UNIVERSITY OF AGDER

BACHELOR THESIS

All in one Servo Lab

Arduino™-based laboratory platform for microcontroller operated servo systems

Version 1.00 Final 2014-05-19

by

Audun Hørthe
Helge Nødland
Bjørnar Preus-Olsen

Supervisors:

Morten Ottestad
Torstein K. Wroldsen

This Bachelor's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2014
Faculty of Engineering and Science
Department of Engineering Sciences

Abstract

Servo technology and servo-control is a key element in Mechatronics, and working with servo systems involves using methods from mechanics, mathematics, electronics and control systems engineering.

In addition, the subject introduces micro-controller programming. Previous to the fifth term, there is only a minuscule element of programming. This means that the learning-curve in Servo technology is quite steep.

To amend this situation, the University of Agder wanted a platform for doing practical laboratory assignments, simulations and demonstrations. Ease of use, "plug-and-work", was a key issue, but most of all: *working with the platform should be fun.*

We started out with the equipment the university already had provided for laboratory assignments. From there, we designed a physical platform, calculated the equipment characteristics and created accessible and relevant assignments.

The end result is delivers what the project title promises; the All in one Servo Lab. It is a single unit platform with servo- and stepper motor, various inputs and feedback options. In addition to this, feedback front-end-software is provided, and laboratory assignments with solutions and manuals are in development.

We believe that the All In One Servo Lab will be an important asset for the university and future students. Furthermore, we also hope that this project leads to similar projects. In our experience as students, we know that a practical and hands-on-experience would be greatly appreciated in most subjects.

Preface

In 2011, a new educational structure for the undergraduate engineering education was implemented at the University of Agder. one of the results of the new structure was the subject MAS220 Servo-technology, which also from now on would incorporate microprocessor technology.

In this context, new lab equipment was obtained to ensure a practical practical approach to microprocessor programming and servo control. The equipment was based on Arduino MEGA 2560 micro-controller and off-the-shelf motors and add-on peripherals. During the fall term, it proved difficult to accomplish lab assignments if the students were to set up the equipment themselves. The small amount of software programming that had been introduced in the curriculum at this point of the education contributed to the challenges for the course.

In an attempt to amend the situation for future students, the University of Agder decided to initiate the bachelor project "All in one Servo Lab". The main goal of this project was to create, build and test a laboratory platform specifically for servo technology assignments. A total of twenty-five units were commissioned for delivery at the end of the spring term. Furthermore, the project called for assignments, with suggested solutions, to be created as a part of the project.

Grimstad, may 2013

Audun Hørthe

Helge Nødland

Bjørnar Preus-Olsen

Contents

List of Figures

List of Tables

List of Documents

CHAPTER 1

Introduction

"All the electronic devices are powered by white smoke. When smoke goes out, device is dead."
–Milan Nikolić

The All in one Servo Lab-project is mainly a product design and development project, initiated by the University of Agder. It was conducted during the spring term of 2014, with a planned implementation at the beginning of the fall term same year. This chapter will lay out the background for the project, as well a short introduction to the world of mechatronics, servo-technology and microcontrollers.

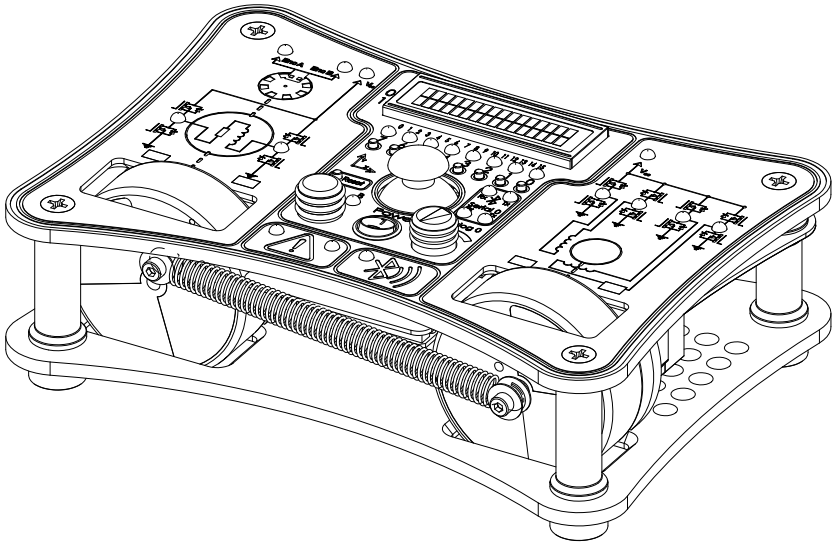


Figure 1.1: The All in one Servolab

1.1 Mechatronics

Mechatronics is a fairly new term that was introduced as a registered trademark of Yaskawa Electric Corporation [1]. According to Yaskawa, mechatronics is "the concept of working smarter – not harder – and to inexpensively get the most done in as little time as possible. The term can be defined in many different ways, but functionally, it is a blend of mechanics and the

[1] The trademark registration number for the term Mechatronics is "46-32714" and was registered in 1973.[1]

synergistic use of precision engineering, control theory, computer science, and finally sensor and actuator technology – all designed to improve products and processes.”.

The field of mechatronics is a multi discipline engineering term and it is a fusion of mechanical engineering, electrical engineering, control systems engineering and computer science. When these fields of engineering come together to form a unified field, the mechatronics engineer is equipped with a toolbox that enable for design of complex computer controlled electromechanical systems where only the sky is the limit.

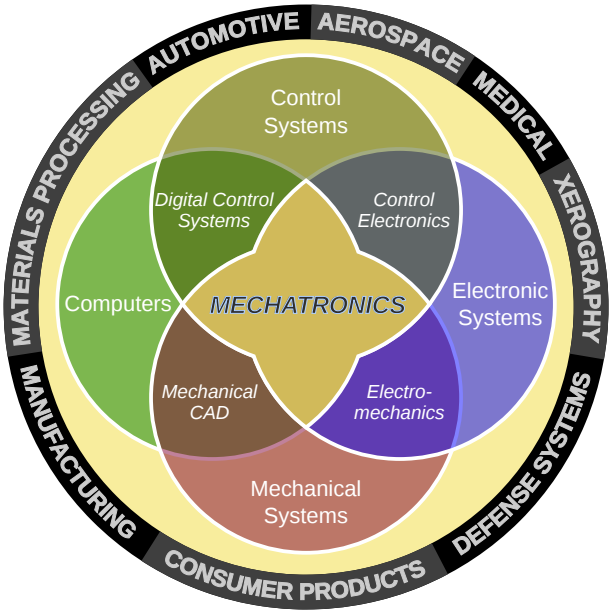


Figure 1.2: Diagram describing the interconnected studies in mechatronics[2]

At the University of Agder, much attention is paid to dynamic systems, system modelling and simulation. This is critical to the field of robotics, heave compensating systems and process automation systems.

One of the reasons for this, is the strong presence of specialized suppliers of off-shore technology in the Agder-region.

1.2 Servo-technology

Servo is a term that comes from the Latin word for ***slave***, servus, which explains its role in the servomechanism. The servomechanism is defined as a feedback controlled system. Modelling and calculating these systems is based on ***Control systems engineering***.

It uses feedback from sensors, typically positioning sensors, to calculate the position, speed and acceleration. The servomechanism uses this data to control the speed and torque of the servomotors. In electrical servo systems, this is done by changing the motor voltage and current respectively. Thereby, much of the theory from ***electric circuits*** and ***sensor technology*** are put to good use.

Most electrical motors, servomotors included, has high speed and low torque and therefore needs to be geared down. Therefore, to be able to design a servo-system, proper calculations based on ***mechanics*** and ***machine design*** must be applied.

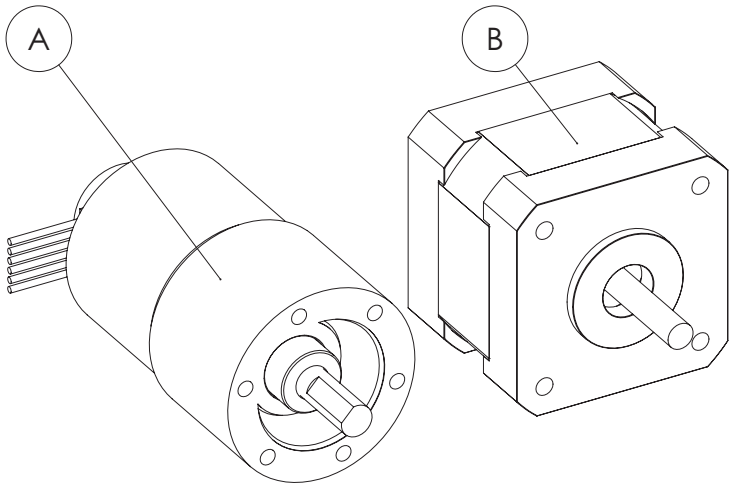


Figure 1.3: Motors used in the project. A: Geared DC-motor with encoder, B: Stepper motor

1.3 Microprocessor technology

Microprocessors are single silicon chip devices[3] that, simply put, arithmetically and logically calculated input and output. In the world of servo-systems, the input is sensor data while servo control signals is the output. The control system model resides on the chip. Microcontrollers, on the other hand, is a device that has at least one microprocessor and various

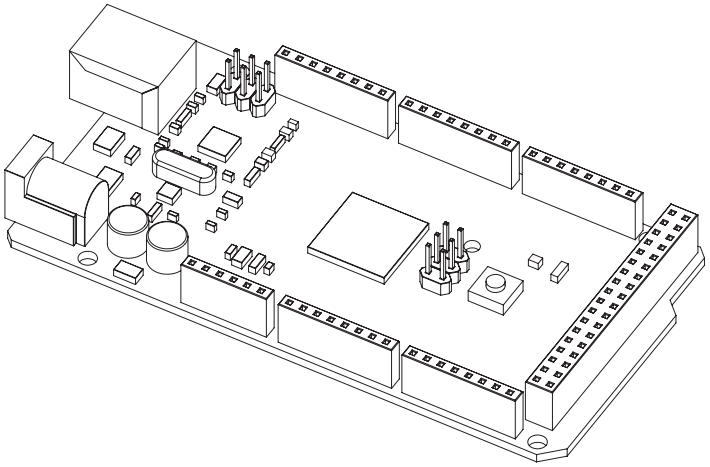


Figure 1.4: The Arduino Mega 2560 rev. 3 microcontroller board

peripheral components. Typically, the microcontroller is fitted with communication ports, different input/output-ports (IO-ports). The All in one Servo Lab uses the Arduino Mega 2560R3 microcontroller-board, which is based on the Atmel® AVR® ATmega2560 microprocessor.

1.4 Project background

In 2011, a new educational structure for the engineering studies was implemented at the University of Agder. This resulted in a fusion of several courses, amongst them Servo technology and microprocessor technology. Although these disciplines are closely related within the study of mechatronics, they don't in fact share much in the theoretical aspect. Practical laboratory work that involves the two parts of the course would be likely to increase the learning output and possibly bridge the gap.

To facilitate the laboratory work, the university acquired the following equipment:

- Arduino MEGA 2560 micro-controller
- Stepper motors (2 types)
- DC-motor with quadrature encoder
- DC power supply

The students was required to hook up the equipment correctly from scratch. It soon became clear that this was cumbersome and impractical, and it caused frustration amongst the students. It was evident that something needed to amend the situation for future students.

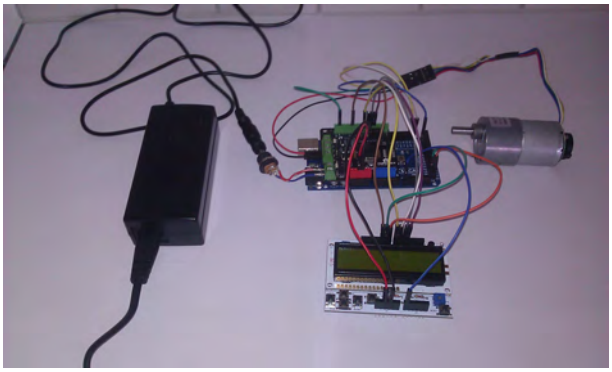


Figure 1.5: Laboratory assignment 5 setup

1.5 Literary review

The All in one Servo Lab is based, and has made the use of, mostly resources that are freely available on the internet. An example of this is the Arduino platform itself; the information on the official web-pages is more up-to-date, more relevant and has better detail than anything published on paper. As a result, there are few literary references in this report. Furthermore, documentation regarding components and equipment are as a rule electronic documents.

Even though there are books available on programming, embedded platforms and servo technology, it would contradict the very essence of the project. Programming in any form cannot be mastered by reading books alone. It is a question of trial and error, seeking help, and trying some more.

That said, literature on control systems engineering was put to good use, as well as lecture notes from earlier courses in the mechatronics engineering studies at the University of Agder.

1.6 Report outline

This report is built up around the project, not the All in one Servo Lab-*product*. It covers the design, initial testing, verifications that were conducted. Also in the report, the finished product is evaluated against initial design specifications and other possible outcomes.

Given the nature of the project, there has been taken liberties in the usual engineering thesis format. It was decided that product development would be the main focus, and has been broken down according to the different aspects of the project. The nature of the project implies that there is few areas that is subject to calculations and testing. Therefore, the amount of results data and analysis thereof is relatively small.

1.7 Problem statement

The project description [2] defines the main design criteria for project. These requirements were subject to change as the project moved forward, and exceeds the initial specifications. The reason for this is due to the chosen approach of the project, but also an adjustment in the desired functional design of the lab platform. The problem statements for the project are:

Problem Statement 1 : ***The All in one Servo Lab must have the capability to control a stepper motor with full-, half- and micro-stepping.***

Problem Statement 2 : ***The All in one Servo Lab must be able to control position, speed and torque of a DC-motor through pulse-width-modulation and quadrature encoder.***

Problem Statement 3 : ***It is desirable with a unit that has Bluetooth®- and wired communication capabilities.***

Problem Statement 4 : ***The physical user-interface must be designed with clarity and simplicity in mind.***

Problem Statement 5 : ***Quality, robustness and aesthetics must be a priority in the look and feel in the product design.***

Problem Statement 6 : ***The product must be delivered with relevant and achievable laboratory assignments, as well as examples, solutions and user manuals.***

Problem Statement 7 : ***There should be software accompanied with the product, that gives extended feedback and demonstration capabilities.***

During the design phase, the university stated that a fully assembled class set of Servo Labs was desired by the end of the term. Originally, the project was limited to designing, testing and verification of a prototype.

Problem Statement 8 : ***A class set of 25 fully assembled units must be delivered upon the conclusion of the project.***

In order to provide a solution to these problem statements, there are different methods that can be applied. Naturally, selecting a feasible method is one of the first issues that need to be addressed. What options, and how they would affect the outcome and the what was finally decided is the subject of the next chapter.

[2] attached document B.1

CHAPTER 2

Design options and considerations

"All of technology, really, is about maximizing free options."

– Nassim Nicholas Taleb

One of the more critical decisions in the project, is choosing the correct approach to meet the project requirements. Of course, there is no *correct* approach, but there are *wrong* ones. Deciding on an approach that yields the desired results, while also being feasible, was of great importance. Should the design rely on off-the-shelf products or custom parts? In this chapter, the possible advantages, and disadvantages, of these approaches are considered.

2.1 Motor drivers

In the All in one Servo Lab, the motor-drivers are a key feature. Without the ability control the parameters of the DC-motor and stepper-motor, it would not be a Servo Lab.

For the Arduino-platform, there is a variety of add-on modules, ***shields***, also motor-drivers. To the 2013 fall-term, the University of Agder acquired to kinds of motor-shields:

- Arduino Motorshield R3 MT0013346
- DFRobot DRI0009

However, these shields does not comply with the requirements of the project (table 2.1). It could be considered to either disregard the lacking features or replace them with solutions that do comply with the requirements. However, the most viable solution seemed to be a design built for the purpose. The downside is of course the time, cost and effort to design and build, as well as replacement parts would not be available. The upside was the freedom to use optimal components in optimal circuitry.

Table 2.1: Motor-driver: requirements and features

Requirement	All in One Servo Lab	Arduino MT0013346	DFRobot DR0009
Microstepping	Yes	No	No
Current sensing	Yes	Yes	No

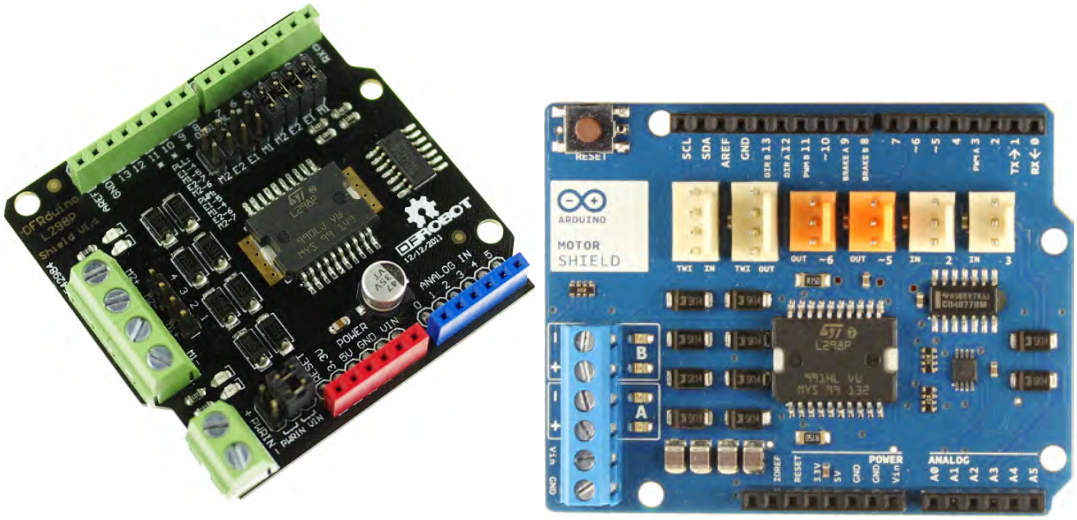


Figure 2.1: DFRobot[4] and Arduino[5] motorshields

Choosing the driving components of the motor-driver feature was made out from the desired specifications and availability of free samples. A total of 10 components, of which four were drivers, was acquired from Texas Instruments.

2.2 Interfacing features

The university also acquired Linksprite 16X2 LCD Keypad-shield model B for the Arduino-platform. Although it serves the purpose of controlling input and output with the Arduino microcontroller, it has some distinct disadvantages and limitations:

- The shield uses pins that also is used by the before-mentioned motor-shields
- It has only five customisable buttons
- It does not have any LEDs usable for programming tasks
- It does not have a feature to control the LCD backlight through pulse-width modulation(PWM).

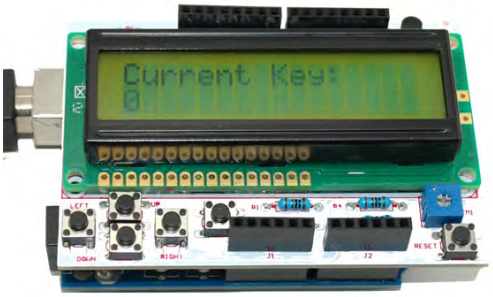


Figure 2.2: Linksprite LCD keypad shield[6]

However, by deciding to design the motor-drivers from scratch, other opportunities became available. This meant that the unit could have the desired features, organised with regard to how it is intended to be operated. In addition to buttons, other inputs could be integrated: potentiometers, encoders. It also enabled the implementation of Bluetooth®-communication module, easier connection to external devices and other units.

2.3 Building from scratch

It is likely that what was expected from the university, who issued the project, was an assembly of the motor shields, power supply, shields and motors. In that regard, the project would yield a different result. It would also mean a lot more work had to be done. And with the additional requirement of delivering a full class set by the end of the project, the workload was substantial. Everything needed to be designed, prototyped, tested, manufactured and assembled within five months. At the same time, it loosened the constraints with regard to the functionality and design of the unit.

Opposed to using off-the-shelf devices, designing the Servo Lab from scratch has the following advantages:

- Purpose-built, features are according to specifications
- Simple operation
- Single unit, plug-and-work.
- Designed with regard easy storage and durability
- No hardware adjustments required
- Clear and functional interface

2.4 Comparison and conclusion

The conclusion, although it is somewhat coloured by how the project has progressed, is that a purpose-built unit has considerable benefits. This was of course the chosen approach, and the following chapters concerns the design and testing of the different features of the All in one Servo Lab.

CHAPTER 3

Motor-driver design

"I never did a day's work in my life. It was all fun."
–Thomas A. Edison

As stated earlier, the motor-driver design is the most critical part of the All in one Servo Lab. It has to be able to perform according to the requirements. Furthermore, it is important that measures are taken to avoid failure of the electronic devices or the motors themselves. This chapter deals with the design of the two motor-driver designs, and how they work.

3.1 Motor-drivers

In order to control motors, there is a need for a circuit for converting the low power logic signals from the microcontroller to the voltage and current which the motor demand. Depending on the requirements of the system, the motor-driver can be as simple as a transistor which allows the motor to spin in one direction, or it can be a more complex circuit which allow bidirectional control, torque control and various feedback signals. A typical servomotor is often controlled by a circuit called a H-bridge.

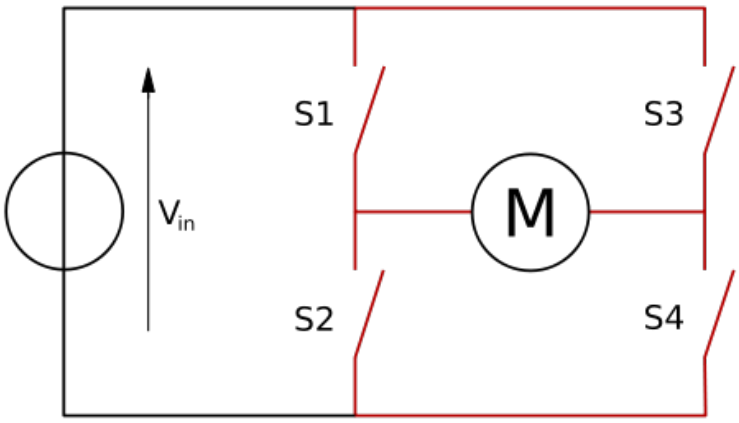


Figure 3.1: H-bridge. Theory of operation[7]

As seen on the picture by closing switch S1 and S4, current will flow trough the motor. By closing switch S2 and S3, current will flow trough the motor in the opposite direction and

the and makes the motor spin the other way. By applying ***pulse width modulation***(PWM) to the switches and thus switching the motor fast on and off, the average voltage over the motor terminals can be lowered and the power output of the motor reduced. By activating both the high side and the low side switches, the motor terminals will be shorted, and the motor would brake.

3.2 Determination of the motor drivers of the Servolab

The Servo Lab got two actuators. One brushed ***direct-current***(DC) motor and one 2-phase stepper-motor. These motors along with power supplies had been purchased in advance by the university, so the motor drives had to be chosen to fit these components. The following list shows the initial specification from the drivers:

- ***Supply voltage up to 20 volt.***
- ***Withstand a 5 Ampere stall current from the DC-motor.***
- ***Support bidirectional motor control and PWM.***
- ***Able to supply 1.2 Ampere continuous, trough both windings of the stepper-motor.***
- ***Provide an un-simplified interface with the step-motor.*** Most newer stepper-motor-drivers provide a simplified interface where only an input of step and direction is necessary. This is not wanted since the goal of the Servo Lab is to demonstrate how a step-motor actually works.
- ***Able to perform some kind of microstepping of the step-motor.***
- ***Singel IC-package***[1] ***with as few external components as possible.***
- ***include some sort of internal protection.***

A series of drivers which meets these specifications, is The DRV88XX family of motor-drivers from Texas Instrument. They also supply free samples of all their drivers, so several drivers was ordered and evaluated. For the DC-motor the *DRV8840* was selected as the driver. The driver selected for the stepper-motor was the *DRV8813*.

3.3 Thermal calculations of the motor drivers

The current a motor driver is capable of delivering is mostly determined by the heat generated inside the driver, and its ability to dissipate that heat. To ensure that the drivers won't overheat during operation, the power generated inside the driver as well as its junction-to-ambient thermal resistance has to be calculated. The main Sources of power dissipation inside the motor driver are[8]:

- **$R_{DS(on)}$DISSIPATION**
  This is caused by the ON-resistance in the switches of the H-bridge. In the *DRV88XX* drivers, both the high-side and low-side switches consists of N-channel ***MOSFETs*** (Metal Oxide Semiconductor Field Effect Transistor). The power dissipated from the ***FETs*** (Field Effect Transistor) in a single H-bridge can be calculated with the following formula:

$$P_{RDS} = (HS - R_{DS(ON)} \times (I_{OUT})^2 + (LS - R_{DS(ON)} \times (I_{OUR(RMS)})^2 \tag{3.1}$$

[1] Integrated Circuit.

P_{RDS} is the power dissipated in the output FETs, $HS - R_{DS(ON)}$ is the resistance of the high side FET, $LS - R_{DS(ON)}$ is the resistance of the low side FET and I_{OUT} is the average current trough the h-bridge.

- ***Switching losses.***
  Switching losses comes when the driving FETs are switched from one state to the other. When the the FETs switches from low to high or high to low, the transistor will move through its linear region. In its linear mode the FET will have low enough resistance for current to flow through it, but to high resistance to do it efficiently. This results in a high power dissipation in the while the FETs changes their states. The Power dissipated in each switching element can be calculated as follows:

$$P_{SW} = \frac{1}{2} \times V_M \times I_{OUT} \times f_{SW} \times (t_R + t_F) \tag{3.2}$$

  P_{SW} is the power dissipated in one output, V_M is the supply voltage, I_{OUT} is the output current, f_{SW} is the switching frequency, t_R and t_F is the rise-time and fall-time of the FET. However this formula is only true when one side of the bridge is switched. The bridge can also be driven in a "high decay" mode, where all four FETs are being switched. This mode is usually used to drive stepper motors. In this case the switching losses have to be multiplied with 2.

- ***Operating supply current dissipation.***
  The internal circuits inside the driver IC consumes some power on its own. This can is calculated as follows:

$$P_{IVM} = V_M \times I_{VM} \tag{3.3}$$

  P_{IVM} is the power dissipation from the internal circuit, V_M is the supply voltage and I_{VM} is the current consumed by the internal logic..

3.4 Calculating power dissipation of the dc-motor

For the dc-motor system we get the following data:

- $\mathbf{V_m} = \mathbf{12V}$. The rated voltage of the motor.
- $\mathbf{I_{OUT}} = \mathbf{2.2A}$. This is two times the rated continuous current of the motor. This is because the motor got a large thermal mass, and is capable of drawing higher currents for shorter periods.
- $\mathbf{HS - R_{DS(ON)}} = \mathbf{LS - R_{DS(ON)}} = \mathbf{0.13\Omega}$. Taken from the DRV8840 data-sheet. typical values at 85°C.
- $\mathbf{f_{SW}} = \mathbf{32kHz}$. This is the max frequency the Arduino is able to deliver.
- ***low decay mode.*** Only two switching FETs
- $\mathbf{t_R} = \mathbf{t_F} = \mathbf{200nS}$. This is the longest possible switching time according to the data-sheet.
- $\mathbf{I_{VM}} = \mathbf{5mA}$. Typical current-consumption according the data-sheet.

Using these values inside equation 3.1, 3.2 and 3.3, the value of P_{RDC} becomes 1.26W, P_{SW} becomes 0.17W, and I_{VM} becomes 0.06W. The complete power dissipation of the dc-motor driver-chip then becomes 1.49W.

3.5 Calculating power dissipation of the step-motor

For the step-motor we got the following data:

- $\mathbf{V_m = 20V}$. Since the driver for the step-motor is a chopper-driver, it would limit the current through the motor, and the supply voltage can be set to max(20V). This allows the motor to spin faster.
- $\mathbf{I_{OUT} = 1.2A}$. This is the rated current through each phase of the step-motor.
- $\mathbf{HS - R_{DS(ON)} = LS - R_{DS(ON)} = 0.25\Omega}$. Taken from the DRV8813 data-sheet. typical values at 85°C.
- $\mathbf{f_{SW} = 50kHz}$. When the driver operate in chopping mode, it will always be switched at its internal current control PWM frequency which is 50kHz according the data-sheet.
- ***High decay mode.*** All four FETs will switch.
- ***2 H-bridges per driver.*** The step-motor needs two H-bridges. both bridges are included in the single driver-chip. Therefore P_{RDC} and P_{SW} needs to be multiplied by 2.
- $\mathbf{t_R = t_F = 200nS}$. This is the longest possible switching time according to the data-sheet.
- $\mathbf{I_{VM} = 5mA}$. Typical current-consumption according the data-sheet.

Following equations 3.1, 3.2 and 3.3, both H-bridges gives gives the following dissipation: The P_{RDC} becomes 1.44 W, P_{SW} becomes 0.96 W, and I_{VM} becomes 0.06 W, giving a complete power dissipation of 2.46 W.

3.6 Calculating the anticipated driver temperature

In the last section the maximum expected power dissipation of the drivers was calculated to 1.49 W for the DC-motor driver and 2.46 W for the step-motor driver. The way the driver dissipate the power is through an exposed pad at the bottom of the chip which is soldered directly to the copper layers on the PCB. This way the PCB works as an heat sink, and an external heat sink is not necessary.

