Pen Plotter Main Code

```
//================================
//ESP32 WebSocket Server: Pen Plotter Web UI
//by: Rajat Joshi
//================================
#include <WiFi.h>
#include <WebServer.h>
#include <WebSocketsServer.h>
#include <ArduinoJson.h>
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//----------NETWORK DETAILS---------------------
const char* ssid = "Bereich 51 24";
const char* password = "BB1010e72#16!#6!5bjGu";
int clients_connected = 0;

//---------- WEB SERVER AND WEBSOCKET ------------
WebServer server(80);
WebSocketsServer webSocket = WebSocketsServer(81);

//------ Task handle for sensor input ------------
TaskHandle_t sensor_input;

//-------------- DEFINE IO PINS -----------------
#define MOTOR_Y_CW  13
#define MOTOR_Y_CCW 12
#define MOTOR_Y_PWM 27
#define MOTOR_X_CW  A5
#define MOTOR_X_CCW A1
#define MOTOR_X_PWM 18
#define Ylim_min    21
#define Ylim_max    17
#define Xlim_min    16
#define Xlim_max    19
#define Ymotor_A0   A3
#define Ymotor_A1   A2
#define Xmotor_A0   32
#define Xmotor_A1   33
#define E_stop_btn  14
#define Z_solenoid  15
#define SCREEN_WIDTH 128    // OLED display width, in pixels
#define SCREEN_HEIGHT 64    // OLED display height, in pixels
#define OLED_RESET     -1   // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Setting PWM properties
const int freq = 2000;
const int pwmChannel_X = 0;
const int pwmChannel_Y = 2;
const int resolution = 8;

const char* web_Button = "";
int web_Jog_step_X    = 20;
int web_Jog_step_Y    = 20;
int web_Jog_speed_XY  = 1000;
int web_Goto_X        = 0;
int web_Goto_Y        = 0;
int web_House_size    = 20;
int web_Draw_speed    = 1000;
int web_Start_pos     = 1;
float web_Cal_X       = 0;
float web_Cal_Y       = 0;
float web_Kpx         = 1;
float web_Kpy         = 1;
float web_Kix         = 0;
float web_Kiy         = 0;
float web_Kdx         = 0;
float web_Kdy         = 0;


int Ymin_limit = 1;
int Ymax_limit = 1;
int Xmin_limit = 1;
int Xmax_limit = 1;
int E_stop    = 1;
int Pen_state  = 1;
const char* Plotter_status = "Idle";
const char* calibrate_axis = "";

float Plot_area_xmax = 220;
float Plot_area_ymax = 245;
float X_coordinate   = 0;
float Y_coordinate   = 0;
int X_dir           = 0;
int Y_dir           = 0;
int X_count         = 0;
int X_count_dec     = 0;
int Y_count         = 0;
int Y_count_dec     = 0;

```

```
float Target_X      = 0;
float Target_Y      = 0;
float Delta_X       = 0;
float Delta_Y       = 0;
float Travel_dist_X  = 10;
float Travel_dist_Y  = 10;

int   X_motor_CW    = 0;
int   X_motor_CCW   = 0;
int   Y_motor_CW    = 0;
int   Y_motor_CCW   = 0;
int   X_motor_PWM   = 0;
int   Y_motor_PWM   = 0;
float d_X = 0.25;
float d_Y = 0.25;
float Kpx = 70;
float Kpy = 70;
float Kix = 100;
float Kiy = 100;
float Kdx = 0.06;
float Kdy = 0.06;

float e_integral_x = 0;
float e_integral_y = 0;
float e_diff_x     = 0;
float e_diff_y     = 0;
float Last_Delta_X = 0;
float Last_Delta_Y = 0;
float ux = 0;
float uy = 0;

float Calibration_sequence[5][2] = { {-220, -245}, {0, 0}, {-10, -10}, {210, 235}, {0, 0} };

int   position_index       = 0; // position 0-8 in drawing sequence and 0-4 Calibration_sequence
int   Draw_sequence_index  = 0;
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},
                              {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},
                              {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},
                              {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},
                              {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},
                              {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},
                              {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},
                              {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},
                              {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},
                              {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},
                              {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}};
float Nikolaus_Haus[5][2]  = { {0, 0}, {20, 0}, {20, 20}, {0, 20}, {10, 30} };
int   Inside_draw_area     =1;

int           Web_update_interval = 200;            // send data to the client every 1000ms -> 1s
unsigned long Web Prev Millis = 0;                  // we use the "millis()" command for time reference and this will output an unsigned long
int           Sensor_update_interval = 100;         // send data to the client every 1000ms -> 1s
unsigned long Sensor_Prev_Micros = 0;               // we use the "millis()" command for time reference and this will output an unsigned long
unsigned long Serial_update_Micros = 0;             // we use the "millis()" command for time reference and this will output an unsigned long

int Estop_reset_time = 2e6;
unsigned long E_stop_Micros = 0;                    // we use the "millis()" command for time reference and this will output an unsigned long

int        Axis_stable_time = 5e4;
unsigned long X_stable_Micros = 0;
unsigned long Y_stable_Micros = 0;                  // we use the "millis()" command for time reference and this will output an unsigned long

int           Max_Pen_Down_Time = 44000;            // send data to the client every 1000ms -> 1s
unsigned long Pen_Down_Time = 0;
//------------ DEFINE FUNCTION FILES-------------
#include "WebUI_HTML.h"
#include "Display.h"
#include "Functions.h"
#include "Input_signal_processing.h"



//=====================================================================
void setup()
{

```

```
    Serial.begin(115200);
    //-----------------------------------------------
    if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {    // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
      Serial.println(F("SSD1306 allocation failed"));
      for(;;);                 // Don't proceed, loop forever
    }
    Welcome_Screen();
    //-----------------------------------------------
    WiFi.begin(ssid, password);
    Serial.println("Establishing connection to WiFi with SSID: " + String(ssid));
    while(WiFi.status() != WL_CONNECTED){Serial.print("."); delay(500);}
    WiFi.mode(WIFI_STA);
    Serial.println();
    Serial.print("Local IP: ");
    Serial.println(WiFi.localIP());
    display.setCursor(4,40);
    display.println("- Local IP:");
    display.setCursor(45,50);
    display.println(WiFi.localIP());
    display.display();
    delay(3000);
    display.clearDisplay();
    //-----------------------------------------------
    server.on("/", webpage);
    //-----------------------------------------------
    server.begin(); webSocket.begin();
    webSocket.onEvent(webSocketEvent);
    //-----------------------------------------------
    pinMode(Xlim_min,  INPUT_PULLUP);
    pinMode(Xlim_max,  INPUT_PULLUP);
    pinMode(Ylim_min,  INPUT_PULLUP);
    pinMode(Ylim_max,  INPUT_PULLUP);
    pinMode(Xmotor_A0, INPUT_PULLUP);
    pinMode(Xmotor_A1, INPUT_PULLUP);
    pinMode(Ymotor_A0, INPUT_PULLUP);
    pinMode(Ymotor_A1, INPUT_PULLUP);
    pinMode(E_stop_btn,INPUT_PULLUP);
    pinMode(Z_solenoid,   OUTPUT   );

    pinMode       (MOTOR_X_CW,   OUTPUT   );
    pinMode       (MOTOR_X_CCW,  OUTPUT   );
    pinMode       (MOTOR_Y_CW,   OUTPUT   );
    pinMode       (MOTOR_Y_CCW,  OUTPUT   );
    ledcSetup     (pwmChannel_X, freq, resolution);
    ledcSetup     (pwmChannel_Y, freq, resolution);
    ledcAttachPin (MOTOR_X_PWM,  pwmChannel_X);
    ledcAttachPin (MOTOR_Y_PWM,  pwmChannel_Y);

    Xmin_limit = digitalRead(Xlim_min);
    Xmax_limit = digitalRead(Xlim_max);
    Ymin_limit = digitalRead(Ylim_min);
    Ymax_limit = digitalRead(Ylim_max);
    E_stop     = digitalRead(E_stop_btn);
    digitalWrite(Z_solenoid, LOW);
    setMotorx();
    setMotorx();
    //-----------------------------------------------
    xTaskCreatePinnedToCore(
      sensor_input_func,        /* Function to implement the task */
      "sensor_input",           /* Name of the task */
      10000,                   /* Stack size in words */
      NULL,                     /* Task input parameter */
      0,                        /* Priority of the task */
      &sensor_input,             /* Task handle. */
    0);                         /* Core where the task should run */
    //-----------------------------------------------
    // Target_X       = Calibration_sequence[position_index][0];
    // Target_Y       = Calibration_sequence[position_index][1];
    // Travel_dist_X  = abs (Target_X - X_coordinate);
    // Travel_dist_Y  = abs (Target_Y - Y_coordinate);
    // Plotter_status = "Calibrating";

  }



  //================================================================
  void loop(){
    webSocket.loop(); server.handleClient();

    if ((unsigned long)(millis() - Web_Prev_Millis) >= Web_update_interval) { // check if "interval" ms has passed since last time the clients were updated

      Web_Prev_Millis = millis();                       // reset previousMillis
      String jsonString = "";                          // create a JSON string for sending data to the client
      StaticJsonDocument<200> doc;                     // create a JSON container
      JsonObject object = doc.to<JsonObject>();         // create a JSON Object
      object["X_coor"] = X_coordinate;                  // write data into the JSON object -> I used "rand1" and "rand2" here, but you can use anything else
      object["Y_coor"] = Y_coordinate;
```

