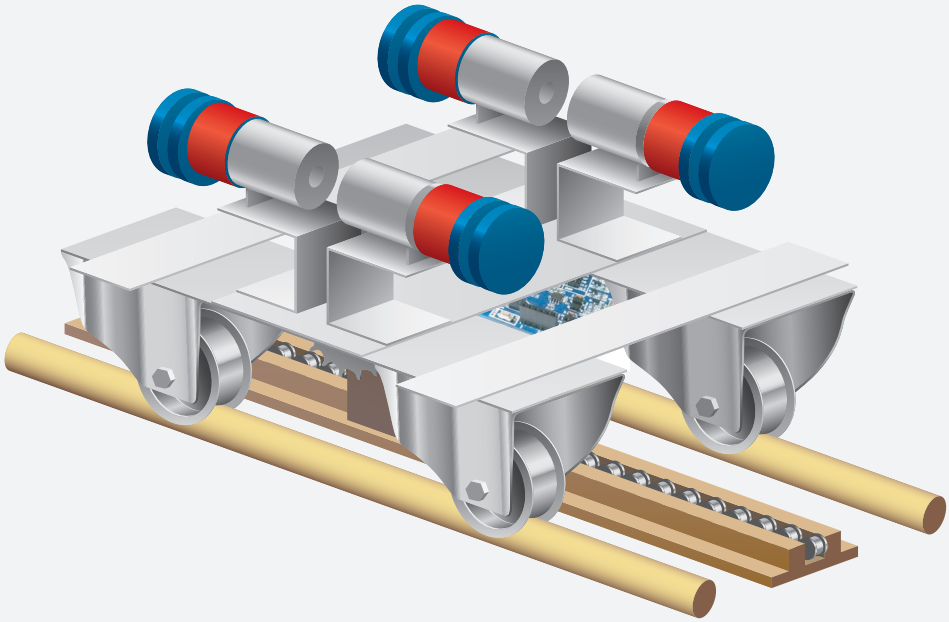
bitplant.de
ATD-Services

powered by TEXML

Twin Robot

STANDARD

Ausgabe: 2014-05-15



Originalbetriebsanleitung
Für künftige Verwendung aufbewahren!

Inhaltsverzeichnis

1 Zu dieser Betriebsanleitung

Die Betriebsanleitung verwendet Symbole und Zeichen, die Ihnen das schnelle Auffinden von Informationen erleichtern. Lesen Sie die Erläuterungen zu den Symbolen im folgenden Abschnitt.

1.1 Aufbau Betriebsanleitung

Kapitel	Inhalt
–	Register Titelblatt Kontrollblatt Vorwort Inhaltsverzeichnis Abbildungsverzeichnis (optional) Tabellenverzeichnis (optional) Abkürzungsverzeichnis (optional)
1	**Zu dieser Betriebsanleitung** Aufbau der Gesamtdokumentation Aufbau der Betriebsanleitung Erläuterung der Kapitelstruktur Erläuterung der verwendeten Zeichen und Symbole
2	**Sicherheit** Warnung vor allgemeinen Gefahren, die von der Maschine/Anlage ausgehen Hinweise zum sicherheitsgerechten Umgang mit der Maschine/Anlage
3	**Technische Daten** Zusammenfassung der Kenndaten der Maschine/Anlage
4	**Beschreibung** Beschreibung der Maschine/Anlage und ihrer Komponenten
5	**Montage** Informationen zur Montage der Maschine/Anlage
6	**Bedienung** Informationen zum Standardbetrieb der Maschine/Anlage für das normale Bedienungspersonal
7	**Instandhaltung** Beschreibung der Instandhaltungsarbeiten, die das Fachpersonal ausführt
8	**Zeichnungen und Schemata** Liste aller mitgelieferten Zeichnungen und Schemata
9	**Anhang** Liste aller mitgelieferten Zulieferdokumente Liste aller mitgelieferten Voith Paper Werknormen Liste aller zitierten Normen
10	**Standardlisten (optional)** Liste aller mitgelieferten Standardlisten

Tab. 1-1 Aufbau Betriebsanleitung

2 Sicherheit

Alle in dieser Betriebsanleitung genannten Höchstwerte für Leistung, Belastung, Druck etc. sind Grenzwerte.

Diese Werte sind Basis für die konstruktive Auslegung und Festigkeitsberechnung der Bauteile.

⚠ GEFAHR

Verletzungsgefahr durch berstende Maschinenteile!

Werden zulässige Grenzwerte überschritten, können Teile der Maschine zerstört werden. In Folge davon können Personen verletzt oder getötet werden!

⇨ Maschine immer innerhalb der zulässigen Grenzwerte betreiben!

Die Überschreitung/Nichtbeachtung der genannten Grenzwerte beim Betrieb der Anlage/Maschine stellt einen bestimmungswidrigen Gebrauch der Anlage/Maschine dar und schließt die Haftung seitens Voith Paper für daraus entstehende Schäden aus.

3 Technische Daten

Technical specifications of the system.

All entries of the following table are filled via external data stored in Excel or exported XML from SAP.

The number of possible entries is not limited.

Parameter	Value
Value 1	–
Value 2	–
Value 3	–

Tab. 3-1 Technical specifications

Parameter	Value
Value 5	55
Value 6	66
Value 7	77

Tab. 3-2 Technical specifications

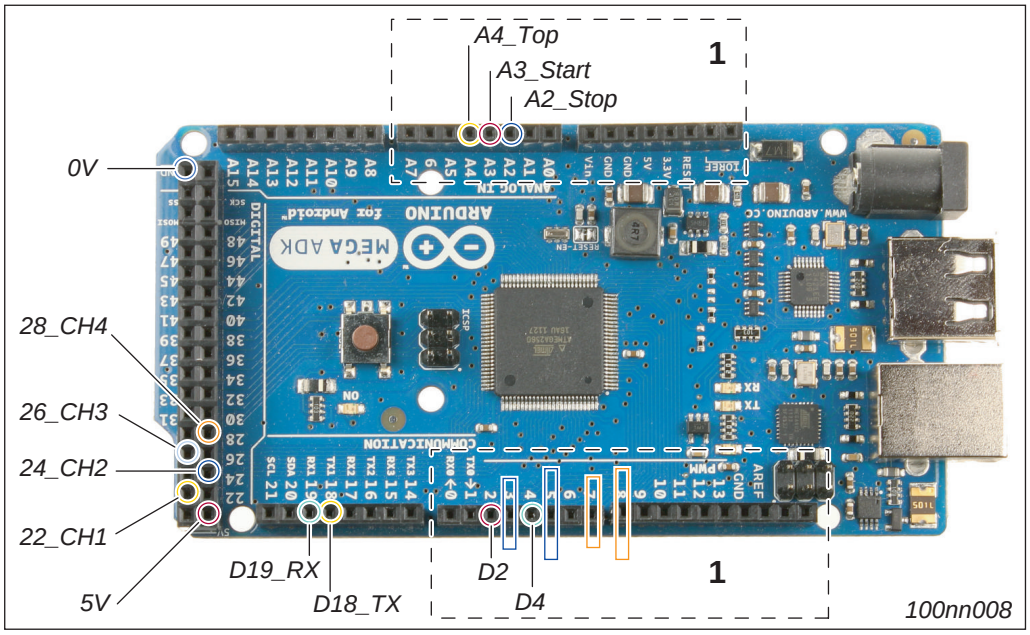


Fig. 3-1

4 Beschreibung

Aufgabe

Die Aufgabe des Systems ist es, einzelne Holzscheite von einem Holzstapel aufzuheben und diese in einen bereitstehenden Korb zu legen.