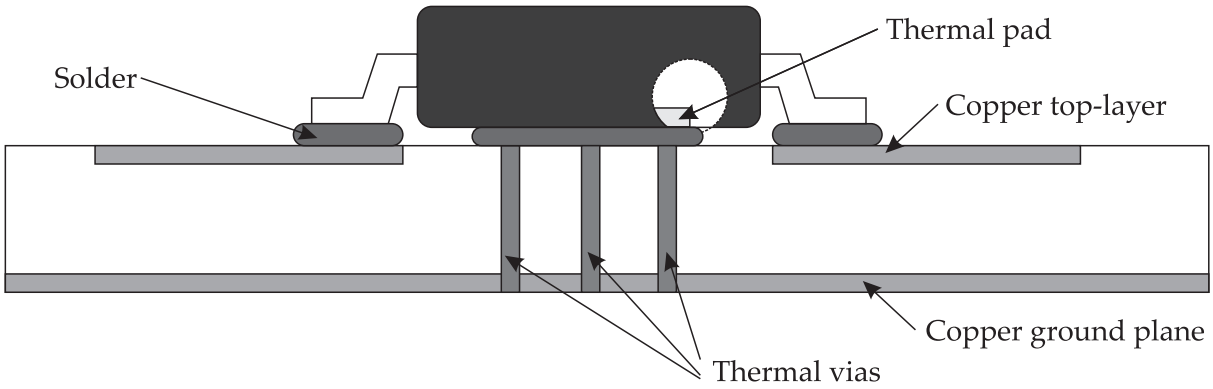


Figure 3.2: Section view of a PowerPAD Package[9]

Since the PCB is used as a heat sink, The amount of power the chip can dissipate is depending on the design of the PCB. In the DRV8840 and DRV8813 data-sheet the thermal information is given in the following table:

The most interesting value in this table is θ_{JA} Junction-to-ambient thermal resistance, which

THERMAL INFORMATION

THERMAL METRIC(1)		DRV8840 PWP 28 PINS	UNITS
θ_{JA}	Junction-to-ambient thermal resistance(2)	31.6	°C/W
θ_{JCtop}	Junction-to-case (top) thermal resistance(3)	15.9	
θ_{JB}	Junction-to-board thermal resistance(4)	5.6	
ψ_{JT}	Junction-to-top characterization parameter(5)	0.2	
ψ_{JB}	Junction-to-board characterization parameter(6)	5.5	
θ_{JCbot}	Junction-to-case (bottom) thermal resistance(7)	1.4	

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.
(2) The junction-to-ambient thermal resistance under natural convection is obtained in a simulation on a JEDEC-standard, high-K board, as specified in JESD51-7, in an environment described in JESD51-2a.

Table 3.1: DRV88XX Thermal information[10]

is 31.6 °C/W meaning that each watt of power dissipated in the driver will raise the junction temperature by 31.6°C. This is however as explained in the footnote only true when using to a JEDEC-standard, high-K board. In this standard it is used a 4 layer board, and the chip thermal pad is connected to a continuous ground plane as showed in figure 3.2. In a 2-layer board with no continuous ground plane the θ_{JA} is likely to be higher. In Texas Instrument AN 1520(Table 2)[11]. A 2-layer 1oz/1oz[2], with a 2 inch square ground plane on the back side, is given a θ_{JA} of 43.4 °C/W. The PCB in the Servo Lab is planned to have 2 layers and 2 oz copper planes and therefore give a lower thermal resistance. The open design of the Servo Lab also allow some airflow over the PCB. The specific θ_{JA} in our system is likely to be between 31.6 and 43.4 °C/W, so the middle value of 37.5°C/W is chosen as a probable value of θ_{JA}. The maximum core temperature in the drivers is given by the following equation:

$$T_{MAX} = P_{DISSIPATED} \times \theta_{JA} + T_{AMBIENT} \tag{3.4}$$

T_{MAX} is the maximum temperature of the die inside the driver chip, $P_{DISSIPATED}$ is the power dissipated by the chip and $T_{AMBIENT}$ is the ambient temperature. With a ambient temperature of 25°Cthe maximum temperature of the dc-motor driver is calculated to 80.9°C, and the maximum temperature of the step-motor driver is calculated to 117.3°C. In the driver chips datasheet, the chip goes into thermal shut-down at a die temperature of minimum 150°C, so the calculated driver temperature are within the allowable operating temperature. These temperatures are also worst case values when the motors are driven at their maximum rated current. The normal operation temperature will be a lot lower.

3.7 Stepper system

Figure 3.5 shows the schematic for the step-motor circuit. The stepper driver is the DRV8813[12], which contain the two H-bridges, driver and chopping circuit. In order to control the chopping current there is also used a MCP4922[13] 2-channel, 12 bit digital to analogue converter (DAC). The schematic can be broken into smaller parts as follows:

Driver Interface

Each winding in the step-motor is controlled by an Enable- and a Phase-Pin. A high signal on Enable, allows current through the winding. The state of Phase determines the direction of the current. These signals are also used to drive the LEDs on the top panel which indicates the state of the bridge. In order to do this the signal had to be manipulated. The following truth-table shows the desirable relationship between the input signals and the LEDs. Figure3.3 shows the circuit designed to make this happen.

[2]copper thickness on a PCB is defined as oz(weight) of copper per square foot of PCB(area)

Table 3.2: H-bridge status LEDs truth table

Input		*Output*	
Enable	***Phase***	***LED D8***	***LED D9***
1	1	0	1
1	0	1	0
0	1	0	0
0	0	0	0

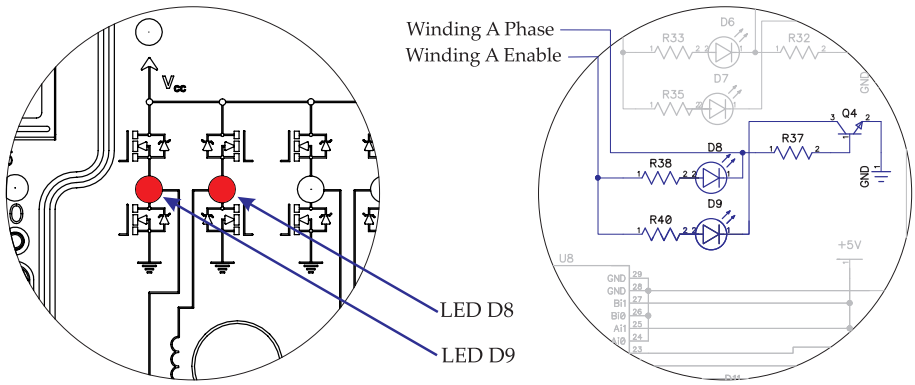


Figure 3.3: H-bridge status LEDs circuit

The circuit works by taking advantage of the diode properties of the LED. The current can only flow from anode to cathode, and will block in the other direction. By connecting the anodes of D8 and D9 to the enable pin, the LEDs can only light up when this pin is high. By connecting the cathode of D8 to the phase pin, the LED will light up when this pin is set to low. The NPN transistor(Q4) will open up when the phase pin is set high. This will open a path from the cathode of D9 to ground, making the LED light up.

Fault and Overheat switch

One of the pins on the motor-driver is the nFAULT pin. If the driver detects en error such as overheat or excessive current, the driver will shut down and the nFault pin is driven low. When the pin goes low the D11 LED located at the top panel will light up, signalling the user that something is wrong. On the PCB there is also support for an external normally open motor over-temperature switch. The switch can be attached to the motor housing, and triggers if the temperature gets to hot. When the switch is triggered, it will pull the driver nRESET pin low, making it shut off the motor. The D3 diode will pull the nFault pin low, making LED D11 light up to warn the user. The temperature switch itself, is not present on the current version of the Servo Lab, but it is easy to add if motor temperature turns out to be a problem.

DAC and current-control

The current through the windings of the motor is measured using a pair of sense-resistor connected to the drivers ISENA and ISENB pins. Once the current hits the chopping threshold, the bridge disables the current until the beginning of the next PWM cycle of the drivers internal oscillator. The chopping threshold is calculated using this equation:

$$I_{CHOP} = \frac{V_{REFX}}{3.1 \times R_{ISENSE}} \tag{3.5}$$

where R_{ISENSE} is the value of the sense resistor. In this circuit it is 0.15Ω. V_{REFX} is the voltage applied to the AVref and BVref pin. This voltage is controlled by the DAC, with its 12 bits

resolution allows for 4096 different current levels. To prevent the user from setting the chopping threshold to high and therefore risk to destroy the motor, the maximum voltage on the drivers Vref-pins is adjustable in hardware. The voltage on the wiper of potentiometer(R58) determines the maximum voltage of the Vref-pins. On the Servo Lab prototype this voltage is adjusted to 0.465V, which according to equation 3.5 allows a maximum current of 1A through each motor winding. Testing at this current setting shows that the maximum temperature will settle at about 65°C.

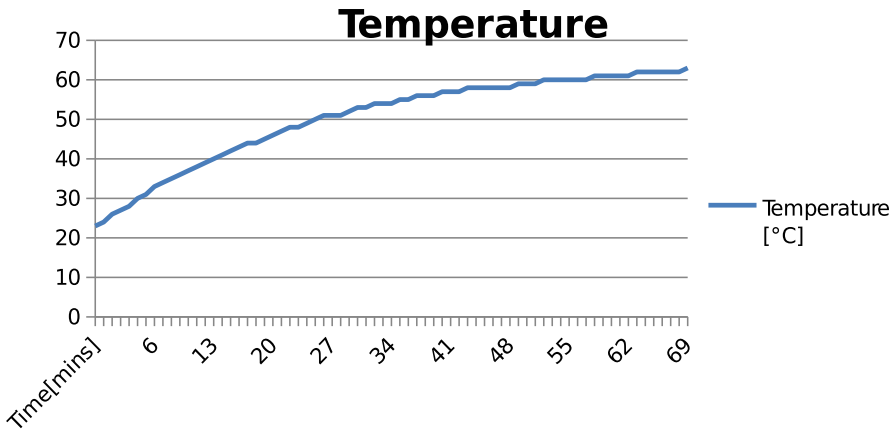


Figure 3.4: Step-motor temperature over time at 1A per winding

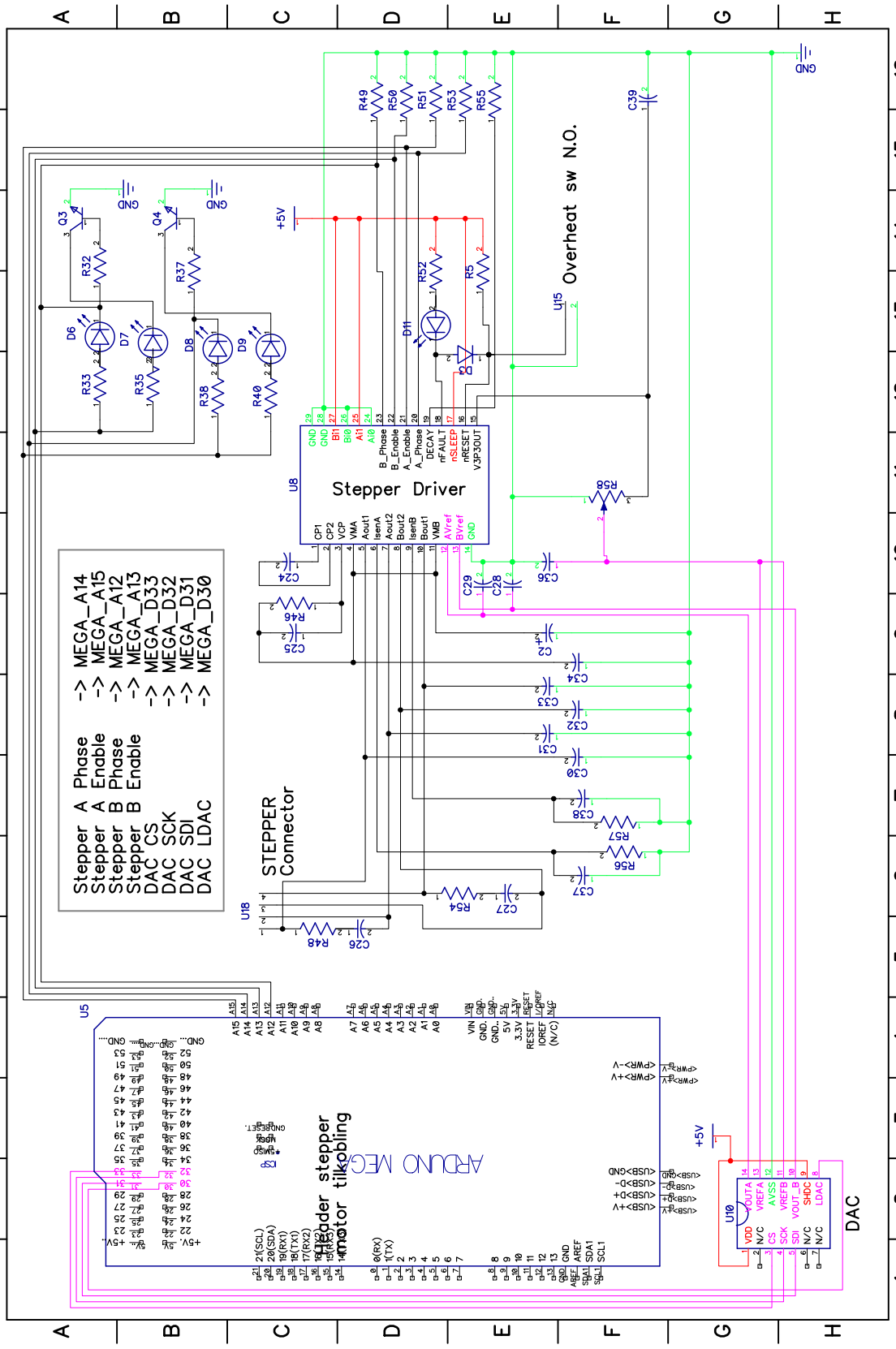


Figure 3.5: Step-motor driver circuit

3.8 DC-motor system

Figure 3.9 shows the schematic for the dc-motor circuit. The motor driver is the DRV8840[10]. In addition an ACS712ELCTR-05B-T[14] Hall Effect-Based Linear Current Sensor IC is used to measure the motor current. Most of the circuits are exactly the same as on the step-motor driver and will therefore not be discussed in this section.

Driver interface

The dc-motor is controlled using a enable and phase pin like the stepper motor. in addition there is a decay-pin, which determines the state of the H-bridge when the enable pin is set low. This pin can be set low for slow decay, or high for fast decay. The decay pin is by default pulled low by a resistor, setting the driving mode to slow decay. figure 3.6 shows the difference of the two decay modes. In short, fast decay disconnect the motor and allows it to coast, while slow decay connects the two motor terminals together, shorting the motors back-EMF causing the motor to brake.

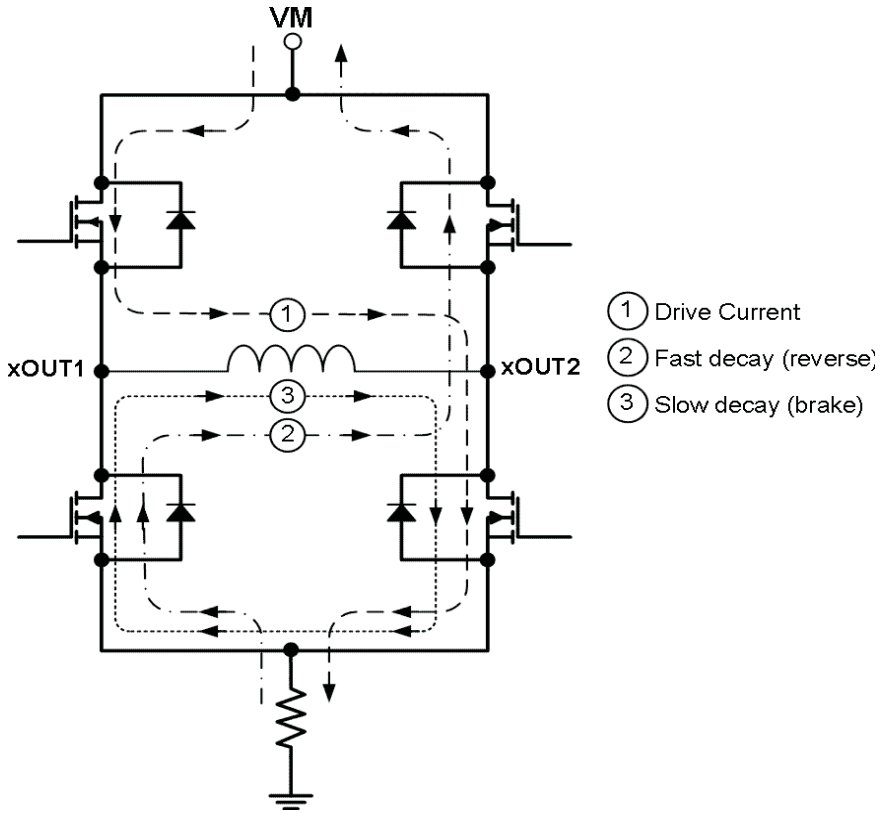


Figure 3.6: Decay mode[10]

There are three methods of controlling the DC-motor:

- ***Unipolar driving with low decay.*** In this driving mode the phase-pin is used to select the direction of the motor. A PWM-signal is applied to the enable pin to control the voltage applied to the motor. The decay pin is set low. this driving method gives a linear relation between the applied PWM-signal and the motor voltage. When the PWM-signal is set to 0, the motor will brake.
- ***Unipolar driving with high decay.*** This driving method is similar to the previous method except the decay pin is set high. When the PWM-signal is set to 0, the motor will coast to a stop. This drive method does not give a linear relation between the PWM-signal and voltage, and is therefore harder to use in a servo system. The difference between the

output voltage of the two unipolar driving modes can be seen in figure 3.7

- ***Bipolar driving.*** In this driving mode the PWM-signal is applied to the phase pin. The enable pin is set to high. In this mode the motor will stand still when the PWM is 50%, and then be able to spin in both directions by raising or lowering the PWM-signal. In this mode it is important to use a high PWM-frequency in order to lower the current consumption of the motor. An advantage with this driving mode is that only one signal is required to control the motor. See figure 3.8 for the relation between PWM-signal and output voltage.

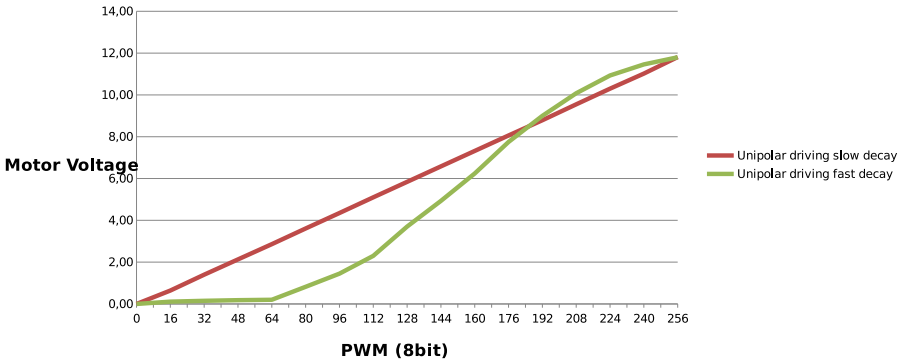


Figure 3.7: Motor voltage in unipolar mode with slow and fast decay

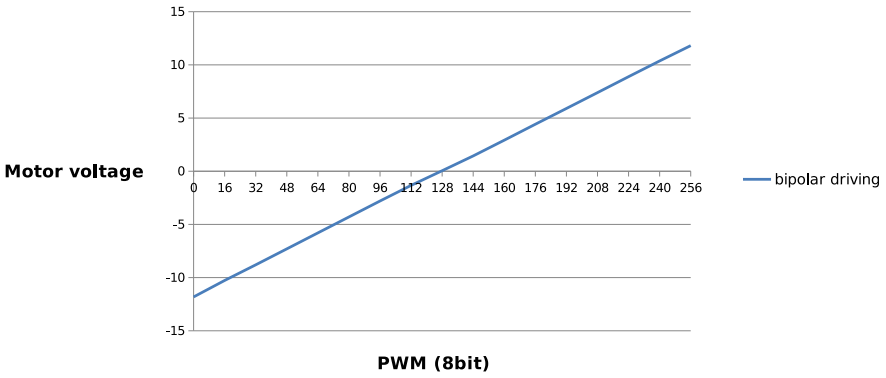


Figure 3.8: Motor voltage in bipolar mode

Current chopping

The DC-motor driver also includes a chopper driver. This is only used to limit the maximum current peaks through the motor. Chopping threshold can be adjusted using potentiometer R20. The DC-motor used in the Servo Lab, has a high thermal resistance between its rotor and the motor housing. If the power dissipated in the rotor is to high, there is a risk that the core will overheat and be damaged before the motor housing gets hot enough to trigger the over-temperature switch. On the Servo Lab prototype the threshold is set to 3 A, which limits the maximum power dissipation from the motor to 21 W.

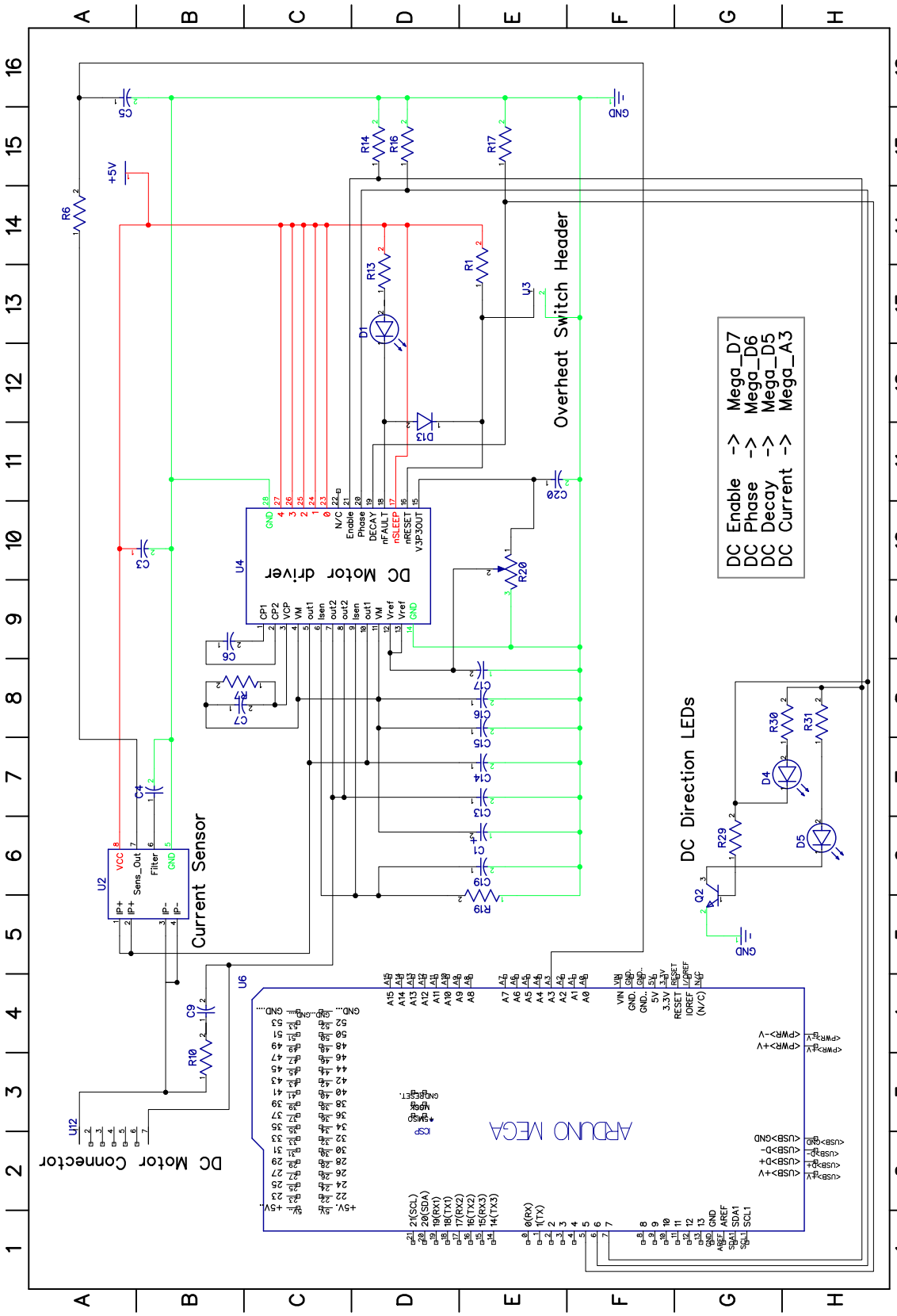


Figure 3.9: Dc-motor driver

3.9 Current sensing of the DC-motor

In order to do torque-control of the DC-motor some form of current-feedback is necessary. One way to read the current is to measure the voltage drop over the sense-resistor already used in the motor-driver chopping-circuit, and use Ohms law: $I = \frac{U}{R}$ to calculate the current. However this resistor is not in series with the motor, but in the H-bridge path to ground(se figure 3.1). This means that the motor current only flows through the sense-resistor when the H-bridge is in its driving state. Figure 3.10 shows a oscilloscope screen shot of the voltage across a 0.47Ω sense resistor connected to the sense-pin of the motordriver. Figure 3.11 shows the voltage over an external 0.235Ω resistor connected in series with the motor. This resistor can measure the real current through the motor.

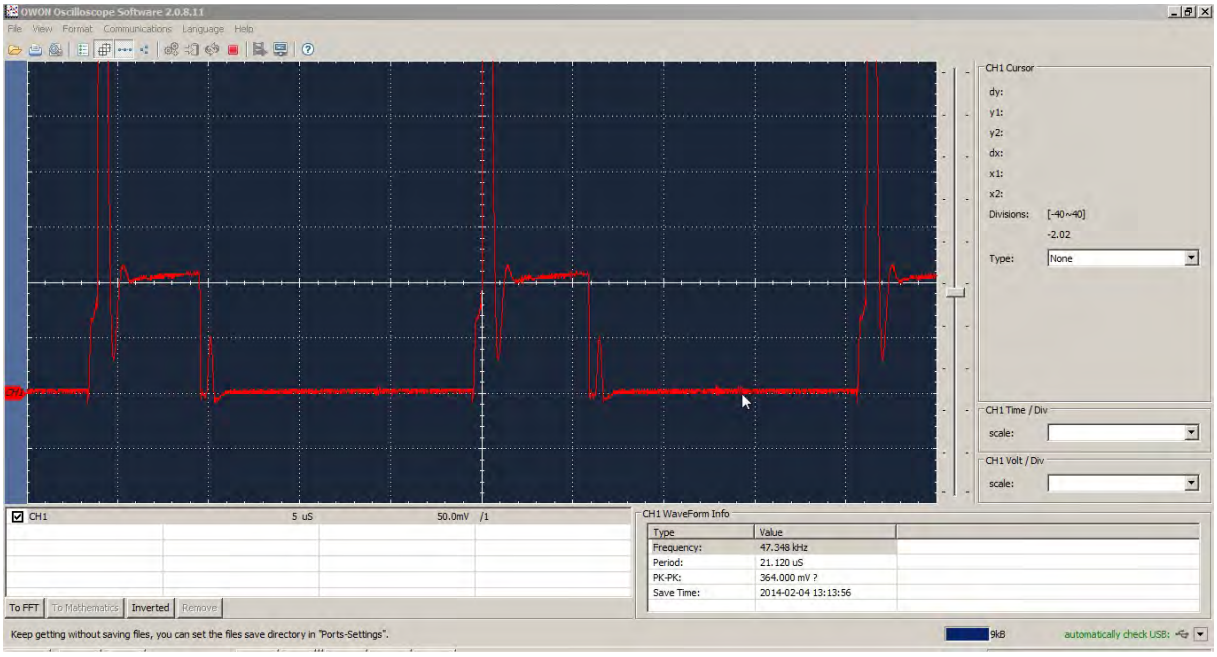


Figure 3.10: Voltage over 0.235Ω driver sense resistor. Slow decay, DC-motor at stall

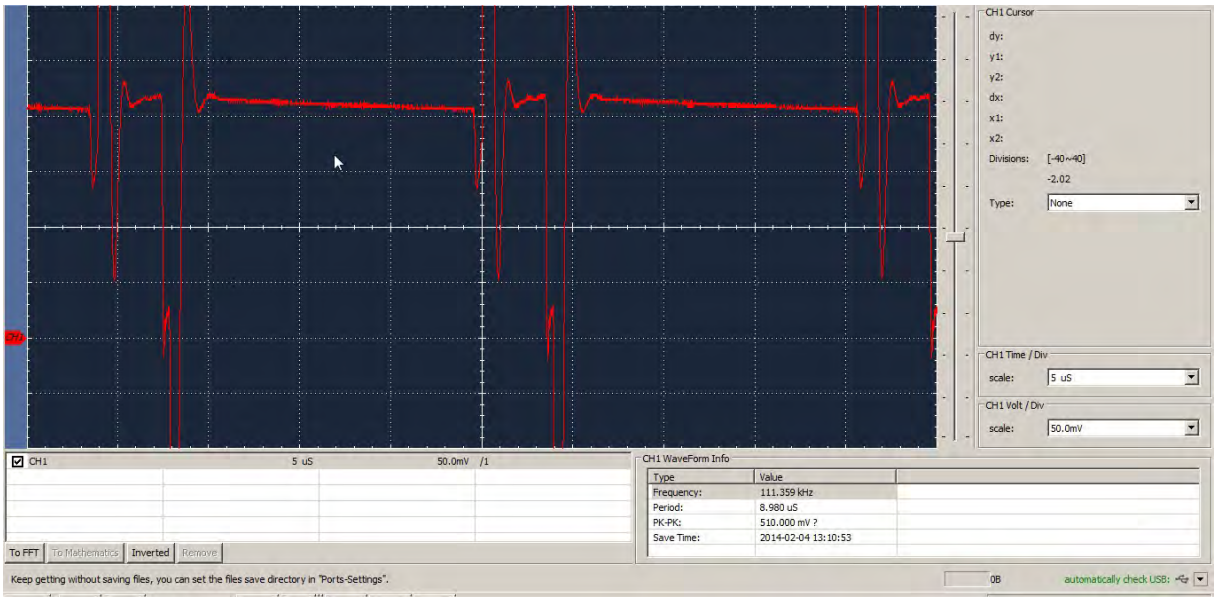


Figure 3.11: Voltage over an external 0.47Ω resistor in series with motor. Slow decay, DC-motor at stall.

In order to use the sense resistor to accurately measure the current, the voltage have to be

sampled exactly in the middle of the current burst. One way to solve this is to use phase correct PWM modulation on the enable pin, and trigger a sample in the middle of the PWM-period. This is not possible to do in the Arduino environment, so this method is desirable.