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

    serializeJson(doc, jsonString);                    // convert JSON object to string
    //Serial.println(jsonString);                     // print JSON string to console for debug purposes (you can comment this
    out)
    webSocket.broadcastTXT(jsonString);                // send JSON string to clients

    Update_Display();


    if (X_stable_Micros > Axis_stable_time && Y_stable_Micros > Axis_stable_time ){

      if (Plotter_status == "Plotting..." ){

        Serial.println("Point reached");

        //point();

        if (web_Start_pos == 1 and position_index != 8){
          position_index++;
          Target_X         = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][0];
          Target_Y         = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][1];
          Travel_dist_X    = abs (Target_X - X_coordinate);
          Travel_dist_Y    = abs (Target_Y - Y_coordinate);

        } else if (web_Start_pos == 1 and position_index == 8){
          position_index=0;
          digitalWrite(Z_solenoid, LOW);
          if(Draw_sequence_index == 43){Draw_sequence_index=0;} else {Draw_sequence_index++;}
          Plotter_status = "Idle";
          // pen up
        }

        if (web_Start_pos == 2 and position_index != 0){
          position_index--;
          Target_X         = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][0];
          Target_Y         = Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][1];
          Travel_dist_X    = abs (Target_X - X_coordinate);
          Travel_dist_Y    = abs (Target_Y - Y_coordinate);

        } else if (web_Start_pos == 2 and position_index == 0){
          digitalWrite(Z_solenoid, LOW);
          if(Draw_sequence_index == 43){Draw_sequence_index=0;} else {Draw_sequence_index++;}
          Plotter_status = "Idle";
          // pen up
        }
        delay(1000);
        Serial.print("DRAW (X,Y) = > ");
        Serial.print(Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][0]);
        Serial.print(",");
        Serial.println(Nikolaus_Haus[Draw_sequence[Draw_sequence_index][position_index]][1]);
      }


      if (Plotter_status == "Calibrating"){

        Plotter_status == "Hold";
        Serial.print(Delta_X);
        Serial.print(" = dx,dy = ");
        Serial.print(Delta_Y);
        Serial.print(" index = ");
        Serial.print(position_index);
        Update_Display();
        if (position_index < 4){

          position_index++;

          if (calibrate_axis == "X") {
            Target_X         = Calibration_sequence[position_index][0];
          } else if (calibrate_axis == "Y") {
            Target_Y         = Calibration_sequence[position_index][1];
          } else {
            Target_X         = Calibration_sequence[position_index][0];
            Target_Y         = Calibration_sequence[position_index][1];
          }

          Travel_dist_X    = abs (Target_X - X_coordinate);
          Travel_dist_Y    = abs (Target_Y - Y_coordinate);

          if (position_index > 2) {e_integral_x = 0; e_integral_y = 0;} else {e_integral_x = 260; e_integral_y = 260;}

          Plotter_status = "Calibrating";
        } else {
          position_index = 0;
```

```
          e_integral_x = 0;
          e_integral_y = 0;
          Plotter_status = "Idle";


        }
      }
    }
  }

  if ((unsigned long)(millis() - Pen_Down_Time) >= Max_Pen_Down_Time){
    digitalWrite(Z_solenoid, LOW);
  }


  if(Kix * e_integral_x >= 255 || Kiy * e_integral_y >= 255){

    if(abs(Delta_X) > 5 && Plotter_status != "Calibrating") {e_integral_x = 0;}
    if(abs(Delta_Y) > 5 && Plotter_status != "Calibrating") {e_integral_y = 0;}

    delay(22);

    if(abs(Delta_X) >= 0.4 && Kix * e_integral_x >= 255){
      X_motor_CW = 0;  X_motor_CCW = 0;
      if (Delta_X > 0 && Xmax_limit == 0)        { X_motor_CW = 1;  X_motor_CCW = 0; }
      else if (Delta_X < 0 && Xmin_limit == 0)   { X_motor_CW = 0;  X_motor_CCW = 1; }
      if(abs(Delta_X) > 0.4) {X_motor_PWM = 240;}
      setMotorx();
    }

    if(abs(Delta_Y) >= 0.4 && Kiy * e_integral_y >= 255){
      Y_motor_CW = 0;  Y_motor_CCW = 0;
      if (Delta_Y > 0 && Ymax_limit == 0)        { Y_motor_CW = 1;  Y_motor_CCW = 0; }
      else if (Delta_Y < 0 && Ymin_limit == 0)   { Y_motor_CW = 0;  Y_motor_CCW = 1; }
      if(abs(Delta_Y) > 0.4) {Y_motor_PWM = 255;}
      setMotory();
    }

    delay(16);

    if(Kix * e_integral_x >= 255){
      X_motor_PWM = 0;
      setMotorx();
    }

    if(Kiy * e_integral_y >= 255){
      Y_motor_PWM = 0;
      setMotory();
    }
  }
}
```

Input Signal Processing.h

```
//=====================================================
//Signal binning: discretization
//=====================================================
int signal_binning( float value){
  if (value >1.8){
    return 2;
  } else if (value < 1.6 && value > 0.6){
    return 1;
  } else if (value < 0.4 && value > -0.4){
    return 0;
  } else if (value < -0.6 && value > -1.6){
    return -1;
  } else if (value <-1.8){
    return -2;
 } else {
    return 5;
  }
}

//=====================================================
//update function: update position and sensor ststus
//=====================================================
void sensor_input_func( void * parameter) {

  float Y_A0_raw[]          = {0,0,0};
  float X_A0_raw[]          = {0,0,0};
  float Y_A1_raw[]          = {0,0,0};
  float X_A1_raw[]          = {0,0,0};
  float Y_A0_filtered[]   = {0,0,0};
  float X_A0_filtered[]   = {0,0,0};
  float Y_A1_filtered[]   = {0,0,0};
  float X_A1_filtered[]   = {0,0,0};
  int Y_A0_binned[]       = {0,0,0};
  int X_A0_binned[]       = {0,0,0};
  int Y_A1_binned[]       = {0,0,0};
  int X_A1_binned[]       = {0,0,0};
  // float Y_CORCTON          = 1;
  // float X_CORCTON          = 1;

  // (second order Butterworth coefficients)
  float b[] = {0.01323107, 0.02646213, 0.01323107};
  float a[] = {1.64927209, -0.70219636};

  int Print_index=0;
  float sens[1000][2];



  for(;;) {

    if ((unsigned long)(micros() - Sensor_Prev_Micros) >= Sensor_update_interval){

      Sensor_Prev_Micros = micros();                      // reset previousMicros
      Print_index++;

      Ymin_limit = digitalRead(Ylim_min);
      Ymax_limit = digitalRead(Ylim_max);
      Xmin_limit = digitalRead(Xlim_min);
      Xmax_limit = digitalRead(Xlim_max);
      E_stop     = digitalRead(E_stop_btn);

      Y_A0_raw[0] = analogRead(Ymotor_A0);
      X_A0_raw[0] = analogRead(Xmotor_A0);
      Y_A1_raw[0] = analogRead(Ymotor_A1);
      X_A1_raw[0] = analogRead(Xmotor_A1);

      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];
      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];
      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];
      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];

      Y_A0_binned[0] = signal_binning((Y_A0_filtered[0]/204.8-9.0 )*1.8);
      Y_A1_binned[0] = signal_binning((Y_A1_filtered[0]/204.8-9.0 )*1.8);
      X_A0_binned[0] = signal_binning((X_A0_filtered[0]/204.8-9.0 )*1.8);
      X_A1_binned[0] = signal_binning((X_A1_filtered[0]/204.8-9.0 )*1.8);

      if (Y_A0_binned[0] == 5) {Y_A0_binned[0] = Y_A0_binned[1];}
      if (Y_A1_binned[0] == 5) {Y_A1_binned[0] = Y_A1_binned[1];}
      if (X_A0_binned[0] == 5) {X_A0_binned[0] = X_A0_binned[1];}
      if (X_A1_binned[0] == 5) {X_A1_binned[0] = X_A1_binned[1];}



      if ((X_A1_binned[0] == 0 && X_A1_binned[1] != 0) || X_A1_binned[0] * X_A1_binned[1] < 0){

        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)){
          X_count++;
```

```
          X_dir = 1;
          if (X_A1_binned[0] == 0) { X_count_dec = 0;} else {abs(X_A1_binned[0]);}

        } else {
          X_count--;
          X_dir = -1;
          if (X_A1_binned[0] == 0) { X_count_dec = 4;} else {4-abs(X_A1_binned[0]);}

        }
      } else if (X_A1_binned[0] != X_A1_binned[1] && X_dir == 1 && X_count_dec < 3 ) {
        X_count_dec++;

      } else if (X_A1_binned[0] != X_A1_binned[1] && X_dir == -1 && X_count_dec > 1 ) {
        X_count_dec--;

      }

      X_coordinate = (X_count * 4 + X_count_dec) * d_X;

      if ((Y_A0_binned[0] == 0 && Y_A0_binned[1] != 0) || Y_A0_binned[0] * Y_A0_binned[1] < 0){

         if ((Y_A0_binned[0] - Y_A0_binned[1] > 0 && Y_A1_binned[0] > 0) || (Y_A0_binned[0] - Y_A0_binned[1] < 0 &&
         Y_A1_binned[0] < 0)){
          if (Y_dir == 1) {Y_count++;}
          Y_dir = 1;
          if (Y_A0_binned[0] == 0) { Y_count_dec = 0;} else {abs(Y_A0_binned[0]);}

