// Define stepper motor connections:

#define dirPinx 8

#define stepPinx 7

#define dirPiny 6

#define stepPiny 5

#define stepPinp 9

#define xhome 11

#define yhome 10

// include libraries

#include <Stepper.h>

// \*\*\*\*\* USER SETTINGS \*\*\*\*\*

int speed = 250; // ms delay speed for discrete motions (lower is faster)

int motor\_speed = 100; // us delay for motor speed (lower is faster)

bool discrete = false; // continuous or discrete movement

bool hardware\_trig = true; // software or hardware trigger

bool y\_down = false; // true = down, false = up

bool x\_right = true; // true = right, false = left

// \*\*\*\*\* USER SETTINGS \*\*\*\*\*

// \*\*\*\* Define SAR scan \*\*\*\* //

// 25,000 microsteps = ~20cm

// 201 x 25 positions -> 20cm x 20cm area

int x\_steps = 291; // 200 og value, 291 full range

int y\_steps = 125; // 100 og value, 125 full range

int x\_stepsize = 125; // 125

int y\_stepsize = 250; // 250

// 401 x 20 positions -> 20cm x 20cm area

// int x\_steps = 400; // ~20cm & ~5.23s

// int y\_steps = 19; //

// int y\_stepsize = 250;

// int x\_stepsize = 62;

// GLOABL VARIABLES

bool y\_start; // variable for storing y start direction

bool x\_start; // variable for storing x start direction

int in;

int yhomebutton = LOW;

int xhomebutton = LOW;

int scanstart = 0;

int scanend = 0;

int scantime = 0;

void setup() {

// Declare pins as output:

pinMode(stepPinx, OUTPUT);

pinMode(dirPinx, OUTPUT);

pinMode(stepPiny, OUTPUT);

pinMode(dirPiny, OUTPUT);

pinMode(stepPinp, OUTPUT);

pinMode(xhome, INPUT\_PULLUP);

pinMode(yhome, INPUT\_PULLUP);

// digitalWrite(xhome,LOW);

//digitalWrite(yhome,LOW);

// Set the spinning direction CW/CCW: high is clockwise

digitalWrite(dirPinx, HIGH); // HIGH = right, LOW =

digitalWrite(dirPiny, HIGH);

// Set all pin states

digitalWrite(stepPinx, LOW);

digitalWrite(stepPiny, LOW);

digitalWrite(stepPinp, LOW);

Serial.begin(115200);

Serial.print("\n\nConnected!\n");

delay(500);

Serial.print("Connected!\n");

delay(500);

Serial.print("Connected!\n");

delay(500);

Serial.print("Connected!\n");

delay(2000); // delay 3s on start

}

void loop() {

x\_start = x\_right; // store x start direction

y\_start = y\_down; // store y start direction

in = mainMenu();

if(in == 1) home();

if(in == 2){

// countdown();

scanstart =millis();

xy\_motion();

scanend =millis();

scantime = scanstart - scanend;

Serial.print("\n\nFull Scan Time: ");

Serial.print(scantime);

//ROM(1);

}

if(in == 3){

int input = 0;

while (input == 0){

Serial.print("\n\nSelect one of the following:");

delay(500);

Serial.print("\n1.X-Axis Calibrate\n2.Y-Axis Calibrate");

input = userinput(1,3);

if(input == 1) x\_calibrate();

if(input == 2) y\_calibrate();

if(input > 2 && input < 1) input = 0;

}

}

if(in == 4){

int input = 0;

while (input == 0){

Serial.print("\n\nSelect one of the following:");

delay(500);

Serial.print("\n1.X-Axis Range of Motion\n2.Y-Axis Range of Motion\n3.Both Axis Range of Motion");

input = userinput(1,3);

if(input == 1) xROM(1);

if(input == 2) yROM(1);

if(input == 3){

xROM(1);

yROM(1);

}

if(input > 3 && input < 1) input = 0;

}

}

if(in == 5){

for(int i=0; i< 40000; i++) {

frame();

delay(25);

}

}

if(in == 6) xOnlyScan();

if(in == 7) xScanWithOneYStep();

if(in == 8) ydown1();

// for manual scanner positioning

// for (int i=0; i<4; i++) move\_left();

// for (int i=0; i<10; i++) move\_right();

// for (int i=0; i<2; i++) move\_up();

// for (int i=0; i<1; i++) move\_down();

// box(); // right, up, down, left

// ROM(1); // Range of motion (n times)

// delay(3000); // delay on completion

// while(1); // hang on completion

}

void frame() { // this sends a pulse to the radar board hardware trigger

digitalWrite(stepPinp, HIGH);

delayMicroseconds(40);

digitalWrite(stepPinp, LOW);

delayMicroseconds(40);