Komponenten

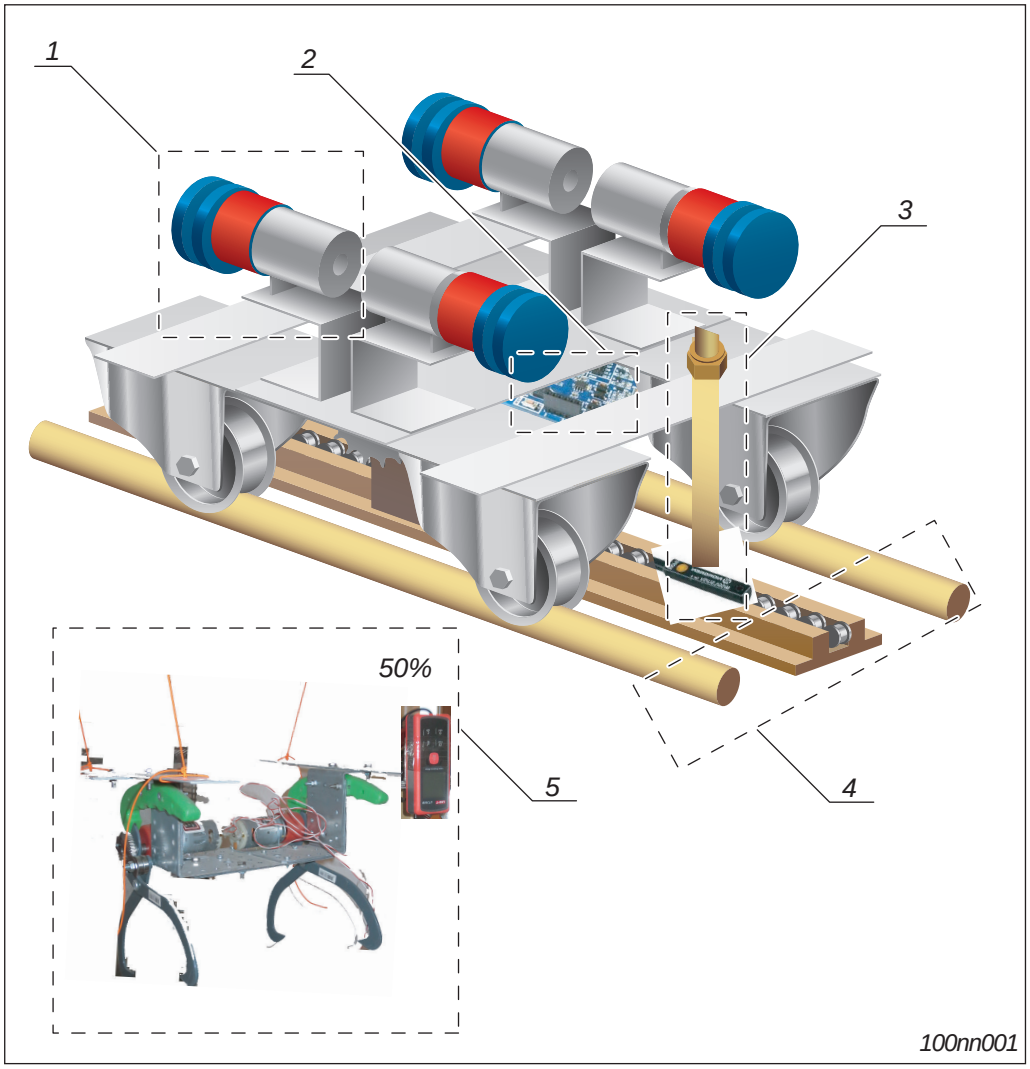


Fig. 4-1 System Overview

1 Gear with Cable Winch
2 MicroController with two MotorShields
3 Carriage with Main Gear and Limit Scitch
4 Rail System with Guiding Chain and Position Magnets (Pos. 1 / Pos. 2)
5 Gripper with Inclinometer and Pressure-Sensor

4.1 Carriage with Main Gear and LimitSwitch

Aufgabe

The Carriage bears the components of the system. It moves forward horizontally on rails.

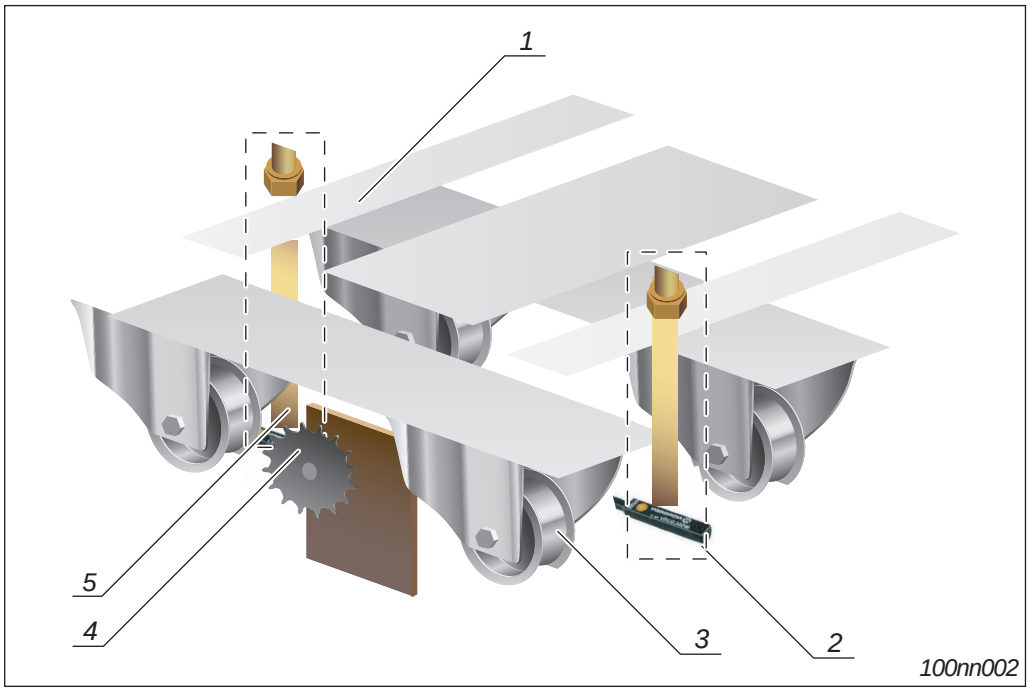


Fig. 4-2

1 Mounting Plate (cross direction)
2 LimitSwitch "EndPosition" with holder
3 Wheel (non-powered)
4 Main Gear with Gear-wheel
5 LimitSwitch "Garage" with holder

Funktion

The Carriage brings the Gripper into optimal position to grab the at time highest wood log.