        } else {
          if (Y_dir == -1) {Y_count--;}
          Y_dir = -1;
          if (Y_A0_binned[0] == 0) { Y_count_dec = 4;} else {4-abs(Y_A0_binned[0]);}
        }
      } else if (Y_A0_binned[0] != Y_A0_binned[1] && Y_dir == 1 && Y_count_dec < 3) {
         Y_count_dec++;
      }else if (Y_A0_binned[0] != Y_A0_binned[1] && Y_dir == -1 && Y_count_dec > 1) {
         Y_count_dec--;
      }

      Y_coordinate = (Y_count * 4 + Y_count_dec) * d_Y;

      if(Plotter_status == "Calibrating"){
        if (Target_X < 0   && Xmin_limit == 1) {
          X_count      = -5;
          X_count_dec  = 0;
          X_coordinate = (X_count * 4 + X_count_dec) * d_X;
          Target_X = X_coordinate;

        } else if (Target_X == 210 && Xmax_limit == 1) { Target_X = X_coordinate; Plot_area_xmax = X_coordinate - 10;
        }//Serial.println("xmax");}
        if (Target_Y < 0   && Ymin_limit == 1) {
          Y_count      = -5;
          Y_count_dec  = 0;
          Y_coordinate = (Y_count * 4 + Y_count_dec) * d_Y;
          Target_Y     = Y_coordinate;

        } else if (Target_Y == 235 && Ymax_limit == 1) { Target_Y = Y_coordinate; Plot_area_ymax = Y_coordinate - 10;
        }//Serial.println("ymax");}
      }

      Delta_X     = Target_X - X_coordinate;
      Delta_Y     = Target_Y - Y_coordinate;

      if (abs(Delta_X) < 0.4){X_stable_Micros += Sensor_update_interval;} else {X_stable_Micros = 0;}
      if (abs(Delta_Y) < 0.4){Y_stable_Micros += Sensor_update_interval;} else {Y_stable_Micros = 0;}

      if (Print_index>2 && (X_A1_binned[0] != X_A1_binned[1] || Y_A0_binned[0] != Y_A0_binned[1])){
        Print_index=0;
        Serial.print(Target_X);
        Serial.print(" ,");
        Serial.print(X_coordinate);
        Serial.print(" ,");
        // Serial.print(Delta_X);
        // Serial.print(" ,");
        // Serial.print(ux);
        // Serial.print(" ,");
        // Serial.print(Kix * e_integral_x);
        // Serial.print(" ,");
        Serial.print(Target_Y);
        Serial.print(" ,");
        Serial.print(Y_coordinate);
        // Serial.print(" ,");
        // Serial.print(Delta_Y);
        // Serial.print(" ,");
        // Serial.println(uy);
        Serial.print(" ,");
        Serial.print(Y_A0_binned[0]);
        Serial.print(" ,");
        Serial.println(Y_A1_binned[0]);

      // //   while(Print_index > 10){
      //     // Serial.print(X_A0_raw[1]);
      //     // Serial.print(" ,");
```

```
//     // Serial.print(X_A0_filtered[1]);
//     // Serial.print(" ,");
//     // Serial.println(X_A1_raw[0]);
// //     Print_index--;
// //   }

}

for(int i = 1; i >= 0; i--){
  Y_A0_raw[i+1]        = Y_A0_raw[i]; // store xi
  X_A0_raw[i+1]        = X_A0_raw[i]; // store xi
  Y_A0_filtered[i+1]   = Y_A0_filtered[i]; // store yi
  X_A0_filtered[i+1]   = X_A0_filtered[i]; // store yi
  Y_A0_binned[i+1]     = Y_A0_binned[i]; // store yi
  X_A0_binned[i+1]     = X_A0_binned[i]; // store yi

  Y_A1_raw[i+1]        = Y_A1_raw[i]; // store xi
  X_A1_raw[i+1]        = X_A1_raw[i]; // store xi
  Y_A1_filtered[i+1]   = Y_A1_filtered[i]; // store yi
  X_A1_filtered[i+1]   = X_A1_filtered[i]; // store yi
  Y_A1_binned[i+1]     = Y_A1_binned[i]; // store yi
  X_A1_binned[i+1]     = X_A1_binned[i]; // store yi
}

if (Ymin_limit + Ymax_limit + Xmin_limit + Xmax_limit != 0 && Plotter_status != "E-Stop !!"){

  if (Plotter_status == "Calibrating" || Plotter_status == "Hold"){
    // do nothing
  } else if (Plotter_status == "Plotting..."){
    Target_X = X_coordinate;
    Target_Y = Y_coordinate;
    Plotter_status = "Warning";

  } else if (Plotter_status == "Jogging"){
    if (Xmax_limit == 1 && Delta_X > 0.4) { Target_X = X_coordinate; }
    else if (Xmin_limit == 1 && Delta_X < -0.4) {
      X_count      = -8;
      X_count_dec  = 2;
      X_coordinate = (X_count * 4 + X_count_dec) * d_X;
      Target_X = X_coordinate;
    }
    if (Ymax_limit == 1 && Delta_Y > 0.4) { Target_Y = Y_coordinate; }
    else if (Ymin_limit == 1 && Delta_Y < -0.4) {
      Y_count      = -8;
      Y_count_dec  = 2;
      Y_coordinate = (Y_count * 4 + Y_count_dec) * d_Y;
      Target_Y = Y_coordinate;
    }
    X_motor_PWM = 0;
    Y_motor_PWM = 0;
    setMotorx();
    setMotorx();
    Plotter_status = "Warning";
  }else{
    Plotter_status = "Warning";
  }
  Delta_X     = Target_X - X_coordinate;
  Delta_Y     = Target_Y - Y_coordinate;
} else if (Plotter_status == "Warning" && Ymin_limit + Ymax_limit + Xmin_limit + Xmax_limit == 0){
  if (abs(Delta_X) < 0.4 && abs(Delta_Y) < 0.4 ) {Plotter_status = "Idle";}
  else {Plotter_status = "Jogging";}
}

if (E_stop == 1){ E_stop_Micros = micros(); }
if (E_stop == 0 && (micros() - E_stop_Micros) >= Estop_reset_time){
  Plotter_status = "Idle";
} else if (E_stop == 0 && (micros() - E_stop_Micros) >= 100){
  Plotter_status = "E-Stop !!";
  Target_X = X_coordinate;
  Target_Y = Y_coordinate;
  Delta_X     = Target_X - X_coordinate;
  Delta_Y     = Target_Y - Y_coordinate;
  setMotorx();
  setMotory();
  digitalWrite(Z_solenoid, LOW);
}

if (Plotter_status != "Idle" && Plotter_status != "Paused" && Plotter_status != "Hold"){



  if(Kix * e_integral_x < 255){

    if (abs(Delta_X) < 2 && abs(Delta_X) > 0.4 ){e_integral_x = e_integral_x + abs(Delta_X) /100;}

    if (abs(Delta_X) < 5 || abs(Delta_X) < 5 + Travel_dist_X*0.2){ e_diff_x     = (Delta_X - Last_Delta_X) *
    10000; } else {e_diff_x     = 0;}
    Last_Delta_X = Delta_X;

    X_motor_CW = 0;  X_motor_CCW = 0;

    if((Travel_dist_X - abs(Delta_X)) < 5 && Travel_dist_X > 10){
```

```
            ux = sgn(Delta_X)*(255 + 4 * (Travel_dist_X - abs(Delta_X))) ;
            if (Delta_X > 0 && Xmax_limit == 0)        { X_motor_CW = 1;  X_motor_CCW = 0; }
            else if (Delta_X < 0 && Xmin_limit == 0)   { X_motor_CW = 0;  X_motor_CCW = 1; }

          } else if (abs(Delta_X) < 5 || abs(Delta_X) < 5 + Travel_dist_X*0.2) {
            ux = Kpx * Delta_X + Kdx * e_diff_x;
            if (ux > 0 && Xmax_limit == 0)        { X_motor_CW = 1;  X_motor_CCW = 0; }
            else if (ux < 0 && Xmin_limit == 0)   { X_motor_CW = 0;  X_motor_CCW = 1; }

          } else {
            ux =  sgn(Delta_X)*255;
            if (Delta_X > 0 && Xmax_limit == 0)        { X_motor_CW = 1;  X_motor_CCW = 0; }
            else if (Delta_X < 0 && Xmin_limit == 0)   { X_motor_CW = 0;  X_motor_CCW = 1; }
          }

          X_motor_PWM = (int) fabs(ux);
          if (X_motor_PWM > 255) {X_motor_PWM = 255;} else if (X_motor_PWM < 150){X_motor_PWM = 0;}
          setMotorx();

        }

        if(Kiy * e_integral_y < 255){

          if (abs(Delta_Y) < 2 && abs(Delta_Y) > 0.4 ){e_integral_y = e_integral_y + abs(Delta_Y) /100;}

          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;}
          Last_Delta_Y = Delta_Y;

          Y_motor_CW = 0;  Y_motor_CCW = 0;

            if ((Travel_dist_Y - abs(Delta_Y)) < 5 && Travel_dist_Y > 10){
              uy = sgn(Delta_Y)*(255 + 6 * (Travel_dist_Y - abs(Delta_Y))) ;
              if (Delta_Y > 0 && Ymax_limit == 0)        { Y_motor_CW = 1;  Y_motor_CCW = 0; }
              else if (Delta_Y < 0 && Ymin_limit == 0)   { Y_motor_CW = 0;  Y_motor_CCW = 1; }