An other method is to use a peak detector circuit to hold the highest voltage over the resistor, so the voltage can be sampled later. Figure 3.12 shows the peak detector circuit that was tested. It has a low-pass filter on the input to cancel out the electrical noise from the motor. Figure 3.13 shows how the peak detector hold its voltage when the voltage over the sense resistor is removed. The disadvantage of the peak detector is if the pulsewith of the sampled signal is to short the peak detector will fail to bring the output signal as high as it should. This is shown in figure 3.14

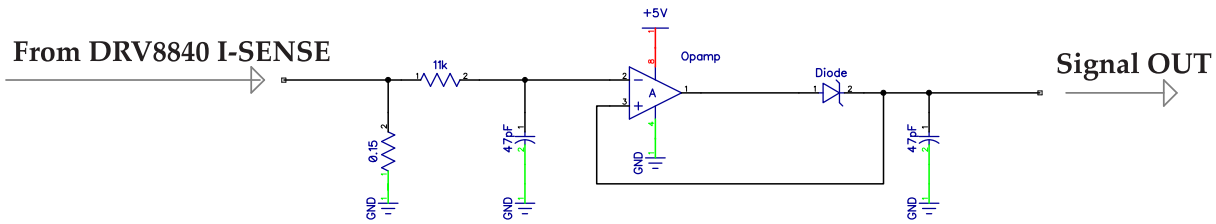


Figure 3.12: Peak detector

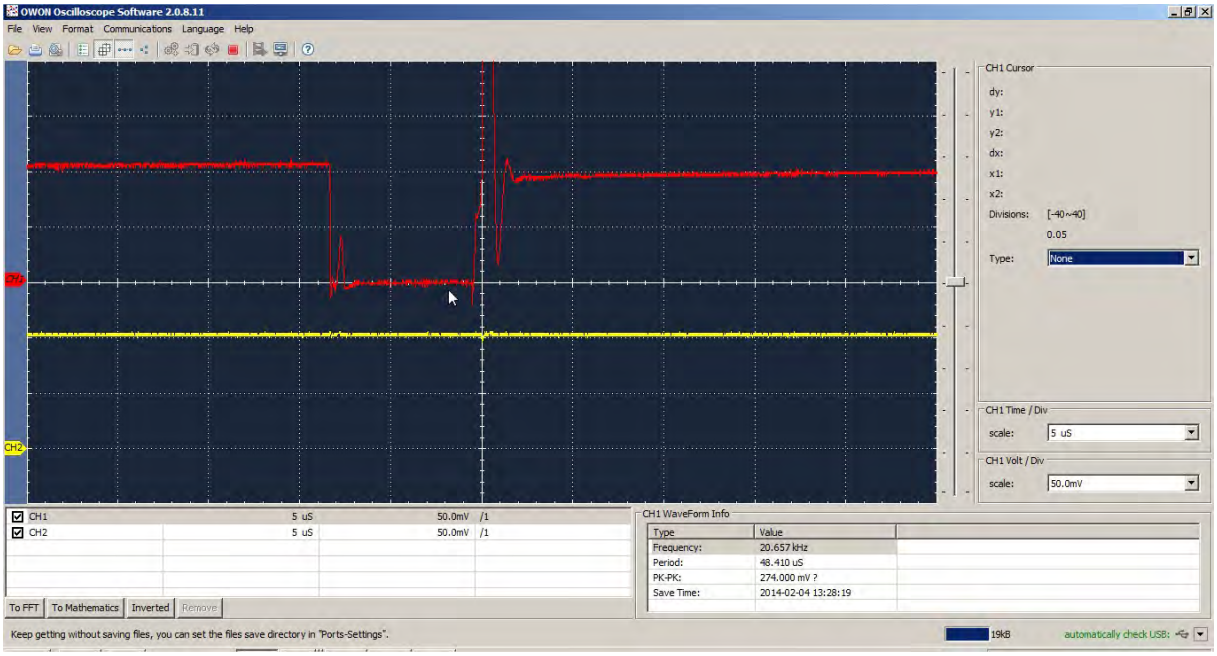


Figure 3.13: Ch1: Voltage over sense resistor. Ch2: voltage from peak-detector

The final solution was to use an dedicated current measurement chip in series with the motor. This adds some cost to the hardware, but provide a simple and accurate way of measuring the current. The IC selected for this was a ACS712ELCTR-05B-T[14] hall-effect based current sensor. The sensitivity of this sensor is 285 mV/A, with a voltage offset of VCC/2. This makes it possible to sense the current in both directions.

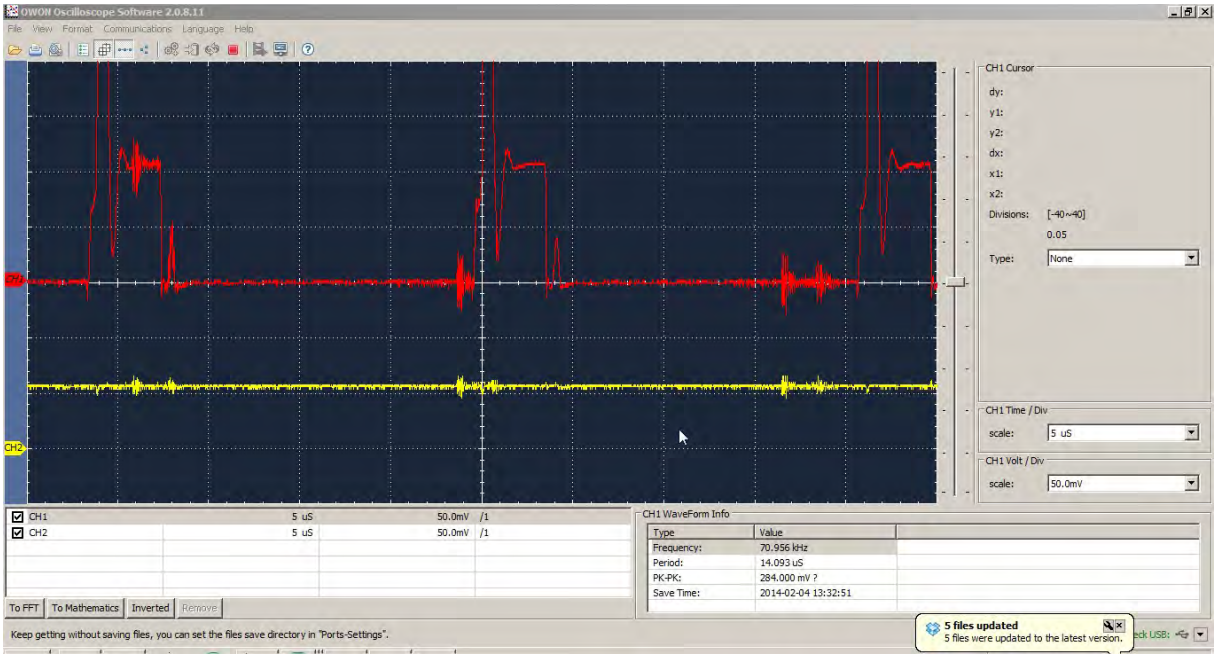


Figure 3.14: Peak-detector failing to maintain the peak voltage

3.10 Modification of the DC-motor connector

The original motor connector is a 6-pin standard header plug, which can be plugged into the socket in both directions. If the plug is connected in the wrong direction, the 5 V and ground on the hall sensor would be reversed. The motor supply voltage would have been connected to the output pins of the hall sensor. This would probably killed the hall sensor. Depending on how the hall sensor would fail, there is also a probability that the "high" voltage from the motor supply would find a connection to the 5V rail, destroying all devices in this rail. This must be prevented, either by changing the existing unpolarized connector with an polarized one, or make the motor connection able to work when it is plugged into the socket in both directions. Figure 3.15 shows how the problem was solved by changing the 6-pin connector to a 7-pin and realign the pins. If the modified motor connector is plugged in backwards, the only result is that the motor will spin in the other direction.

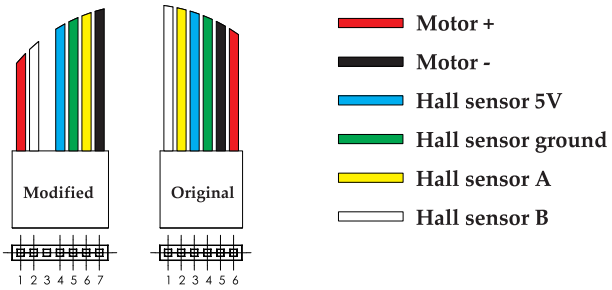


Figure 3.15: Original and modified motor connector

CHAPTER 4

Chassis and circuit board design

*"Design is not just what it looks like and feels like.
Design is how it works."*

–Steve Jobs

4.1 Design process

The unit to be developed shall have an easy to use interface. It must also be ready to use right out of the box without the need for cabling or assembling. This type of preparation is time consuming and not really the essence of the intended usage.

The physical unit itself should have a form factor that enables easy and quick set-up, this is quite important since the students only have a finite amount of laboratory time and to be most effective the students should be able to set up the system and start their experiments.

In order to collect all of the sub systems into one single unit, there is a need for a specifically designed printed circuit board (PCB). This feature will enable the designer to create a compact design, while having the flexibility to design the circuit board to be embedded into the physical unit.

In order to enable the designer to create and order printed circuit boards, there is a need for a computer aided design (CAD) software. Depending on what is to be designed, there are several software providers to choose from. To design the PCB, some considerations had to be taken. First of all, the software had to be able to render the standard file format, *gerber-files*, that is sent to the PCB manufacturer. These files contain all the necessary information to print and assemble the circuit board. Furthermore, the software also had to have a free student version, or provide a trial version with adequate features.

Lastly, for the end user/student there will be a need to have an electrical schematic available in order to understand the laboratory assignments given. Therefore a good PCB-CAD software should have the option to export the PCB schematic together with circuit details to a suitable format.

Much the same way as the design of the circuit board, the design of the educational unit itself will require the designer to use a computer design tool. The field of 2D and 3D CAD tools have come a long way, and the ability to perform a top-down design[*glossary*] of the physical unit enables the designer to implement frequent design changes while the dependent part re-

lations are automatically updated. In the market of 2D 3D design tools there are CAD systems available for pretty much all levels of complexity, ranging from simple 2D drawing software to more complex 3D systems with the ability to assemble multiple part components into an assembly of the finished product, in addition to other high level simulation capabilities such as flow, stress and fatigue to name a few.

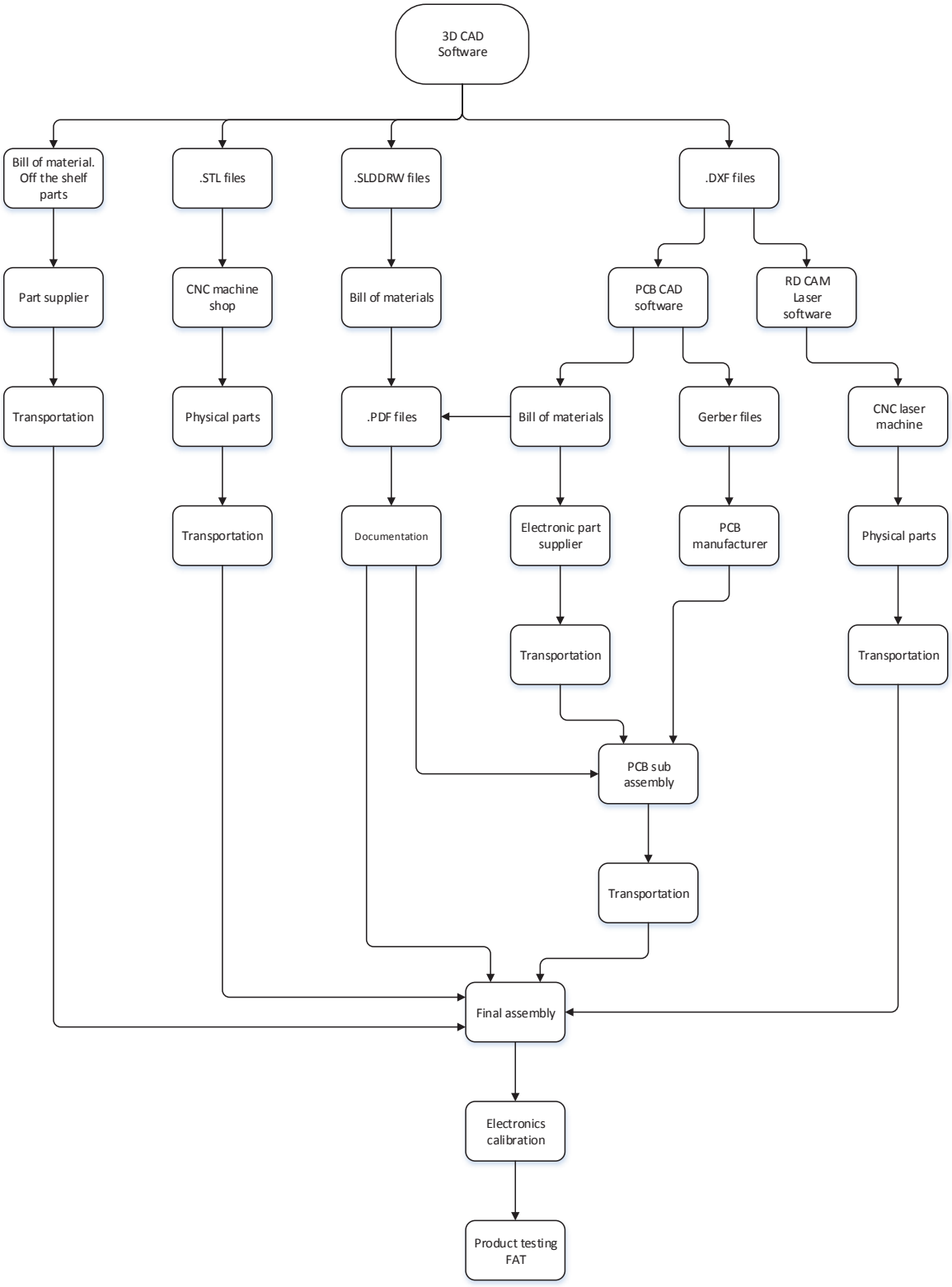


Figure 4.1: Schematic of the design workflow

4.2 3D CAD design

The design of the physical unit was done using a CAD package form Solidworks®2014. The design was to include one four lead bipolar stepper motor, one DC gear motor with 16 line quadrature encoder, a Bluetooth®TM interface module, a 16 character by two line liquid crystal display ,eight buttons and eight light emitting diodes connected to their respective microprocessor ports, in addition to this we decided to include a 2 axis joystick, a rotary potentiometer and a mechanical rotary encoder.

The integration of the sub systems into a final product was a driving force for the design, the design needed to be functional and provide the user with the appropriate feedback from the respective sub systems, with this in mind the unit went through many fairly different layouts before finally ending up with a design that was both simple in construction and fulfilled the technical requirements given.

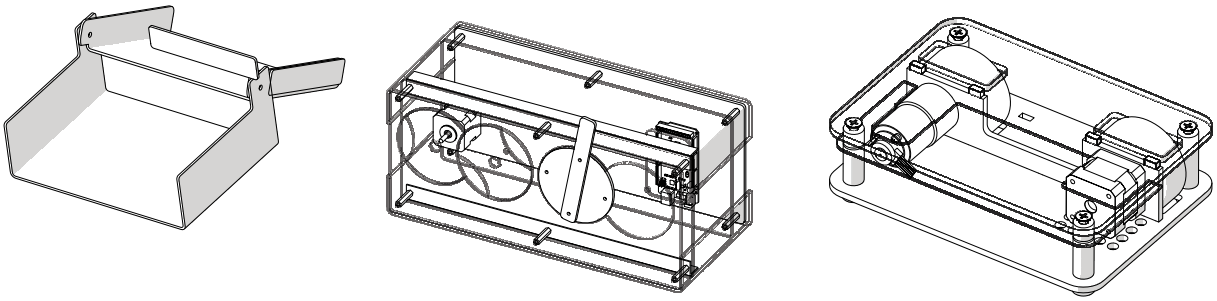


Figure 4.2: Suggested designs early in the project

The area of the design that was given the most attention was the ability for the unit to convey meaningful feedback to the user as to what was going on. The top plate of the device was effectively transformed into a schematic representation of the sub systems involved, by means of sectioning out the human-machine-interface (HMI) part, DC-motor/encoder and the Stepper-motor, also the flywheels of the motors were made accessible so that the user will be able to apply manual disturbance to the system.

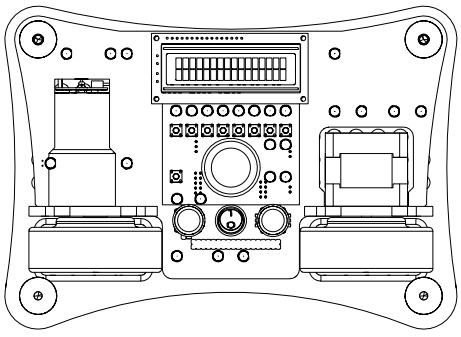


Figure 4.3: Servo Lab with top removed

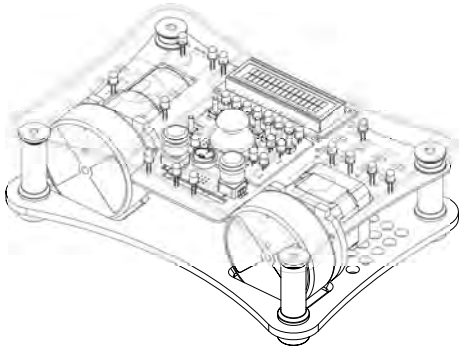


Figure 4.4: Isometric view with top removed

4.3 Arduino power supply design

The Arduino board needs a 5 volt power supply. This power is desirable to be drawn from the same power supply that supplies the motors. The Arduino board contains a linear voltage regulator, which accepts voltages between 6-20V according the Arduino web page technical spec[15]. The regulator circuit on the Arduino board is built around the NCP1117ST50T3G linear voltage regulator.

A linear regulator is very ineffective. It works by dissipate the excess voltage into heat inside

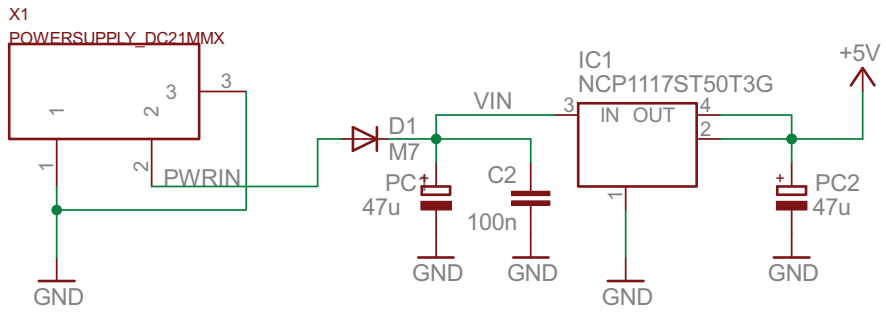


Figure 4.5: Arduino 2560 board voltage regulator circuit [16]

the regulator. The power dissipated inside the regulator can be calculated by the following formula:

$$P_{DISSIPADED} = I_{IN} \times (V_{IN} - V_{OUT}) \tag{4.1}$$

Where $P_{DISSIPADED}$ is the power dissipated inside the regulator, I_{IN} is the current going into the regulator. It is virtually the same as the current out of the device. V_{IN} is the voltage going in, and V_{OUT} is the voltage coming out of the regulator.

The highest current consumption of the 5 V rail on the Servo Lab is calculated to approximately 200 mA. This includes the microcontroller, the Bluetooth module, LCD, encoders, sensors, and all the LEDs. Using a 20V power supply equation 4.1 gives a power dissipation of 3 W. In the data-sheet of the NCP1117ST50T3G-regulator this table is shown:

NCP1117, NCV1117

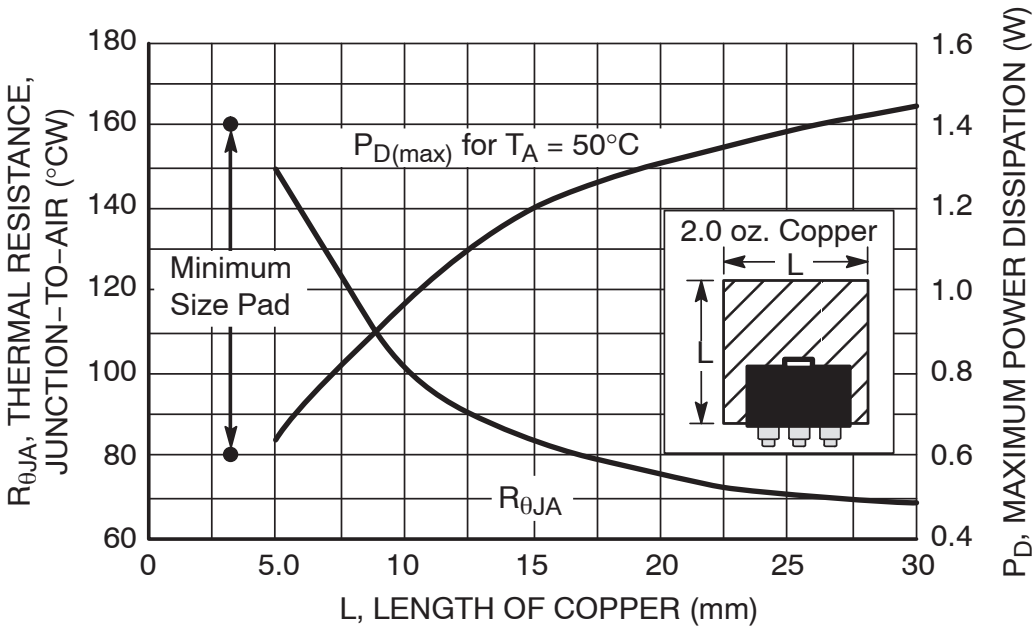


Figure 4.6: Thermal Resistance and Maximum Power Dissipation vs. PCB Copper Length [17]

The copper plane underneath the regulator is approximately 12.5 mm. This gives a maximum power dissipation of 1.1 watt with an ambient temperature of 50°C. This means that the input voltage into the regulator have to be lowered in order to prevent it from overheating. This have to be done using an other voltage regulator in front of the Arduino regulator. This should be a switching type regulator to minimize the power dissipation of the system.

According the NCP1117ST50T3G data-sheet the maximum drop-out voltage[1] at 800 mA output current, is 1.2V. This means that in order to get the lowest possible power dissipation through the linear regulator, the input voltage should be set to 1.2 V above the output voltage. In this case according to equation 4.1, the Arduino voltage regulator would only dissipate 0.24 watt. well within it's operational range.

4.4 PCB/schematic CAD design

The printed circuit board CAD package selection landed on a software package called DipTrace®from Novarm Limited. The company offers a 30 day trial period of their full version of the project, in addition to a version that is not time limited, but limited by the amount of component pads on the board. Both these versions are completely free. The version we selected for this project was the component limited version. However during the design process we exceeded the free amount of components, and was forced to start the 30 day full version to complete the design.

The schematics of the different sub systems was combined into one large schematic, and pin mapped to the Arduino connections according to the individual requirements such as PWM hardware interrupt and communication functionality.

The PCB outline was previously modelled in the Solidworks environment and exported as a .DXF file, this file was then imported into the PCB software and mapped to form the actual PCB outline, the schematic was then converted into a collection of components that was automatically interconnected with thin lines called "ratlines" representing the copper traces to be routed.

The layout of the components on the circuit board was carefully placed into groups similar to their sub system origin, in order to keep the copper traces as short as possible. The components and traces was moved around many times during the routing of the actual copper traces, while continuously having to make compromises between keeping analogue, power and digital signals separate.

4.5 Assembly of the prototype circuit board

To assure that there was no problems with the designed circuit board, the design had to be tested. There is no viable way to simulate it, so testing required a complete and assembled PCB. Five circuit boards were ordered and delivered in mid-April. This is the minimum order, but that also ensures redundancy. There is always a chance of production errors, damage during transport and errors made during assembly and soldering.

Before placing the components, the soldering paste was applied to the PCB using the stencil. Placing the surface mounted devices (SMDs) was done manually, placing each component with tweezers. The next stage was to heat the circuit board, so that the SMDs are soldered and fixated. There was no soldering oven available, but this was solved by using a re-flow heat gun. Also the SMD-soldering of the HC-05 Bluetooth module worked satisfactory without any extra soldering. After the soldering of the top-layer components was completed, the procedure was repeated for the bottom layer.

Next, the through-hole components needed to be soldered in place. This included headers, LEDs, joystick, encoder and potentiometers. Soldering these was done conventionally using soldering-iron and soldering wire. The correct offset for the LEDs was achieved by using a custom distance gage. One of the LEDs has bent legs, thereby translating the position, and so

[1] Drop-out voltage is the minimum voltage drop through the regulator

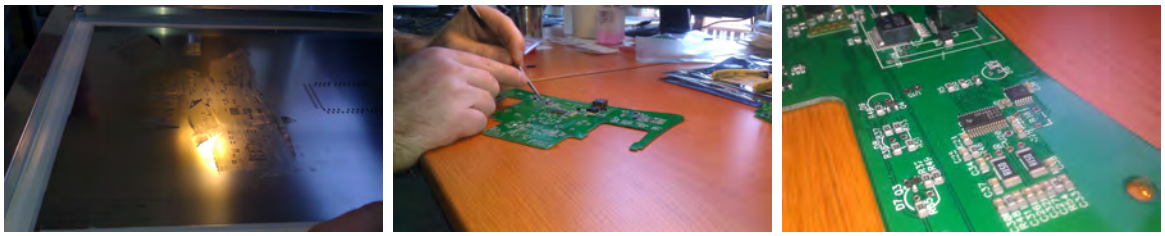


Figure 4.7: Applying solder-paste and placing components

needed a larger offset. In the procured, fully assembled circuit boards, this special offset has been taken in to account.

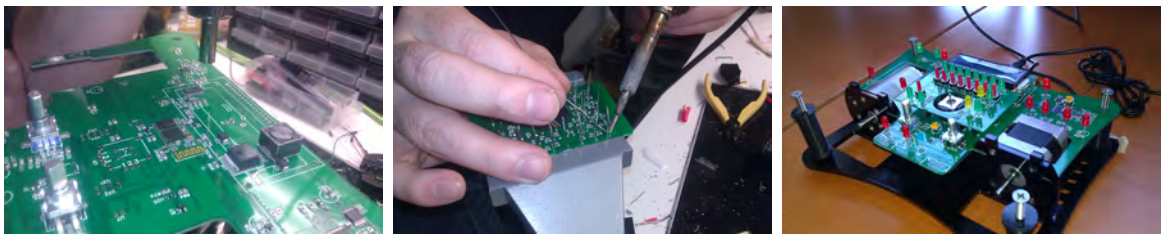


Figure 4.8: reflow soldering, through-hole soldering and the assembled PCB

The main circuit board is positioned with regard to the appropriate placement of the joystick. Because of this, there are no commonly available tactile switches that could be used directly. Therefore, a second, smaller PCB was designed. In addition to the buttons, this circuit board is used for most of the LEDs, the LED series resistors and button de-bounce circuitry.

The small circuit board was assembled and soldered using the same procedure as that of the main circuit board. Finally, the appropriate headers on the main circuit board was soldered to the small circuit board, along with the LCD circuit board.

4.6 Bluetooth and communication system

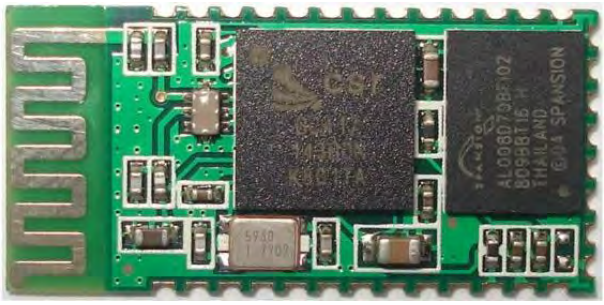


Figure 4.9: Hc-05 Bluetooth module[18]

The Servo Lab is equipped with a HC-05[19] Bluetooth serial module. This module allows the Arduino to send serial strings over Bluetooth to another devise that supports Bluetooth serial data. The module can act both as a slave and a master. figure 4.10 shows the schematic of the Bluetooth circuit.

Interface

The Bluetooth module works on voltages from 3.0-4.2 volts. Because the Arduino system runs on 5 volts, there is circuits to convert the voltage between the two units. The 3.3 volt supply

for the module, is taken for the 3.3V rail on the Arduino board. The RX and TX signals are buffered, and used to drive two LEDs on the top panel to visualize the data traffic. There is also a LED on the top panel which light up blue, when the Bluetooth is connected with another devise.

User interface

The Bluetooth module is connected to “Serial port 2” on the Arduino. Default baudrate is 9600 bit/s, and the default mode is as a slave unit. To change roles, devise name or similar, the module can be set in AT-mode by using the BT-key and BT-reset pins. This is further described in the module data-sheet.

