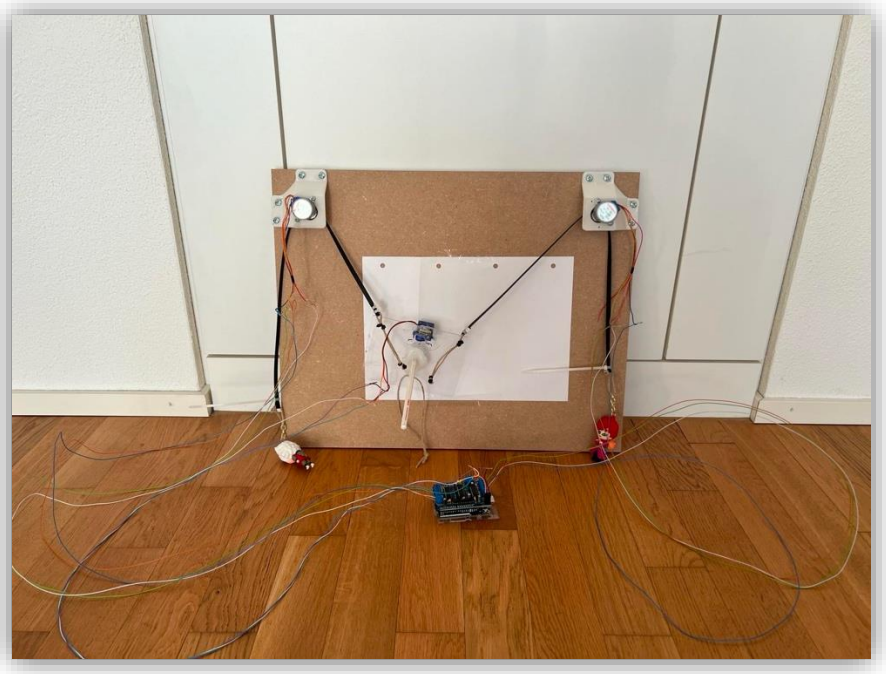
Projektarbeit - XY-Vertical Plotter



TEKO Zürich, 22. Januar 2024

5. Semester

Fach

Mikroprozessor

Dozent

Christian Meier

Projektleiter

Festim Elmazi

Inhaltsverzeichnis

1. Einleitung

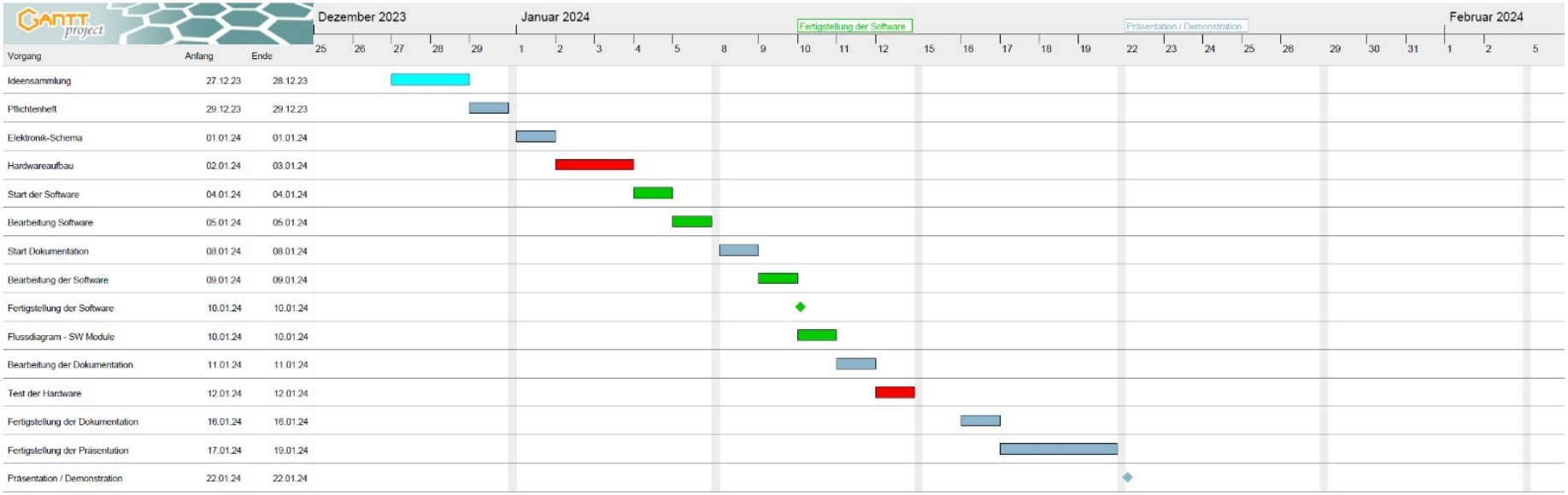
Im Rahmen des Mikroprozessor-Fachs an der TEKO ZH sind die Studierenden im fünften Semester dazu verpflichtet, eine Projektarbeit zu verfassen. Diese kann entweder allein oder in Zweiergruppen durchgeführt werden. Die nachfolgende Beschreibung und Struktur des Dokuments erläutert die Vorgehensweise. Abschließend wird das Projekt in einer Präsentation vorgestellt und demonstriert. Mein Ziel war es, ein praxisrelevantes Projekt zu entwickeln, das in der realen Anwendung finden kann.