            } else if (abs(Delta_Y) < 5 || abs(Delta_Y) < 5 + Travel_dist_Y*0.2){
              uy = Kpy * Delta_Y + Kdy * e_diff_y;
              if (uy > 0 && Ymax_limit == 0)        { Y_motor_CW = 1;  Y_motor_CCW = 0; }
              else if (uy < 0 && Ymin_limit == 0)   { Y_motor_CW = 0;  Y_motor_CCW = 1; }

            } else {
              uy =  sgn(Delta_Y)*255;
              if (Delta_Y > 0 && Ymax_limit == 0)        { Y_motor_CW = 1;  Y_motor_CCW = 0; }
              else if (Delta_Y < 0 && Ymin_limit == 0)   { Y_motor_CW = 0;  Y_motor_CCW = 1; }
            }

            Y_motor_PWM = (int) fabs(uy);
            if (Y_motor_PWM > 255) {Y_motor_PWM = 255;} else if (Y_motor_PWM < 150){Y_motor_PWM = 0;}
            setMotory();
          }

        }

        if(Plotter_status != "Idle" && X_stable_Micros > Axis_stable_time && Y_stable_Micros > Axis_stable_time){
          X_motor_CW = 0;  X_motor_CCW = 0; X_motor_PWM = 0;
          Y_motor_CW = 0;  Y_motor_CCW = 0; Y_motor_PWM = 0;
          e_integral_x = 0;
          e_integral_y = 0;
          setMotorx();
          setMotory();
          if (Plotter status == "Jogging") { Plotter status = "Idle";}
            }
          }
        }
      }
```

WebUI Functions.h

```
//======================================
//handle function: send webpage to client
//======================================
void webpage()
{
  server.send(200,"text/html", webpageCode);
}

//======================================
//Set motors
//======================================
void setMotorx(){
  if(Plotter_status != "E-Stop !!"){
    ledcWrite    (pwmChannel_X, X_motor_PWM );
    digitalWrite (MOTOR_X_CW,  X_motor_CW  );
    digitalWrite (MOTOR_X_CCW, X_motor_CCW );
  } else if (Plotter_status == "E-Stop !!" || Plotter_status == "Paused" ){
    ledcWrite    (pwmChannel_X, 0 );
    digitalWrite (MOTOR_X_CW,   0 );
    digitalWrite (MOTOR_X_CCW,  0 );
  }

}
void setMotory(){
  if(Plotter_status != "E-Stop !!"){
    ledcWrite    (pwmChannel_Y, Y_motor_PWM );
    digitalWrite (MOTOR_Y_CW,  Y_motor_CW  );
    digitalWrite (MOTOR_Y_CCW, Y_motor_CCW );
  } else if (Plotter_status == "E-Stop !!" || Plotter_status == "Paused" ){
    ledcWrite    (pwmChannel_Y, 0 );
    digitalWrite (MOTOR_Y_CW,   0 );
    digitalWrite (MOTOR_Y_CCW,  0 );
  }

}

//================================================
//sign of variable
//================================================
int sgn(float val) {
  if (val > 0) return 1;
  return -1;
}

//================================================
//Mark a point
//================================================
void point() {
  digitalWrite(Z_solenoid, HIGH);
  delay(3000);
  digitalWrite(Z_solenoid, LOW);
}

//======================================
//Calculate haus points 1-5
//======================================
int Calc_nik_points()
{
  if(web_Start_pos == 1) {
    Nikolaus_Haus[0][0] = X_coordinate;                    Nikolaus_Haus[0][1] = Y_coordinate;
    Nikolaus_Haus[1][0] = X_coordinate + web_House_size;    Nikolaus_Haus[1][1] = Y_coordinate;
    Nikolaus_Haus[2][0] = X_coordinate + web_House_size;    Nikolaus_Haus[2][1] = Y_coordinate + web_House_size;
    Nikolaus_Haus[3][0] = X_coordinate;                    Nikolaus_Haus[3][1] = Y_coordinate + web_House_size;
    Nikolaus_Haus[4][0] = X_coordinate + web_House_size/2;  Nikolaus_Haus[4][1] = Y_coordinate + web_House_size * 1.5
    ;

  } else if((web_Start_pos == 2)) {
    Nikolaus_Haus[0][0] = X_coordinate - web_House_size;    Nikolaus_Haus[0][1] = Y_coordinate;
    Nikolaus_Haus[1][0] = X_coordinate;                    Nikolaus_Haus[1][1] = Y_coordinate;
    Nikolaus_Haus[2][0] = X_coordinate;                    Nikolaus_Haus[2][1] = Y_coordinate + web_House_size;
    Nikolaus_Haus[3][0] = X_coordinate - web_House_size;    Nikolaus_Haus[3][1] = Y_coordinate + web_House_size;
    Nikolaus_Haus[4][0] = X_coordinate - web_House_size/2;  Nikolaus_Haus[4][1] = Y_coordinate + web_House_size * 1.5
    ;

  }

  Serial.print("(1 ,2 ,Plot_area_xmax ,5 ,Plot_area_ymax) = ( ");
  Serial.print(Nikolaus_Haus[0][0]);
  Serial.print(" ,");
  Serial.print(Nikolaus_Haus[1][0]);
  Serial.print(" ,");
  Serial.print(Plot_area_xmax);
  Serial.print(" ,");
  Serial.print(Nikolaus_Haus[4][1]);
  Serial.print(" ,");
  Serial.print(Plot_area_ymax);
  Serial.println(" )");

  if (Nikolaus_Haus[0][0]<0 || Nikolaus_Haus[1][0]>Plot_area_xmax || Nikolaus_Haus[4][1]>Plot_area_ymax){return 0;}
  else {return 1;}


```

```
}

//=======================================================
//function process event: new data received from client
//=======================================================
void webSocketEvent(byte num, WStype_t type, uint8_t * payload, size_t length) {      // the parameters of this callback function are always the same -> num: id of the client who send the event, type: type of message, payload: actual data sent and length: length of payload
  switch (type) {                                     // switch on the type of information sent
    case WStype_DISCONNECTED:                         // if a client is disconnected, then type == WStype_DISCONNECTED
      Serial.println("Client " + String(num) + " disconnected");
      clients_connected--;
      break;
    case WStype_CONNECTED:                            // if a client is connected, then type == WStype_CONNECTED
      Serial.println("Client " + String(num) + " connected");
      clients_connected++;
      // optionally you can add code here what to do when connected
      break;
    case WStype_TEXT:                                 // if a client has sent data, then type == WStype_TEXT
      // try to decipher the JSON string received
      StaticJsonDocument<200> doc;                    // create a JSON container
      DeserializationError error = deserializeJson(doc, payload);
      if (error) {
        Serial.print(F("deserializeJson() failed: "));
        Serial.println(error.f_str());
        return;
      }
      else {
        // JSON string was received correctly, so information can be retrieved:
        web_Button        = doc["Button"];
        web_Jog_step_X    = doc["Jog_step_X"];
        web_Jog_step_Y    = doc["Jog_step_Y"];
        web_Jog_speed_XY  = doc["Jog_speed_XY"];
        web_Goto_X        = doc["Goto_X"];
        web_Goto_Y        = doc["Goto_Y"];
        web_House_size    = doc["House_size"];
        web_Draw_speed    = doc["Draw_speed"];
        web_Start_pos     = doc["Start_pos"];
        web_Cal_X         = doc["Cal_X"];
        web_Cal_Y         = doc["Cal_Y"];
        web_Kpx           = doc["Kp_X"];
        web_Kpy           = doc["Kp_Y"];
        web_Kix           = doc["Ki_X"];
        web_Kiy           = doc["Ki_Y"];
        web_Kdx           = doc["Kd_X"];
        web_Kdy           = doc["Kd_Y"];

        if (String(web_Button) == "Jog_X_+" && Xmax_limit == 0 && (Plotter_status == "Idle" || Plotter_status == "Jogging" || Plotter_status == "Warning")){
          Target_X        = Target_X + web_Jog_step_X;
          Target_Y        = Target_Y;
          Travel_dist_X   = abs (Target_X - X_coordinate);
          Travel_dist_Y   = abs (Target_Y - Y_coordinate);
          Plotter_status  = "Jogging";

        } else if (String(web_Button) == "Jog_X_-" && Xmin_limit == 0 && (Plotter_status == "Idle" || Plotter_status == "Jogging" || Plotter_status == "Warning")){
          Target_X        = Target_X - web_Jog_step_X;
          Target_Y        = Target_Y;
          Travel_dist_X   = abs (Target_X - X_coordinate);
          Travel_dist_Y   = abs (Target_Y - Y_coordinate);
          Plotter_status = "Jogging";

        } else if (String(web_Button) == "Jog_Y_+" && Ymax_limit == 0 && (Plotter_status == "Idle" || Plotter_status == "Jogging" || Plotter_status == "Warning")){
          Target_X        = Target_X;
          Target_Y        = Target_Y + web_Jog_step_Y;
          Travel_dist_X   = abs (Target_X - X_coordinate);
          Travel_dist_Y   = abs (Target_Y - Y_coordinate);
          Plotter_status = "Jogging";