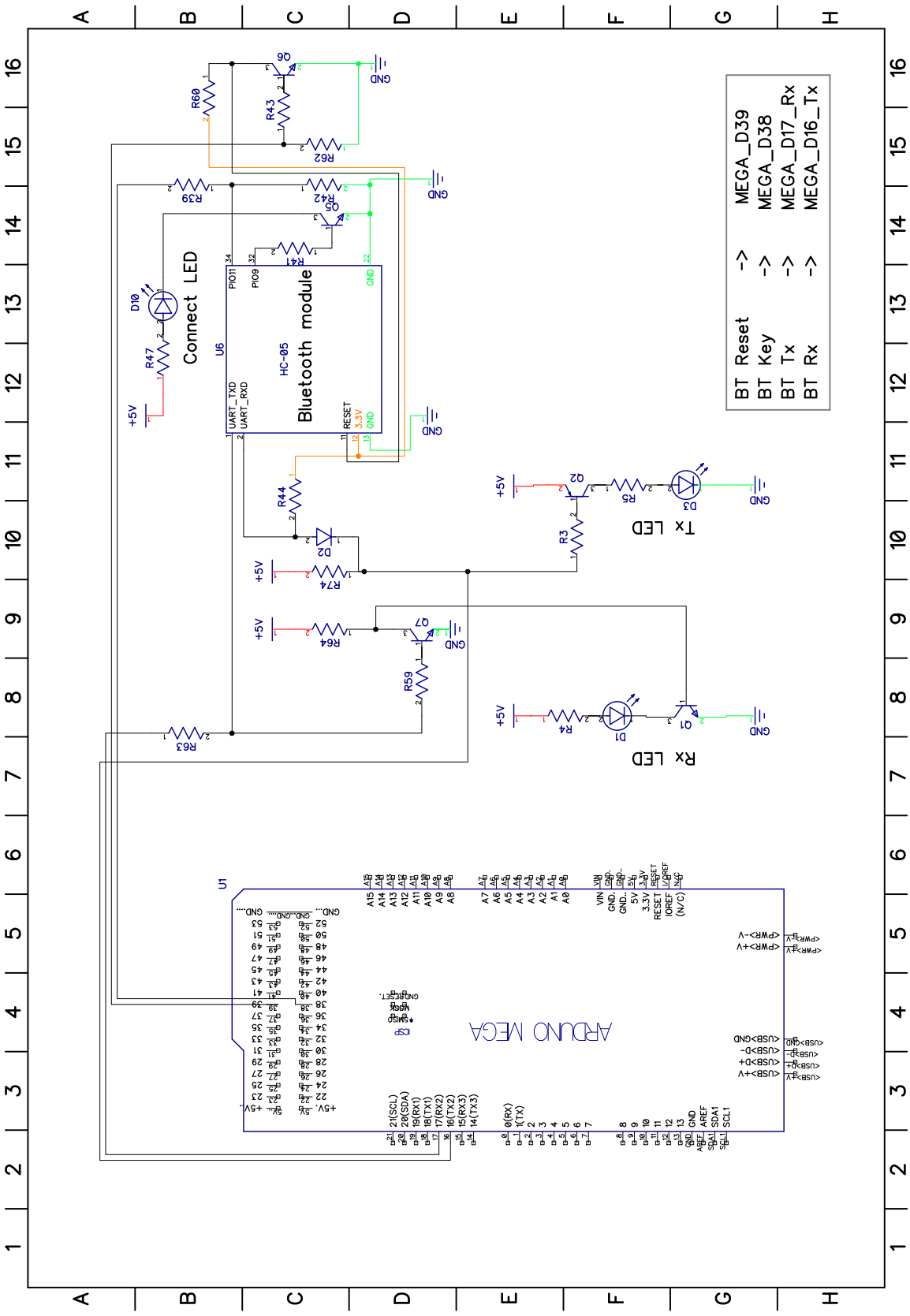


Figure 4.10: Bluetooth module schematic

4.7 Human-Machine interface system

In order for the students to interact with and get feedback from the developed equipment, an interface must be developed in accordance to the requirements given for each sub system. The

system can in general be divided into two main types of interface.

The first type being input interface from the operator to the machine, this interface will typically be digital and analogue signals given from the operator to the micro controller device.

The second type of interface will be a set of output signals from the micro processor unit that will need to be communicated to the operator, this interface will convey both binary signals and analogue signals. The analogue signals will be represented by means of PWM modulation of the front panel lights to emulate a continuously variable signal.

There will also be a display conveying information as plain text on the front panel, the text is written to the display by the microcontroller and will display two lines of text 16 characters long, the display can represent 255 different characters given by the ASCII table.

In addition to these interface methods, there is also the option to control the device via a bidirectional serial link, where graphical applications can be written on a computer either in the Processing software, visual basic or equivalent software packages, this will enable for both commanding to and receiving information from the All in one Servo Lab.

Figure 4.11 shows the schematic of the 8 programmable LEDs on the top panel. The LEDs are connekted to a single port on the microcontroller. This allow the processor to write to all the LEDs in one single operation.

Figure 4.12 shows the schematic of the 8 switches on the top panel. Each switch is provided with a pull-down resistor and a de-bounce[2] circuit with a 10 mS time constant.

Figure 4.16 shows the schematic of the motor encoder circuit. and the analogue sensors.

[2]A de-bounce circuit is circuit that cancel out noise when the contact points in a mechanical switch is bouncing against each other

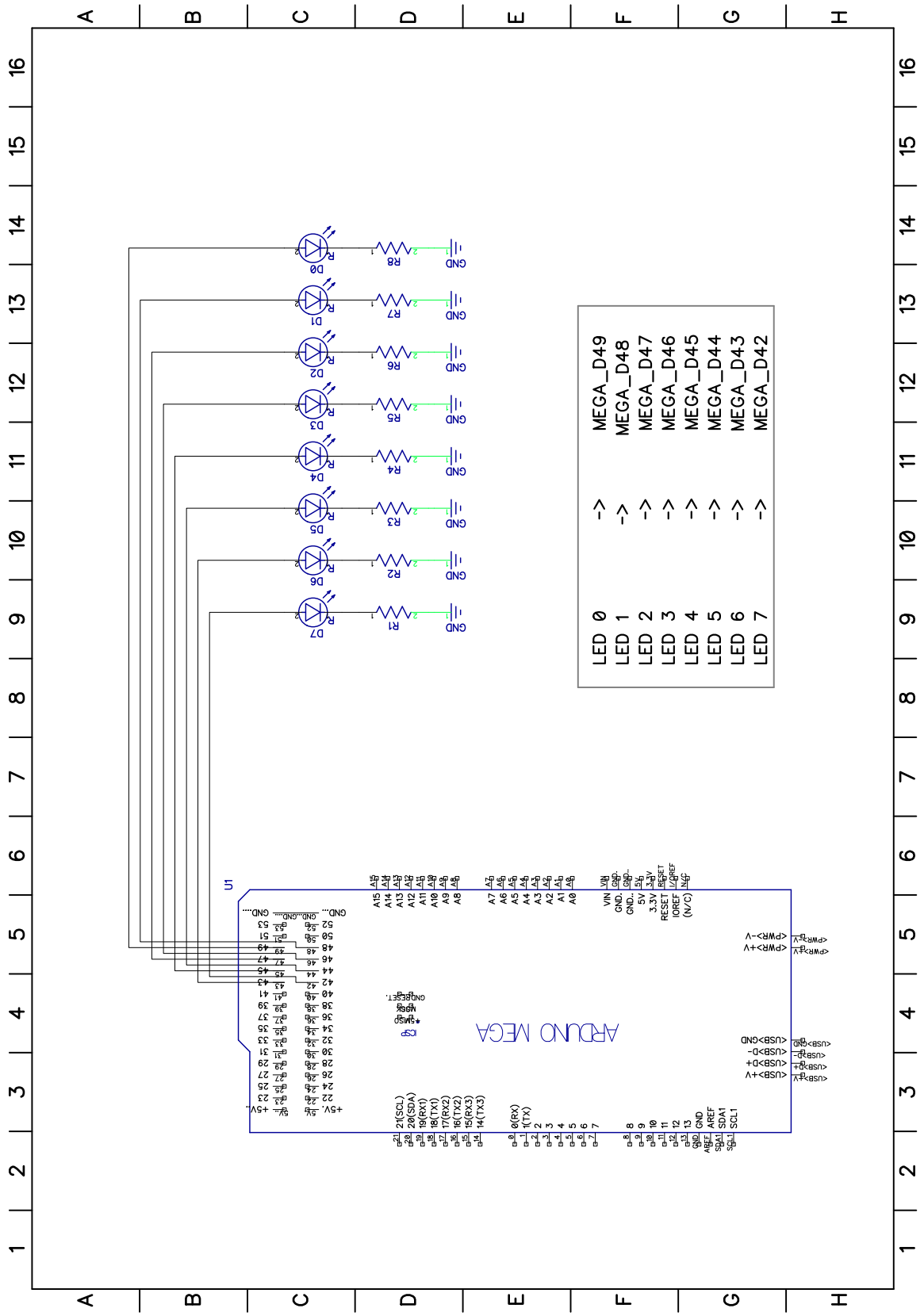


Figure 4.11: LEDs

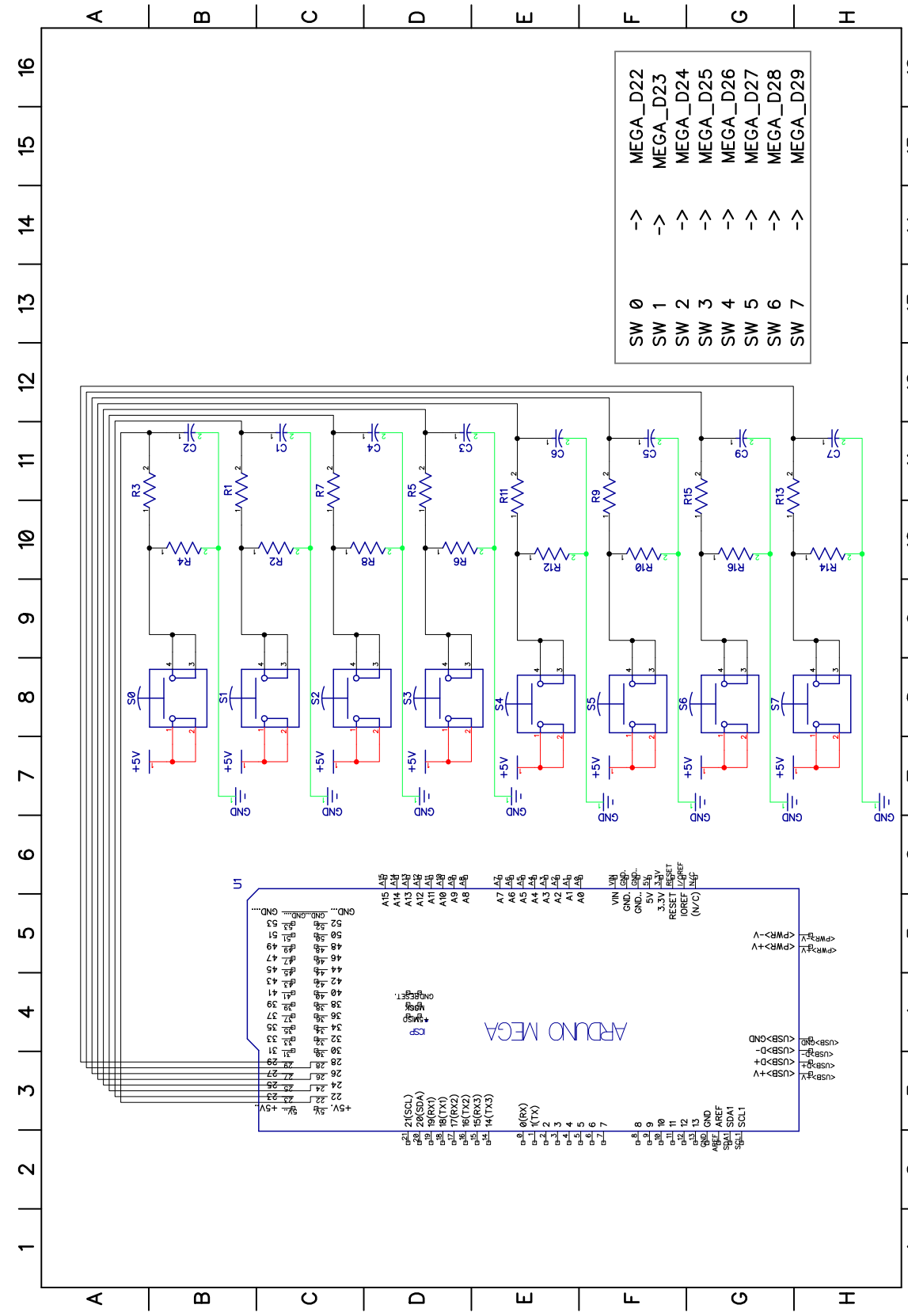


Figure 4.12: Switch schematic

4.8 Motor encoder

The encoder is built up around two Hall effect sensors which senses the fields of a rotating disc with a magnetically polarized pattern. The encoder outputs a 2-bit grey-code, which is corresponding to the reading of the hall sensors. The rotating disc is polarized with 16 north- and south-poles. Depending on how the grey-code is read by the microcontroller different resolutions of the encoder are possible. By only reading the encoder at the rising edge of one signal line, the resolution is 16 Counts per rotation (CPR). Reading the encoder at both rising and falling edge, gives a resolution of 32 CPR. Reading the encoder on both rising and falling edge on both signal lines, gives the resolution of 64 CPR. Two LEDs are added to the top panel to visualise the gray-code coming out of the encoder. The output signals from the encoder is connected to two of the interrupt pins of the Arduino.

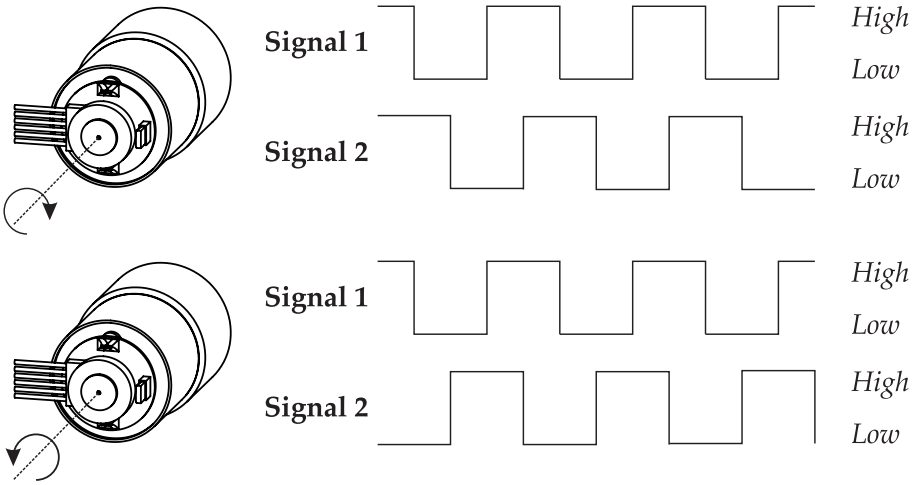


Figure 4.13: DC-motor encoder signal

4.9 Analogue sensors

The potentiometers inside the joystick and the rotating potentiometer on the top panel, are connected between 5 V and ground. The wiper pin is connected to the analogue pins on the Arduino through 220 Ω resistor. The purpose of the resistor is to prevent the Arduino pin from sinking or souring excessive current, if it is accidentally set as an output when the potentiometer is at the end of its range.

4.10 External interface pins

The Arduino pins which is not used by the Servo Lab is brought out to external headers. On the back side of the Servo Lab, two analogue pins ar brought out on headers which is pin compatible with the Tinkerkit™. This provides an easy interface to external sensors4.14.

The rest of the free pins is brought out to two double headers in the front of the Servo Lab. The pins available on these headers are: SPI-port, a full 8-bit port, Serial port 3, one interrupt pin, 4 PWM supported pins and 6 analogue pins.

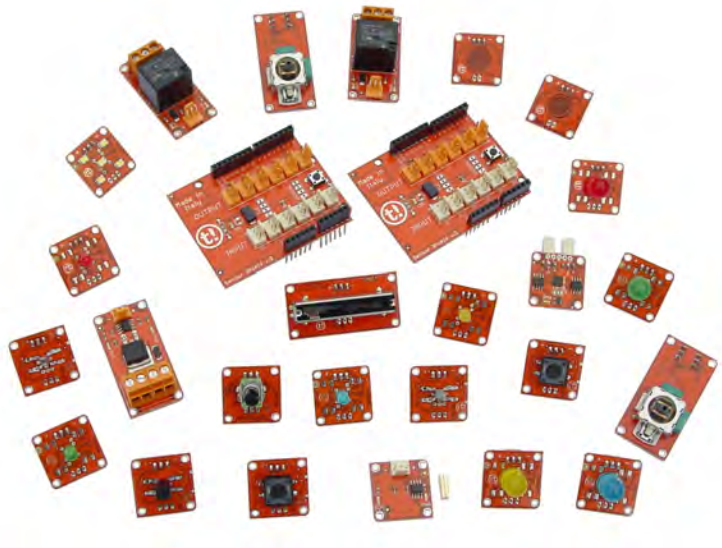


Figure 4.14: Tinkerkit Pro for Arduino [20]

4.11 Mechanical encoder

Figure 4.17 show the circuit of the mechanical encoder. It outputs the same grey-code as the motor encoder, and uses two LEDs on the top panel to visualise the code. Sins the encoder uses mechanical switches, a de-bounce circuit is needed filter the switching noise, to ensure the encoder won't skip ticks. The encoder also contains a switch which is operated by pushing the encoder-button down. The three outputs of the encoder is connected to interrupts pins on the Arduino.

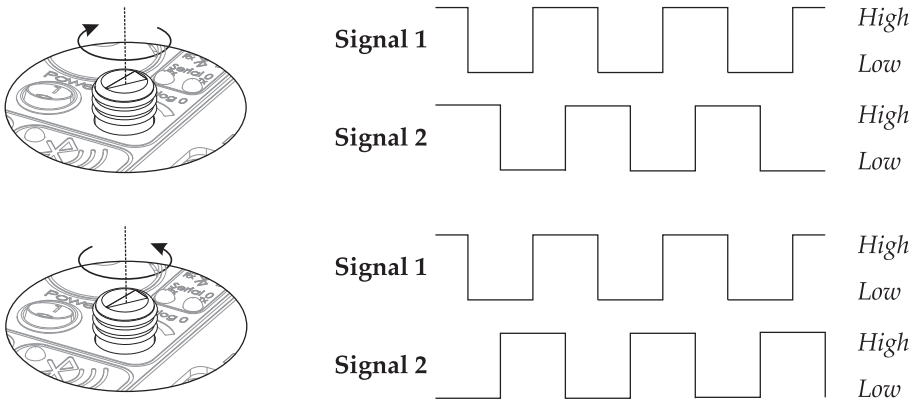


Figure 4.15: Mechanical encoder and signals

4.12 LCD-display

Figure 4.18 shows the LCD circuit. The LCD-display is a white on blue background monochrome character display, able to show 16 characters on two lines. The display uses the "Hitachi HD44780"[21] LCD-driver. The driver support both a 8-bit and a 4-bit parallel interface. On the Servo Lab the 4-bit interface is used in order to save Arduino pins and to have better compatibility with existing code libraries for the Arduino board. the potentiometer(R3) is used to adjust the contrast of the display. The transistor(Q1) is used to control the back-light on the display. The back-light can be swished on or of by writing the base pin of the transistors high or low. By writing a PWM-signal to the transistor the back-light can be adjusted to any levels.

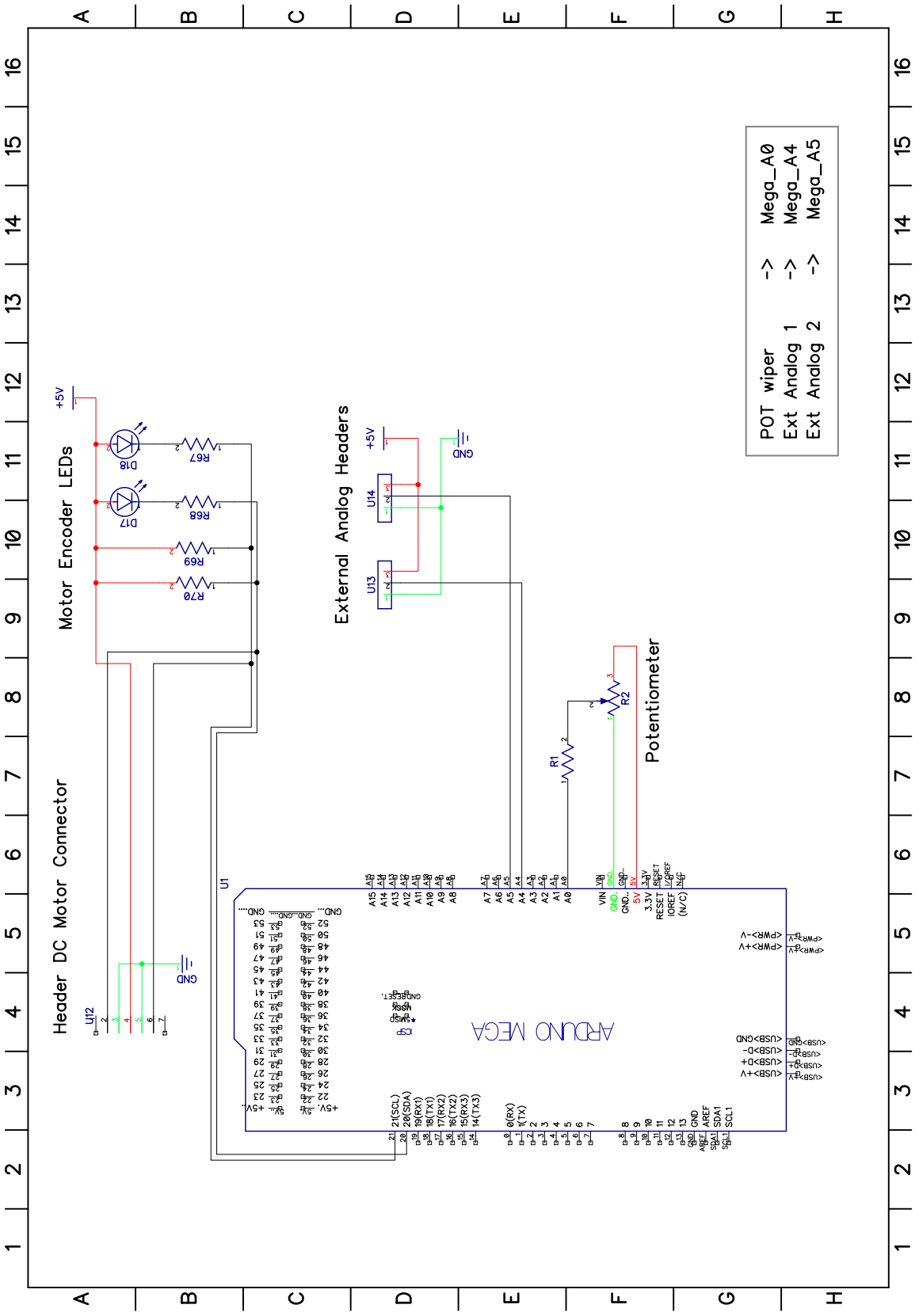


Figure 4.16: Motor encoder, joystick and potentiometer circuit

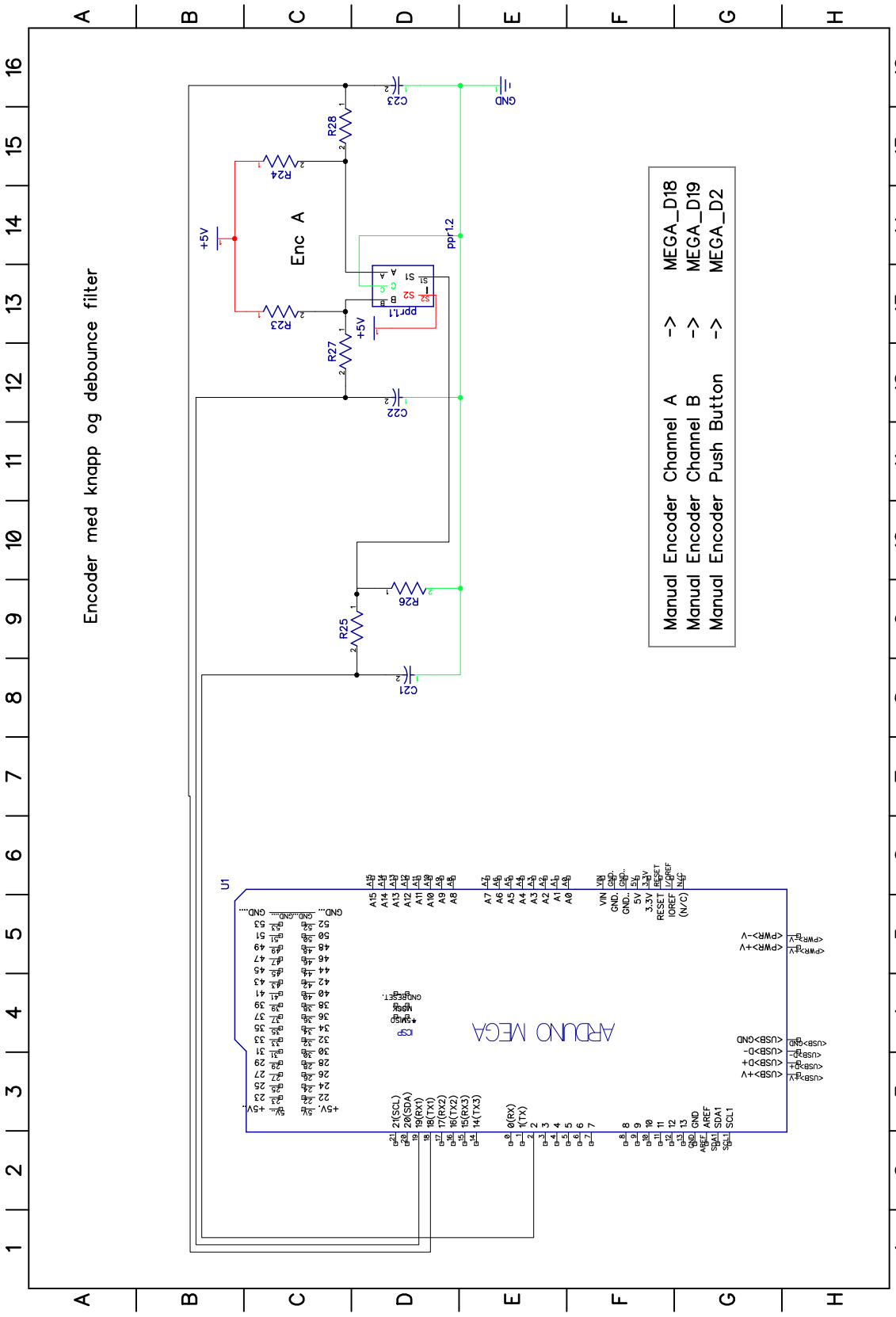


Figure 4.17: Mecanical encoder

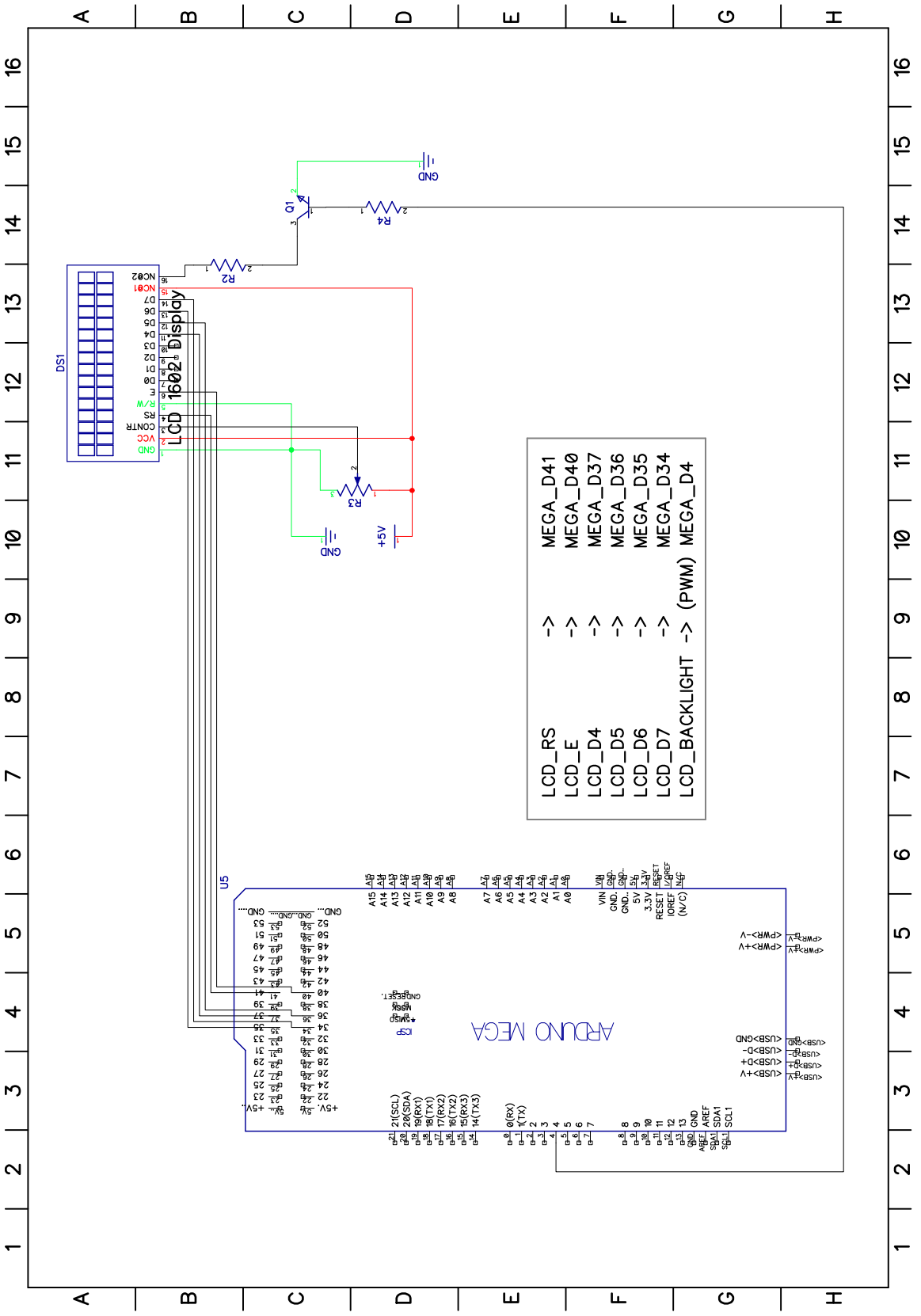


Figure 4.18: LCD

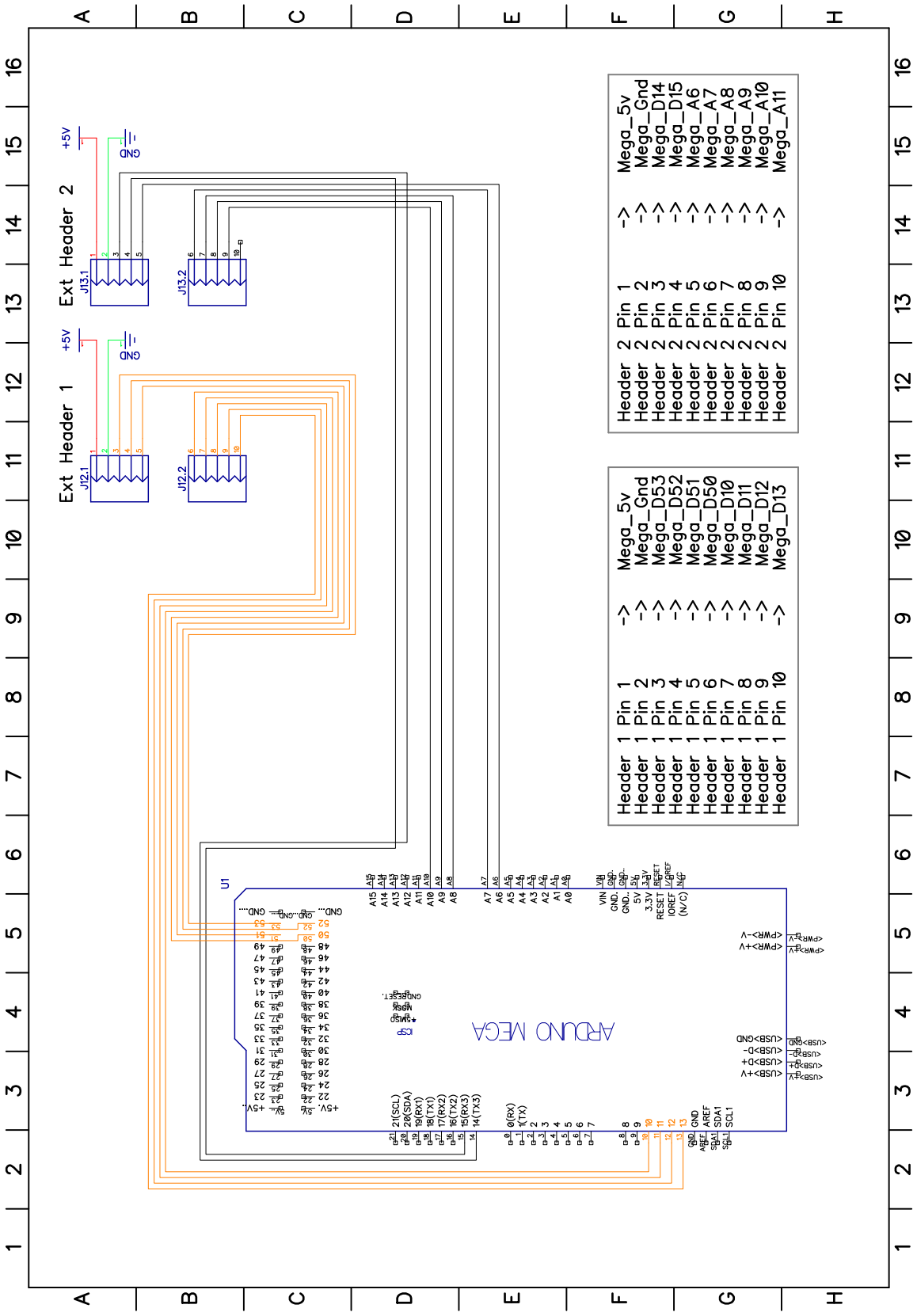


Figure 4.19: External headers

4.13 Assembly and main parts

The All in one Servo Lab have a number of different parts(table 4.1) that makes up the final assembly, and in order to make the product easy to assemble, the core design have been developed around an idea that the parts should be assembled into sub-assemblies that will slide into place and be fixed by the top plate interlocking design. The steps are described below, and illustrated in figure 4.20.