2. Terminplan

(siehe Tabelle nächste Seite)

XY Plotter

Gantt-Diagramm

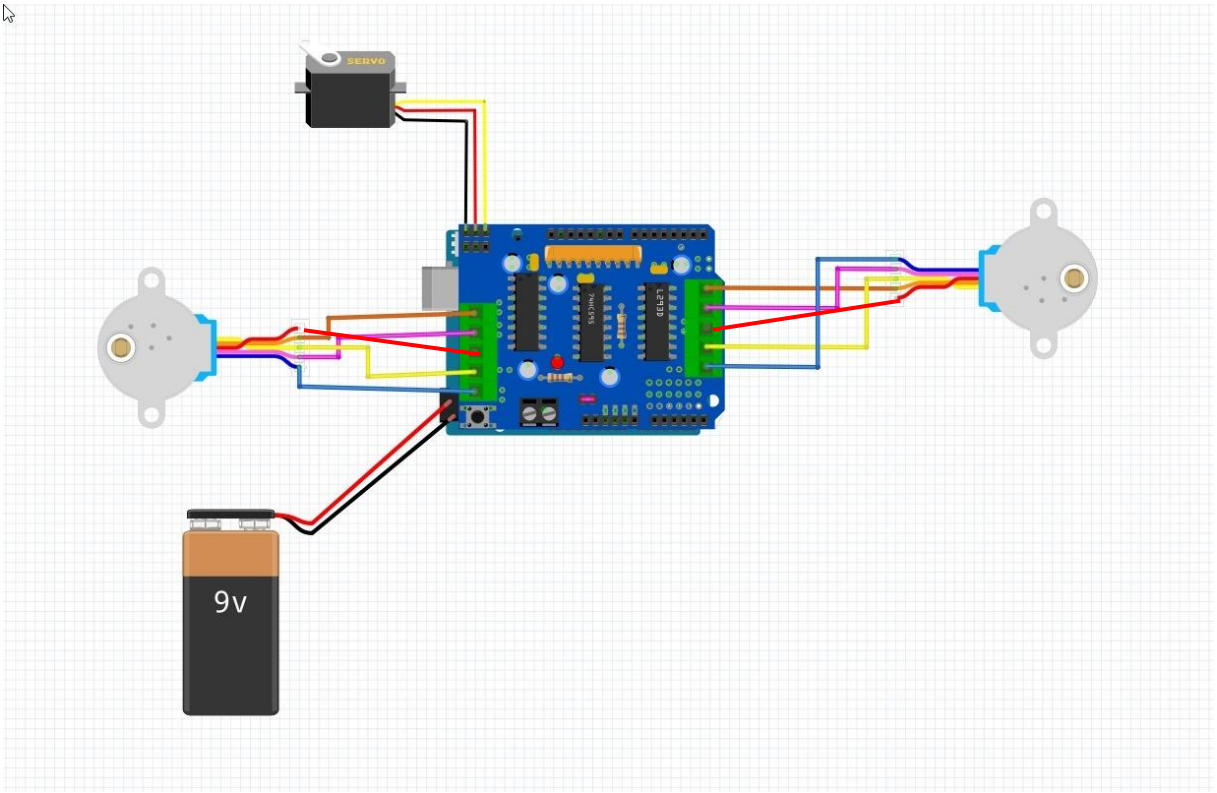


Vorgang	Anfang	Ende
Ideensammlung	27.12.23	28.12.23
Pflichtenheft	29.12.23	29.12.23
Elektronik-Schema	01.01.24	01.01.24
Hardwareaufbau	02.01.24	03.01.24
Start der Software	04.01.24	04.01.24
Bearbeitung Software	05.01.24	05.01.24
Start Dokumentation	08.01.24	08.01.24
Bearbeitung der Software	09.01.24	09.01.24
Fertigstellung der Software	10.01.24	10.01.24
Flussdiagram - SW Module	10.01.24	10.01.24
Bearbeitung der Dokumentation	11.01.24	11.01.24
Test der Hardware	12.01.24	12.01.24
Fertigstellung der Dokumentation	16.01.24	16.01.24
Fertigstellung der Präsentation	17.01.24	19.01.24
Präsentation / Demonstration	22.01.24	22.01.24