The horizontal position of the carriage is calculated by analyzing the time travelled over the entire length.

StartPosition of the carriage (Pos.1, "Garage") and EndPosition (Pos. 2, "EndPosition") are defined via two LimitSwitches.

4.1.1 Main Gear

Energy Supply LiPo 3200mAh 3S 11.1V 20C

The used model is available at:

- http://www.conrad.de/ce/de/product/198411

100nn010

4.2 Microcontroller Arduino Mega2560 – Carriage

Aufgabe

The Microcontroller Arduino Mega2560 contains the Central Prosessing Unit, it controls all other elements which are connected to it. The program is loaded directly to the Arduino Mega2560.

Übersicht

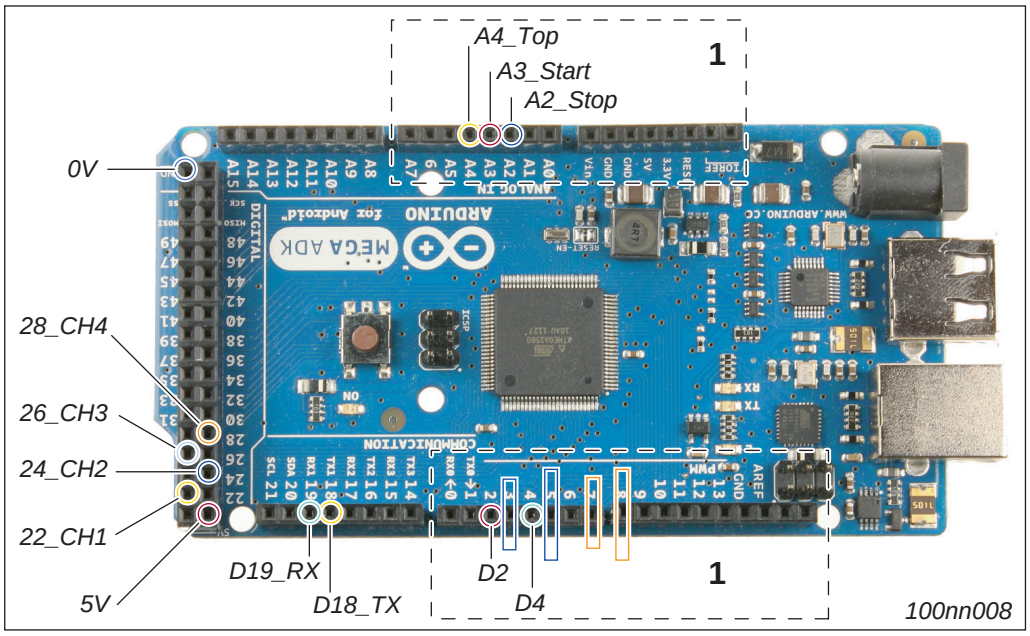


Fig. 4-3

1 THESE PINS ARE ACCESSIBLE ONLY VIA THE UPMOST 10 A DC SHIELD

Funktion

Controlled by the Arduino Mega2560 installed **on the carriage** are:

- 5 Volt power supply for sensors and secondary boards (excludes engines)
- 10A DC Motor Driver Shield (2 units)
  - first Shield controls the Main Gear
  - second shield controls the four Winders (together with the Relay Board)
- Relay Board (1 unit)
- Limit switches (2 units)
- Inclinometer (2 units)
- Pressure Sensors (4 units)
- Control diodes (2 units)

4.2.1 Assigned PINs

Unit	Name	Analog PIN	Digital PIN	Task
1	**Power Supply**	5 V	–	• provides a stable 5.0 Volt output for needed sensor voltage
		GND	–	• provides grounding (0 Volt)
2	**10A DC Motor Driver Shield 1**	–	D_3	• communicates with port "PWM" (Pulse Width Modulation)
			D_7	• communicates with port "Direction" of the Main Gear
3	**10A DC Motor Driver Shield 2**	–	D_5	• communicates with port "PWM" (Pulse Width Modulation)
			D_8	• communicates with port "Direction"
4	**Relay Unit**	–	D_22	• communicates with Relay Unit's "Channel 1"
			D_24	• communicates with Relay Unit's "Channel 2"
			D_26	• communicates with Relay Unit's "Channel 3"
			D_28	• communicates with Relay Unit's "Channel 4"
5	**Limit Switch Stop**	A_2	–	• communicates with Limit Switch "Stop" (connection via blue cable)
6	**Limit Switch Start**	A_3	–	• communicates with Limit Switch "Start" (connection via blue cable)
7	**Limit Switch Top**	A_4	–	• communicates with Limit Switch "Top" (connection via blue cable)
8	**Laser Meter UT390B**	–	D_18 TX	• communicates with port TX (connection via yellow cable)
			D_19 RX	• communicates with port RX (connection via green cable)
9	**Gears for Grippers**	5 V	–	• SWITCH; TESTING provides a stable 5.0 Volt output for needed sensor voltage
		GND	–	• SWITCH; TESTING provides grounding (0 Volt)
10	**Inclinometer 1**	A_5	–	• LATER; SKIPPING; communicates with Inclinometer XY
11	**Inclinometer 2**	A_6	–	• LATER; SKIPPING; communicates with Inclinometer XZ
12	**Sensor 1**	A_7	–	• communicates with Gripper 1 pincer 1
13	**Sensor 2**	A_8	–	• communicates with Gripper 1 pincer 2
14	**Sensor 3**	A_9	–	• communicates with Gripper 2 pincer 1
15	**Sensor 4**	A_10	–	• communicates with Gripper 2 pincer 2
16	**LED resistor 1**	A_11	–	• indicates end position Gripper 1
17	**LED resistor 2**	A_12	–	• indicates end position Gripper 2
18	**LED Red**	–	D_2	• communicates with port "PWM" (Pulse Width Modulation)
19	**LED Green**		D_4	• communicates with port "Direction" of the Main Gear

Tab. 4-1 Assigned PINs

4.2.2 10A DC Motor Driver Arduino Shield

The used model is available at:

- http://www.robotshop.com/en/10a-dc-motor-driver-arduino-shield.html

Aufgabe

The **first** 10A DC Motor Driver Arduino Shield controls:

- speed of the Main Gear
- voltage inversion

The **second** 10A DC Motor Driver Arduino Shield controls:

- speed of all four LMC gear motors (each activated individually via Relay)
- voltage inversion of all four LMC gear motors (activation individually)

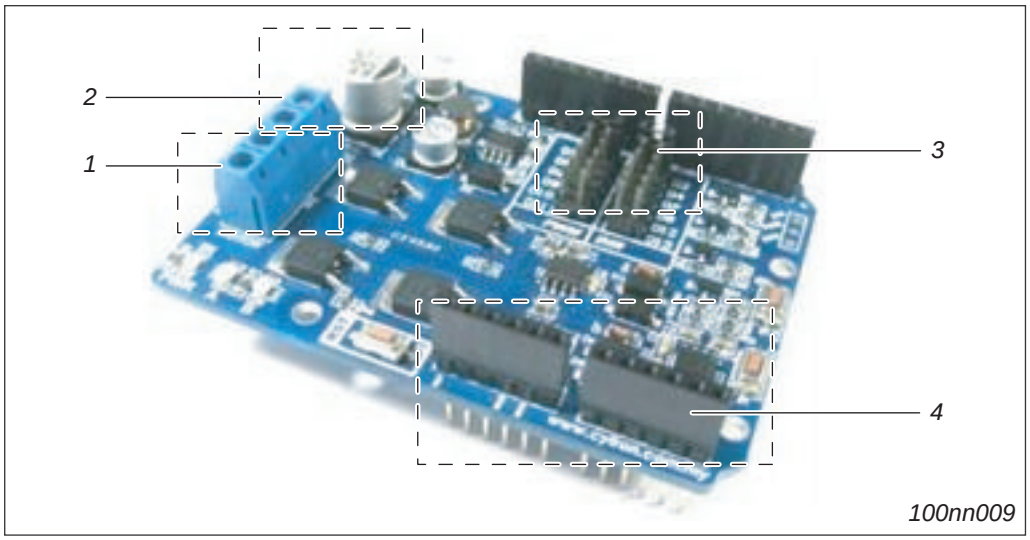


Fig. 4-4 10A DC Motor Driver Arduino Shield

1 POWER IN
2 POWER OUT

3 ACTIVATION OF ARDUINO PINS **FOR CONNECTIVITY**

4 CONNECTION POINTS FOR HIDDEN ARDUINO PINS

4.3 Microcontroller Arduino Mega2560 – Gripper

Aufgabe

The Microcontroller Arduino Mega2560 contains the Central Prosessing Unit, it controls all other elements which are connected to it. The program is loaded directly to the Arduino Mega2560.

Übersicht

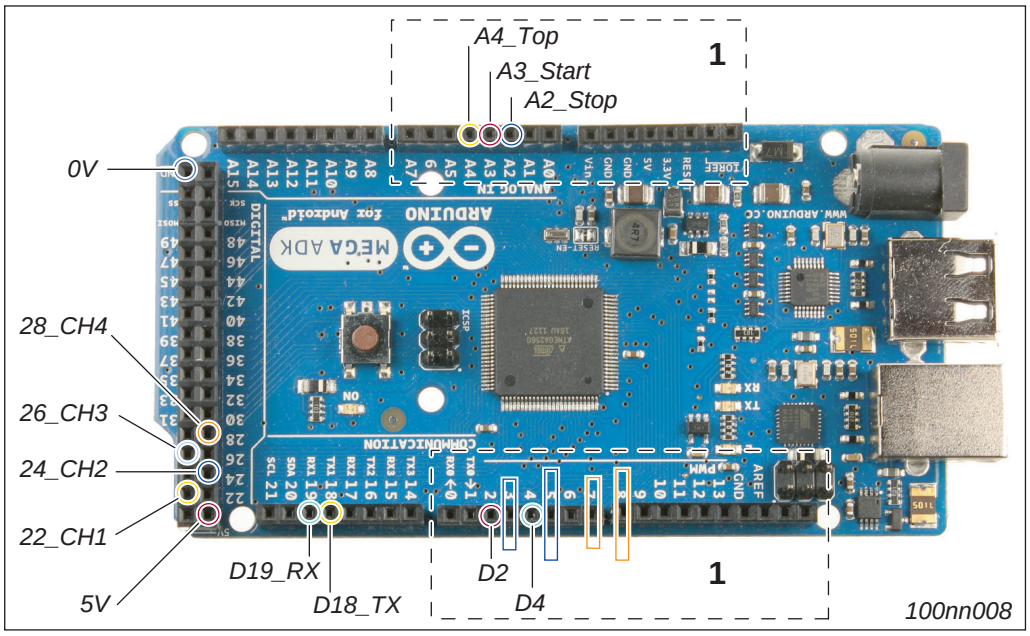


Fig. 4-5

1 THESE PINS ARE ACCESSIBLE ONLY VIA THE UPMOST 10 A DC SHIELD

Funktion

Controlled by the Arduino Mega2560 installed **on the gripper** are:

- 5 Volt power supply for sensors and secondary boards (excludes engines)
- 10A DC Motor Driver Shield (1 unit)
  - first shield controls the two gripper motor [OPTIONAL, later: Laser-Servo]
- Relay Board (1 unit)
- Laser meter UT390B
- gears for Grippers (2 units)
- inclinometer (2 units)
- Pressure Sensors (4 units)
- Control diodes (2 units)

4.3.1 Assigned PINs

Unit	Name	Analog PIN	Digital PIN	Task
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•
1			–	•

Tab. 4-2 Assigned PINs

4.3.2 10A DC Motor Driver Arduino Shield

The used model is available at:

- http://www.robotshop.com/en/10a-dc-motor-driver-arduino-shield.html

Aufgabe

The **first** 10A DC Motor Driver Arduino Shield controls:

- speed of the Main Gear
- voltage inversion

The **second** 10A DC Motor Driver Arduino Shield controls:

- speed of all four LMC gear motors (each activated individually via Relay)
- voltage inversion of all four LMC gear motors (activation individually)

