// Definitionen

#include <Servo.h> // servo treiber

Servo servo\_0;

Servo servo\_1;

Servo servo\_2;

Servo servo\_3;

int sensorPin0 = A0; // Schulter

int sensorPin1 = A1; // Handfind

int sensorPin2 = A2; // Ellbogen

int sensorPin3 = A3; // Zange

int count0, arrayStep, arrayMax, countverz, Taster, stepsMax, steps, time = 1000, del = 1000, temp;

// arraystep = memory what pos in the array

// arrayMax = max steps we safed to array

// countverz = seems to be something to calculate the delay between complete moves

// Taster = Button

// stepsMax = longest way a servo have to travel

// steps = single steps for a move between stored positions

unsigned int verz = 0;

long previousMillis1 = 0;

long previousMillis2 = 0;

long previousMillis3 = 0;

long previousMillis4 = 0;

long previousMicros = 0;

unsigned long currentMillis = millis();

unsigned long currentMicros = micros();

// arrays

int Delay[7] = {0,0,1,3,15,60,300}; // array to map gripper pot to delay in seconds

int SensVal[4]; // sensor value

float dif[4], ist[4], sol[4], dir[4]; // difference between stored position and momentary position

int joint0[180];// array for servo(s)

int joint1[180];

int joint2[180];

int joint3[180];

int top = 179; // we should not write over the end from a array

// status

boolean playmode = false, Step = false;

void setup()

{

 pinMode(4, INPUT); // sets the digital pin 4 as input

 pinMode(6, INPUT);

 pinMode(13, OUTPUT); // sets the digital pin 13 as outtput

 digitalWrite(13, HIGH); // sets the LED on

 servo\_0.attach(3); // attaches the servo

 servo\_1.attach(10);

 servo\_2.attach(9);

 servo\_3.attach(11);

 Serial.begin(115200); // Baudrate have to be same on the IDE

 Serial.println("mini robot ready...");

 //delay(1000);

 digitalWrite(13, LOW);

}

void loop() // here we go!

{

 currentMillis = millis(); // all is about timing

 currentMicros = micros();

 // read the button

 Button();

 if(!playmode) // manualy modus

 {

 if(currentMillis - previousMillis1 > 25) // 25miliseconds until next manual mode update

 {

 if (arrayStep < top)

 {

 previousMillis1 = currentMillis; //reset

 readPot(); // get the value from potentiometers

 mapping(); // map to milliseconds for servos

 move\_servo(); // setz newservo position

 //record();

 } // end counter < max

 } // end step check

 } // ende manualy move

 else if(playmode) // play

 {

 if (Step) // next step read from array

 {

 digitalWrite(13, HIGH); //LED

 if (arrayStep < arrayMax) // we not reach the end from stored data

 {

 arrayStep += 1; // next array pos

 Read(); // from the arrays

 calculate(); // find biggest travel distance and calculate the other 3 servos (the have to do smaler steps to be finished at same time!)

 Step = 0;

 digitalWrite(13, LOW);

 }

 else // array read finished > start over

 {

 arrayStep = 0; //

 calc\_pause(); // delay between moves read from potentiometer

 countverz = 0; // used for the delay

 while(countverz < verz) // verz = time getting from calc\_pause();

 { // here we do loop and wait until next start over

 countverz += 1;

 calc\_pause();

 digitalWrite(13, HIGH); delay(25);

 digitalWrite(13, LOW); delay(975);

 }

 }

 //Serial.println(arrayStep);

 }

 else // do the servos!