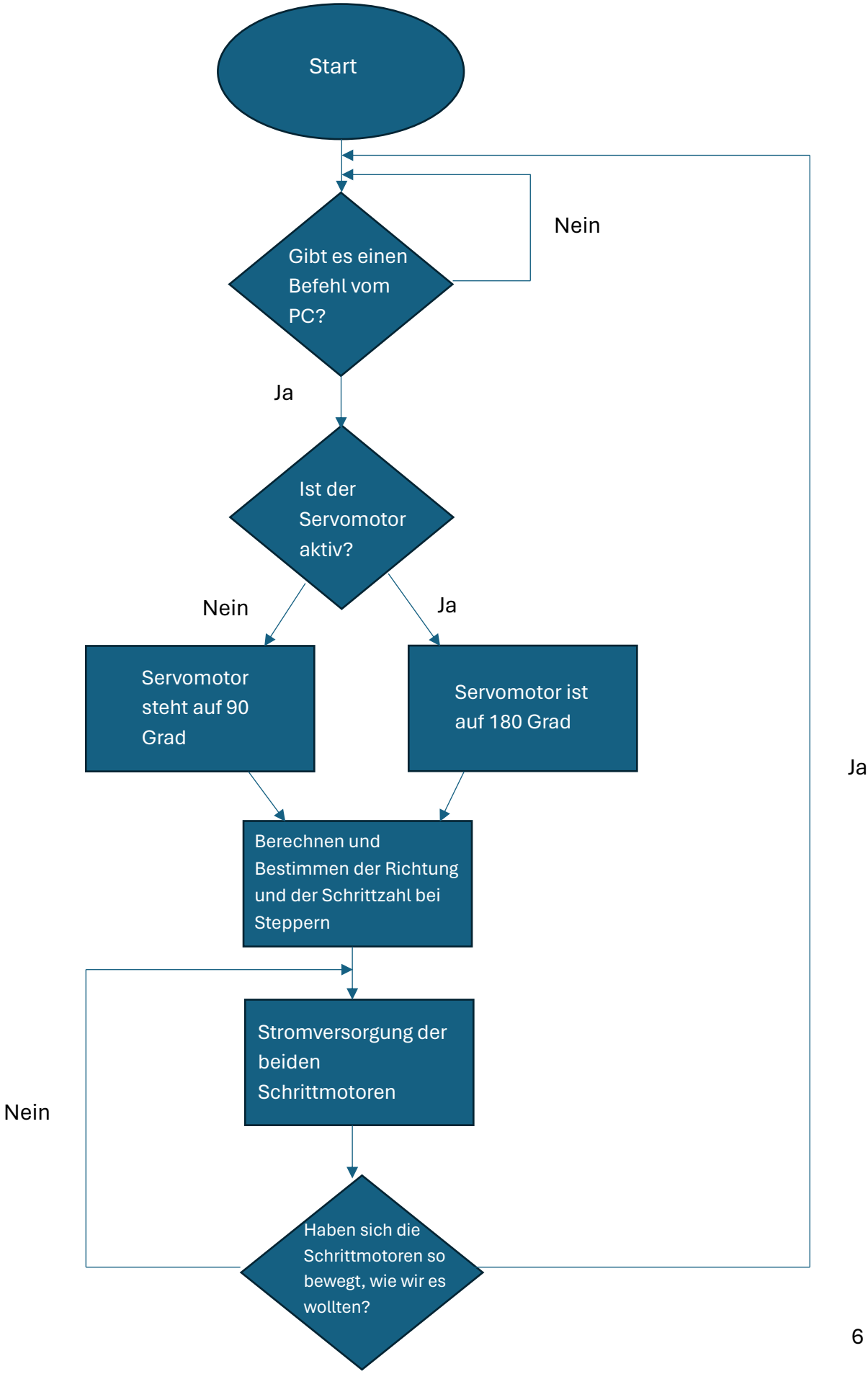
3. Pflichtheft Aufgabenstellung

Ich möchte einen XY-Plotter bauen, der Bilder genau zeichnet und direkt vom PC gesteuert wird. Dafür brauche ich eine gute Verbindung zum PC, passende Motoren und einfach zu bedienende Software. Ich werde Fehlerprüfungen einbauen, gründliche Tests machen und alles detailliert dokumentieren. Bei der Vorstellung zeige ich dann live, wie der XY-Plotter mit PC-Steuerung funktioniert.

4. Elektronik Schema



5. Flussdiagramm



6. Code Modularisierung / Regeln / Cleancode

Polargrafcontroller-Code

Der gegebene Code ist eine Arduino-Skizze, die verschiedene Definitionen, Konstanten und Variablen enthält. Es werden Bibliotheken wie AFMotor, AccelStepper, Servo und EEPROM verwendet. Der Code enthält auch Funktionen für die Konfiguration des Motors, das Laden von Maschinenspezifikationen aus dem EEPROM und die Kommunikation über die serielle Schnittstelle.

Es gibt auch verschiedene Befehle, die über die serielle Schnittstelle empfangen und ausgeführt werden können, wie das Ändern der Länge, die Steuerung des Stifts (Penlift), das Zeichnen von Pixeln und Vektoren, das Setzen der Maschinengröße und das Zurücksetzen des EEPROMs.