1. Assemble the DC-motor and Stepper motor to the plates using countersunk flathead M3x8mm screws.

2. Assemble the flywheels to the motor shafts by a method of gluing with Loctite®-2701 Thread locking solution and setting the clearance to 5.7+- 1mm. Set it to dry for one hour.

3. Place both motor assemblies in the mating holes of the bottom plates.

4. Assemble the lower four stand-offs and feet using four countersunk flat-head M6x12mm screws, do not tighten the screws yet.

5. Slide the complete circuit board into the mating slots in the motor plates and place the two top stand-offs on top of the pcb.

6. Place the top plate on top of the circuit board while being careful to align all of the buttons and LEDs to their mating holes, and at the same time fitting the motor plate notches into the mating slots in the top plate. Insert two M5x12 and two M5x20 countersunk flat head screws.
   Place the complete assembly on a flat surface and tighten the four top screws first, then put the assembly on the side and tighten the four bottom screws.

7. Mount the O-rings onto the knobs and glue the knobs to the encoder/potentiometer shafts by method of gluing with Loctite-2701 and leave to dry for one hour.

Table 4.1: All in one Servo Lab: Bill of materials (BOM)

Description	BOM ID	Quantity
Alu_ feet	2	4
Bottom_ plate	3	1
Button	4	1
Countersunk flat head cross recess screw	25	10
DC_ Plate	6	1
DFRobot_ FIT0127	7	1
Flywheel_ DC	8	1
Flywheel_ Stepper	9	1
Knob	12	2
NEMA17	11	1
PCB	14	1
Socket countersunk head screw_ am	26	8
Spacer_ Bottom	19	2
Spacer_ Front	20	2
Spacer_ Top	18	2
StepperPlate	21	1
TopPlate	22	1
V4GB37Y_ DC12v_ 83rpm_ Motor	24	1

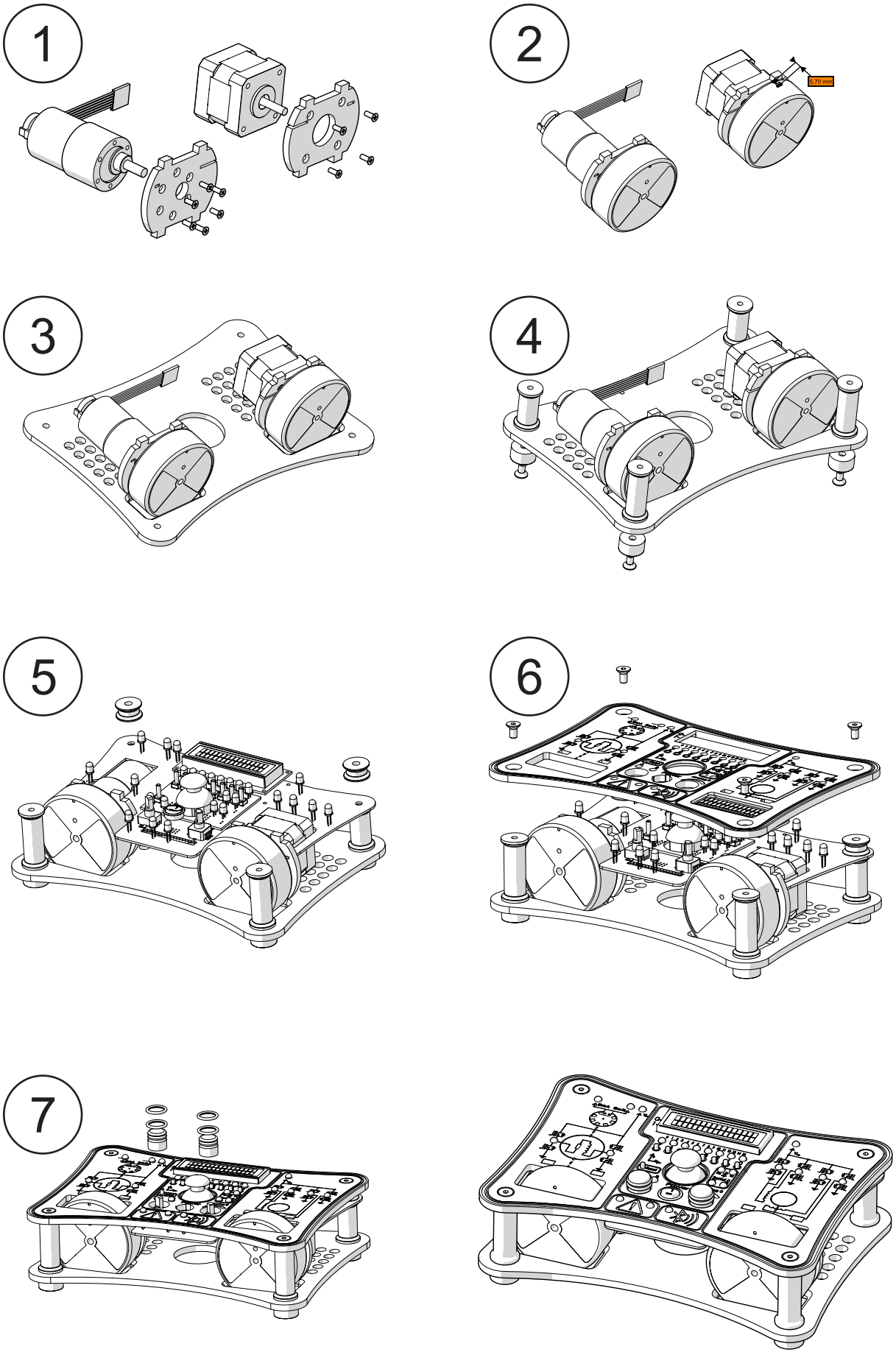


Figure 4.20: Assembly step-by-step

CHAPTER 5

Thermal dissipation test of the stepper driver

"Testing leads to failure, and failure leads to understanding."

– Burt Rutan

In order to verify the thermal calculations of the motor drivers, an experiment was done on the stepper driver to measure the real junction temperature and compare it with the calculated one.

5.1 Setup of the experiment

A program was written on the Arduino to activate both of the windings on the stepper-motor. The chopper threshold was set to 1A per winding. The program was uploaded to the Servo Lab unit, which was connected to a 12 V powersupply. 2 hours later the top panel was removed and a thermal picture was taken with a Fluke TiR32 IR-camera. The ambient temperature in the test environment was 23°C.

Program code 5.1: stepperMaxTest1A.ino

```
#include <dac.h>
void setup()
{
dac_init();
set_dac(4095,4095);
pinMode(66,OUTPUT);
pinMode(67,OUTPUT);
pinMode(68,OUTPUT);
pinMode(69,OUTPUT);
digitalWrite(66,HIGH);
digitalWrite(67,HIGH);
digitalWrite(68,HIGH);
digitalWrite(69,HIGH);
}
void loop()
{
}
```

Table 5.1: System data

$HS\_R_{DS(ON)}$	0.25Ω
$LS\_R_{DS(ON)}$	0.25Ω
$I_{OUT} - combined$	$2.0A$
V_m	$12V$
f_{SW}	$50kHz$
t_R	$200nS$
t_F	$200nS$
I_{VM}	$5mA$
$decay\ mode$	$fast$
$number\ of\ H-bridges$	2
$Ambient\ temperature$	23°C

5.2 Calculated driver temperature

The following data was collected for the system and the DRV8813 driver:

Using equation 3.1, 3.2 and 3.3 from the last chapter, the complete power dissipation of the driver is calculated to 1.3 watt. In last chapter the junction-to-ambient thermal resistance(θ_{JA}) was estimated to be about 37.5°C/W. With an ambient temperature of 23°C, the estimated junction temperature is calculated using equation 3.4 to 71.75°C.

5.3 Measured driver temperature

As seen on the thermal picture on figure 5.2 the temperature on the top of the IC is measured to 67.2°C. According to the thermal information in the data-sheet(figure 3.1) the "junction-to-top" characterization parameter(Ψ_{JC}) is 0.2 °C/W. With a power dissipation of 1.3 W, the junction temperature will be about 0.26°C warmer then the top of the IC. This gives an actual junction temperature in the step-motor driver of 67.5°C.

5.4 Thermal experiment conclusion

The thermal experiment shows that the conservative estimation of the driver temperature was correct. Since the thermal calculations of the dc-motor driver is mostly the same as for the stepper driver, the thermal experiment also confirms the temperature estimation of the DC-motor driver. The test also shows that the stepper-motor cannot handle much more than 1 A per phase without overheating.

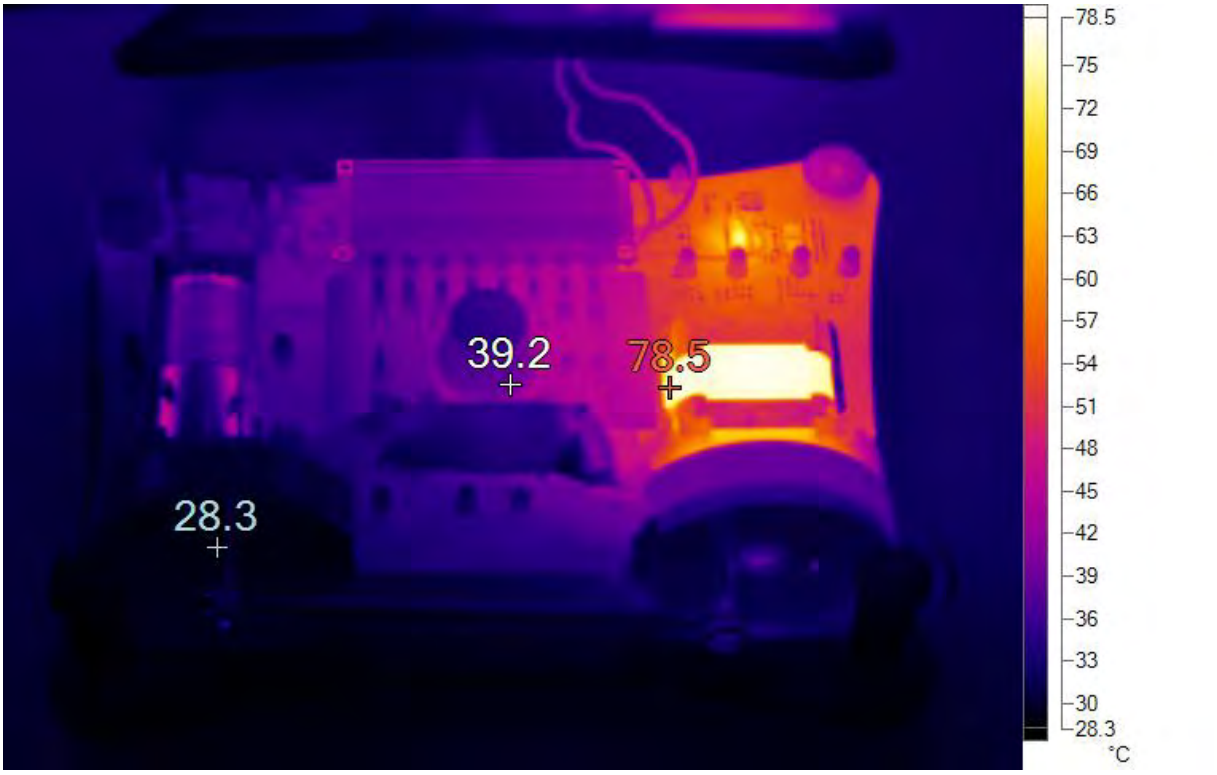


Figure 5.1: Thermal picture of Servo Lab

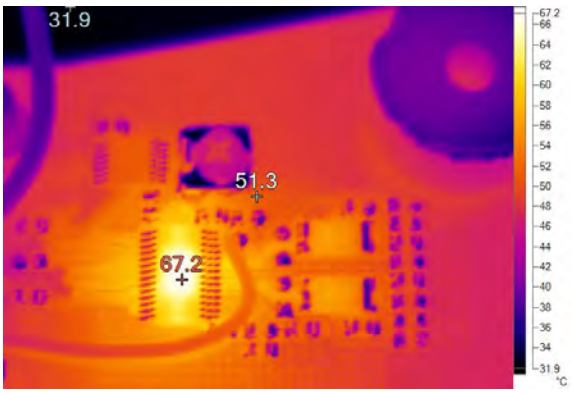


Figure 5.2: Th.pic. of step-motor driver IC

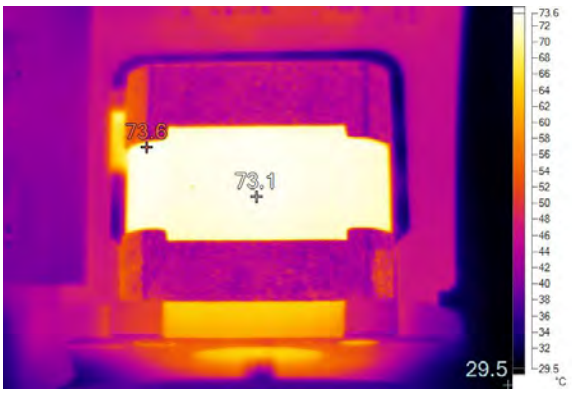


Figure 5.3: Thermal picture of step-motor

CHAPTER 6

Software development

"My favorite programming language is solder."
–Bob Pease

The software front-end, i.e. the software running on a computer connected to the Servo Lab, was developed in the Processing Integrated Development Environment (IDE). This is an open-source programming environment that uses the Processing programming language. Both language and software is based on the Java programming language, which means that the software will run on just about any platform. Furthermore, it is developed with graphic user interface (GUI) in mind, making it relatively easy to create a visual feedback of what goes on in the Servo Lab.

6.1 The interactive display-module

The preparation of the software modules started with simple figures that changed its size and relative position according to the joystick and potentiometer values. Using graphic editing software to create moving and changing elements, a representation of the current state of the Servo Lab was created. The module does not really do anything useful in terms of actuator control, but it demonstrates how sensor measurements can be transmitted to and interpreted on a computer.

Figure 6.1: All in one Servo Lab interactive display-module

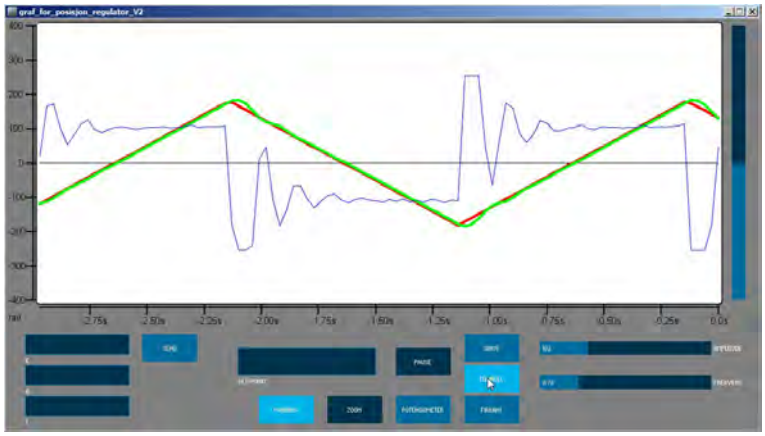


Figure 6.2: Screenshot of Lead/Lag controller HMI

6.2 DC-servo motor control and feedback

The more sophisticated servo control and feedback modules has the ability to display position, speed and control effort of the DC-motor. This is done graphically in the time domain, giving the user an impression of the time response of the servo-system. Moreover, the user can adjust the set-up of the lead-lag control system to optimise performance of the system. Finally, the user can set various periodic functions to the servo-system that is crucial to interpret the system and optimise it. To adjust the parameters of the lead-lag controller the user can chose to enter the K, a and T variables for the compensator, or enter the A, B and C values for the discrete controller directly6.3. The control system operates with a 1 µs sampling-time. When the set point is set as a continuous function, a new set point is calculated every 2 µs. Data is uploaded and plotted on the real-time graph every 30 ms. Figure 6.2 shows a screen-shot of the Human-Machine Interface (HMI). The red line shows the set point, the green line shows the actual position of the motor and the blue line shows the motor-driver control effort.

6.3 Implementing a Lead/Lag controller on the Arduino

The block diagram of an Lead/Lag, position regulator is showed in figure 6.3.

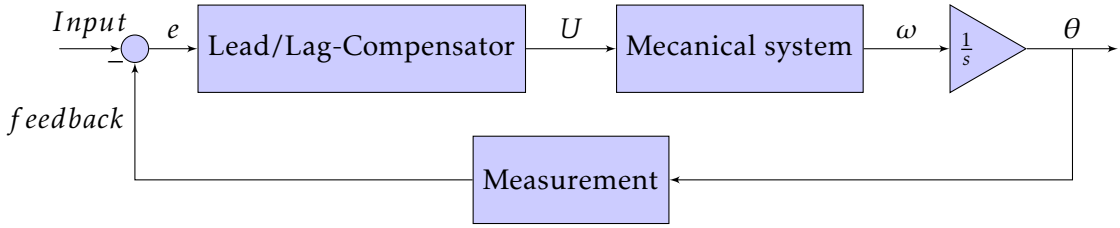


Figure 6.3: block diagram of a positioning servo system

The transfer function of the compensator can be written as:

$$\frac{u(s)}{e(s)} = K\frac{a \times Ts + 1}{Ts + 1} \tag{6.1}$$

A simplified transferfunction of the mechanical system can be written as:

$$\frac{\omega}{u(s)} = \frac{\frac{1}{K_e}}{\tau_m s + 1} \tag{6.2}$$

where K_e is the motor voltage constant and τ_m is the mechanical time constant of the system. Filling inn all the known values, gave the block diagram shown in figure 6.4. The position

feedback signal from the motor-encoder gives an output of 64 "ticks" per revolution. By multiplying this with $\frac{\pi}{32}$ the feedback signal is converted to radians.

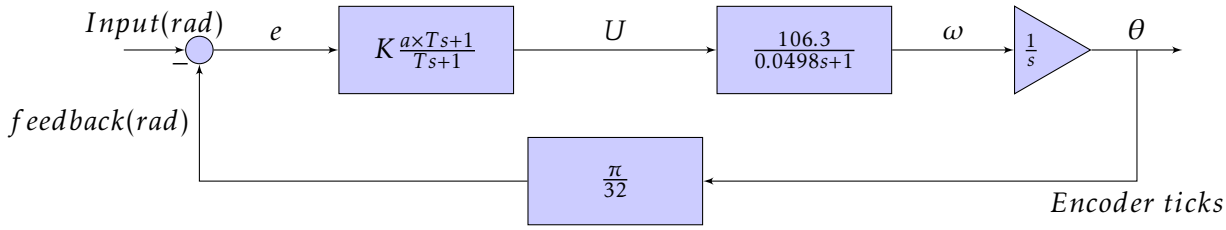


Figure 6.4: block diagram for the actual servo positioning system

In order to be implemented on a microcontroller, the transfer-function of the compensator needs to be converted from the frequency-domain to the discrete time-domain. The transfer function in equation 6.1 can be converted to the following equation:

$$u_n = A \times u_{n-1} + B \times e_n + C \times e_{n-1} \tag{6.3}$$

Where u_n and u_{n-1}, is the present and previous voltage applied to the motor. e_n and e_{n-1} is the

$$A = \frac{\frac{T}{\Delta t}}{(1+\frac{T}{\Delta t})} \qquad B = K\frac{(1+\frac{aT}{\Delta t})}{(1+\frac{T}{\Delta t})} \qquad C = -K\frac{\frac{aT}{\Delta t}}{(1+\frac{T}{\Delta t})}$$

present and previous error between the set point and the actual value. Δt is the sampling time, which is determined by the microcontroller. Once the K, a and T value is found, the A, B and C value can be calculated and saved as constants on the Arduino. In the software, equation 6.3 calculates the voltage the system needs. The voltage is then constrained to ± the voltage of the power supply, to prevent wind-up of the controller. Then the voltage is converted to a PWM-signal using equation 6.4 and sent to the motor-driver.

$$PWM_{duty} = u_n \times \frac{PWM_{resolution}}{u_{Powersupply}} \tag{6.4}$$

[22] [23]

CHAPTER 7

Development of laboratory assignments and documentation

"Great ability develops and reveals itself increasingly with every new assignment."

– Baltasar Gracian

In the project assignment [B.1], some laboratory assignment keywords were mentioned. This was regarded as a minimum, additional assignments would be desirable. It also became clear that simple programming tasks would be preferable as a starting point for laboratory work.

7.1 The programming assignments

General programming is essential in getting to grips with the All in one Servo Lab. The Arduino IDE was chosen as the programming environment. This software already has basic samples included that are valid also for Servo Lab. With just a few adjustments to the selected pins, the students will be able to make things blink and react to pushing and turning buttons.

The level of complexity is gradually increased, but there has been a general idea of reusing code and methods from the preceding assignments. Likewise, assignments make use of Arduino-functions where possible. The alternative is to use native C or C++. That would in turn implicate that the solution would be more complicated and documentation from the online Arduino resources would be none or at least scarce. On the other side, C/C++ is more effective, and, in most cases more relevant for real-world applications. With regard to the course outcome, it is a question of what the outcome should and could be. The limiting factors are term length, number of lectures and laboratory hours per week. It was therefore decided that laboratory assignments should rely on Arduino Sketch.

7.2 Servo-related programming assignments

The first actuator assignment takes on the DC-servo motor. Step-by-step, the students will design motor control software with increasing sophistication. The assignment provides useful hints to guide the students in the right direction, and apply techniques that has been used in earlier assignments.

Next up is the control of the stepper-motor. Because of the relative complexity of programming micro-stepping and different ramping methods, much of the assignment rely on attached code. The students are expected to complete the code and make the necessary adjustments. Therefore, the stepper-motor assignment is an achievable transition from basic input/output control to actuator control.

Finally, the students face the task of applying lead-lag control to the DC-motor. It is necessary to use calculations that are a part of servo-technology to design and optimise the controller. However, it is regarded to be to complicated to write all the code needed to create a software controller. A basic framework for the software controller is therefore supplied with the assignment. Nevertheless, understanding how the program works is needed to complete the task.

7.3 Document layout and functionality

Great consideration was taken in regard to the layout of the laboratory assignments. In order to clarify what was pure code and commands, the listings-package was used in LaTeX. Because there is no defined Arduino Sketch-dialect defined in listings, this had to be created. That enabled the use of in-line styling of programming code, e.g. **`int varint = 100`**, using the \inlineino-macro.

Being able to distinctly describe programming code made the process of writing the documents a lot easier, but there was also necessary to insert larger portions of programming code. To fully understand how a program or function works, it needs the full context. Therefore, further adaptations to the listings package was implemented. As a result, Arduino-sketches can be directly linked into the document without hassle, a macro called \inoblock. Because it can split the code up in multiple blocks while still have the correct line numbering, it ensures that there is coherence between the actual code as seen in the programming environment and the document. It is also styled in a manner which is very close to the default style of the Arduino IDE. Although it has not been used, similar configurations has been used for Processing-code.

```
18 // the loop routine runs over and over a
19 void loop() {
20   digitalWrite(led, HIGH);   // turn the
       voltage level)
21   delay(1000);               // wait for
22   digitalWrite(led, LOW);    // turn the
       voltage LOW
23   delay(1000);               // wait for
24 }
```

Figure 7.1: Styled programming block

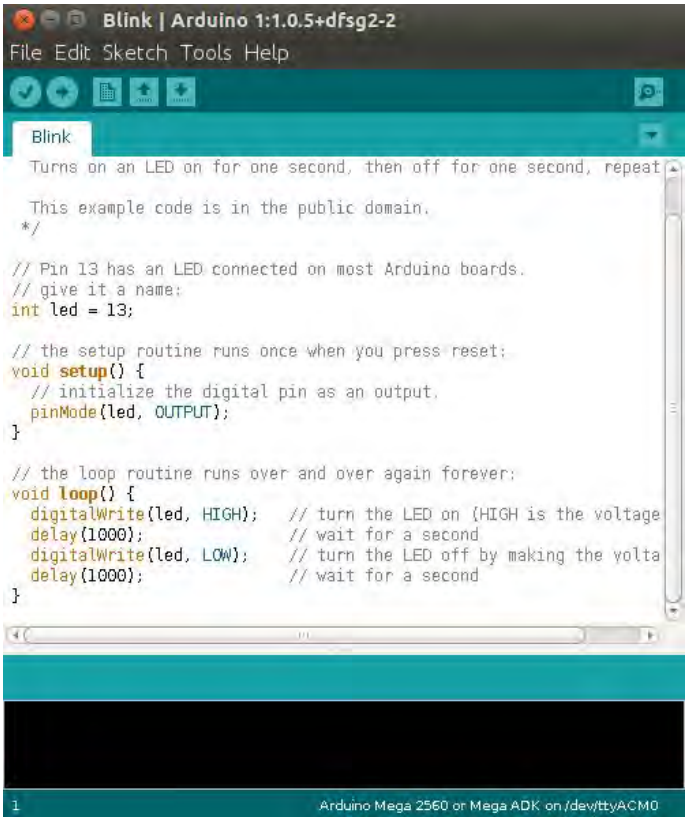


Figure 7.2: Arduino IDE, blink program

CHAPTER 8

Discussion

"Obstacles are those frightful things you see when you take your eyes off your goal."

–Henry Ford

The preceding chapters describe how the project was completed. However, it is useful to look at other possible solutions. Some of these has already been described in chapter 2. How would different a different approach affect the product and the project? In this chapter, these opportunities are evaluated against the chosen solution.

8.1 Motor-drivers

The decision to design and build a stepper driver circuit from scratch was necessary to meet the project requirements. Off-the-shelf devices was considered, but they lacked the functionality to limit the current and had no chopper drive. Another consequence of using Arduino-shields was the predefinition of hardware pin-mapping and the physical layout itself. Tables 8.1 and 8.2 compares the designed solution with the motor-drivers that was acquired by the university in 2013.

Table 8.1 shows that in regard to performance, reliability and feedback functionality, the All in one Servo Lab would be preferable choice. The downside is of course that it relies on custom parts, and by that making the unit harder to retrofit or repair. Furthermore, the fact that micro-stepping was a prerequisite disqualifies both commercially available alternatives.

The DC motor driver integrated circuit was selected around the same criteria as the stepper motor integrated circuit. In addition to the chopper drive that limits the current to a safe level, there is also implemented a hall effect current sensor in line with the DC motor winding to measure the true current by excluding the flyback current from the H-bridge, neither the Arduino motor shield nor the DFRobot shield have this feature of "true" current sensing.

Clearly the decision to build from scratch directly influences the workload. On the other hand, relying on commercially available devices would lead to lacking functionality and constrained design. Most of all, the use of the already acquired devices would mean that the Servo Lab would not comply with the requirements.

Table 8.1: Comparison of stepper motor-driver alternatives

	All in one Servo Lab	Arduino Motor Shield	DFRobot motor shield
Microstepping	Yes	No	No
Power disappation@1A load	0.5W	2.55W	2.55W
Driving transistors	N-channel MOSFETs	Darlington	Darlington
Chopper drive	Yes	No	No
Current sensing	Yes	Yes	No
Over-current protection	Yes	No	No
Fault indicator	Yes	No	No
Over temperature protection	Yes	Yes	Yes
Motor overheat protection	Optional	No	No
Driver type	DRV8813	L298	L298
Voltage range	8.2-35V	5-12V	4.8-35V
Format	Integrated	Shield	Shield
Flexible pin configuration	No	No	No
Note: Comparison table are limited to the shields acquired in 2013. Other commercially shields may have extended functionality			

Table 8.2: Comparison of DC motor-driver alternatives

	All in one Servo Lab	Arduino Motor Shield	DFRobot motor shield
Power disappation@1A load	0.13W	2.55W	2.55W
Driving transistors	N-channel MOSFETs	Darlington	Darlington
Chopper drive	Yes	No	No
Current sensing	Yes	Yes	No
In-line current sensing	Yes	No	No
Over-current protection	Yes	No	No
Fault indicator	Yes	No	No
Over temperature protection	Yes	Yes	Yes
Motor overheat protection	Optional	No	No
Driver type	DRV8840	L298	L298
Voltage range	8.2-35V	5-12V	4.8-35V
Format	Integrated	Shield	Shield
Flexible pin configuration	No	No	No
Note: Comparison table are limited to the shields acquired in 2013. Other commercially shields may have extended functionality			

8.2 Functionality and design

The All in on Servo Lab have been designed around the idea of being robust and at the same time have an appealing design. The decision to sandwich the electronics and motors between two plates of plastic makes the design both aesthetically pleasing and at the same time acting as an outer shell to provide structural integrity to the motor mounts, LEDs, switches and all of the internal electronics. With the shield approach there is no level of protection for the electronics, there is no external chassis and there is no method for mounting the motors.