 {

 if (currentMicros - previousMicros > time) // here we do a single micro step

 { //

 previousMicros = currentMicros;

 play\_servo();

 }

 }

 }// ende playmode

// ---------------------------------------------------------------------------------Hardware pause switch PIN 6

 while (digitalRead(4) == false)

 {

 digitalWrite(13, HIGH); delay(500);

 digitalWrite(13, LOW); delay(500);

 }

// ---------------------------------------------------------------------------------- Textout serial

 // serial ausgabe 1 sek

 /\*if(currentMillis - previousMillis2 > 5000)

 {

 previousMillis2 = currentMillis;

 /\*count0 = 0;

 while(count0 < 4)

 {

 int val = SensVal[count0];

 // val = map(val, 142, 888, 0, 180);

 Serial.println(val);

 //Serial.println("test");

 count0 += 1;

 }

 Serial.println(playmode);

 Serial.println(arrayStep);

 Serial.println(arrayMax);

 Serial.println(" ");

 }\*/

}

// ---------------------------------------------------------------------------------------- sub routinen

void calc\_pause() // read pot and map to usable delay time after a complete move is done

{

 readPot();

 temp = SensVal[3];

 if (temp < 0) temp = 0;

 temp = map(temp, 0, 680, 0 ,6);

 verz = Delay[temp]; // verz = delay in second

 /\*Serial.print(temp);

 Serial.print(" ");

 Serial.print(verz);

 Serial.print(" ");

 Serial.println(countverz);\*/

}

void readPot() // read analog inputs and add some offsets (mechanical corrections)

{

 SensVal[0] = analogRead(sensorPin0); //SensVal[0] += -10; // rotate

 SensVal[1] = analogRead(sensorPin1); //SensVal[1] += 280; // Shoulder

 SensVal[2] = analogRead(sensorPin2); //SensVal[2] += -50; // hand

 SensVal[3] = analogRead(sensorPin3); // SensVal[3] += 0;// gripper

 Serial.print(SensVal[2]);Serial.print(" "); // CHECK

}

void mapping() // we need microsecond for the servos instead potentiometer values

{

 ist[0] = map(SensVal[0], 150, 900, 600, 2400);// drehen

 ist[1] = map(SensVal[1], 1000, 100, 550, 2400);// Schulter

 ist[2] = map(SensVal[2], 120, 860, 400, 2500);// Hand

 ist[3] = map(SensVal[3], 1023, 0, 500, 2500);// Zange

 Serial.println(ist[2]); // CHECK

}

void record()

{

 joint0[arrayStep] = ist[0]; // write positions in servo array

 joint1[arrayStep] = ist[1];

 joint2[arrayStep] = ist[2];

 joint3[arrayStep] = ist[3];

}

void Read()

{

 sol[0] = joint0[arrayStep]; // read from the array

 sol[1] = joint1[arrayStep];

 sol[2] = joint2[arrayStep];

 sol[3] = joint3[arrayStep];

}

void move\_servo()

{

 servo\_0.writeMicroseconds(ist[3]); // send milissecond values to servos

 servo\_1.writeMicroseconds(ist[2]);

 servo\_2.writeMicroseconds(ist[0]);

 servo\_3.writeMicroseconds(ist[1]);

}

// ------------------------------------------------------------ single steps calculating

void calculate()

{

 // travel distance for each servo

 dif[0] = abs(ist[0]-sol[0]);

 dif[1] = abs(ist[1]-sol[1]);

 dif[2] = abs(ist[2]-sol[2]);

 dif[3] = abs(ist[3]-sol[3]);

 // biggest travel way from all 4 servos

 stepsMax = max(dif[0],dif[1]);

 stepsMax = max(stepsMax,dif[2]);

 stepsMax = max(stepsMax,dif[3]);

 // stepsMax is the biggest distance a servo have to do beween momentary position and new pos read from the array

 //Serial.println(stepsMax);

 if (stepsMax < 500) // del(ay) between a single step is bigger is move is smaler. just looks cool

 del = 1200;

 else

 del = 600;

 // calculating single (micro) step for each servo

 // need that to do move all servos in a loop (stepsMax times done) with different values.