        } else if (String(web_Button) == "Jog_Y_-" && Ymin_limit == 0 && (Plotter_status == "Idle" || Plotter_status == "Jogging" || Plotter_status == "Warning")){
          Target_X        = Target_X;
          Target_Y        = Target_Y - web_Jog_step_Y;
          Travel_dist_X   = abs (Target_X - X_coordinate);
          Travel_dist_Y   = abs (Target_Y - Y_coordinate);
          Plotter_status = "Jogging";

        } else if (String(web_Button) == "Pen_UP"){
          digitalWrite(Z_solenoid, LOW);

        } else if (String(web_Button) == "Pen_Down"){
          digitalWrite(Z_solenoid, HIGH);
          Pen_Down_Time = millis();

        } else if (String(web_Button) == "X0" && Xmin_limit == 0 && (Plotter_status == "Idle" || Plotter_status == "Jogging")){
          Target_X        = 0;
          Target_Y        = Y_coordinate;
          Travel_dist_X   = abs (Target_X - X_coordinate);
          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 ==
            "Jogging")){
178           Target_X        = X_coordinate;
179           Target_Y        = 0;
180           Travel_dist_X   = abs (Target_X - X_coordinate);
181           Travel_dist_Y   = abs (Target_Y - Y_coordinate);
182           Plotter_status = "Jogging";
183
184         } else if (String(web_Button) == "Home" && Xmin_limit == 0 && Ymin_limit == 0 && (Plotter_status == "Idle" ||
            Plotter_status == "Jogging")){
185           Target_X        = 0;
186           Target_Y        = 0;
187           Travel_dist_X   = abs (Target_X - X_coordinate);
188           Travel_dist_Y   = abs (Target_Y - Y_coordinate);
189           Plotter_status = "Jogging";
190
191         } else if (String(web_Button) == "Go_To_XY" && (Plotter_status == "Idle" || Plotter_status == "Jogging")){
192           if (web_Goto_X >= 0 && web_Goto_X <= Plot_area_xmax && web_Goto_Y >= 0 && web_Goto_Y <= Plot_area_ymax){
193             Target_X        = web_Goto_X;
194             Target_Y        = web_Goto_Y;
195             Travel_dist_X   = abs (Target_X - X_coordinate);
196             Travel_dist_Y   = abs (Target_Y - Y_coordinate);
197             Plotter_status = "Jogging";
198           } else {
199             Inside_draw_area = 0;
200           }
201
202         } else if (String(web_Button) == "Draw" && Plotter_status == "Idle"){
203           if(web_Start_pos == 1){position_index=0;} else {position_index=8;}
204           Inside_draw_area = Calc_nik_points();
205           if(Inside_draw_area==1){
206             digitalWrite(Z_solenoid, HIGH);
207             Pen_Down_Time = millis();
208             delay(50);
209             Target_X = X_coordinate;
210             Target_Y = Y_coordinate;
211             Travel_dist_X   = abs (Target_X - X_coordinate);
212             Travel_dist_Y   = abs (Target_Y - Y_coordinate);
213             Plotter_status = "Plotting...";
214           }
215
216         } else if (String(web_Button) == "Pause" && Plotter_status == "Plotting..."){
217           Plotter_status = "Paused";
218           digitalWrite(Z_solenoid, LOW);
219
220         }else if (String(web_Button) == "Resume" && Plotter_status == "Paused"){
221           digitalWrite(Z_solenoid, HIGH);
222           Pen_Down_Time = millis();
223           delay(50);
224           Plotter_status = "Plotting...";
225
226         } else if (String(web_Button) == "Stop" && Plotter_status != "E-Stop !!"){
227           Plotter_status = "Idle";
228           Target_X = X_coordinate;
229           Target_Y = Y_coordinate;
230           Travel_dist_X   = abs (Target_X - X_coordinate);
231           Travel_dist_Y   = abs (Target_Y - Y_coordinate);
232           e_integral_x = 0;
233           e_integral_y = 0;
234           digitalWrite(Z_solenoid, LOW);
235
236         } else if (String(web_Button) == "E_Stop_btn"){
237           Plotter_status = "E-Stop !!";
238           digitalWrite(Z_solenoid, LOW);
239
240         } else if (String(web_Button) == "Cal_X_btn" && Plotter_status != "E-Stop !!" && (Plotter_status == "Idle" ||
            Plotter_status == "Jogging")){
241           position_index = 0;
242           calibrate_axis = "X";
243           Target_X       = Calibration_sequence[position_index][0];
244           Target_Y       = Y_coordinate;
245           Travel_dist_X  = abs (Target_X - X_coordinate);
246           Travel_dist_Y  = abs (Target_Y - Y_coordinate);
247           Plotter_status = "Calibrating";
248           digitalWrite(Z_solenoid, LOW);
249
250         } else if (String(web_Button) == "Cal_Y_btn" && Plotter_status != "E-Stop !!" && (Plotter_status == "Idle" ||
            Plotter_status == "Jogging")){
251           position_index = 0;
252           calibrate_axis = "Y";
253           Target_X       = X_coordinate;
254           Target_Y       = Calibration_sequence[position_index][1];
255           Travel_dist_X  = abs (Target_X - X_coordinate);
256           Travel_dist_Y  = abs (Target_Y - Y_coordinate);
257           Plotter_status = "Calibrating";
258           digitalWrite(Z_solenoid, LOW);
259
260         } else if (String(web_Button) == "Cal_XY_btn" && Plotter_status != "E-Stop !!" && (Plotter_status == "Idle" ||
             Plotter_status == "Jogging")){
261           position_index = 0;
```

```
            calibrate_axis = "";
            Target_X       = Calibration_sequence[position_index][0];
            Target_Y       = Calibration_sequence[position_index][1];
            Travel_dist_X  = abs (Target_X - X_coordinate);
            Travel_dist_Y  = abs (Target_Y - Y_coordinate);
            Plotter_status = "Calibrating";
            digitalWrite(Z_solenoid, LOW);

          } else if (String(web_Button) == "Update_pid"){
            Kpx = web_Kpx;
            Kpy = web_Kpy;
            Kix = web_Kix;
            Kiy = web_Kiy;
            Kdx = web_Kdx;
            Kdy = web_Kdy;
            Serial.print("(Kp X ,Kp Y ,Ki X ,Ki Y) = ( ");
            Serial.print(Kpx);
            Serial.print(" ,");
            Serial.print(Kpy);
            Serial.print(" ,");
            Serial.print(Kix);
            Serial.print(" ,");
            Serial.print(Kiy);
            Serial.print(" ,");
            Serial.print(Kdx);
            Serial.print(" ,");
            Serial.print(Kdy);
            Serial.println(" )");
          }
        }
        //Serial.println("");
        break;
      }
    }
```

Web Page HTML Code

```
//=====================
//HTML code for webpage
//=====================

const char webpageCode[] PROGMEM = R"=====(

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Pen Plotter Web UI</title>
    <style>

      *{
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: Arial, Helvetica, sans-serif;
      }
      body{
        background-color: rgb(206, 231, 255);
        display: flex;
        flex-direction: column;
        justify-content: flex-start;
        align-items: center ;
        height: 100px;
      }

      h1{
        color: rgb(255, 255, 255);
        background-color: rgb(37, 78, 150);
        box-shadow: inset  -5px 0px 15px rgba(0, 0, 0, 0.623);
        border-radius: 0px 10px 10px 0px;
        font-size: 30px;
        padding: 20px 85px;
        margin: 0px;
        text-align: center;
      }

      .container{
        width: 800px;
        margin-top: 100px;
        background-color: white;
        padding: 0px;
        box-shadow: 0 2px 16px rgb(0,0,0,0.3);
        border-radius: 0px 20px 0px 20px;
        animation-duration: 0.5s;
      }

      .H_container{
        margin: 10px;
        display: flex;
        flex-direction: row;
        background-color: white;
        border: 0.8px solid rgb(170, 170, 170);
        padding: 0px;
        box-shadow: 2px 2px 5px rgba(4, 74, 179, 0.5);
        border-radius: 10px;
      }

      .V_container{
        margin: 10px;
        display: flex;
        flex-direction: column;
        flex-wrap: wrap;
        justify-content: flex-start;
        background-color: white;
        border: 0.8px solid rgb(170, 170, 170);
        padding: 0px;
        box-shadow: 2px 2px 5px rgba(4, 74, 179, 0.5);
        border-radius: 10px;
      }

      .Status_container{
        color: #000000;
        font-size: 20px;
        font-weight: 600;
        text-align:center;
        justify-content: center;
        padding: 15px 15px 10px 20px;
        margin: 10px;
        justify-content: flex-start;
        background-color: rgb(133, 253, 129);
        border: 0.8px solid rgb(73, 73, 73);
        box-shadow: inset 2px 2px 5px rgba(105, 105, 105, 0.5);
        border-radius: 10px;
      }

      .Status_container.idle{
        color: #0045db;
        background-color: rgb(255, 255, 255);
      }
```

```
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;
```

```
        }

        input {
          height: 28px;
          margin: 0;
          padding-right: 32px;
          text-align: center;
        }

        .blue_button {
          color: #FFFFFF;
          width: 22.5%;
          font-size: 16px;
          border-radius: 5px;
          border: solid 1px #3866a3;
          padding: 10px 10px;
          margin: 5px;
          text-shadow: 1.2px 1.2px 1.5px #172e44;
          box-shadow: inset 1px 1px 0px 0px #BBDAF7;
          text-decoration: none;
          cursor: pointer;
          position: relative;
          overflow: hidden;
          font-family: Arial;
          background: linear-gradient(180deg, #63B8EE 10%, #468CCF 100%);
          display: inline-block;
          vertical-align: middle;
        }
        .blue_button:active {
          background: linear-gradient(180deg, #468CCF 10%, #63B8EE 100%);
        }
        .blue_button-text {
          position: relative;
          display: inline-block;
        }

