x18, 0x70, 0x00, 0x10, 0x06, 0x18, 0x60, 0x0c, 0x1c, 0x03, 0x18, 0xc0,
24     0x38, 0x0e, 0x01, 0x00, 0xc0, 0x70, 0x03, 0x80, 0x00, 0x01, 0xc0, 0x00, 0xe0, 0x00, 0x07, 0x80,
25     0x00, 0x71, 0xff, 0x8e, 0x00, 0x00, 0x13, 0xff, 0xc8, 0x00, 0x00, 0x03, 0x7f, 0xc0, 0x00, 0x00,
26     0x02, 0x7f, 0xc0, 0x00, 0x00, 0x02, 0x7f, 0xc0, 0x00, 0x07, 0xe2, 0x7f, 0xe7, 0xe0, 0x07, 0xe6,
27     0x7f, 0xe7, 0xe0, 0x00, 0x06, 0x7f, 0xe0, 0x00, 0x00, 0x06, 0xff, 0xe0, 0x00, 0x00, 0x04, 0xff,
28     0xe0, 0x00, 0x00, 0x04, 0xff, 0xe0, 0x00, 0x00, 0x04, 0xff, 0xf0, 0x00, 0x00, 0x0c, 0xff, 0xf0,
29     0x00, 0x00, 0x0c, 0xff, 0xf0, 0x00, 0x00, 0x0f, 0xff, 0xf0, 0x00, 0x00, 0x0f, 0xff, 0xf0, 0x00,
30     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x7f, 0xff, 0xfe, 0x00, 0x00,
31     0x7f, 0xff, 0xfe, 0x00, 0x00, 0x7f, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
32     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
33 };
34
35
36 void Welcome_Screen(){
37     display.clearDisplay();
38
39     display.drawBitmap( 4, 20, logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
40
41     display.setTextColor(WHITE);
42     display.setCursor(1,1);
43     display.setTextSize(2);
44     display.println("PEN");
45     display.setCursor(44,1);
46     display.println("PLOTTER");
47     display.setCursor(50,20);
48     display.setTextSize(1);
49     display.println("HOCHSCHULE");
50     display.setCursor(50,30);
51     display.println("SCHMALKALDEN");
52     display.setCursor(60,52);
53     display.println("BY:TEAM #14");
54
55     display.display();
56     // Invert and restore display, pausing in-between
57     display.invertDisplay(true);
58     delay(4000);
59     display.invertDisplay(false);
60     delay(4000);
61
62     display.clearDisplay();
63
64     display.setTextColor(WHITE);
65     display.setCursor(1,1);
66     display.setTextSize(2);
67     display.println(">>STARTING");
68     display.setCursor(4,20);
69     display.setTextSize(1);
70     display.println("- Initialising WEB      Server.....");
71     display.display();
72 }
73
74 void Update_Display(){
75     display.clearDisplay();
76
77     if (Plotter_status != "E-Stop !!" && clients_connected==0){
78
79         display.setTextColor(WHITE);
80
81         display.setCursor(1,1);
82         display.setTextSize(2);
83         display.println("PEN");
84         display.setCursor(44,1);
85         display.println("PLOTTER");
86
87         display.setCursor(0,18);
88         display.println("LOCAL IP:");
89
90         display.setCursor(0,35);
91         display.println(WiFi.localIP());
92

```

```

93 } else if (Plotter_status != "Warning" && Plotter_status != "E-Stop !!"){
94
95     display.setTextColor(WHITE);
96
97     display.setCursor(1,1);
98     display.setTextSize(2);
99     display.println("PEN");
100    display.setCursor(44,1);
101    display.println("PLOTTER");
102
103    display.setCursor(0,18);
104    display.println("X:");
105    display.setCursor(25,18);
106    display.println(X_coordinate);
107    display.setCursor(90,18);
108    display.println(" mm");
109
110    display.setCursor(0,35);
111    display.println("Y:");
112    display.setCursor(25,35);
113    display.println(Y_coordinate);
114    display.setCursor(90,35);
115    display.println(" mm");
116
117    if (Plotter_status == "Plotting..."){display.fillRect(0, 51, 128, 13, WHITE); display.setTextColor(BLACK);}
118
119    display.setCursor(0,54);
120    display.setTextSize(1);
121    display.println("STATUS:");
122    display.setCursor(45,54);
123    display.setTextSize(1);
124    display.println(Plotter_status);
125
126 } else if (Plotter_status == "E-Stop !!") {
127
128     display.fillRect(0, 0, 128, 16, WHITE);
129     display.setTextColor(BLACK);
130
131     display.setCursor(5,1);
132     display.setTextSize(2);
133     display.println(">>E-STOP<<");
134
135     display.drawBitmap( 2, 22, logo_estop, LOGO_WIDTH, LOGO_HEIGHT, 1);
136
137     display.setTextColor(WHITE);
138
139     display.setCursor(44,20);
140     display.setTextSize(1);
141     display.println("E-STOP ACTIVE!");
142
143     display.setCursor(46,40);
144     display.setTextSize(1);
145     display.println(" LONGPRESS");
146     display.setCursor(44,50);
147     display.setTextSize(1);
148     display.println("TO DEACTIVATE");
149
150 } else if (Plotter_status == "Warning"){
151     display.fillRect(0, 0, 128, 16, WHITE);
152     display.setTextColor(BLACK);
153
154     display.setCursor(11,1);
155     display.setTextSize(2);
156     display.println("!WARNING!");
157
158     display.setTextColor(WHITE);
159
160     display.setCursor(8,18);
161     display.setTextSize(2);
162     display.println("X");
163     display.setCursor(21,18);
164     display.setTextSize(1);
165     display.println("MIN");
166     display.setCursor(43,18);
167     display.setTextSize(2);
168     display.println(Xmin_limit);
169
170     display.setCursor(75,18);
171     display.setTextSize(2);
172     display.println("X");
173     display.setCursor(89,18);
174     display.setTextSize(1);
175     display.println("MAX");
176     display.setCursor(110,18);
177     display.setTextSize(2);
178     display.println(Xmax_limit);
179
180     display.setCursor(8,35);
181     display.setTextSize(2);
182     display.println("Y");
183     display.setCursor(21,35);
184     display.setTextSize(1);

```

```
185     display.println("MIN");
186     display.setCursor(43,35);
187     display.setTextSize(2);
188     display.println(Ymin_limit);
189
190     display.setCursor(75,35);
191     display.setTextSize(2);
192     display.println("Y");
193     display.setCursor(89,35);
194     display.setTextSize(1);
195     display.println("MAX");
196     display.setCursor(110,35);
197     display.setTextSize(2);
198     display.println(Ymax_limit);
199
200     display.setCursor(6,53);
201     display.setTextSize(1);
202     display.println("LIMIT SWITCH ACTIVE!");
203 }
204 display.display();//Finally display the created image
205 }
```