Es gibt auch spezifische Teile des Codes, die nur für den Arduino Mega relevant sind, wie das Testen der Stiftbreite, das Zeichnen von Kreisen und das Lesen von Bitmap-Dateien von einer SD-Karte.

```
#ifndef MICROCONTROLLER
#define MICROCONTROLLER MC_UNO
//#define MICROCONTROLLER MC_MEGA
#endif

#define ADAFRUIT_MOTORSHIELD_V1
#include <AFMotor.h>
#define PIXEL_DRAWING
#define PENLIFT
#define VECTOR_LINES



#define MC_UNO 1
#define MC_MEGA 2

#include <AccelStepper.h>
#include <Servo.h>
#include <EEPROM.h>
#include "EEPROMAnything.h"

const String FIRMWARE_VERSION_NO = "1.2.1";

//  EEPROM addresses
```

```
const byte EEPROM_MACHINE_WIDTH = 0;
const byte EEPROM_MACHINE_HEIGHT = 2;
const byte EEPROM_MACHINE_MM_PER_REV = 14; // 4 bytes (float)
const byte EEPROM_MACHINE_STEPS_PER_REV = 18;
const byte EEPROM_MACHINE_STEP_MULTIPLIER = 20;

const byte EEPROM_MACHINE_MOTOR_SPEED = 22; // 4 bytes float
const byte EEPROM_MACHINE_MOTOR_ACCEL = 26; // 4 bytes float
const byte EEPROM_MACHINE_PEN_WIDTH = 30; // 4 bytes float

const byte EEPROM_MACHINE_HOME_A = 34; // 4 bytes
const byte EEPROM_MACHINE_HOME_B = 38; // 4 bytes

const byte EEPROM_PENLIFT_DOWN = 42; // 2 bytes
const byte EEPROM_PENLIFT_UP = 44; // 2 bytes

// Pen raising servo
Servo penHeight;
const int DEFAULT_DOWN_POSITION = 90;
const int DEFAULT_UP_POSITION = 180;
static int upPosition = DEFAULT_UP_POSITION; // defaults
static int downPosition = DEFAULT_DOWN_POSITION;
static int penLiftSpeed = 3; // ms between steps of moving motor
const byte PEN_HEIGHT_SERVO_PIN = 9; //UNL2003 driver uses pin 9
boolean isPenUp = false;

// Machine specification defaults
const int DEFAULT_MACHINE_WIDTH = 650;
const int DEFAULT_MACHINE_HEIGHT = 650;
const int DEFAULT_MM_PER_REV = 95;
const int DEFAULT_STEPS_PER_REV = 400;
const int DEFAULT_STEP_MULTIPLIER = 1;

// working machine specification
static int motorStepsPerRev = DEFAULT_STEPS_PER_REV;
static float mmPerRev = DEFAULT_MM_PER_REV;
static byte stepMultiplier = DEFAULT_STEP_MULTIPLIER;
static int machineWidth = DEFAULT_MACHINE_WIDTH;
static int machineHeight = DEFAULT_MACHINE_HEIGHT;


static float currentMaxSpeed = 800.0;
static float currentAcceleration = 400.0;
static boolean usingAcceleration = true;

int startLengthMM = 800;

float mmPerStep = 0.0F;
float stepsPerMM = 0.0F;
```

```
long pageWidth = machineWidth * stepsPerMM;
long pageHeight = machineHeight * stepsPerMM;
long maxLength = 0;

//static char rowAxis = 'A';
const int INLENGTH = 50;
const char INTERMINATOR = 10;
const char SEMICOLON = ';';

float penWidth = 0.8F; // line width in mm

boolean reportingPosition = true;
boolean acceleration = true;

extern AccelStepper motorA;
extern AccelStepper motorB;

boolean currentlyRunning = true;

static char inCmd[10];
static char inParam1[14];