The high level of user feedback incorporated into our design gives the user a meaningful and easy way to understand the system as it shows both the graphical representation of the motors in addition to the direction and magnitude of the current passing through the motor. The way all of this information is passed on to the user would be problematic perhaps even impossible to represent by only using off the shelf components.

The design also enable the user to get a hands on feel of the motors by putting disturbance to the flywheels as these are directly accessible on the front panel. The shield approach used in the laboratory assignments up until now relied on the motors being unsupported on the workbench.

As stated in the preceding section, the decision to build the motor-driver circuits from scratch opened up the design possibilities. It would be feasible to build a functional Servo Lab using available Arduino-shields. However, it would lack functionality, and most probably the design would be flawed. Because of the characteristics of the Arduino-shields, reconfiguration of the pin-mapping would be necessary. This could be achieved by cabling, further complicating the design.

8.3 Software and written material

It was a prerequisite that the Arduino IDE should be used as the primary programming environment. Also, the functions used in the assignments and solutions should be Arduino Sketch. In regard to the computer front-end software, any software was applicable. When it was decided to use Processing, this was due to the following characteristics:

- Arduino IDE is based on Processing IDE, making the transition easier for future use and development
- Processing is based on, and runs within, the Java runtime environment. This makes the software run on more or less any platform.
- Processing is designed with the purpose of effective graphical programming.
- It shares many programming similarities with Arduino Sketch.

Alternatives to Processing are programming languages like Visual Basic, C in various forms and Java to name a few. With the exception of Java, these alternatives needs to be compiled for each operating platform operating system (and sometimes for different versions). Java would be a good candidate, but with no experience with Java-programming, Processing was the preferred solution.

The assignments and documentation is a considerable part of the project. To develop this material, it was needed to have a document processor that could handle collaborate work to some extent. LaTeX has been recommended for writing technical reports and thesis. Even though it is a flexible, highly configurable way of writing documents, much is to be desired when it comes to user friendliness. Microsoft Word is on the other hand very easy to master, but has many

shortcomings in regard to strict and consistent layouts and flexible input methods. In regard to future development of the laboratory assignments, Word would probably be the preferred editing platform.

CHAPTER 9

Conclusions

*"It's more fun to arrive a conclusion
than to justify it."*

– Malcolm Forbes

How does the All in one Servo Lab measure up to the expectations of the project owner, the University of Agder? And has the project answered the problems stated in chapter 1? This is the final chapter in the report, and conclusions as to how the outcome of the project fulfils the project assignment are drawn and the need or possibility for further work is assessed.

Problem Statement 1 : ***The All in one Servo Lab must have the capability to control a stepper motor with full-, half- and micro-stepping.***

Problem Statement 2 : ***The All in one Servo Lab must be able to control position, speed and torque of a DC-motor through pulse-width-modulation and quadrature encoder.***

The decision to design and build the motor-drivers from scratch, with all the implications that followed, had a great impact on the workload of the project. However, the motor-drivers does provide correct operation and correct feedback, which really is the point of the Servo Lab. All the functions of the motor-drivers has been tested, and the performance and results are as expected.

Problem Statement 3 : ***It is desirable with a unit that has Bluetooth- and wired communication capabilities.***

Problem Statement 4 : ***The physical user-interface must be designed with clarity and simplicity in mind.***

The Servo Lab provides communication through wire and Bluetooth that can be used for transferring signals and data between units or to a computer. The wired communication is native to the Arduino-platform, and the wireless communication is based on a commercially available module. This solution ensures ease of use and configuration, and a substantial amount of online examples and tutorials.

In regard to the physical user-interface, the All in one Servo Lab's significantly surpasses the project outline. It is configured in a manner that makes programming easier, more understandable and provides many different functions. Although the interfacing is a secondary feature,

it does provide a basic and accessible introduction to microcontroller-programming. This is in many cases vital when it comes to understanding how hardware programming works.

Problem Statement 5 : ***Quality, robustness and aesthetics must be a priority in the look and feel in the product design.***

The aesthetic design has been a priority from the very start of the product development. It has a level of detail and careful consideration in every aspect and stage of the design process that hopefully will respond well with the future users. Keywords in the design process has been *simplicity*, *robustness*, *aesthetics*. Measuring these properties is possible would be possible by performing studies and getting feedback from users. But with the limitations of the project, especially in regard to time, this was not feasible.

Problem Statement 6 : ***The product must be delivered with relevant and achievable laboratory assignments, as well as examples, solutions and user manuals.***

The laboratory assignments that has been developed for the project are within the requirements for the project. It has also been created assignments that exceed the requirements. However, more effort could have been assigned to this part of the project in order to have them ready for print at the deadline for the thesis. Moreover, the solutions and complete user manual are still in development which clearly is not optimal.

The effort that has been put into the document layout and configuration will hopefully be an asset when changing code, adding or editing the documentation in the future. Furthermore, the direct input of code ensures an easy way of updating documents and source-code when needed.

Problem Statement 7 : ***There should be software accompanied with the product, that gives extended feedback and demonstration capabilities.***

The software that has been developed for the project do comply with the requirements. However, as pointed out in chapter 8, more and more advanced software could have been provided. But the software that accompanies the Servo Lab shows some of the possibilities then lie in the Arduino-Processing environments. In that regard, the project has provided a foundation for further development.

Problem Statement 8 : ***A class set of 25 fully assembled units must be delivered upon the conclusion of the project.***

As stated in the preceding chapter, the parts are still in production when this thesis is due. There is always the possibility of errors in the manufacturing process, and there is a small chance of errors in the design. However, this is mitigated by the functionality testing of fully featured prototype mark II. The design changes from prototype to the final design were minimal, so there should be no flaws in the final version. There is therefore confidence in the ability to hand over the full class set of the All in one Servo Lab at the end of the term.

9.1 Further work

When this report is published, the complete class set of All in one Servo Lab is yet not assembled and delivered. However, all the parts and components are in-house or en route. Anyhow, this implies that further work in the project is still required. But there is great confidence in the ability to hand over the units by the time the project is to be presented.

In the preceding section, it was stated that there could be a more sophisticated and advanced software back-end. Throughout the project, the functionality and possibilities with the Processing development environment was discussed. However, the realization of the Servo Lab

had to be top priority. Additional software-modules and adaptation of the existing modules will benefit from further work.

An optional accessory in form of a ball-track which could be mounted on the Servo Lab was also planned. An optical range sensor should be used to monitor the position on the ball on the track. The track could then be tilted to either side by the stepper or DC-motor, trying control the position and motion of the ball.

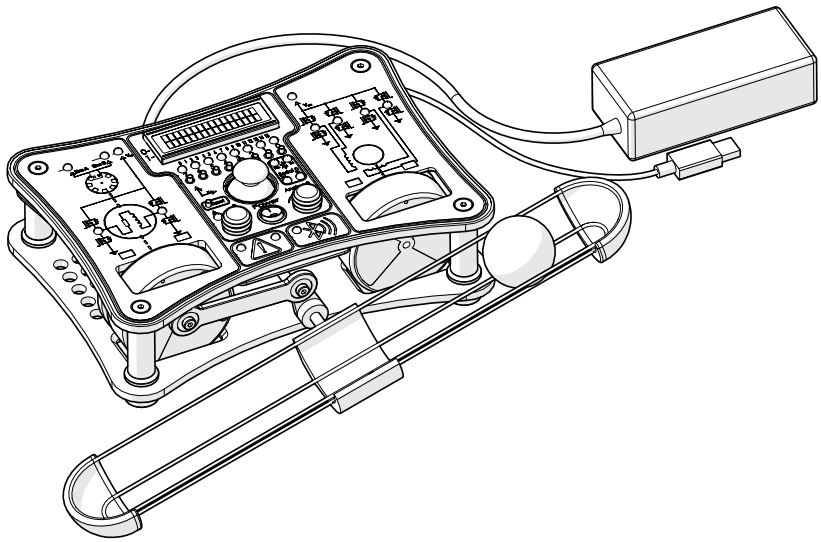


Figure 9.1: Optional ball-track accessory for the Servo Lab

At the start of the project, or even before, the level of ambitions were high. There was a general idea that a successful project could pave the way for similar projects at the university. The All in one Servo lab does deliver according to the project specifications, and more. Finding more ways to make use of the Servo Lab is up to the lecturers, staff and students at the university. Moreover, the project does show that designing customized products for laboratory work is viable practically and economically.

Therefore, the university is challenged to deploy more bachelor projects like the All in one Servo Lab in the future. Courses like control systems engineering, mechatronics and electric circuits to name a few, would most likely benefit greatly from dedicated laboratory equipment.

Bibliography

[1] T. Rohde, "The straight dope on mechatronics." http://www.yaskawa.com/site/products.nsf/staticPagesNewWindow/mechatronics.html. Accessed: 18 May 2014.

[2] Wikimedia/Ahm2307, "Mechatronics diagram." http://en.wikipedia.org/wiki/File:Mecha_workaround.svg. Accessed: 18 May 2014.

[3] M. D. licker et al., *Dictionary of Engineering*. McGraw-Hill, 2003.

[4] DFRobot, "2a motor shield for arduino." http://www.dfrobot.com/image/cache/data/DRI0009/_IGP6780-600x600.png. Accessed: 18 May 2014.

[5] Arduino.cc, "Arduino motor shield." http://arduino.cc/en/uploads/Main/MotorShield_R3_Front.jpg. Accessed: 18 May 2014.

[6] Linksprite, "16 x 2 lcd keypad shield for arduino." http://linksprite.com/wiki/index.php5?title=File:1602_lcd_keypad_shield-02.jpg. Accessed: 18 May 2014.

[7] C. BUTTAY, "Structure of an h-bridge." http://en.wikipedia.org/wiki/File:H_bridge.svg. Accessed: 7 May 2014.

[8] P. Millet, *Calculating Motor Driver Power Dissipation*. 2012.

[9] T. Instruments, "A guide to board layout for best thermal resistance for exposed packages." Texas Instrument app note: AN-1520.

[10] T. Instruments, "Drv8840 datasheet." http://www.ti.com.cn/general/cn/docs/lit/getliterature.tsp?genericPartNumber=drv8840&fileType=pdf.

[11] T. Instruments, "An-1520 a guide to board layout for best thermal resistance for exposed packages." http://www.ti.com/lit/an/snva183b/snva183b.pdf.

[12] T. Instruments, "Drv8813 datasheet." http://www.ti.com.cn/general/cn/docs/lit/getliterature.tsp?genericPartNumber=drv8813&fileType=pdf.

[13] Microchip, "Mcp4922 datasheet." http://ww1.microchip.com/downloads/en/DeviceDoc/22250A.pdf.

[14] A. MicroSystems, "Acs712 datasheet." http://www.allegromicro.com. Accessed: 19 May 2014.

[15] Arduino.cc, "Arduino mega 2560." http://arduino.cc/en/Main/ArduinoBoardMega2560. Accessed: 9 May 2014.

[16] Arduino, "arduino-mega2560-r3-schematic.pdf." http://arduino.cc/en/uploads/Main/arduino-mega2560_R3-sch.pdf.

[17] O. Seniconductor, "Ncp1117 datasheet." http://www.onsemi.com/pub_link/Collateral/NCP1117-D.PDF.

[18] G. H. C. I. T. Co., "Hc-05." http://www.wavesen.com/probig.asp?id=10. Accessed: 12 May 2014.

[19] G. H. C. I. T. Co., "Hc-05 datasheet." http://wavesen.com/downloadDis.asp?id=22. Accessed: 18 May 2014.

[20] Lextronic, "Tinkerkit - pro kit for arduino." http://www.lextronic.fr/doc/produit/img1_22993_1326719613.jpg. Accessed: 18 May 2014.

[21] Hitachi, "Hd44780 dot matrix liquid crystal display controller driver." https://www.sparkfun.com/datasheets/LCD/HD44780.pdf. Accessed: 18 May 2014.

[22] M. Ottestad, "Mas220 forelesningsnotater." Lecture notes distributed at the University of Agder.

[23] M. Ottestad, "Lead/lag kompensator diskret." Not published.

Glossary

actuator *"A mechanism to activate process control equipment by use of pneumatic, hydraulic, or electronic signals(...)"*[3]

cascade compensation *"Compensation in which the compensator is placed in series with the forward transfer function. Also known as series compensation; tandem compensation."*[3]

cascade control *"An automatic control system in which various control units are linked in sequence, each control unit regulating the operation of the next control unit in line."*[3]

computer-aided design *"The use of computers in converting the initial idea for a product into a detailed engineering design. Computer models and graphics replace the sketches and engineering drawings traditionally used to visualize products and communicate design information. Abbreviated CAD."*[3]

computer-aided engineering *"The use of computer-based tools to assist in solution of engineering problems."*[3]

computer-aided manufacturing *"The use of computers in converting engineering designs into finished products. Computers assist managers, manufacturing engineers, and production workers by automating many production tasks, such as developing process plans, ordering and tracking materials, and monitoring production schedules, as well as controlling the machines, industrial robots, test equipment, and systems that move and store materials in the factory. Abbreviated CAM"*[3]

computer control *"Process control in which the process variables are fed into a computer and the output of the computer is used to control the process."*[3]

computer-controlled system *"A feedback control system in which a computer operates on both the input signal and the feedback signal to effect control."*[3]

control circuit *"A circuit that controls some function of a machine, device, or piece of equipment."*[3]

driver *"The amplifier stage preceding the output stage in a reciever or transmitter."*[3]

field-effect transistor *"A transistor in which the resistance of the current path from source to drain is modulated by applying a transverse electric field between the grid or gate electrodes; the electric field varies the thickness of the depletion layer between the gates, thereby reducing the conductance. Abbreviated FET"*[3]

flywheel *"A rotating element attached to the shaft of a machine for the maintenance of uniform angular velocity and revolutions per minute. Also known as balance wheel."*[3]

forward transfer function *"In a feedback control loop, the transfer function of the forward path."*[3]

integral compensation *"Use of compensator whose output changes at a rate proportional to its input."*[3]

integral control *"Use of a control system in which the control signal changes at a proportional to the error signal."*[3]

integral network *"A compensating network which produces high gain at low input frequencies and low gain at high frequencies, and is therefore useful in achiev-*

ing low steady-state errors. Also known as lagging network, lag network."[3]

lead compensation *"A type of feedback compensation primarily employed for stabilization or for improving a system's transient response; it is generally characterized by a series compensation transfer functions of the type*
$G_c(s) = \kappa \frac{s-z}{s-p}$"
where $z < p$ and κ is a constant.[3]

lead-lag network *"Compensating network which combines the characteristics of the lag and lead networks, and in which the phase of a sinusoidal response lags a sinusoidal input at low frequencies and leads it at high frequencies. Also known as lag-lead network."*[3]

light-emitting diode *"A rectifying semiconductor device that converts electric energy into electromechanic radiation. The wavelength of the emitted radiation ranges from the near-ultraviolet to the near-infrared, that is from about 400 to over 1500 nm. Abbreviated LED"*[3]

mechatronics *"A branch of engineering that incorporates the ideas of mechanical and electronic engineering as a whole, and, in particular, covers those areas of engineering concerned with the increasing integration of mechanical, electronic and software engineering into a production process."*[3]

metal oxide semiconductor field-effect transistor *"A field-effect transistor having a gate that is insulated from the semiconductor substrate by a thin layer of silicon dioxide. Abbreviated MOSFET(...)"*[3]

microcontroller *"A microcomputer, microprocessor or other equipment used for precise process control in data handling, communication and manufacturing."*[3]

Microprocessor *"A single silicon chip on which the arithmetic and logic funtions of a computer are placed."*[3]

reflowing *"Melting and resoldering an electrodeposited or other type of coating."*[3]

Servomechanism *"An automatic feedback control system for mechanical motion; it applies to those systems in which the controlled quantity or output is mechanical position or one of its derivatives (velocity, acceleration and so on). Also known as servo system."*[3]

Servo(motor) *"The electric, hydraulic, or other type of motor that serves as the final control element in a servomechanism. It receives power from the amplifier element and drives the load with a linear or rotary motion."*[3]

stepper motor *"A motor that rotates in short and essentially uniform angular movements rather than continuously; typical steps are 30, 45 and 90°; the angular steps are obtained by the ratchet and pawl mechanisms of stepping relays. Also known as magnetic stepping motor, stepping motor, step-servo motor."*[3]

surface-mount technology *"The technique of mounting electronic circuit components and their electrical connections on the surface of a printed circuit board rather than through holes."*[3]

top-down design *"A design methodology that proceeds from the highest level to the lowest and from the general to the particular, and that provides a formal mechanism for breaking complex process designs into functional descriptions, reviewing progress, and allowing modifications."*[3]

APPENDIX A

Drawings

This appendix contains the technical drawings of the All in one Servo Lab. The electric schematics are found in 4, as well as the attached CD-ROM.