}

void step\_x(bool dir) { // true = right, false = left

// Set the direction (HIGH = clockwise, LOW = counterclockwise)

if (dir) {

digitalWrite(dirPinx, HIGH); // clockwise

}

else {

digitalWrite(dirPinx, LOW); //counterclockwise

}

// Create a single step:

digitalWrite(stepPinx, HIGH);

delayMicroseconds(motor\_speed); // Adjust this value to control the speed

digitalWrite(stepPinx, LOW);

delayMicroseconds(motor\_speed); // This delay ensures proper timing between steps

}

void step\_y(bool dir) { // true = down, false = up

// Set the direction (HIGH = clockwise, LOW = counterclockwise)

if (dir) {

digitalWrite(dirPiny, HIGH); // clockwise

}

else {

digitalWrite(dirPiny, LOW); //counterclockwise

}

// Create a single step:

digitalWrite(stepPiny, HIGH);

delayMicroseconds(motor\_speed); // Adjust this value to control the speed

digitalWrite(stepPiny, LOW);

delayMicroseconds(motor\_speed); // This delay ensures proper timing between steps

}

void move\_x(bool dir) { // true = right, false = left

for (int i=0; i<x\_stepsize; i++) {

step\_x(dir);

}

}

void move\_y(bool dir) { // true = down, false = up

for (int i=0; i<y\_stepsize; i++) {

step\_y(dir);

}

}

// helper functions

void move\_right() { move\_x(true); }

void move\_left() { move\_x(false); }

void move\_up() { move\_y(false); }

void move\_down() { move\_y(true); }

void y\_motion() { // this executes a low level y motion

for(int i=0; i<y\_steps; i++) {

// signal radar board to capture a frame

if (hardware\_trig) frame();

if (discrete) delay(speed); // delay before moving

// step Y motor

move\_y(y\_down);

if (discrete) delay(speed); // delay after moving

}

// signal radar board to capture last frame

if (hardware\_trig) frame();

y\_down = !(y\_down); // change direction

if (discrete) delay(speed); // delay before moving

}

void x\_motion() { // this executes a low level x motion

for(int i=0; i<x\_steps; i++) {

// signal radar board to capture a frame

if (hardware\_trig) frame();

if (discrete) delay(speed); // delay before moving

// step Y motor

move\_x(x\_right);

if (discrete) delay(speed); // delay after moving

}

// signal radar board to capture last frame

if (hardware\_trig) frame();

x\_right = !(x\_right); // change direction

if (discrete) delay(speed); // delay before moving

}

void reset\_yx() { // this resets an xy-axis scan moving vertically first

// reset x axis

for(int i=0; i<x\_steps; i++) {

// step x motor

move\_x(!x\_start);

}

// reset y axis

if(y\_start != y\_down) {

for(int i=0; i<y\_steps; i++) {

// step y motor

move\_y(y\_down);

}

}

}

void reset\_xy() { // this resets an xy-axis scan moving horizontally first

// reset y axis

for(int i=0; i<y\_steps; i++) {

// step y motor

move\_y(!y\_start);

}

// reset x axis

if(x\_start != x\_right) { // new?

for(int i=0; i<x\_steps; i++) {

// step x motor

move\_x(x\_right);

}

}

// if(x\_right) { // OG

// for(int i=0; i<x\_steps; i++) {

// // step x motor

// move\_x(x\_right);

// }

// }

}

void yx\_motion() { // this is an xy-axis scan moving vertically first

for(int i=0; i<x\_steps; i++) {

// do Y motion

y\_motion();

// step X motor

move\_x(x\_right);

if (discrete) delay(speed);

}

// do final Y motion

y\_motion();

reset\_yx(); // reset x-y motors to original position

}

void xy\_motion() { // this is an xy-axis scan moving horizontally first

for(int i=0; i<y\_steps; i++) {

// do x motion

x\_motion();

// step y motor

move\_y(y\_down);

if (discrete) delay(speed);

}

// do final Y motion

x\_motion();

reset\_xy(); // reset x-y motors to original position

}

void box() { // for testing purposes

for (int i=0; i<4; i++) move\_right();

if (discrete) delay(speed);

for (int i=0; i<4; i++) move\_up();

if (discrete) delay(speed);

for (int i=0; i<4; i++) move\_left();

if (discrete) delay(speed);

for (int i=0; i<4; i++) move\_down();

if (discrete) delay(speed);

}

void countdown() { // for testing purposes

for (int i=0; i<500; i++) step\_x(true);

delay(speed);

for (int i=0; i<500; i++) step\_y(false);

delay(speed);

for (int i=0; i<500; i++) step\_x(false);

delay(speed);

for (int i=0; i<500; i++) step\_y(true);

delay(speed);

}

void ROM(int n) {

for (int j=0; j<n; j++) {

for (int i=0; i<x\_steps; i++) move\_right();

//delay(speed);

for (int i=0; i<y\_steps; i++) move\_up();

delay(speed);

/\*for (int i=0; i<x\_steps; i++) move\_left();

delay(speed);

for (int i=0; i<y\_steps; i++) move\_down();

delay(speed);

\*/

}

}

void xROM(int n) {

for (int j=0; j<n; j++) {

for (int i=0; i<x\_steps; i++) move\_right();