static char inParam2[14];
static char inParam3[14];
static char inParam4[14];

static byte inNoOfParams;

char lastCommand[INLENGTH+1];
boolean commandConfirmed = false;

int rebroadcastReadyInterval = 5000;
long lastOperationTime = 0L;
long motorIdleTimeBeforePowerDown = 600000L;
boolean automaticPowerDown = true;
boolean powerIsOn = false;

long lastInteractionTime = 0L;

#ifdef PIXEL_DRAWING
static boolean lastWaveWasTop = true;

//  Drawing direction
const static byte DIR_NE = 1;
const static byte DIR_SE = 2;
const static byte DIR_SW = 3;
const static byte DIR_NW = 4;

static int globalDrawDirection = DIR_NW;

const static byte DIR_MODE_AUTO = 1;
```

```
const static byte DIR_MODE_PRESET = 2;
static byte globalDrawDirectionMode = DIR_MODE_AUTO;
#endif

#if MICROCONTROLLER == MC_MEGA
  #define READY_STR "READY_100"
#else
  #define READY_STR "READY"
#endif

#define RESEND_STR "RESEND"
#define DRAWING_STR "DRAWING"
#define OUT_CMD_SYNC_STR "SYNC,"

char MSG_E_STR[] = "MSG,E,";
char MSG_I_STR[] = "MSG,I,";
char MSG_D_STR[] = "MSG,D,";

const static char COMMA[] = ",";
const static char CMD_END[] = ",END";
const static String CMD_CHANGELENGTH = "C01";
const static String CMD_CHANGEPENWIDTH = "C02";
#ifdef PIXEL_DRAWING
const static String CMD_DRAWPIXEL = "C05";
const static String CMD_DRAWSCRIBBLEPIXEL = "C06";
const static String CMD_CHANGEDRAWINGDIRECTION = "C08";
const static String CMD_TESTPENWIDTHSQUARE = "C11";
#endif
const static String CMD_SETPOSITION = "C09";
#ifdef PENLIFT
const static String CMD_PENDOWN = "C13";
const static String CMD_PENUP = "C14";
const static String CMD_SETPENLIFTRANGE = "C45";
#endif
#ifdef VECTOR_LINES
const static String CMD_CHANGELENGTHDIRECT = "C17";
#endif
const static String CMD_SETMACHINESIZE = "C24";
const static String CMD_GETMACHINEDETAILS = "C26";
const static String CMD_RESETEEPROM = "C27";
const static String CMD_SETMACHINEMMPERREV = "C29";
const static String CMD_SETMACHINESTEPSPERREV = "C30";
const static String CMD_SETMOTORSPEED = "C31";
const static String CMD_SETMOTORACCEL = "C32";
const static String CMD_SETMACHINESTEPMULTIPLIER = "C37";

void setup()
{
  Serial.begin(57600);          // set up Serial library at 57600 bps
  Serial.println("POLARGRAPH ON!");
```

```
  Serial.print("Hardware: ");
  Serial.println(MICROCONTROLLER);

  #if MICROCONTROLLER == MC_MEGA
  Serial.println("MC_MEGA");
  #elif MICROCONTROLLER == MC_UNO
  Serial.println("MC_UNO");
  #else
  Serial.println("No MC");
  #endif
  configuration_motorSetup();
  eeprom_loadMachineSpecFromEeprom();
  configuration_setup();

  motorA.setMaxSpeed(currentMaxSpeed);
  motorA.setAcceleration(currentAcceleration);
  motorB.setMaxSpeed(currentMaxSpeed);
  motorB.setAcceleration(currentAcceleration);

  float startLength = ((float) startLengthMM / (float) mmPerRev) * (float)