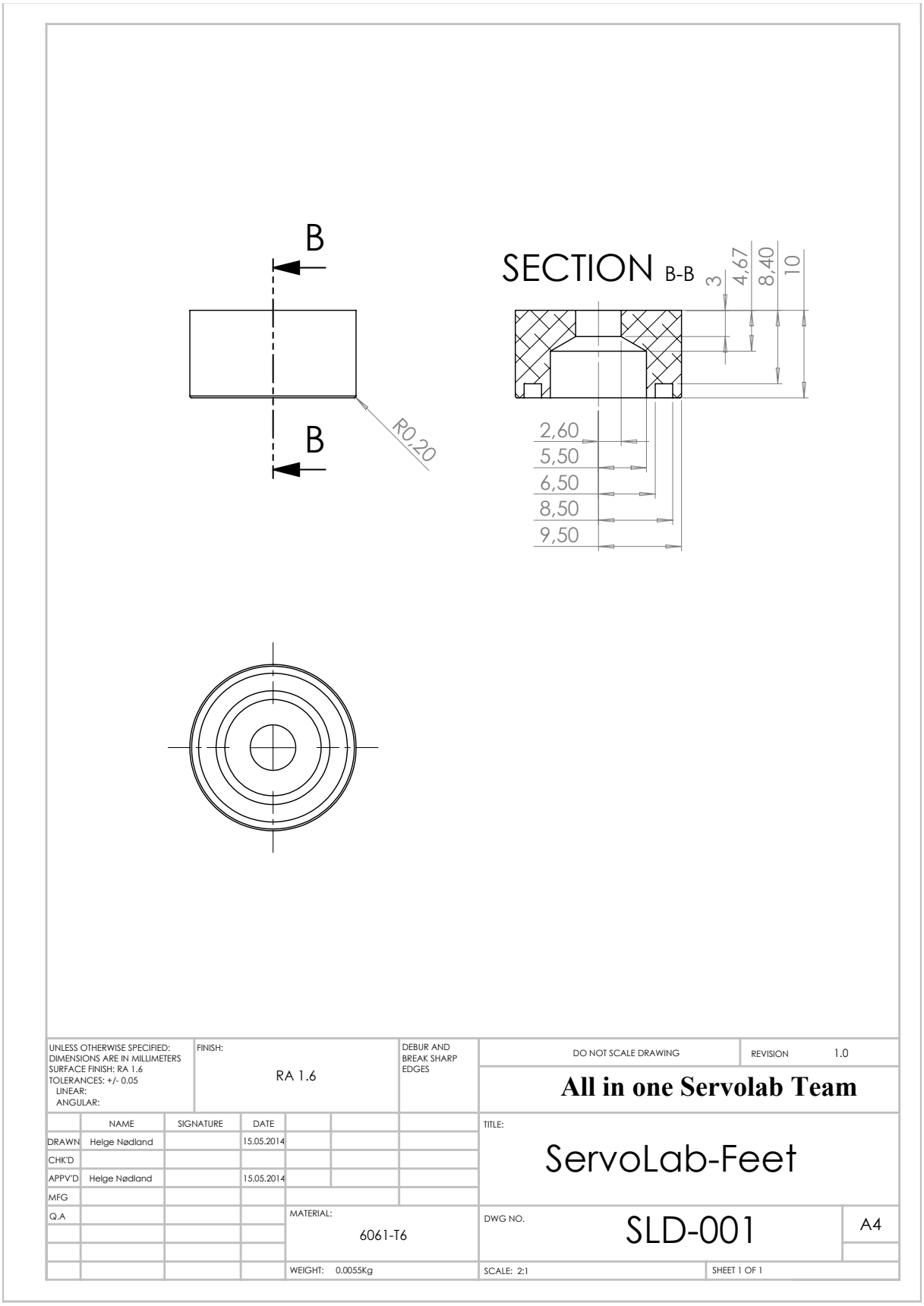


Figure A.1: All in one Servo Lab : Feet

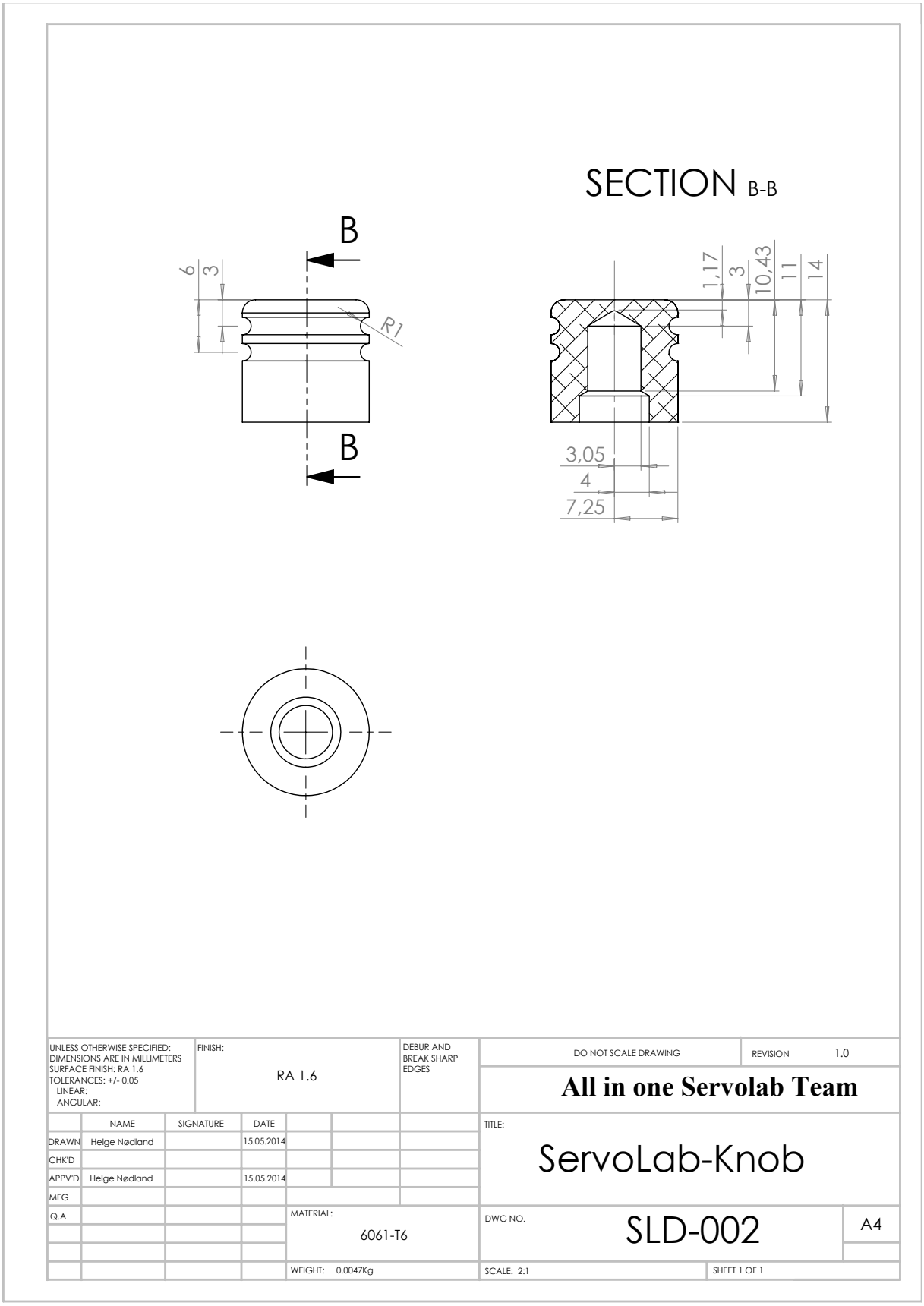


Figure A.2: All in one Servo Lab : Knob

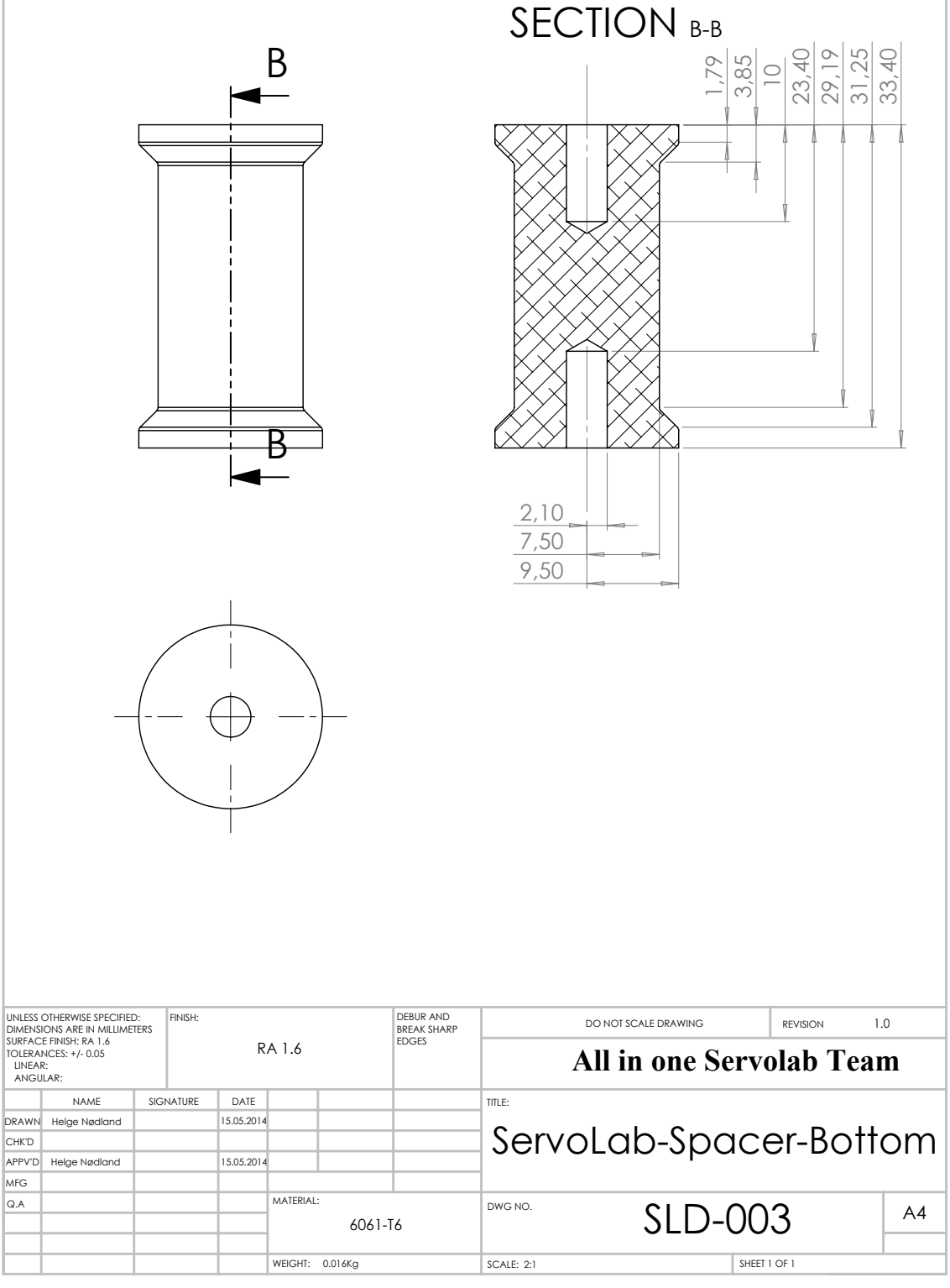


Figure A.3: All in one Servo Lab : 33.4 mm spacer

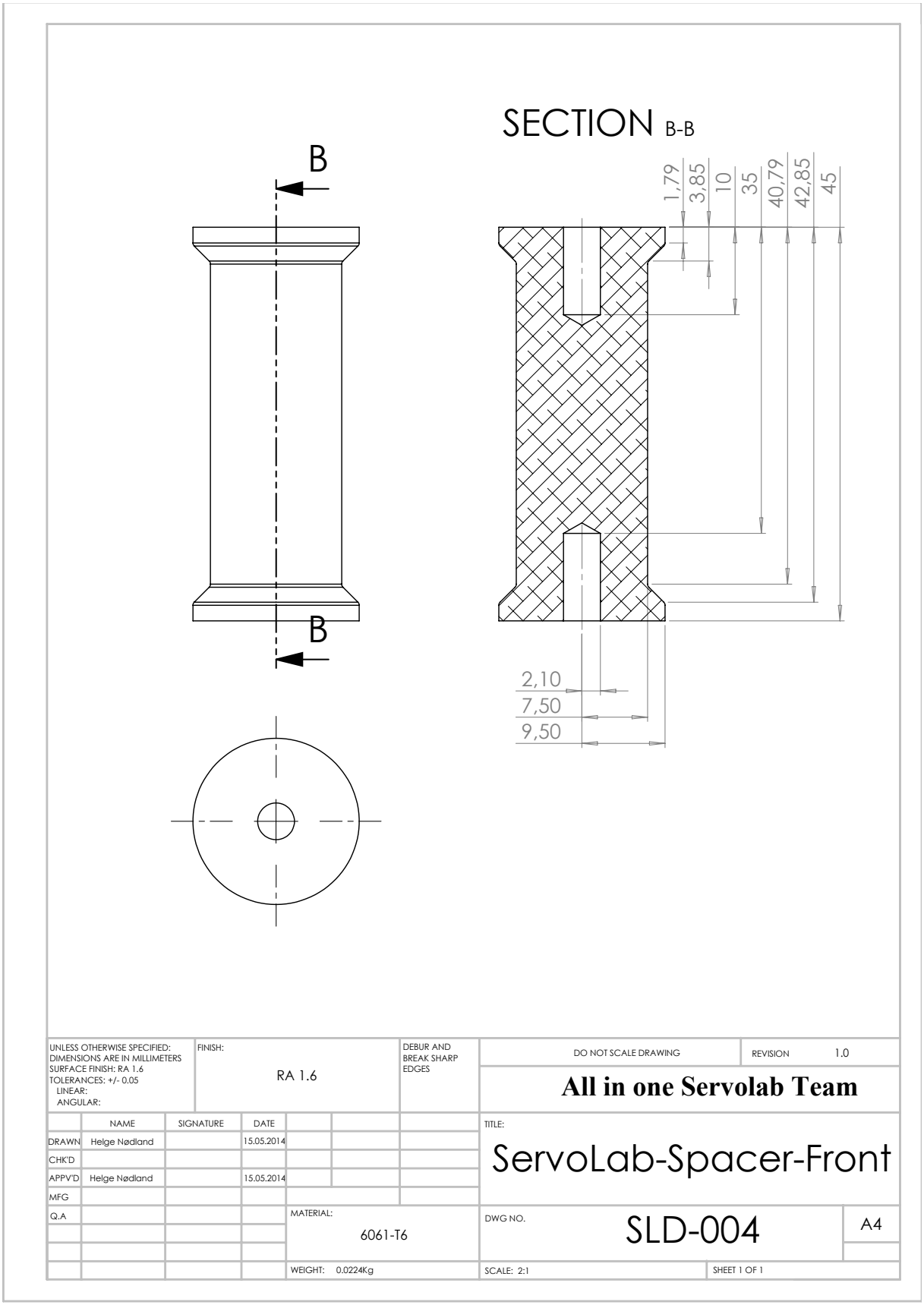


Figure A.4: All in one Servo Lab : 45 mm spacer

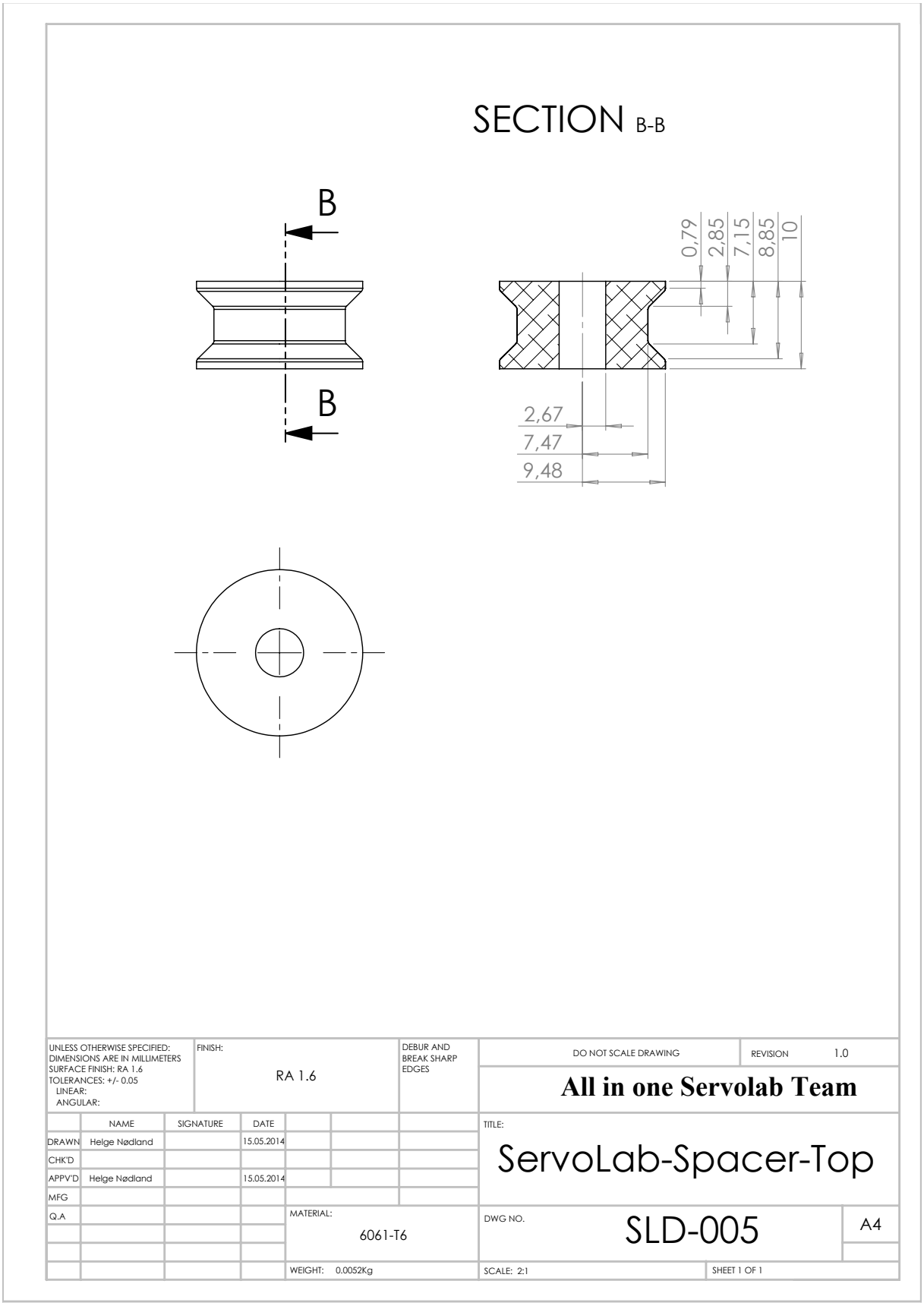


Figure A.5: All in one Servo Lab : 10 mm spacer

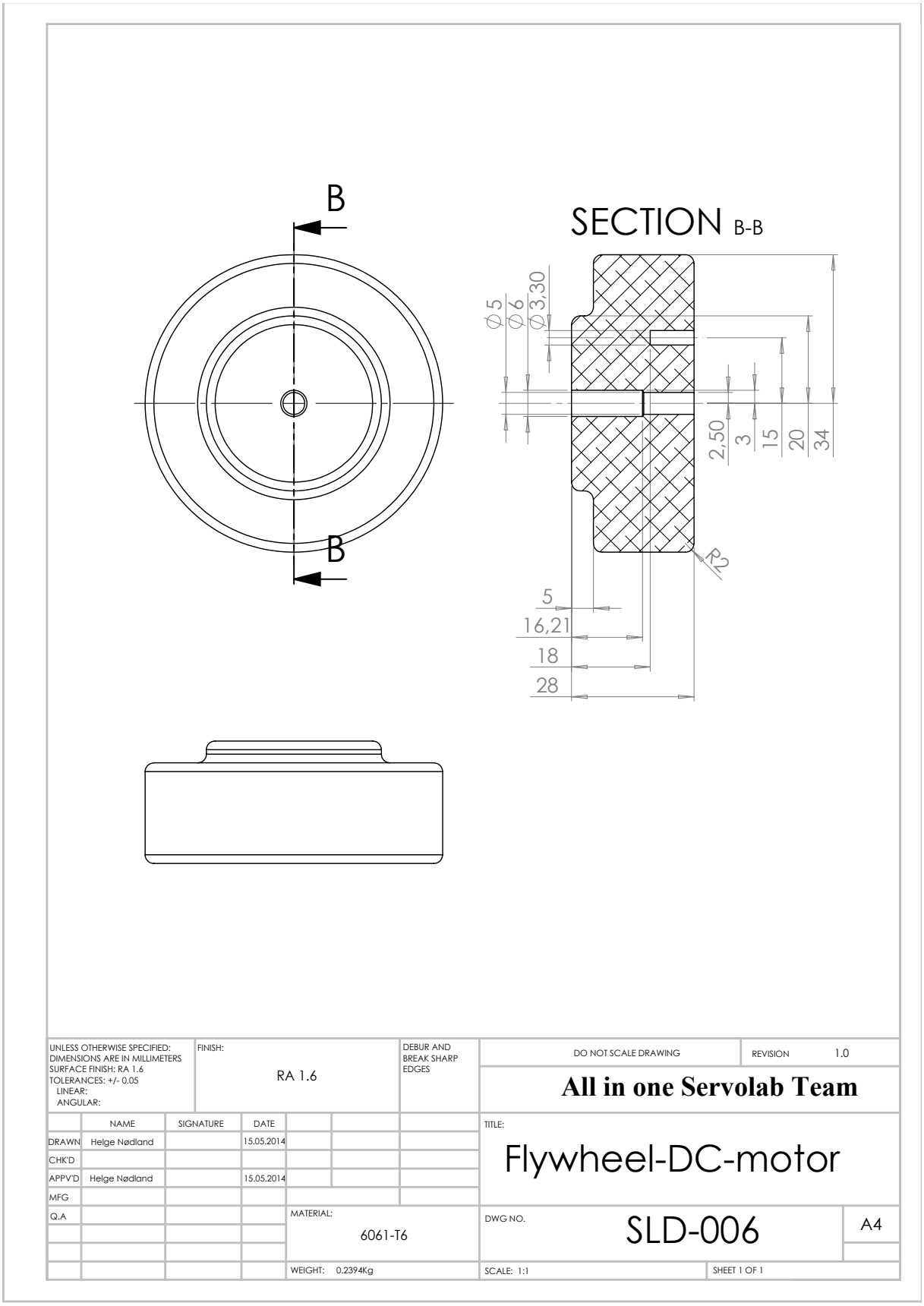
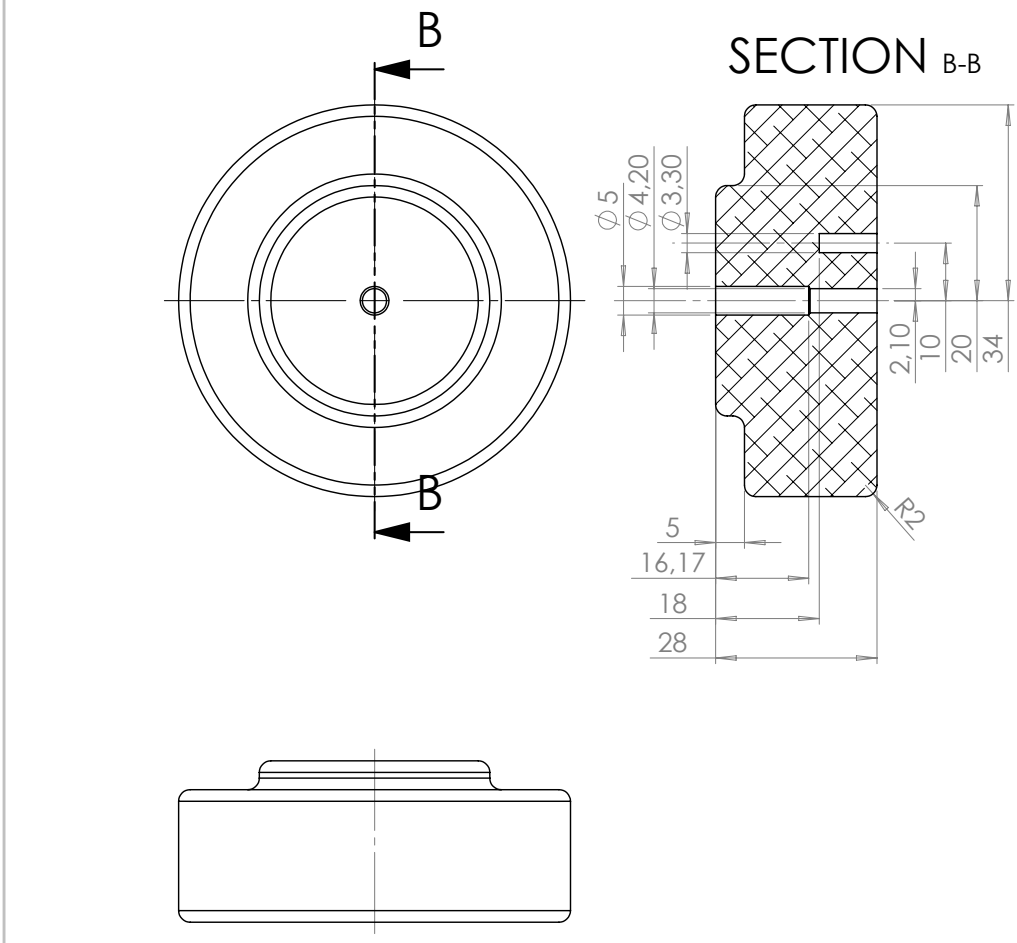


Figure A.6: All in one Servo Lab : DC-motor flywheel



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: RA 1.6 TOLERANCES: +/- 0.05 LINEAR: ANGULAR:	FINISH: RA 1.6				DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION 1.0
						All in one Servolab Team
	NAME	SIGNATURE	DATE			TITLE: Flywheel-Stepper-motor
DRAWN	Helge Nødland		15.05.2014			
CHK'D						
APPV'D	Helge Nødland		15.05.2014			
MFG						
Q.A				MATERIAL: 6061-T6		DWG NO. SLD-007 / A4
WEIGHT: 0.2400Kg						SCALE: 1:1 / SHEET 1 OF 1

Figure A.7: All in one Servo Lab : Stepper motor flywheel

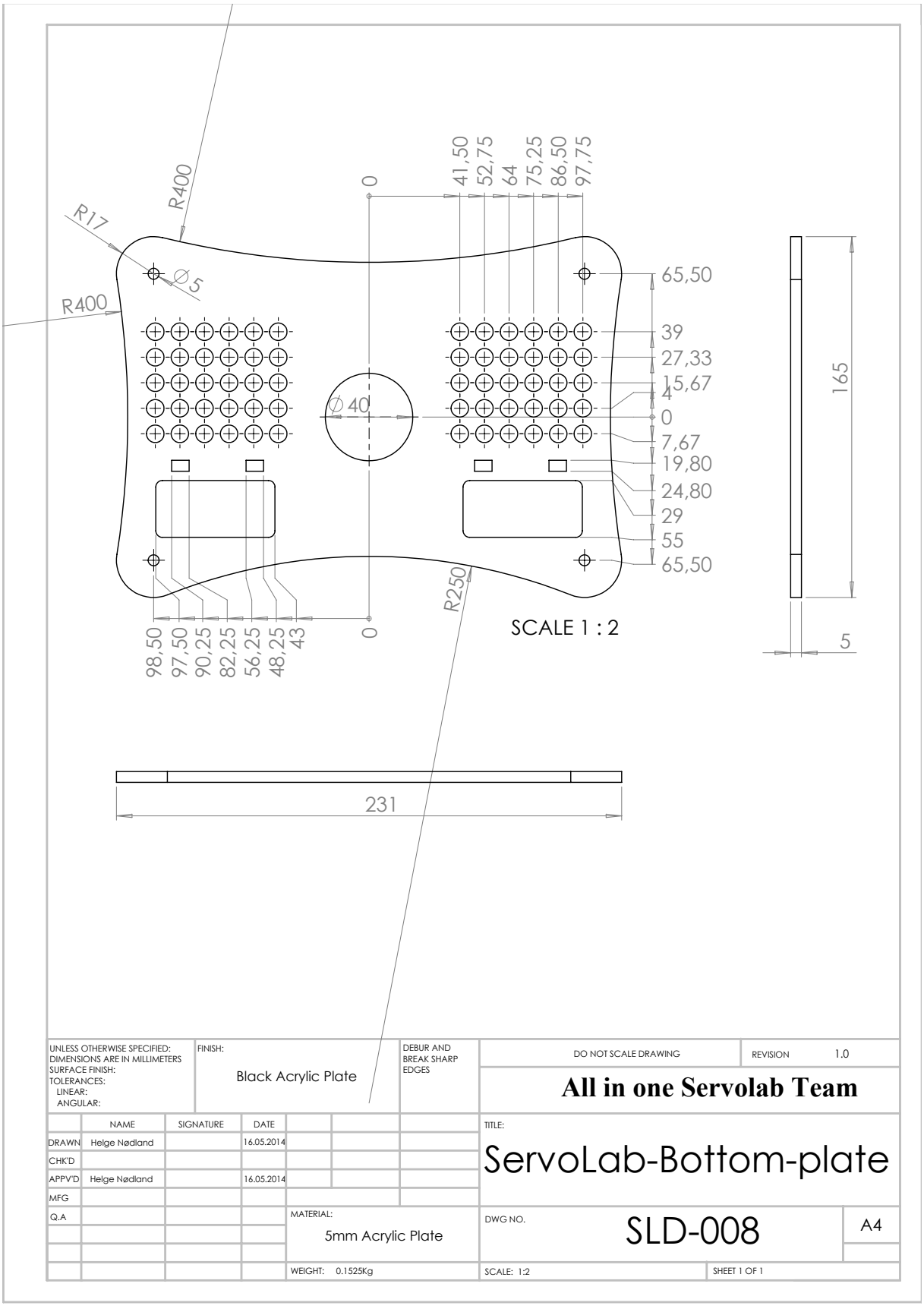


Figure A.8: All in one Servo Lab : Bottom plate

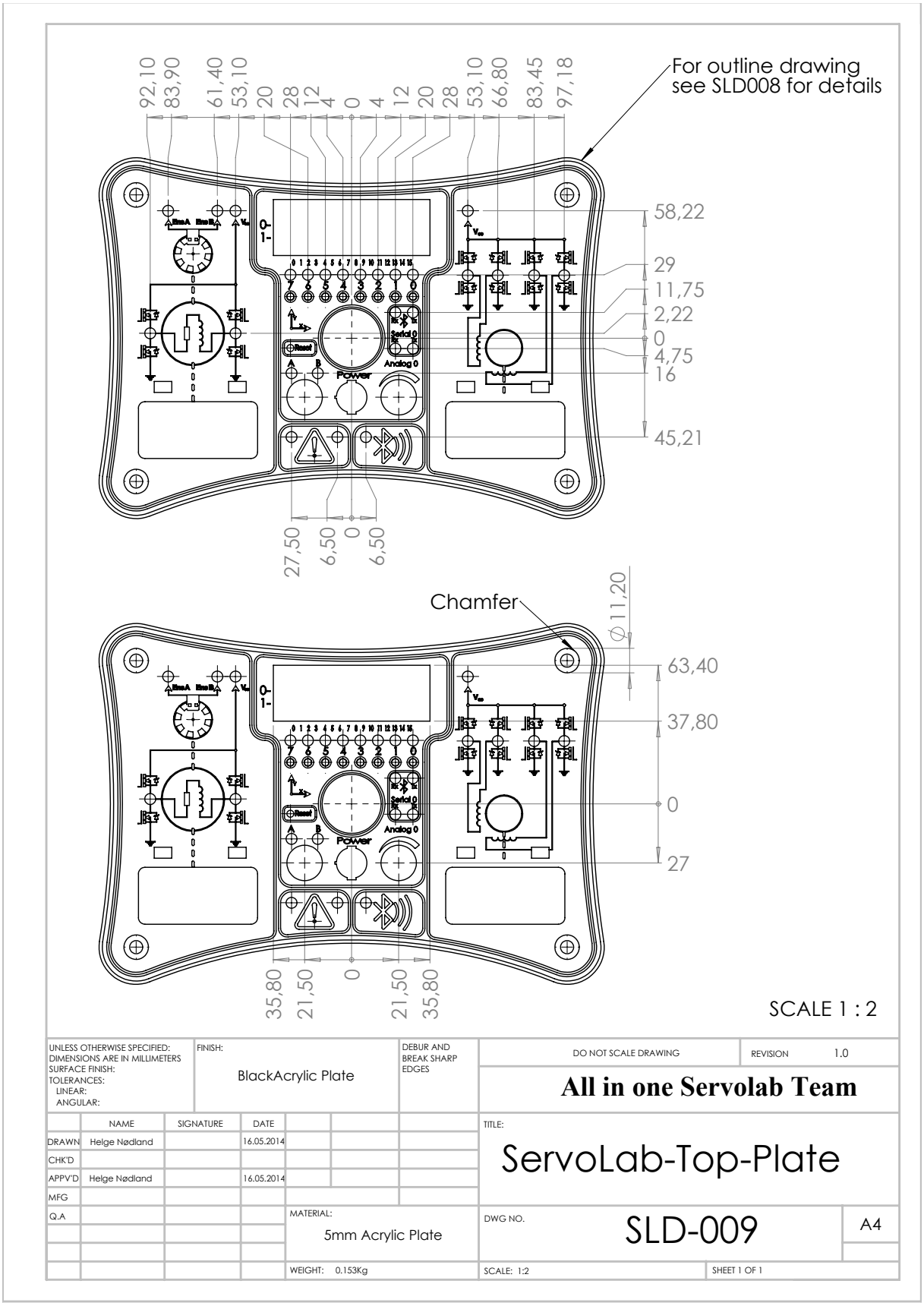


Figure A.9: All in one Servo Lab : Top plate

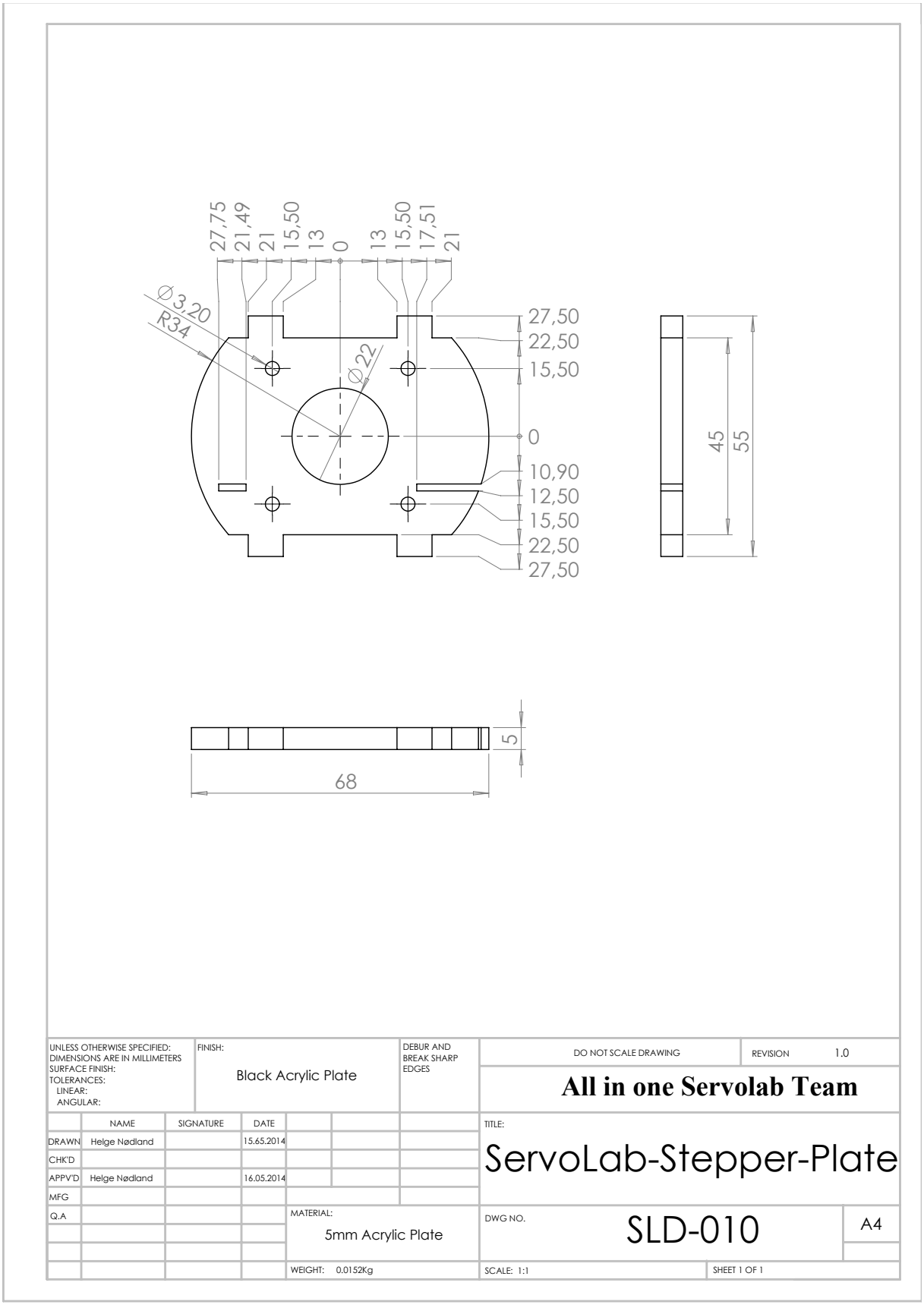
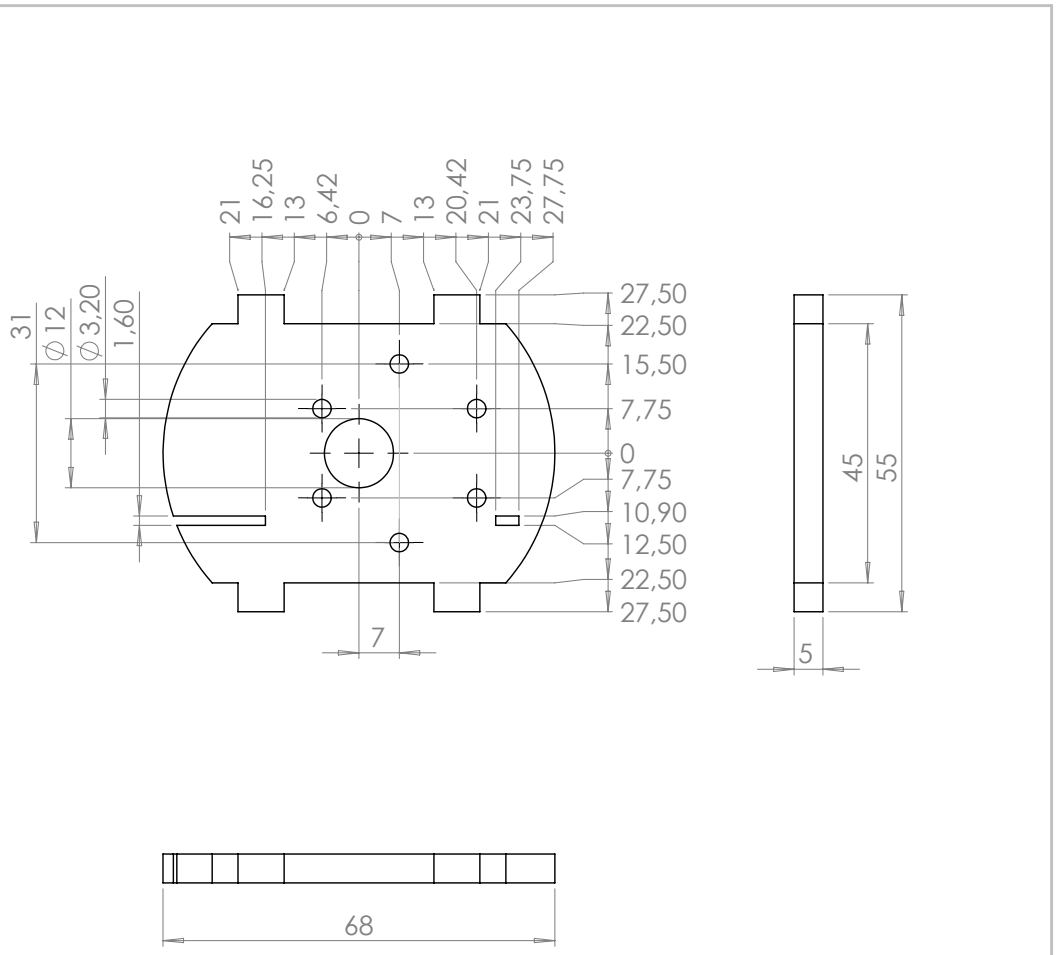


Figure A.10: All in one Servo Lab : Stepper motor mounting plate [not to scale]



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH: Black Acrylic Plate			DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION 1.0
						All in one Servolab Team
	NAME	SIGNATURE	DATE			TITLE:
DRAWN	Helge Nødland		16.05.2014			ServoLab-DC-Plate
CHK'D						
APPV'D	Helge Nødland		16.05.2014			
MFG						
Q.A				MATERIAL: 5mm Black Acrylic		DWG NO. SLD-011 / A4
				WEIGHT: 0.0167Kg		SCALE: 1:1 / SHEET 1 OF 1