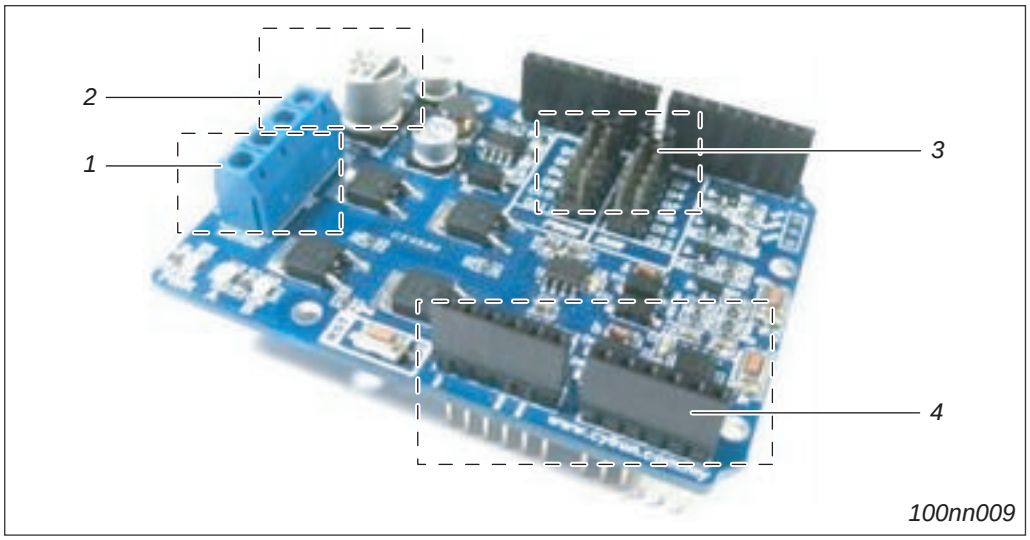


Fig. 4-6 10A DC Motor Driver Arduino Shield

1 POWER IN
2 POWER OUT

3 ACTIVATION OF ARDUINO PINS

FOR CONNECTIVITY
4 CONNECTION POINTS FOR HIDDEN ARDUINO PINS

4.4 Gear with Cable Winch

The used model [Gear] with **gear ratio 810:1** is available at:

- http://www.conrad.de/ce/de/product/222367/

Aufgabe

The four winches are designed to lower the gripper vertically / to lift the grippper vertically.

Komponenten

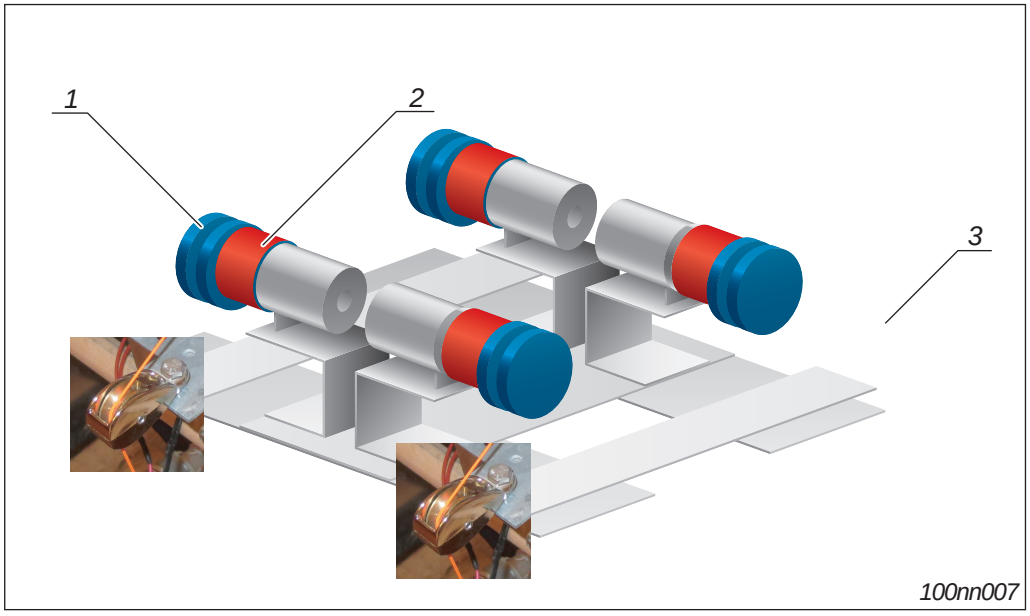


Fig. 4-7 Seilwinde

1 Winch / Winder
2 Gear motor
3 Deflector roll

Funktion

Each Gear Motor is triggered individually by one of the four Channels of the Relay.

Speed and Direction of the entire group are controlled by the second Motor Shield.

4.5 Rail System with Guiding Chain and Inductive Limit Switch

The used model [Norgren Limit Switch] is available at:

- http://www.conrad.de/ce/de/product/580622/

Aufgabe

The Rail System is designed to keep the carriage on track and to move it forward (including position control).

Komponenten

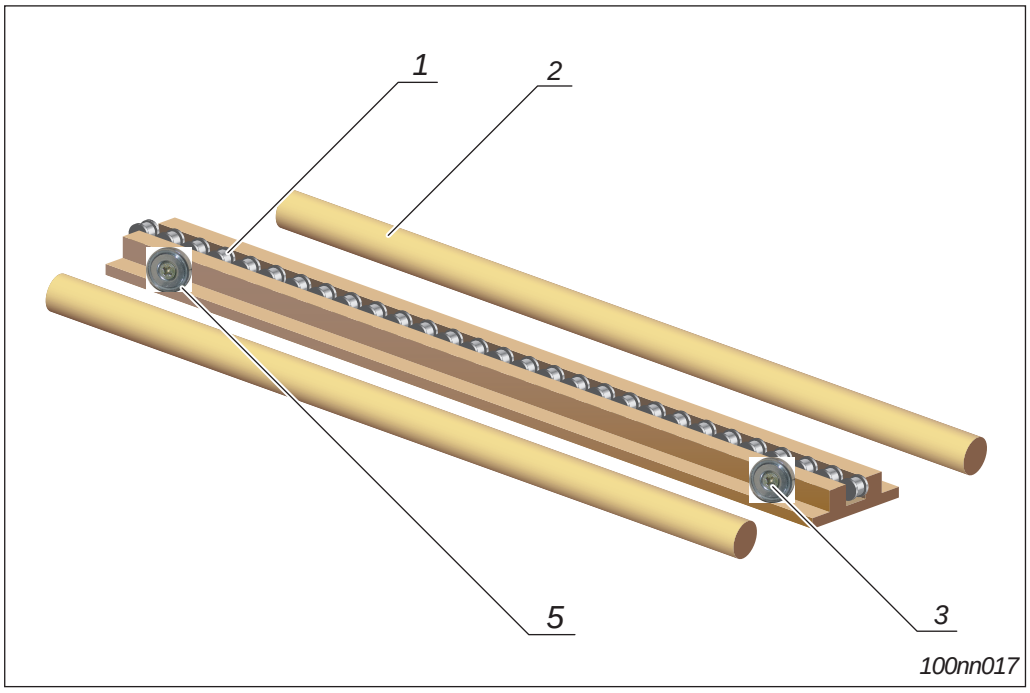


Fig. 4-8 Seilwinde

1 Chain
2 Rail
3 Guide
4 Inductive Limit Switch Stop
5 Inductive Limit Switch Start

Funktion

By transmitting traction to the Guiding Chain the carriage...