motorStepsPerRev;
  motorA.setCurrentPosition(startLength);
  motorB.setCurrentPosition(startLength);
  for (int i = 0; i<INLENGTH; i++) {
    lastCommand[i] = 0;
  }
  comms_ready();

#ifdef PENLIFT
  penlift_penUp();
#endif
  delay(500);

}

void loop()
{
  if (comms_waitForNextCommand(lastCommand))
  {
#ifdef DEBUG_COMMS
    Serial.print(F("Last comm: "));
    Serial.print(lastCommand);
    Serial.println(F("..."));
#endif
    comms_parseAndExecuteCommand(lastCommand);
  }
}

#if MICROCONTROLLER == MC_MEGA
const static String CMD_TESTPENWIDTHSCRIBBLE = "C12";
```

```
const static String CMD_DRAWSAWPIXEL = "C15,";
const static String CMD_DRAWCIRCLEPIXEL = "C16";
const static String CMD_SET_ROVE_AREA = "C21";
const static String CMD_DRAWDIRECTIONTEST = "C28";
const static String CMD_MODE_STORE_COMMANDS = "C33";
const static String CMD_MODE_EXEC_FROM_STORE = "C34";
const static String CMD_MODE_LIVE = "C35";
const static String CMD_RANDOM_DRAW = "C36";
const static String CMD_START_TEXT = "C38";
const static String CMD_DRAW_SPRITE = "C39";
const static String CMD_CHANGELENGTH_RELATIVE = "C40";
const static String CMD_SWIRLING = "C41";
const static String CMD_DRAW_RANDOM_SPRITE = "C42";
const static String CMD_DRAW_NORWEGIAN = "C43";
const static String CMD_DRAW_NORWEGIAN_OUTLINE = "C44";

/*  End stop pin definitions  */
const int ENDSTOP_X_MAX = 17;
const int ENDSTOP_X_MIN = 16;
const int ENDSTOP_Y_MAX = 15;
const int ENDSTOP_Y_MIN = 14;

long ENDSTOP_X_MIN_POSITION = 130;
long ENDSTOP_Y_MIN_POSITION = 130;

// size and location of rove area
long rove1x = 1000;
long rove1y = 1000;
long roveWidth = 5000;
long roveHeight = 8000;

boolean swirling = false;
String spritePrefix = "";
int textRowSize = 200;
int textCharSize = 180;

boolean useRoveArea = false;

int commandNo = 0;
int errorInjection = 0;

boolean storeCommands = false;
boolean drawFromStore = false;
String commandFilename = "";

// sd card stuff
const int chipSelect = 53;
boolean sdCardInit = false;

// set up variables using the SD utility library functions:
```

```
File root;
boolean cardPresent = false;
boolean cardInit = false;
boolean echoingStoredCommands = false;

// the file itself
File pbmFile;

// information we extract about the bitmap file
long pbmWidth, pbmHeight;
float pbmScaling = 1.0;
int pbmDepth, pbmImageoffset;
long pbmFileLength = 0;
float pbmAspectRatio = 1.0;

volatile int speedChangeIncrement = 100;
volatile int accelChangeIncrement = 100;
volatile float penWidthIncrement = 0.05;
volatile int moveIncrement = 400;

boolean currentlyDrawingFromFile = false;
String currentlyDrawingFilename = "";

static float translateX = 0.0;
static float translateY = 0.0;
static float scaleX = 1.0;
static float scaleY = 1.0;
static int rotateTransform = 0;

#endif
```