        .yellow_button {
          color: #333333;
          width: 22.5%;
          font-size: 16px;
          border-radius: 5px;
          border: solid 1px #533300;
          padding: 10px 10px;
          margin: 5px;
          text-shadow: 1px 1px 0px #ffed66;
          box-shadow: inset 1px 1px 0px 0px #fff6af;
          text-decoration: none;
          cursor: pointer;
          position: relative;
          overflow: hidden;
          font-family: Arial;
          background: linear-gradient(180deg, #ffed64 10%, #ffab23 100%);
          display: inline-block;
          vertical-align: middle;
        }
        .yellow_button:active {
          background: linear-gradient(180deg, #ffab23 10%, #ffed64 100%);
        }
        .yellow_button-text {
          position: relative;
          display: inline-block;
        }

        .red_button {
          color: #ffffff;
          font-weight: 600;
          width: 80%;
          height: 65px;
          font-size: 22px;
          border-radius: 5px;
          border: solid 1px #530000;
          padding: 10px 10px;
          margin: 5px;
          margin-top: 30px;
          text-shadow: 1px 1px 0px #ff6666;
          box-shadow: inset 1px 1px 0px 0px #ffafaf;
          text-decoration: none;
          cursor: pointer;
          position: relative;
          overflow: hidden;
          font-family: Arial;
          background: linear-gradient(180deg, #ff6464 10%, #ff2323 100%);
          display: inline-block;
          vertical-align: middle;
        }
        .red_button:active {
          background: linear-gradient(180deg, #ff2323 10%, #ff6464 100%);
        }
        .red_button-text {
          position: relative;
          display: inline-block;
        }
```

```
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           <img src="https://alumni.hs-schmalkalden.de/data/images/2022/09/18/1-6327830894f05.png" alt="logo" height="50"
            style="margin: 30px;">
421         </div>
422         <div class = "tab_box">
423           <button class="tab_btn active">Home</button>
424           <button class="tab_btn">Calibrate</button>
425           <button class="tab_btn">About</button>
426           <button class="tab_btn">Contact Us</button>
427           <div class="line"></div>
428         </div>
429
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:  </label>
459                   <div class="TH_container" style="margin-top: 6px;">
```

```
                <input style="width: 80px;" type="number" id="Jog_step_X" name="Jog_step_X" min="0" max="100" step="10"
                value="20" placeholder="Step size"><span style="margin-left:-32px;">mm</span>
              </div>
            </div>
            <div class="TH_container" style="align-items: center;">
              <label for="Jog_step_Y">Jog Y:  </label>
              <div class="TH_container" style="margin-top: 6px;">
                <input style="width: 80px;" type="number" id="Jog_step_Y" name="Jog_step_Y" min="0" max="100" step="10"
                value="20" placeholder="Step size"><span style="margin-left:-32px;">mm</span>
              </div>
            </div>
          </div>
          <div class="TH_container" style="flex-wrap: wrap; margin-top: 10px;">
            <button type="button" id='Jog_X_+' class="blue_button" style="padding: 5px;">
              <span class="blue_button-text" ><font size="+2">&#10236;</font> X +</span>
            </button>
            <button type="button" id='Jog_Y_+' class="blue_button" style="padding: 5px;">
              <span class="blue_button-text"><font size="+2">&#8613;</font> Y +</span>
            </button>
            <button type="button" id='Pen_UP' class="blue_button" style="width: 30%; padding: 5px;">
              <span class="blue_button-text">Pen UP <font size="+2">&#8613;</font></span>
            </button>
            <button type="button" id='Jog_X_-' class="blue_button" style="padding: 5px;">
              <span class="blue_button-text"><font size="+2">&#10235;</font> X -</span>
            </button>
            <button type="button" id='Jog_Y_-' class="blue_button" style="padding: 5px;">
              <span class="blue_button-text"><font size="+2">&#8615;</font> Y -</span>
            </button>
            <button type="button" id='Pen_Down' class="blue_button" style="width: 30%; padding: 5px;">
              <span class="blue_button-text">Pen Down <font size="+2">&#10515;</font></span>
            </button>
            <button type="button" id='X0' class="yellow_button">
              <span class="yellow_button-text"> X0 <font size="+1.5"><b>&#171;</b></font> &#127968;</span>
            </button>
            <button type="button" id='Y0' class="yellow_button">
              <span class="yellow_button-text"> Y0 <font size="+1.5"><b>&#171;</b></font> &#127968;</span>
            </button>
            <button type="button" id='Home' class="yellow_button" style="width: 30%;">
              <span class="yellow_button-text">&#127968; Home XY</span>
            </button>
          </div>
          <div class="TH_container" style="margin-bottom: 20px;">
            <div class="TV_container" style="margin-right: 10px;">
              <div class="TH_container" >
                <label for="Goto_X">Goto X:  </label>
                <input style="width: 120px;" type="number" id="Goto_X" name="Goto_X" min="0" max="300" step="10"
                placeholder="X position"><span style="margin-left:-32px;">mm</span>
              </div>
              <div class="TH_container" style="margin-top: 6px;">
                <label for="Goto_Y">Goto Y:  </label>
              <input style="width: 120px;" type="number" id="Goto_Y" name="Goto_Y" min="0" max="300" step="10"
              placeholder="Y position"><span style="margin-left:-32px;">mm</span>
              </div>
            </div>
            <button type="button" id="Go_To_XY" class="yellow_button" style="width: 25%;">
              <span class="yellow_button-text">Go To XY &#128663;   &#128663;...</span>
            </button>
          </div>
        </div>
        <div class="V_container" style="width: 395px;">
          <h3 style="margin-top: 20px; margin-bottom: 20px;">Plotter Status / Warnings</h3>
          <div id="Status container" class="Status container idle" style="margin: 0px 30px 0px 30px; align-items: center;">
            <span id="status">
              Initialising...
            </span>
          </div>
          <h3 style="margin-top: 30px; margin-bottom: 12px;">Draw Controls</h3>
          <div class="TH_container" style="align-items: baseline;">
            <label for="House_size">House Width</label>
            <div class="TH_container" style="margin: 6px; justify-content: flex-start;">
              <input style="width: 98px; padding-right: 40px;" type="number" id="House_size" name="House_size" min="10"
              max="60" step="5" value="20" ><span style="margin-left:-35px;">mm</span>
            </div>
          </div>
          <div class="TH_container" style="margin: 0px 30px;">
            <p style="color: #007032; font-size: 16px; font-weight: 600; margin: 0;">Start Position : </p>
            <div class="switch-field">
              <input type="radio" id="Start_pos_1st" name="Start_pos" value="1st" checked/>
              <label for="Start_pos_1st">1st</label>
              <input type="radio" id="Start_pos_2nd" name="Start_pos" value="2nd" />
              <label for="Start_pos_2nd">2nd</label>
            </div>
          </div>
          <div class="TH_container" style="flex-wrap: wrap;">
            <button type="button" id="Draw" class="blue_button">
              <span class="blue_button-text">Draw <br> &#9998;</span>
            </button>
            <button type="button" id="Pause_Resume" class="blue_button" style="width: 90px;">
              <span class="blue_button-text" id="Pause_Resume_text">Pause <br> ⏸</span>
            </button>
            <button type="button" id="Stop" class="blue_button">
              <span class="blue_button-text">Stop <br> &#9724;</span>
```

```
            </button>
            <button type="button" id="E_Stop_btn" class="red_button">
              <span class="red_button-text">&#9940; Emergency Stop !!</span>
            </button>
          </div>
        </div>
      </div>