4.6 Gripper with InclinoMeter, LaserMeter, PressureSensor and Position Indicator

The used model with **gear ratio 3000:1** is available at:

- http://www.conrad.de/ce/de/product/222368/

The used model [Gripper] is available at:

- http://www.amazon.de/gp/product/B007L4FNPU/ref=oh_details_o09_s00_i00?ie=UTF8&psc=1

The used model [pressure sensor] is available at:

- http://www.conrad.de/ce/de/product/182519/

The used model [inclinometer] is available at:

- http://de.mouser.com/ProductDetail/Murata-Electronics/SCA100T-D01-1/?Murata-Electronics/SCA100T-D01-1/&qs=sGAEpiMZZMuRFIlaTiDGn3QasMrWIkqQdoHPq7%252bXssI=
  http://de.mouser.com/ProductDetail/SCA100T-D01-1/

Aufgabe

The Rail System is designed to keep the carriage on track and to move it forward (including position control).

Komponenten

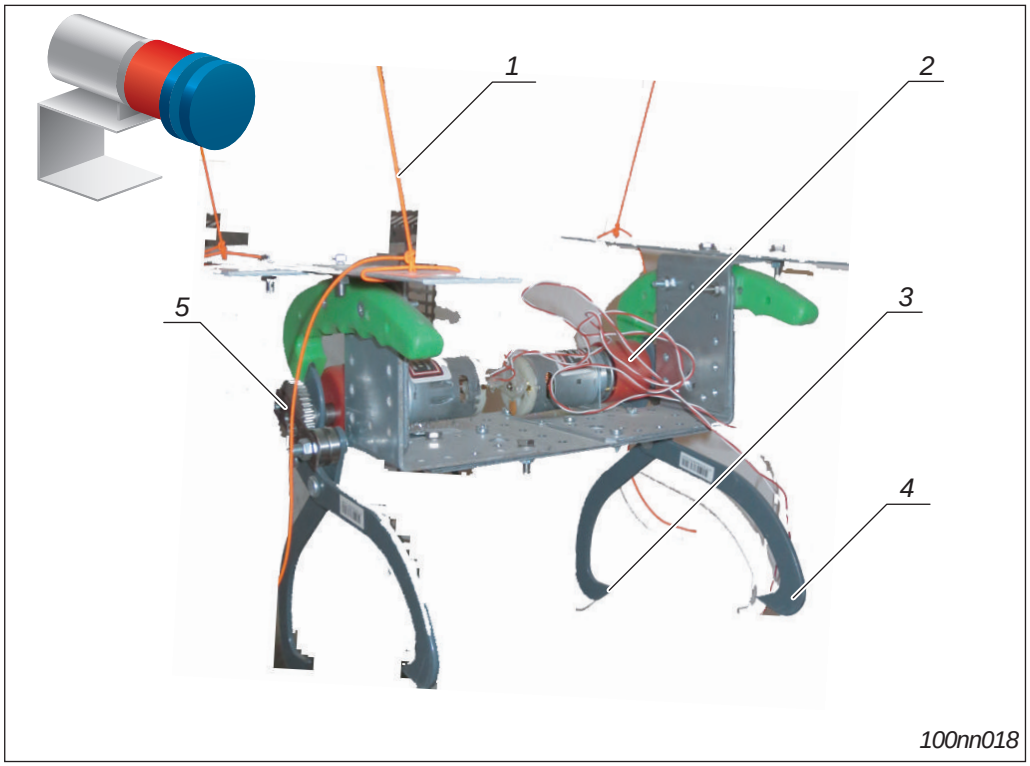


Fig. 4-9 Gripper with Inclinometer and PressureSensor

1 Hebeseil
2 Gear motor
3 Location of Pressure Sensor
4 Gripper (pinces)
5 Transmission
6 **To to: LED (photo resistor)**

Funktion

4.6.1 LaserMeter

The used model is available at:

- www.amazon.de/Laser-Entfernungsmesser-UNI-T-UT391-Genauigkeit-600002/dp/B007FZIEJW/

100nn005

4.6.2 InclinoMeter

The used model is available at:

- www.amazon.de/Laser-Entfernungsmesser-UNI-T-UT391-Genauigkeit-600002/dp/B007FZIEJW/

100nn005

4.6.3 Pressure Sensor

The used model is available at:

- www.amazon.de/Laser-Entfernungsmesser-UNI-T-UT391-Genauigkeit-600002/dp/B007FZIEJW/

100nn005

4.6.4 Position Indicator

The used model is available at:

- www.amazon.de/Laser-Entfernungsmesser-UNI-T-UT391-Genauigkeit-600002/dp/B007FZIEJW/

100nn005

5 Montage

5.1 Montageanleitung nach Anhang VI (EG-RL 2006/42/EG)

Damit die unvollständige Maschine ordnungsgemäß und ohne Beeinträchtigung der Sicherheit und Gesundheit von Personen mit den anderen Teilen zur vollständigen Maschine zusammengebaut werden kann, müssen alle anderen Teile (wie Antriebe, Medienversorgung...) gemäß der Maschinenrichtlinie (EG-RL 2006/42/EG) des Europäischen Parlaments und des Rates über Maschinen konstruiert, gebaut und sachgerecht installiert und gewartet werden.

6 Bedienung

This manual serves as standard manual.

Safety instructions for operation

Observe the general safety instructions and protective measures covered in chapter 2 as well as the applicable legal regulations.

6.1 Height Differential Analysis (HDA)

Aufgabe (Coding)

The traversing carriage on rails moves forward and backwards, using distance sensor TC.

A second distance sensor HD pointing downwards measures the changes of the height differential on the ground.

The unevenness of the ground surface is – in this case – due to the different positions of wood logs, which are piled into a heap.

- The software analyses the travelled distance from wall to wall (travel): x-axis
- The software analyses the height differential (height): y-axis

6.1.1 Distance Sensor HD

The distance sensor HD is pointing directly downwards.

This second sensor is – together with Distance Sensor TC @ Analog Pin_A1 – also connected to the Microcontroller Arduino Mega2560 @ Analog Pin_A3.

- Distance Sensor Sonar MB7060
  http://www.maxbotix.com/documents/MB7060-MB7070_Datasheet.pdf

6.1.2 Height differential recognition

For testing the equipment some wood logs are placed onto a blanket.

100nn012

The Laser Meter is pointing directly downwards to the wood logs on the blanket.