 // This makes all servos have done the traveling distance at same time

 if (sol[0] < ist[0]) dir[0] = 0-dif[0]/stepsMax; else dir[0] = dif[0]/stepsMax;

 if (sol[1] < ist[1]) dir[1] = 0-dif[1]/stepsMax; else dir[1] = dif[1]/stepsMax;

 if (sol[2] < ist[2]) dir[2] = 0-dif[2]/stepsMax; else dir[2] = dif[2]/stepsMax;

 if (sol[3] < ist[3]) dir[3] = 0-dif[3]/stepsMax; else dir[3] = dif[3]/stepsMax;

 //Serial.println(dir4);

}

void play\_servo()

{

 steps += 1;

 if (steps < stepsMax) // sure we not reach the end from a move

 {

 //time = del\*5;// anfahr rampe

 if(steps == 20) time = del\*4; // ramp up

 else if(steps == 40) time = del\*3; // time is the delay in microsecns we wait in the mainloop until

 else if(steps == 80) time = del\*2; // a micro step will be done

 else if(steps == 100) time = del-1; // cannot explain here is not del\*1

 if(steps == stepsMax-200) time = del\*2; // stop ramp down (200 microsteps before end time will be increased

 else if(steps == stepsMax-80) time = del\*3;

 else if(steps == stepsMax-40) time = del\*4;

 else if(steps == stepsMax-20) time = del\*5;

 ist[0] += dir[0]; // set new pos

 ist[1] += dir[1];

 ist[2] += dir[2];

 ist[3] += dir[3];

 servo\_0.writeMicroseconds(ist[3]); // Zange //anschlÃ¼sse gemappt!

 servo\_1.writeMicroseconds(ist[2]); // Hand

 servo\_2.writeMicroseconds(ist[0]); // Schulter

 servo\_3.writeMicroseconds(ist[1]); // Ellbogen

 }

 else

 {

 Step = 1; // next step aus array lesen

 steps = 0; // servo zwischenschritte

 }

}

void data\_out() // just to write the recorded data to serial

{

 int i = 0;

 while(i < arrayMax)

 {

 digitalWrite(13, HIGH);

 i += 1;

 Serial.print(joint0[i]); Serial.print(", ");

 }

 Serial.println("Joint0");

 i = 0;

 while(i < arrayMax)

 {

 digitalWrite(13, HIGH);

 i += 1;

 Serial.print(joint1[i]); Serial.print(", ");

 }

 Serial.println("Joint1");

 i = 0;

 while(i < arrayMax)

 {

 digitalWrite(13, HIGH);

 i += 1;

 Serial.print(joint2[i]); Serial.print(", ");

 }

 Serial.println("Joint2");

 i = 0;

 while(i < arrayMax)

 {

 digitalWrite(13, HIGH);

 i += 1;

 Serial.print(joint3[i]); Serial.print(", ");

 }

 Serial.println("Joint3");

}

void Button() // check buttons for single and doubleclick

{

 if (digitalRead(6) == false)

 {

 delay(1);

 if (digitalRead(6) == true) // taster losgelassen

 {

 if (Taster == 0)

 {

 Taster = 1;

 previousMillis3 = currentMillis;

 //Serial.print("Status Record "); Serial.println(Taster);

 }

 else if ((Taster == 1) && (currentMillis - previousMillis3 < 250))

 {

 Taster = 2;

 //Serial.println(Taster);

 }

 /\*else if ((Taster == 2) && (currentMillis - previousMillis3 < 500))

 {

 Taster = 3;

 Serial.println(Taster);

 }\*/

 }

 }

 if ((Taster == 1) && (currentMillis - previousMillis3 > 1000)) // write to array

 {

 arrayStep += 1;

 arrayMax = arrayStep;

 record();

 Taster = 0;

 playmode = false;

 Serial.print("Record Step: "); Serial.println(arrayStep);

 digitalWrite(13, HIGH);

 delay(100);

 digitalWrite(13, LOW);

 }

 else if (Taster == 2)

 {

 arrayStep = 0;

 playmode = true;

 Taster = 0;

 Step = 1;

 Serial.println("playmode ");

 data\_out();

 delay(250);

 digitalWrite(13, LOW);

 }

 /\*if (Taster == 3)

 {

 // ++ arrayStep

 // playmode = 1;

 Taster = 0;

 Serial.println("Clear ");

 }\*/

 if (currentMillis - previousMillis3 > 2000) // button Status clear

 {

 Taster = 0;

 //Serial.println("restart ");

 }

}