      <div class="content">
        <div class="TV_container" style="width: 760px;">
          <div class="TH_container">
            <div class="V_container" style="width: 760px; height: 518.5px; padding-bottom: 10px;" >
              <h3>Calibration Instructions</h3>
              <p>
                <b>Make sure there is paper in the plotting area before beginning the calibration process!!</b><br><br>
                <b>Step 1A :</b>  To calibrate the X-Axis, Press the Calibrate X Button.<br>
                <b>Step 2A :</b>  Wait for the machine to finishes the calibration sequence for the X-axis.<br><br>
                <b>Step 1B :</b>  To calibrate the Y-Axis, Press the Calibrate Y Button.<br>
                <b>Step 2B :</b>  Wait for the machine to finishes the calibration sequence for the Y-axis.<br><br>
                <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>
            </div>
            <div class="V_container" style="width: 100%; height: 96%;">
              <h3>Sensor Status</h3>
              <table class="myTable" id="SensorTable">
                <colgroup>
                  <col style="width:40%">
                  <col style="width:30%">
                  <col style="width:30%">
                </colgroup>
                <thead>
                  <tr>
                    <th>Sensor</th>
                    <th>Min</th>
                    <th>Max</th>
                  </tr>
                </thead>
                <tbody>
                  <tr>
                    <td>Limit Switch <br>X-Axis</td>
                    <td><span id='X_Lim_min'>OK &#9989;</span></td>
                    <td><span id='X_Lim_max'>--</span></td>
                  </tr>
                  <tr>
                    <td>Limit Switch <br>Y-Axis</td>
                    <td><span id='Y_Lim_min'>Active &#128680;</span></td>
                    <td><span id='Y_Lim_max'>--</span></td>
                  </tr>
                  <tr>
                    <td>Emergency Switch</td>
                    <td colspan="2"><span id='E_Stop_status' style="font-size: 20px;">&#128721; ACTIVE! &#128219;</span></td>
                  </tr>
                </tbody>
              </table>
              <h3 style="margin-top: 45px; margin-bottom: 0px;">Calibration Controls</h3>
              <div class="TH_container" style="flex-wrap: wrap; ">
                <button type="button" id="Cal_X_btn" class="blue_button" style="width: 35%; margin-left: 0;">
                  <span class="blue_button-text">&#128295; Calibrate X</span>
                </button>
                <button type="button" id="Cal_Y_btn" class="blue_button" style="width: 35%; margin-right: 0;">
                  <span class="blue button-text">&#128296; Calibrate Y</span>
                </button>
              </div>
              <button type="button" id="Cal_XY_btn" class="yellow_button" style="width: 290px; height: 60px; margin: 0px
              50px 20px 50px;">
                <span class="yellow_button-text" style="font-size: 20px;">&#128295; Calibrate X-Y &#128296;</span>
              </button>
            </div>
          </div>
          <div class="H_container" >
            <h3 style="margin: 15px 10px 15px 20px; ">PID <br>Controll</h3>
            <div class="TH_container" style="justify-content: space-between;">
              <div class="TH_container" style="width: 400px; margin-bottom: 15px; margin-top: 15px; flex-wrap: wrap">
                <label for="Kp_X">Kp X: </label>
                <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">
                <label for="Ki_X">Ki X: </label>
                <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">
                <label for="Ki_X">Kd X: </label>
                <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">
                <label for="Kp_Y">Kp Y: </label>
                <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">
                <label for="Ki_Y">Ki Y: </label>
                <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">
                <label for="Ki_Y">Kd Y: </label>
                <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">
```

```
                </div>
                <button type="button" id="Update_pid" class="yellow_button" style="width: 30%; height: 50px; margin: 20px;">
                  <span class="yellow_button-text" style="font-size: 20px;">&#128260; Update PID</span>
                </button>
              </div>
            </div>
          </div>

          <div class="content">
            <div class="V_container">
              <h3>About</h3>
              <p class="p1">
                <b>Welcome to PenPlotter WebUI</b>, a project born out of a mechatronics workshop led by university students.
              </p>
              <p class="p1">
                PenPlotter WebUI is a user-friendly web-based interface designed specifically for pen plotters which
                are remarkable electromechanical device used to create precise and high-quality line drawings of the iconic
                Nikolaus house.
              </p>
              <p class="p1">
                Tailored for university students between the ages of 20 and 30, PenPlotter WebUI is very simple and
                straightforward to
                use with little to no prior knowledge of pen plotters, 3D printers or programming required. We've designed our
                interface
                with simplicity and efficiency in mind, allowing you to start drawing straight away.
              </p>
              <p class="p1">
                The PenPlotter provides an accessible and engaging platform for indoor use. Whether
                it's in an office or an educational institution, our platform encourages you to explore the potential of pen
                plotting technology.
              </p>
              <p class="p1">
                We highly value your input, as it shapes the future of PenPlotter WebUI. Our team aims to continuously
                improve and develop the platform based on user feedback. We encourage you to contribute your ideas, report any
                issues you encounter,
                or even become part of our open-source community.
              </p>
              <p class="p1">
                Experience the joy of pen plotting with PenPlotter WebUI, created by passionate university students with a focus
                on mechatronics
                and creativity. Explore the precise art of line drawings, and witness your Nikolaus house take shape.
              </p>
            </div>
          </div>

          <div class="content">
            <div class="V_container">
              <h3>Contact Us</h3>
              <p class="p1">
                Thank you for your interest in PenPlotter WebUI. We value your feedback and are here to assist you with
                any inquiries or concerns you may have.
              </p>
              <p class="p1">
                For general inquiries or information about PenPlotter WebUI, please feel free to reach out to us via email
                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
                information you need.
              </p>
              <p class="p1">
                If you require technical support or have specific questions regarding the usage of PenPlotter WebUI, our
                team is ready to assist you. Please send your support-related queries to <b>r.joshi@stud.fh-sm.de</b> We will
                prioritize
                your request and ensure that you receive the necessary assistance to make the most out of the platform.
              </p>
              <p class="p1">
                We greatly appreciate your feedback and suggestions for improving PenPlotter WebUI. If you have ideas for new
                features, enhancements, or any other ways we can enhance your experience, please don't hesitate to let us know.
                Your input plays a crucial role in shaping the future of our project.
              </p>
              <p class="p1">
                PenPlotter WebUI is a project fueled by your passion and creativity. We are excited to have you as part of our
                community and look forward to hearing from you. Your support and involvement are vital to our continued success.
              </p>
              <p class="p1">
                <b>Thank you for choosing PenPlotter WebUI.</b>
              </p>
            </div>
          </div>
        </div>
      </div>
      <p class="footer_des">Developed by Team #14 MERO 22/23 HS Schmalkalden</p>
      <p class="footer_des" style="margin-top: 0px;">Web UI V1.5 (HTML, CSS, JS)</p>
      <p class="footer_des" style="margin-top: 0px;">ESP32 Web Server</p>
      <script>
        const tabs= document.querySelectorAll('.tab_btn');
        const all_content= document.querySelectorAll('.content');

        tabs.forEach((tab, index)=>{
          tab.addEventListener('click', (e)=>{
            tabs.forEach(tab=>{tab.classList.remove('active')});
            tab.classList.add('active');

```

```
            var line=document.querySelector('.line');
            line.style.width = e.target.offsetWidth + "px";
            line.style.left = e.target.offsetLeft + "px";

            all_content.forEach(content=>{content.classList.remove('active')});
            all_content[index].classList.add('active');
          })
        })

        var Socket;
        let btns = document.querySelectorAll('button');

        btns.forEach(function (i) {
          i.addEventListener('click', function() {
            if(i.id != "" && document.getElementById('status').innerHTML != "E-Stop !!"){
              if(i.innerText == "Pause\n▮▮"){
                var Button_ID = "Pause";
              } else if (i.innerText == "Resume\n□") {
                var Button_ID = "Resume";
              } else {
                var Button_ID = i.id;
              };

              var msg = { Button: Button_ID, };

              if(Button_ID == "Pause" || Button_ID == "Stop" || Button_ID == "E_Stop_btn" ){
                msg = { Button: Button_ID, };
              } else if (Button_ID == "Draw" || Button_ID == "Resume" ){
                msg = {
                  Button: Button_ID,
                  Start_pos:    Start_pos_1st.checked*1 + Start_pos_2nd.checked*2,
                  House_size:   House_size.valueAsNumber,
                };

              } else if (Button_ID == "Update_pid"){
                msg = {
                  Button: Button_ID,
                  Kp_X :        Kp_X.valueAsNumber,
                  Ki_X :        Ki_X.valueAsNumber,
                  Kd_X :        Kd_X.valueAsNumber,
                  Ki_Y :        Ki_Y.valueAsNumber,
                  Kp_Y :        Kp_Y.valueAsNumber,
                  Kd_Y :        Kd_Y.valueAsNumber,
                }
              } else {
                msg = {
                  Button: Button_ID,
                  Jog_step_X:   Jog_step_X.valueAsNumber,
                  Jog_step_Y:   Jog_step_Y.valueAsNumber,
                  Goto_X:       Goto_X.valueAsNumber,
                  Goto_Y:       Goto_Y.valueAsNumber,
                }
              };

              console.log(msg);
              Socket.send(JSON.stringify(msg));
            };
          });
        });

        function init() {
          Socket = new WebSocket('ws://' + window.location.hostname + ':81/');
          Socket.onmessage = function(event) {
            processCommand(event);
          };
        }

        function processCommand(event) {
          var obj = JSON.parse(event.data);
          document.getElementById('X_coor').innerHTML = Math.round((obj.X_coor + Number.EPSILON) * 100) / 100 + " mm";
          document.getElementById('Y_coor').innerHTML = Math.round((obj.Y_coor + Number.EPSILON) * 100) / 100 + " mm";
          document.getElementById('Pen_up_down').innerHTML = obj.Pen_up_down;
          document.getElementById('status').innerHTML = obj.status;
          switch(obj.status) {
            case "Plotting...":
              document.getElementById('UI_content').className = "content_box";
              document.getElementById('Status_container').className = "Status_container plotting";
              break;
            case "Warning":
              document.getElementById('UI_content').className = "content_box warning";
              document.getElementById('Status_container').className = "Status_container warning";
              break;
            case "E-Stop !!":
              document.getElementById('UI_content').className = "content_box estop";
              document.getElementById('Status_container').className = "Status_container estop";
              break;
            default:
              document.getElementById('UI_content').className = "content_box";
              document.getElementById('Status_container').className = "Status_container idle";
          };
          if(obj.status == "Paused"){
            document.getElementById('Pause_Resume_text').innerHTML = "Resume <br> □";
```