100nn005

6.1.3 Height differential visualization

Gnuplot Version 4.2.5

The visualization is done via the terminal program "gnuplot"

Used version is:

- gnuplot-4.2.5-i386.dmg

The Arduino software should read/write:

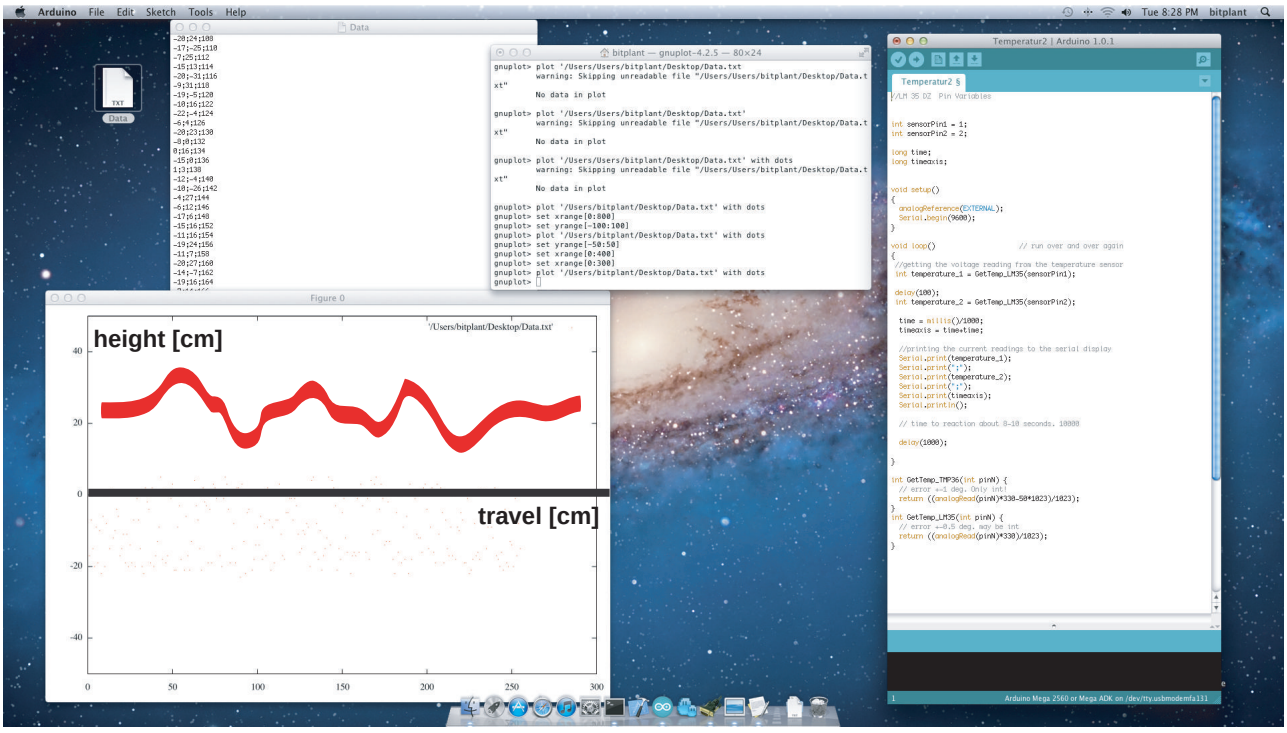
- Serial.print(travel);
- Serial.print(";");
- Serial.print(height);

CoolTerm Version 1.4.1

The data are stored in a text file, using CoolTerm Version 1.4.1.

This text file is the analyzed by "gnuplot".

Übersicht



100nn014

7 Instandhaltung

7.1 General Safety Instructions regarding Operation

HINWEIS

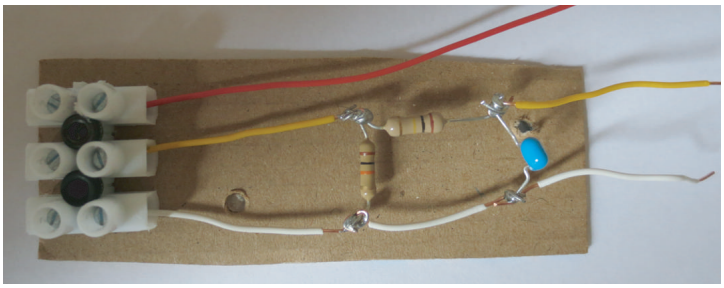
⇨

7.2 Low-pass filter

Aufgabe

The readings of the TC Sonar and the HD Sonar are influenced by noise or dropping voltage when the DC motor is working.

Komponenten

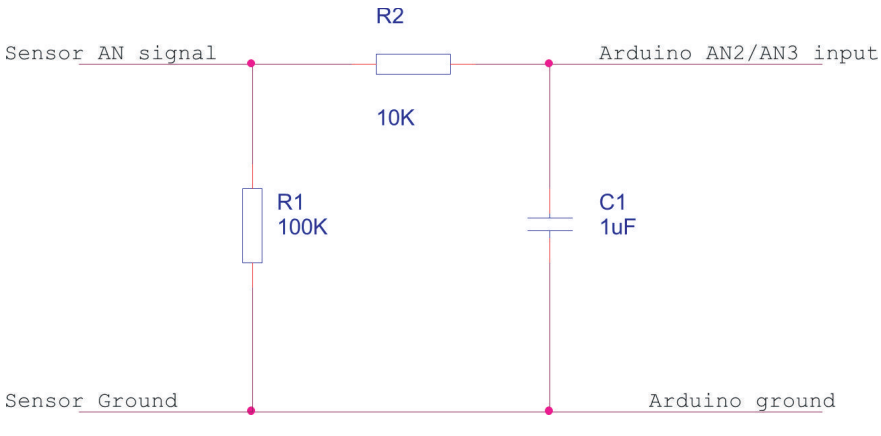


100nn016

Funktion

The low-pass filter does eliminate all frequencies above the cutoff frequency while passing those below remain unchanged.

In order to avoid a drop in power supply (when other electronic devices are turned on), a separate power supply of these devices will be installed.



100nn015

8 Zeichnungen

Designation	Drawing/schematic number	Revision	Where filed
–	–	–	–

Tab. 8-1 Drawings and schematics table

9 Anhang

9.1 Documentation by subsuppliers

The following documentation can be found in the supplied folders for the present Operation Manual.