7. Hardware

	Purecrea Motor Control Shield für Arduino V1.0 mit L293D	1x
	Purecrea Stepper Motor Schrittmotor 5V 1/64 (28BYJ-48)	2x
	Purecrea Z20-GT2-6 Riemenscheibe 5mm Bohrung 16mm Sitz	2x
	Purecrea GT2 Riemen 6mm	1x
	WaveShare Micro Servo 9G / SG90	1x
	Arduino Uno Rev3	1x

8. Ergebnisse (Test-Protokoll)

Frage (Testpunkt)	Antwort	Bemerkungen
Funktioniert die Stromversorgung einwandfrei?	Ja	--
Bewegt sich die X-Achse reibungslos?	Nein	--
Bewegt sich die Y-Achse reibungslos?	Nein	--
Sind alle Kabel sicher und korrekt angeschlossen?	Ja	Nach Schema
Reagiert die Softwaresteuerung auf Befehle?	Nein	--
Sind die Zeichnungen präzise und entsprechen den erwarteten Koordinaten?	Nein	Konnte nicht getestet werden.
Kann der Stift problemlos gewechselt werden?	Nein	--
Ist die Dokumentation vollständig?	Ja / Nein	Die schriftliche Dokumentation ist fertig. Die praktische Arbeit ist noch nicht betriebsbereit, da Probleme mit der Software aufgetreten sind.

9. SOLL- / IST-Abgleich

Soll: Der XY-Vertical Plotter sollte eine präzise und zuverlässige Zeichenmaschine sein. Er sollte in der Lage sein, komplexe und detaillierte Zeichnungen auf verschiedenen Oberflächen zu erstellen. Der Plotter sollte einfach zu bedienen sein und eine benutzerfreundliche Software bieten, um die Erstellung und Bearbeitung von Zeichnungen zu erleichtern. Außerdem sollte er über eine hohe Geschwindigkeit und Genauigkeit verfügen, um effizient arbeiten zu können.

Ist: Derzeit habe ich einige Herausforderungen mit dem XY-Vertical Plotter. Die Software zur Steuerung des Plotters ist komplex und nicht intuitiv, was die Bedienung erschwert. Der XY-Vertical Plotter nimmt keine Befehle vom PC an. Da ich das Problem nicht identifiziert habe ist mein Roboter nicht funktionstüchtig.

10. Quellenverzeichnis

https://www.arduino.cc/en/software

https://processing.org/download

https://github.com/euphy/polargraphcontroller/releases/tag/2017-11-01-20-30

https://www.thingiverse.com/thing:2371117