```
          } else if (obj.status == "Plotting...") {
            document.getElementById('Pause_Resume_text').innerHTML = "Pause <br> ⏸";
          };
          document.getElementById('X_Lim_min').innerHTML = obj.X_Lim_min;
          document.getElementById('Y_Lim_min').innerHTML = obj.Y_Lim_min;
          document.getElementById('X_Lim_max').innerHTML = obj.X_Lim_max;
          document.getElementById('Y_Lim_max').innerHTML = obj.Y_Lim_max;
          document.getElementById('E_Stop_status').innerHTML = obj.E_Stop_status;
          if(obj.Inside_draw_area == "Error"){
            alert(" ERROR ⁉️ : Drawing/Move not possible. 💔 \n\n Figure/point is out of bounds🚧. Change starting position or
            house size and try again later.");
          };
          console.log(event.data);
        }
        window.onload = function(event) {
          init();
        }
      </script>
    </body>
  </html>

  )=====";
```

OLED Display Code

```
#define LOGO_HEIGHT   40
#define LOGO_WIDTH    40
static const unsigned char PROGMEM logo_bmp[] ={
    0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x00, 0x00
    0x00, 0x00, 0x07, 0xff, 0x80, 0x00, 0x00, 0x3f, 0xf9, 0x80, 0x00, 0x01, 0xff
    0x0f, 0xfe, 0x01, 0x80, 0x00, 0x3f, 0xf0, 0x01, 0x80, 0x00, 0xff, 0x80, 0x01
    0x00, 0x07, 0x80, 0x0f, 0xf8, 0x00, 0x3f, 0x80, 0x1f, 0xc0, 0x03, 0xff, 0x80, 0x7f, 0x80, 0x1f,
    0xfd, 0x80, 0xfe, 0x00, 0xff, 0xc1, 0x80, 0xfc, 0x07, 0xfe, 0x01, 0x81, 0xf8, 0x1f, 0xf0, 0x01,
    0x83, 0xf0, 0x7f, 0x80, 0x01, 0x83, 0xe1, 0xfe, 0x00, 0x07, 0x83, 0xc7, 0xf0, 0x00, 0x3f, 0x81,
    0xdf, 0xc0, 0x03, 0xff, 0x81, 0xff, 0x80, 0x1f, 0xfd, 0x80, 0xfe, 0x00, 0xff, 0xc1, 0x80, 0xfc,
    0x03, 0xfe, 0x01, 0x81, 0xf8, 0x1f, 0xf0, 0x01, 0x83, 0xf0, 0x7f, 0x80, 0x01, 0x83, 0xe1, 0xfe,
    0x00, 0x01, 0x83, 0xc7, 0xf0, 0x00, 0x01, 0x81, 0xdf, 0xc0, 0x00, 0x01, 0x81, 0xff, 0x80, 0x00,
    0x01, 0x80, 0xfe, 0x00, 0x00, 0x01, 0x80, 0xfc, 0x00, 0x00, 0x01, 0x81, 0xf0, 0x00, 0x00, 0x01,
    0x83, 0xf0, 0x00, 0x00, 0x01, 0x83, 0xe0, 0x00, 0x00, 0x01, 0x83, 0xc0, 0x00, 0x00, 0x01, 0x81,
    0xc0, 0x00, 0x00, 0x01, 0x81, 0xc0, 0x00, 0x00, 0x01, 0x80, 0xc0, 0x00, 0x00, 0x01, 0x80, 0x40,
    0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff
};

const unsigned char PROGMEM logo_estop[] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00,
    0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00,
    0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x0c, 0x18,
    0x30, 0x00, 0x00, 0x06, 0x18, 0x70, 0x00, 0x10, 0x06, 0x18, 0x60, 0x0c, 0x1c, 0x03, 0x18, 0xc0,
    0x38, 0x0e, 0x01, 0x00, 0xc0, 0x70, 0x03, 0x80, 0x00, 0x01, 0xc0, 0x00, 0xe0, 0x00, 0x07, 0x80,
    0x00, 0x71, 0xff, 0x8e, 0x00, 0x00, 0x13, 0xff, 0xc8, 0x00, 0x00, 0x03, 0x7f, 0xc0, 0x00, 0x00,
    0x02, 0x7f, 0xc0, 0x00, 0x00, 0x02, 0x7f, 0xc0, 0x00, 0x07, 0xe2, 0x7f, 0xe7, 0xe0, 0x07, 0xe6,
    0x7f, 0xe7, 0xe0, 0x00, 0x06, 0x7f, 0xe0, 0x00, 0x00, 0x06, 0xff, 0xe0, 0x00, 0x00, 0x04, 0xff,
    0xe0, 0x00, 0x00, 0x04, 0xff, 0xe0, 0x00, 0x00, 0x04, 0xff, 0xf0, 0x00, 0x00, 0x0c, 0xff, 0xf0,
    0x00, 0x00, 0x0c, 0xff, 0xf0, 0x00, 0x00, 0x0f, 0xff, 0xf0, 0x00, 0x00, 0x0f, 0xff, 0xf0, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x7f, 0xff, 0xfe, 0x00, 0x00,
    0x7f, 0xff, 0xfe, 0x00, 0x00, 0x7f, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};


void Welcome_Screen(){
  display.clearDisplay();

  display.drawBitmap( 4, 20, logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);

  display.setTextColor(WHITE);
  display.setCursor(1,1);
  display.setTextSize(2);
  display.println("PEN");
  display.setCursor(44,1);
  display.println("PLOTTER");
  display.setCursor(50,20);
  display.setTextSize(1);
  display.println("HOCHSCHULE");
  display.setCursor(50,30);;
  display.println("SCHMALKALDEN");
  display.setCursor(60,52);;
  display.println("BY:TEAM #14");

  display.display();
   // Invert and restore display, pausing in-between
  display.invertDisplay(true);
  delay(4000);
  display.invertDisplay(false);
  delay(4000);

  display.clearDisplay();

  display.setTextColor(WHITE);
  display.setCursor(1,1);
  display.setTextSize(2);
  display.println(">>STARTING");
  display.setCursor(4,20);
  display.setTextSize(1);
  display.println("- Initialising WEB     Server.....");
  display.display();
}

void Update_Display(){
  display.clearDisplay();

  if (Plotter_status != "E-Stop !!" && clients_connected==0){

    display.setTextColor(WHITE);

    display.setCursor(1,1);
    display.setTextSize(2);
    display.println("PEN");
    display.setCursor(44,1);
    display.println("PLOTTER");

    display.setCursor(0,18);
    display.println("LOCAL IP:");

    display.setCursor(0,35);
    display.println(WiFi.localIP());

```

```
  } else if (Plotter_status != "Warning" && Plotter_status != "E-Stop !!"){

    display.setTextColor(WHITE);

    display.setCursor(1,1);
    display.setTextSize(2);
    display.println("PEN");
    display.setCursor(44,1);
    display.println("PLOTTER");

    display.setCursor(0,18);
    display.println("X:");
    display.setCursor(25,18);
    display.println(X_coordinate);
    display.setCursor(90,18);
    display.println(" mm");

    display.setCursor(0,35);
    display.println("Y:");
    display.setCursor(25,35);
    display.println(Y_coordinate);
    display.setCursor(90,35);
    display.println(" mm");

    if (Plotter_status == "Plotting..."){display.fillRect(0, 51, 128, 13, WHITE); display.setTextColor(BLACK);}

    display.setCursor(0,54);
    display.setTextSize(1);
    display.println("STATUS:");
    display.setCursor(45,54);
    display.setTextSize(1);
    display.println(Plotter_status);

  } else if (Plotter_status == "E-Stop !!") {

    display.fillRect(0, 0, 128, 16, WHITE);
    display.setTextColor(BLACK);

    display.setCursor(5,1);
    display.setTextSize(2);
    display.println(">>E-STOP<<");

    display.drawBitmap( 2, 22, logo_estop, LOGO_WIDTH, LOGO_HEIGHT, 1);

    display.setTextColor(WHITE);

    display.setCursor(44,20);
    display.setTextSize(1);
    display.println("E-STOP ACTIVE!");

    display.setCursor(46,40);
    display.setTextSize(1);
    display.println("  LONGPRESS");
    display.setCursor(44,50);
    display.setTextSize(1);
    display.println("TO DEACTIVATE");

  } else if (Plotter_status == "Warning"){
    display.fillRect(0, 0, 128, 16, WHITE);
    display.setTextColor(BLACK);

    display.setCursor(11,1);
    display.setTextSize(2);
    display.println("!WARNING!");

    display.setTextColor(WHITE);

    display.setCursor(8,18);
    display.setTextSize(2);
    display.println("X");
    display.setCursor(21,18);
    display.setTextSize(1);
    display.println("MIN");
    display.setCursor(43,18);
    display.setTextSize(2);
    display.println(Xmin_limit);

    display.setCursor(75,18);
    display.setTextSize(2);
    display.println("X");
    display.setCursor(89,18);
    display.setTextSize(1);
    display.println("MAX");
    display.setCursor(110,18);
    display.setTextSize(2);
    display.println(Xmax_limit);

    display.setCursor(8,35);
    display.setTextSize(2);
    display.println("Y");
    display.setCursor(21,35);
    display.setTextSize(1);
```

```
      display.println("MIN");
      display.setCursor(43,35);
      display.setTextSize(2);
      display.println(Ymin_limit);

      display.setCursor(75,35);
      display.setTextSize(2);
      display.println("Y");
      display.setCursor(89,35);
      display.setTextSize(1);
      display.println("MAX");
      display.setCursor(110,35);
      display.setTextSize(2);
      display.println(Ymax_limit);

      display.setCursor(6,53);
      display.setTextSize(1);
      display.println("LIMIT SWITCH ACTIVE!");
    }
    display.display();//Finally display the created image
  }
```