Supplier	Subject	Where filed
–	–	–

Tab. 9-1 Subsuppliers table

10 Software (code)

10.1 Carriage

```
#include "Arduino.h"
#include "Carriage.h"
#include "ServoMotor.h"
#include "Sonar.h"
#include "Led.h"
#include "TimerOne.h"
#include "UT390B.h"
#include "TickClient.h"
#include "TickServer.h"
#include "Constants.h"

//#undef DEBUG
#define DEBUG

// Helper function for debug
void debugHelper(int actualPosition, int wantedPosition);

// Carriage Constructor
Carriage::Carriage(const enum taskIds taskId):
    carriageState_(CarriageIdle),
    actualPosition_(0, CarriagePos::posUnknown),
    wantedPosition_(0, CarriagePos::posUnknown),
    reliefAnalysis_(false),
    reliefAnalyzer_(),
    carriageMotor_(carragePWMPin, carrageDirectionPin,
                   carriageMotorMaxSpeed, carriageMotorAccelSteps,
carriageMotorTid),
    startPosSwitch_(carriageStartPosPin, "Start Position"),
    endPosSwitch_(carriageStopPosPin, "End Position"),
    redLED_(redLedPin),
    greenLED_(greenLedPin)
{
  tickServer.registerTickClient(this, 5, 1, taskId);
  carriageMotor_.periodicTick();
}

// Help function to set motor direction
void Carriage::setMotorDirection(Direction_t direction)
{
  switch(direction)
  {
  case DirectionA:
    if(DirectionA != carriageMotor_.getCurrentDirection())
    {
      carriageMotor_.motorEvent(EventReverse);
    }
    break;
  case DirectionB:
    if(DirectionB != carriageMotor_.getCurrentDirection())
    {
      carriageMotor_.motorEvent(EventReverse);
    }
```

```
    break;
  default:
    break;
  }
}

// Set carriage position to relief top
void Carriage::setReliefTopPosition()
{
  setWantedPosition(reliefAnalyzer_.getDistanceToReliefMax());

  // Testing purposes, set position to 5 seconds from start
  // setWantedPosition(500);
}

// Set wanted carriage position from wall
void Carriage::setWantedPosition(CarriagePos position)
{
  wantedPosition_=position;
}

// Get distance from carriage to relief top
int Carriage::getReliefTopHeight() const
{
  return reliefAnalyzer_.getReliefTopHeight();
}

// Get wanted carriage position
CarriagePos Carriage::getWantedPosition() const
{
  return wantedPosition_;
}

void Carriage::startReliefAnalysis()
{
  reliefAnalyzer_.resetReliefAnalysis();
  reliefAnalysis_ = true;
}

void Carriage::stopReliefAnalysis()
{
//  reliefAnalyzer_.dumpProfile();
  reliefAnalyzer_.findReliefMaximum();
  Serial.print("Determined top of relief to be at distance: ");
  Serial.println(reliefAnalyzer_.getDistanceToReliefMax());
  reliefAnalysis_ = false;
}

// Returns actual state of the carriage
CarriageState Carriage::getState() const
{
  return carriageState_;
}

// Main function to run the Carriage state machine
void Carriage::periodicTick()
{
```

```
  switch(wantedPosition_.posType_) {
  case CarriagePos::posStartSwitch:
  {
    //run Direction B (towards "start")
    setMotorDirection(DirectionB);

    if (startPosSwitch_.isClosed())
    {
      carriageMotor_.motorEvent(EventStop);
      carriageState_ = CarriageInPosition;
      actualPosition_= CarriagePos(0, CarriagePos::posStartSwitch);
    }
    else
    {
      carriageMotor_.motorEvent(EventStart);
      carriageState_ = CarriageInTransition;
      actualPosition_.posType_=CarriagePos::posAbs;
      actualPosition_.position_--;
    }
    break;
  }
  case CarriagePos::posEndSwitch:
  {
    //run Direction A (towards "end")
    setMotorDirection(DirectionA);

    if (endPosSwitch_.isClosed())
    {
      carriageMotor_.motorEvent(EventStop);
      carriageState_ = CarriageInPosition;
      actualPosition_.posType_ = CarriagePos::posEndSwitch;
    }
    else
    {
      carriageMotor_.motorEvent(EventStart);
      carriageState_ = CarriageInTransition;
      actualPosition_.posType_=CarriagePos::posAbs;
      actualPosition_.position_++;
    }

    break;
  }
  case CarriagePos::posAbs:
  {
    if(CarriagePos::posStartSwitch == actualPosition_.posType_)
    {
      //run Direction A (towards "end")
      setMotorDirection(DirectionA);
      carriageMotor_.motorEvent(EventStart);
      carriageState_ = CarriageInTransition;
      actualPosition_.posType_ = CarriagePos::posAbs;
      actualPosition_.position_ = 0;
    }
    else if (CarriagePos::posEndSwitch == actualPosition_.posType_)
    {
      Serial.println("Internal error, Can't move from end switch
to abs pos");
```

```
      while (true) {};
    }
    else if (CarriagePos::posAbs == actualPosition_.posType_)
    {
      if (wantedPosition_.position_ == actualPosition_.position_)
      {
        carriageMotor_.motorEvent(EventStop);
        carriageState_ = CarriageInPosition;
      }
      else if ((wantedPosition_.position_ > actualPosition_.position_)
&&
               (CarriageInPosition == carriageState_))
      {
        //Restart motor in run Direction A (towards "end")
        setMotorDirection(DirectionA);
        carriageMotor_.motorEvent(EventStart);
        carriageState_ = CarriageInTransition;
      }

      if (CarriageInTransition == carriageState_)
      {
        actualPosition_.position_++;
      }
    }
    break;
  }
  case CarriagePos::posUnknown:
  {
    break;
  }
  default:
    break;
  }

  // Add relief analysis data
  if (reliefAnalysis_)
  {
    reliefAnalyzer_.addHeightAtIndex(actualPosition_.position_);
  }

  // Set the LEDs
  ledHelper();

  // Call debug helper
  //debugHelper(actualPosition_, wantedPosition_);
}

// Helper function to set LEDs
void Carriage::ledHelper()
{
  switch(carriageState_)
  {
  case CarriageInPosition:
#if 0
    // What to do with our LEDs?
    if(maxPosition_ == wantedPosition_)
    {
```

```
      greenLED_.setState(LedOn);
      redLED_.setState(LedOff);
    }
    else if (minPosition_ == wantedPosition_)
    {
      greenLED_.setState(LedOff);
      redLED_.setState(LedOn);
    }
#endif
    break;
  case CarriageInTransition:
  case CarriageIdle:
    greenLED_.setState(LedOff);
    redLED_.setState(LedOff);
    break;
  default:
    break;
  }
}

// Helper function to print out status
void Carriage::debugHelper(int actualPosition, int wantedPosition)
{
#ifdef DEBUG
  static int debugCounter=0;
  if (debugInterval <= debugCounter)
  {
    if (wantedPosition != 0)
    {
      Serial.print("Actual pos ");
      Serial.print(actualPosition);
      Serial.print(" Wanted pos ");
      Serial.print(wantedPosition);
      Serial.println("");
    }
    debugCounter=0;
  }
  debugCounter++;
#endif
}

CarriagePos::CarriagePos(int position, enum CarriagePosType posType):
  position_(position),
  posType_(posType)
{ }

CarriagePos::CarriagePos(enum CarriagePosType posType):
  position_(0),
  posType_(posType)
{ }
```