}

}

void yROM(int n) {

for (int j=0; j<n; j++) {

for (int i=0; i<y\_steps; i++) move\_up();

}

}

void home() {

int xy = 0;

xhomebutton = digitalRead(xhome);

yhomebutton = digitalRead(yhome);

//reset both axis to save time

while(xhomebutton == HIGH && yhomebutton == HIGH) {

//Serial.print("xhome and yhome running:\n");

// step x motor

step\_x(!x\_start);

xhomebutton = digitalRead(xhome);

// step y motor

step\_y(!y\_start);

yhomebutton = digitalRead(yhome);

}

// reset x axis alone

xhomebutton = digitalRead(xhome);

//Serial.print("xhomebutton: ");

//Serial.print(xhomebutton);

while(xhomebutton == HIGH) {

// step x motor

//Serial.print("xhome running:\n");

step\_x(!x\_start);

xhomebutton = digitalRead(xhome);

}

// reset y axis alone

yhomebutton = digitalRead(yhome);

//Serial.print("yhomebutton: ");

//Serial.print(yhomebutton);

while(yhomebutton == HIGH) {

// step y motor

//Serial.print("yhome running:\n");

step\_y(!y\_start);

yhomebutton = digitalRead(yhome);

}

for(xy = 0;xy <=15; ++xy){

step\_x(x\_start);

step\_y(y\_start);

}

}

int mainMenu(){

int userInput = 0;

Serial.print("\n\nPlease select one of the following options:\n1.Home Position\n2.Run Program\n3.Calibrate\n4.ROM\n5.Frame Trigger Test\n6.X-Axis Only Scan\n7.X-Axis with Single Y-Axis Step\n8.Y-Axis 1 Step Down");

while (userInput == 0 ) {

// Wait for user input (for example, from a serial interface)

if (Serial.available() > 0) {

userInput = Serial.parseInt(); // Read an integer from the serial input

if (userInput >= 1 && userInput <= 8) {

// Process the user input (valid range)

// You can use userInput as needed

Serial.print("\n\nReceived input: ");

Serial.println(userInput);

} else {

// Handle input outside the valid range

Serial.print("\n\nReceived input: ");

Serial.println(userInput);

Serial.println("Invalid input. Enter a number between 1 and 7.");

int userInput = 0;

delay(3000);

}

}

}

return userInput;

}

int userinput(int x, int y){

int userInput = 0;

while (userInput == 0 ) {

// Wait for user input (for example, from a serial interface)

if (Serial.available() > 0) {

userInput = Serial.parseInt(); // Read an integer from the serial input

if (userInput >= x && userInput <= y) {

// Process the user input (valid range)

// You can use userInput as needed

Serial.print("\n\nReceived input: ");

Serial.println(userInput);

} else {

// Handle input outside the valid range

Serial.print("\n\nReceived input: ");

Serial.println(userInput);

Serial.print("Invalid input. Enter a number between ");

Serial.print(x);

Serial.print(" and ");

Serial.print(y);

delay(3000);

}

}

}

return userInput;

}

void x\_calibrate(){

int startcount = 0;

int endcount = 0;

int xtime = 0;

unsigned int j = 0;

//reset x axis alone

xhomebutton = digitalRead(xhome);

startcount =millis();

//Serial.print("xhomebutton: ");

//Serial.print(xhomebutton);

while(xhomebutton == HIGH) {

// step x motor

//Serial.print("xhome running:\n");

step\_x(!x\_start);

xhomebutton = digitalRead(xhome);

j = j+ 1;

}

endcount =millis();

xtime = endcount - startcount;

Serial.print("xtime: ");

Serial.print(xtime);

Serial.print(" ms\n");

Serial.print(j);

Serial.print(" Steps");

}

void y\_calibrate(){

int startcount = 0;

int endcount = 0;

int xtime = 0;

unsigned int j = 0;

//reset x axis alone

yhomebutton = digitalRead(yhome);

startcount =millis();

//Serial.print("xhomebutton: ");

//Serial.print(xhomebutton);

while(yhomebutton == HIGH) {

// step x motor

//Serial.print("xhome running:\n");

step\_y(!y\_start);

yhomebutton = digitalRead(yhome);

j = j+ 1;

}

endcount =millis();

xtime = endcount - startcount;

Serial.print("xtime: ");

Serial.print(xtime);

Serial.print(" ms\n");

Serial.print(j);

Serial.print(" Steps");

}

void xOnlyScan(){

for (int i=0; i<x\_steps; i++) move\_right();

delay(500);

for (int i=0; i<x\_steps; i++) move\_left();

}

void xScanWithOneYStep(){

for (int i=0; i<x\_steps; i++) move\_right();

delay(500);

for (int i=0; i<x\_steps; i++) move\_left();

delay(500);

move\_y(y\_down);

}

void ydown1(){

yhomebutton = digitalRead(yhome);

if(yhomebutton == HIGH) move\_y(!y\_down);

}