Figure A.11: All in one Servo Lab : DC-motor mounting plate

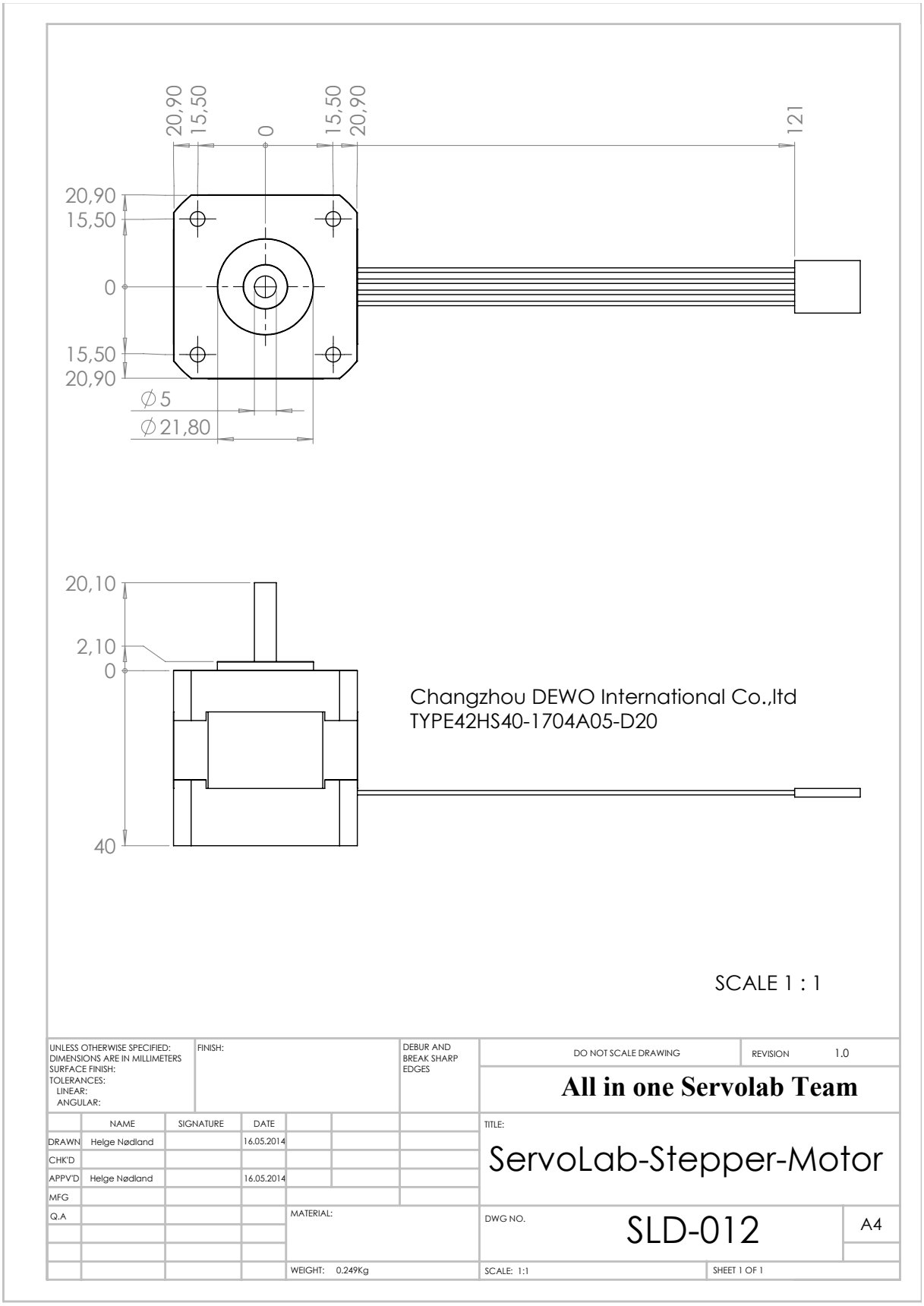


Figure A.12: All in one Servo Lab : Stepper motor

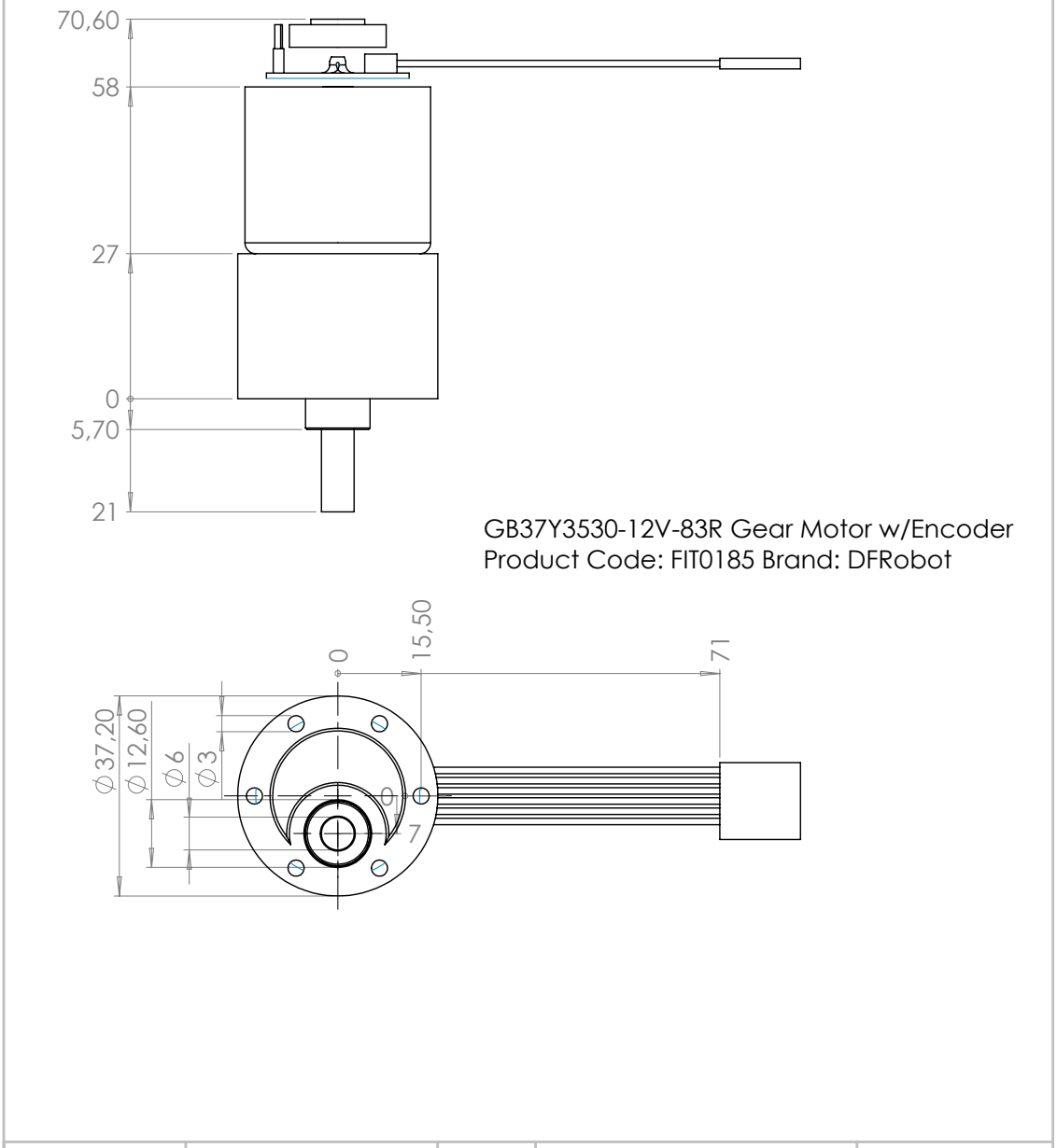


Figure A.13: All in one Servo Lab : DC-motor

APPENDIX B

Project description

Document B.1: Project assignment

Prosjektoppgave

All in one Servo Lab
Faget MAS220 omfatter blant annet mikrokontroller basert motor styring, der følgende styringsoppgave skal realiseres

- Turtallsstyring av servomotorer ved PWM
- Posisjonsstyring av servomotorer ved inkrementell enkoder og PWM
- Momentstyring av servomotorer ved strømmåling og PWM
- Posisjonsstyring av stepp motorer
- Posisjonsstyring av stepp motorer med mikrostepp ved strømkontroll

Det er en Arduino Mega som skal benyttes som basis i oppgaven.

Det vi ønsker å få utviklet er et laboppsett «all in one», der bare støpsel og USB-stikk skal plugges inn (plug and work)

- Det legges stor vekt på god visuell feedback, brukervennlig og kompakt design
- Det skal utarbeides laboppgaver med god beskrivelse og gode løsningsforslag.
- Design prosessen må skje i tett samarbeid med de fagansvarlige

UiA-kontaktperson: Morten Ottestad

APPENDIX C

Laboratory assignments

This appendix contains the latest version of the laboratory assignments. It should be noted that it is a document written in Norwegian, and that it is not by far the completed version.

Document C.2: Laboratory Assignments for the All in one Servo Lab

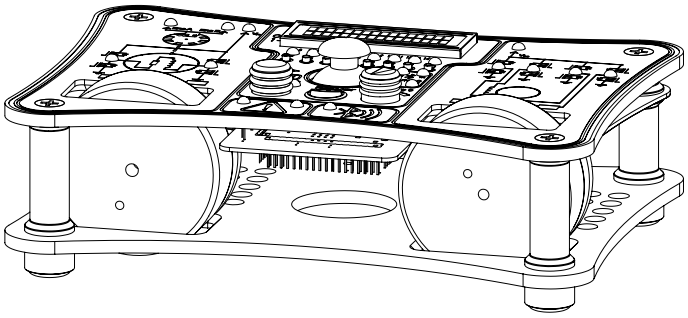
UNIVERSITETET I AGDER

LABORATORIEOPPGAVER

All-In-One Servolab

Arduino™-basert laboratorieplattform for mikrokontroller styring av servosystemer

0.21 Tentativ 2014-04-22



av

Audun Hørthe
Helge Nødland
Bjørnar Preus-Olsen

Veiledere:

Morten Ottestad
Torstein K. Wroldsen

Oppgaveheftet er en del av et bachelorprosjekt som er gjennomført som ledd i utdanningen ved Universitetet i Agder og er godkjent som del av denne utdanningen. Denne godkjenningen innebærer ikke at universitetet innestår for de metoder som er anvendt og de konklusjoner som er trukket.

Universitetet i Agder, 2014
Fakultet for teknologi og realfag
Institutt for ingeniørvitenskap

Document C.2: Laboratory Assignments for the All in one Servo Lab

Versjonskontroll

Version	Status	Dato	Endring	Forfatter
0.10	Tentativ	2014-02-04	Nytt dokument	AH
0.11	Tentativ	2014-03-24	Oppdatert utforming	AH
0.20	Tentativ	2014-04-07	Nye oppgaver	BPO
0.21	Tentativ	2014-04-22	Foreløpig korrektur	AH

Dette dokumentet er skrevet i LaTeX, med TeX Live og TeXmaker, og satt i 11pt kpfonts.

Kompilert med BibTeX, MakeIndex og PDFLaTeX.

Dokumentet kan lastes ned fra:

http://www.NOURLYET.no/aiosl/oppgaver.pdf (pdf)
http://www.NOURLYET.no/aiosl/oppgaver.zip (zip/tex)

Document C.2: Laboratory Assignments for the All in one Servo Lab

Sammendrag

(AH: Kun skisse, fjern dobbelt opp i forord og sjekk innhold!!!)
Servoteknikk og styring av servomotorer er et kjerneelement innen ekatronikk. Faget innebærer mye av elementene fra blant annet matematikk, reguleringsteknikk og elektriske kretser.

"All-In-One Servolab" er ment som et verktøy til å gjøre seg kjent med mikrokontroller og programmering av disse. Oppgavene begynner med små og forholdsvis enkle programmer for å gi en innføring av de viktigste grunnelementene innen programmering. Ved å bygge videre på disse grunnelementene, vil oppgavene innebære det faget handler om; styring av servomotorer.

Selve lab-plattformen er bygget rundt mikrokontrolleren Arduino Mega 2560. I tillegg til motorene (en steppmotor og en likestrømsmotor), så er den utstyrt med blant annet lamper, knapper og lcd-skjerm. De aller fleste funksjonene vil bli tatt i bruk i oppgavene, og det er fullt mulig å bevege seg utenfor oppgavene og lage egne programmer.

Bak i heftet er det stikkordsregister, ordliste og en liten oversikt over de mest sentrale funksjonene. Likevel anbefaler vi at dere også bruker ressursene på nettet, slik som arduino.cc og arduino playground.

Plattformen har en del tilhørende programvare for elektronisk feedback på PC, laget i programmeringsmiljøet Processing. Processing er ikke en del av kurset, men for de interesserte så er det nok en gang fullt mulig å endre, teste og lage sine egne løsninger.

Vi håper dere får godt utbytte av faget og "All-In-One Servolab"!

Document C.2: Laboratory Assignments for the All in one Servo Lab

Forord

(AH: Kun skisse, fjern dobbelt opp i sammendrag og sjekk innhold!!!)
Oppgaveheftet er en del av bachelorprosjektet "All-In-One Servolab", som ble gjennomført våren 2014 ved Universitetet i Agder. Prosjektet tar utgangspunkt i faget MAS220 Servoteknikk, et fag som dekker både mikroprosessorteknikk og servoteknikk.

Selv om universitetet hadde det nødvendige utstyret for å gjennomføre laboratorieoppgaver innen dette faget, så viste det seg høsten 2013 at det ville være hensiktsmessig å ha en ferdig oppsatt plattform, en del ferdig programvare og oppgavesett som ville binde de to fagområdene bedre sammen.

Vi jobbet en del med utviklingen av plattformen, spesielt med tanke på hvilke funksjoner den skulle dekke, hvordan den skulle fungere i forhold til et menneskelig grensesnitt og fysisk utforming. Fremfor å velge eksisterende produkter til Arduino-formatet, valgte vi å basere oss på et fullstendig tilpasset format. Dette innebar testing og design av driverkretser, utvikling av kretskort og chassis.

I tillegg til de innebygde funksjonene, så er det fullt mulig å koble til eksterne komponenter. Dette kan være potensiometre, sensorer, knapper, lamper, seriekommunikasjon eller kommunikasjon mellom flere plattformer. Vi mener derfor at produktet har høy grad av fleksibilitet, kombinert med et robust og driftssikkert ytre.

Målsetningen er at studentene skal ha utstyr å jobbe med som gjør den teoretiske delen av faget mer tilgjengelig og mer relevant. Det er et vesentlig poeng at plattformen virker og føles ordentlig og profesjonell, at den er klar til bruk med en gang og har gode muligheter for å forstå hva som skjer innen programvare og elektronikk.

Arbeidet med dette prosjektet har vært veldig givende. Det føltes som et viktig oppdrag, og vi er selv veldig fornøyd med sluttresultatet. Det håper vi at studenter og ansatte ved Universitetet i Agder også blir. Vi håper faktisk at dette prosjektet fører til liknende bachelorprosjekter for å utvikle lab-utstyr innen fag som mekatronikk/sensorteknikk, reguleringsteknikk, elektriske kretser og sikkert enda flere fag.

Vi ønsker Universitetet i Agder lykke til med "All-In-One Servolab", og takker for et interessant oppgave!

Grimstad, mai 2013

Audun Hørthe
Helge Nødland
Bjørnar Preus-Olsen

Document C.2: Laboratory Assignments for the All in one Servo Lab

Innhold

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

Figurer

Document C.2: Laboratory Assignments for the All in one Servo Lab

Tabeller

Document C.2: Laboratory Assignments for the All in one Servo Lab

Dokumenter

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 1

Brukerveiledning

Formålet med dette kapittelet er å gi en kort innføring av de forskjellige modulene og funksjonene til servo-laben og hvordan de brukes. Se http://arduino.cc/en/Reference/HomePage for en bedre forklaring av arduinoens funksjoner

1.1 LEDs

De 8 lysdiodene er koblet direkte til Arduino pinnene D42-D49(PORT L). Før en LED brukes må den aktuelle pinnen settes som en utgang. Dette gjøres med kommandoen **`pinMode(x,OUTPUT)`**, hvor x er pin-nummeret til den LEDen som skal brukes(eks. D49). Dette gjøres normalt i **`void Setup()`**. Etter dette kan lysdiodene tennes og slukkes hved hjelp av **`DigitalWrite(x,HIGH)`** og **`DigitalWrite(x,LOW)`**. Det kan også brukes port manipulering for å skrive til alle lysdiodene samtidig. f.eks **`DDRL 0xFF`** setter alle lysdiodene til utganger, og **`PORTL 0xFF`** setter alle utgangene høye. Pin 44, 45 og 46 støtter også pulsbreddemodulering for å endre lysstyrke. Dette gjøres med funksjonen: **`analogWrite()`**.

1.2 Trykknapper

De 8 trykknappene er koblet til Arduino pinnene D22-D29(PORT A). Alle knappene har en ekstern "debounce"-krets og en "pulldown"-resistor som sørger for å trekke signalet lavt når knappen ikke trykkes. For å kunne lese statusen til en pinne må den settes til en inngang. Dette gjøres med kommandoen **`pinMode(x,INPUT)`**. Dette gjøres normalt i **`void Setup()`**. En bryter leses med funksjonen: **`digitalRead(x)`**, hvor den returnerer 1 dersom knappen er trykket inn, og 0 dersom den ikke er trykket. Det kan også brukes portmanipulering for å lese alle trykknappene på en gang.

1.3 Analoge sensorer

De analoge sensorene består av et potentiometer og en joystick. I tilleg er det to ekstra innganger bak på servo-laben som kan brukes til eksterne sensorer. De analoge sensorene kan leses med funksjonen: **`analogRead()`**. Denne funksjonen returnerer en 10-bit(0-1023) verdi, avhengig av den stenningen på den analoge inngangen(0-5V).

Document C.2: Laboratory Assignments for the All in one Servo Lab

1.4 LCD

LCD pin	*Arduino pin*	*Funksjon*
LCD_RS	D41	velger mellom kommando og data register
LCD_E	D40	Starter data read/write
LCD_D4	D39	paralell data
LCD_D5	D38	paralell data
LCD_D6	D37	paralell data
LCD_D7	D36	paralell data
LCD_backlight	D4	styrer bakgrunnsbelysningen

Tabell 1.1: LCD pinne oppsett

For å kontrollere LCD-displayet er det lettest å bruke Arduino-biblioteket: LiquidCrystal. Under ***File->Examples->LicuidCrystal*** i Arduino IDE-en er det ferdige eksempler hvor bare pinneopsettet må endres. For å slå på bakgrunnsbelysningen må D4 konfigureres som en utgang og skrives høy(ventuelt med PWM). for mer informasjon se http://arduino.cc/en/Reference/LiquidCrystal

1.5 Mekanisk encoder

1.6 Bluetooth

BT-modul	*Arduino Pin*	*Funksjon*
RX	D16 (TX2)	Seriell komunkasjon
TX	D17 (RX2)	Seriell komunkasjon
RESET	D39	Resetter BT-modul
KEY	D38	Åpne AT-modus
PIO9		BT-connected LED

Tabell 1.2: Bluetooth pinne oppsett

Blåtann-modulen er koblet til Arduinoens seriell port 2, og har en baud-rate på 9600kps. Den kan sende og motta tekst-strenger fra andre enheter med en Bluetooth-terminal. Modulens enhetsnavn er Servolab_x og paringskoden er 1234. For å "åpne" serieporten til arduinoen skriver vi kommandoen: `Serial2.begin(9600)` i `void Setup()`. For å skrive til modulen bruker vi `Serial2.print()` og `Serial2.write()`. For å lese data bruker vi `Serial2.available()` og `Serial.read()`. KEY og RESET pinnene brukes for å komme inn i AT-modus(kommandomodus for endring av BT-moduens oppsett) og trenger nirmalt ikke å konfigureres. Enheten som skal

Document C.2: Laboratory Assignments for the All in one Servo Lab

kommunisere med servo-laben, må ha en Bluetooth-terminal installert. Dette kan være en laptop, men aller enklest er det å bruke en smart-telefon med en bluetooth-terminal app. Iphone kan ikke brukes da den kun tillater audio over bluetooth.

1.7 Dc-motor

Motordriveren til dc-motoren er Texas Instruments DRV8840. datablad: http://www.ti.com/lit/ds/symlink/drv8840.pdf. For å ikke overbelaste motoren, vil driveren begrense maks tillat motorstrøm til 3A.

Pin forklaring

- ***Motor phase:*** Koblet til Arduino pin D6. Bestemmer strømretningen gjennom motoren. Brukes til å bestemme motorens rotasjonsretning.
- ***Motor enable:*** Koblet til Arduino pin D7. Med et pulsbredde-modulert signal på denne pinnen bestemmes pådraget.
- ***Motor encoder:*** Koblet til Arduino pin D21(INT2) og D22(INT3). Encoderen brukes for å bestemme motorens posisjon og hastighet, og har 16 linjer pr.omdreining. Ut av disse kan man få opp til 64 pulser pr. omdreining ved bruk av to interrupts som trigger på hver flanke.
- ***Motor current sense*** Koblet til Arduino pin A3 og lesses av funksjonen `analogRead(A3)`. Sensoren gir ut er signal på 185 mV pr. A, med en offset spenning på 2,5V For å finne motorstrømmen brukes formelen: $I_{Motor} = (AnalogValue - 512) * 37,89$ Strømmen kan være både positiv og negativ avhengig av strømretningen.
- ***Motor decay*** Koblet til Arduino D5. kan brukes til å bremse eller "frikoble" motoren. Denne pinnen trenger normalt ikke å brukes. Se motordriverens datablad for mer informasjon.

Drivmetoder

Decay pin	***Enable pin***	***Phase pin***	***Motor terminal A***	***Motor terminal B***
0	0	X	LOW	LOW
1	0	X	HIGH IMPEDANCE	HIGH IMPEDANCE
X	1	1	HIGH	LOW
X	1	0	LOW	HIGH

Tabell 1.3: Motordriver sannhetstabell

Den vanligste måten å drive motoren på er å bestemme rotasjonsretningen med ***Moter phase*** og å styre pådraget med PWM på ***Motor enable***. Denne drivmetoden kalles "Unipolar switching".

Det er også mulig å drive motoren med et PWM signal på ***Moter phase*** og sette ***Motor enable*** høy. Dette kalles "Bipolar switching". Fordelen med denne drivmetoden er at motoren kan

styres av bare et signal. Når pådrager er under 50% vil motoren gå en vei, og når pådraget er over 50% vil motoren gå den andre veien.

Det kan også være lurt å øke Arduinoens PWM frekvens. Dette gir en jevnere motorstrøm, høyere båndbredde og mindre motorstøy. ved å sette inn linjen **`TCCR4B = TCCR4B & 0b11111000 | 0x01;`** i **`void Setup()`** øker vi PWM frekvensen fra 490 Hz til 32 kHz.

1.8 step-motor

Step-motoren støtter fullstepp, halvstep og mikrostep. Den brir drevet av en "chopper"-driver som regulerer strømmen gjennom motorfasene uavhengig av drivspenningen. Motordriveren er en Texas Instrument DRV8813. datablad: http://www.ti.com/lit/ds/symlink/drv8813.pdf. Det blir også brukt en 2 kanals DAC[1] for å styre strømmen gjennom motoren og dermed gjøre det mulig å mikrosteppe motoren. DAC-en er en 12-bit mcp4922. datablad: http://ww1.microchip.com/downloads/en/DeviceDoc/21897a.pdf. Biblioteket "DAC_bibliotek" for å bestemme motorstrømmen. For å kjøre motoren med full- eller halvstep, trenger strømmen kun å settes i **`void Setup()`**, som vist i følgende eksempel.

Programkode 1.1: DACbibliotek.ino

```
1 #include <dac.h>
2
3 void setup()
4 {
5 dac_init();
6 set_dac(4095,4095);//Allow 1A per motor phase
7 }
```

Funksjonen **`set_dac(a,b)`** setter tillat strøm gjennom de to motorfasene. strømmen kan settes mellom 0(0A) og 4095(1A), og dette blir brukt når man skal mikrosteppe motoren.

Step sekvens

xenable	*xphase*	*xout1*	*xout2*
0	X	HIGH IMPEDANCE	HIGH IMPEDANCE
0	1	HIGH	LOW
0	0	LOW	HIGH

Tabell 1.4: Sannhetstabell stepperdriver

Step nr	*A enable (D69)*	*B enable (D67)*	*A phase (D68)*	*B phase (D66)*
1	1	1	0	0
2	1	1	0	1
3	1	1	1	1
4	1	1	1	0

Tabell 1.5: Eksempel på kjøring i fullstep

[1] Digital til analog converter

Document C.2: Laboratory Assignments for the All in one Servo Lab

Step nr	*A enable (D69)*	*B enable (D67)*	*A phase (D68)*	*B phase (D66)*
1	1	1	0	0
2	0	1	X	0
3	1	1	1	0
4	1	0	1	X
5	1	1	1	1
6	0	1	X	1
7	1	1	0	1
8	1	0	0	X

Tabell 1.6: Eksempel på kjøring i halvstep. X betyr at retningen til pinnen er irrelevant

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 2

Komme igang

Formålet med denne oppgaven er å laste ned og installere arduino IDE. Vi skal også lage et lite program(blink) for å verifisere at arduino IDE virker som det skal og får kontakt med arduinoen.

2.1 Laste ned programvare

Gå til http://arduino.cc/en/Guide/HomePage og følg step-by-step guiden for installering og oppsett av arduino programvaren.

2.2 åpne "Blink" programmet

Når programvaren er lastet ned og installert, åpne arduino IDE ved å dobbeltklikke arduino.exe i mappen du lastet ned (AH: Foreslår generealisering win/mac/linux). I verktøylinja trykker du **Tools -> Board** og velger hvilket arduino-brett du vil bruke. Her velger vi **arduino 2560 or mega ADK** Nå vil vi prøve å laste opp et av eksempelprogrammene til arduinoen. trykk **File -> Examles -> Basic -> Blink**. Dette programmet vil sette pin 13 på arduinoen høy(5V) og lav(0V) med en frekvens på 0,5 Hz. vi ønsker å bytte pin 13 til pin 49 siden denne er koblet til LED-0 på frontpanelet til servolaben. Dette gjør vi ved å endre linje 10: **`int led = 13;`** til: **`int led = 49;`**

Programkode 2.1: Blink.ino

```
1  /*
2    Blink
3    Turns on an LED on for one second, then off for one second,
        repeatedly.
4
5    This example code is in the public domain.
6   */
7
8  // Pin 13 has an LED connected on most Arduino boards.
9  // give it a name:
10 int led = 49;
11
12 // the setup routine runs once when you press reset:
13 void setup() {
14   // initialize the digital pin as an output.
```

Document C.2: Laboratory Assignments for the All in one Servo Lab


```
15   pinMode(led, OUTPUT);
16 }
17
18 // the loop routine runs over and over again forever:
19 void loop() {
20   digitalWrite(led, HIGH);   // turn the LED on (HIGH is the
       voltage level)
21   delay(1000);               // wait for a second
22   digitalWrite(led, LOW);    // turn the LED off by making the
       voltage LOW
23   delay(1000);               // wait for a second
24 }
```

2.3 Modifisere blink

Vi ønsker nå å utvide blink programmet slik at Led-1(pin 48) også blinker. vi ønsker at denne skal blinke motsatt av LED-0. Vi ønsker også å øke blinkefrekvensen til 5 Hz. Gjør de nødvendige modifikasjonene i koden.

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 3

LCD-biblioteket

Formålet med denne oppgaven er å lære å bruke biblioteker. I dette tilfellet bruker vi *LiquidCrystal*-biblioteket for å skrive til et 16x02 LCD.

3.1 Hello World

For å komme raskt igang begynner vi med et ferdig eksempel. trykk **File -> Examples -> LiquidCrystal** og åpne programmet: **HelloWorld**. Noen små endringer må gjøres for å få koden til å kjøre. Interfacen mellom LCD og arduinoen er litt anderledes siden forskjellige pinner er brukt. Vi må også aktivere bakgrunns-belysningen til displayet. Dette gjøres ved å sette pin 4, høy.

Gjør de nødvendige endringene for å kunne kjøre "HelloWorld" på LCD-displayet.

3.2 Serial til display

Lag et program som mottar og lagrer tekst som blir sent fra pc-en. Hvis programmet mottar en ***Linefeed*** (
n) tegn, skal teksten skrives på første linje på displayet. Samtidig skal det som opprinnelig stod på første linje flyttes ned til andre linje.

HINT: lagre inkommende tegn i et array. For å kunne sende en ***Linefeed***-kommandoen fra serie-terminalen nedtrekksmeny. Hvis du velger ***Newline*** i nedtrekksmenyen i bunnen av arduino serie-terminalen, vil terminalen automatisk legge til et ***Linefeed***-tegn på slutten når du trykker enter.

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 4

Vanlige Arduino funksjoner

Formålet med denne oppgaven er å lære å bruke de vanligste funksjonene i Arduino.

4.1 Serial.print()

En metode for å få kommunikasjon mellom arduinoen og andre enheter, er med funksjonen `Serial.print()`. Lag et program som skriver "hei" til serieterminalen hvert sekund.

Hva er forskjellen på `Serial.print("hei");` *og* `Serial.println("hei");` *?*

Hva skjer dersom du skriver `Serial.print("hei t");` *eller* `Serial.print("hei n");` *?*

4.2 analogRead()

`analogRead()` er en funksjon som leser arduinoens ADC-converter. Den måler spenningen 0-5V og returnerer en 10 bit verdi(0-1023). Lag et program som måler Joystickens X og Y verdi, og printer dem ut til serie terminalen. teksten som sendes til terminalen skal ha følgende format: **"X-verdi = (x-verdi) Y-verdi = (y-verdi)"**

4.3 analogWrite()

`analogWrite()` er en funksjon som genererer et 8 bit(0-255) pulsbreddemodulert signal. dette kan f.eks brukes til variere lysstyrken på lysdioder. Vær obs på at ikke alle pinner støtter analogWrite(). Utvid programmet fra ***analogRead()*** slik at lysstyrken på LED-3 styres av Joystickens X verdi, og LED-4 styres av Joystickens Y verdi. HINT: 10 bit signalet fra `analogRead()` må konverteres til et 8 bit signal for `analogWrite()` funksjonen.

4.4 Løkker

Lag et program som begynner med å tenne LED-0. etter en delay() slukkes LED-0 og LED-1 tennes. Slik skal det fortsette til vi kommer til LED-7. Da skifter vi retning og går tilbake til

Document C.2: Laboratory Assignments for the All in one Servo Lab

LED-0. Bruk for- eller while- løkker for å gjøre koden mer kompakt. Bruk Potensiometeret for å kunne justere delay() mellom 0 og 1023 ms.

4.5 millis()/multitasking

Dersom vi går tilbake og ser på arduino Blink eksempelet, ser vi at **`delay(1000)`** er brukt for å lage en pause på 1000 ms mellom tenning og slukking av lysdioden. Dette går greit så lenge programmet vårt ikke skal gjøre annet enn å blinke en lysdiode. Dersom vi ønsker å gi prosessoren flere oppgaver, må vi kvitte oss med **`delay()`**. Vi må derfor finne en ny metode for å få lysdioden til å blinke.

Lag et program som tenner og slukker LED-0 hvert sekund. Samtidig skal resten av lampene styres av de tillhørende bryterene. Lampene skal lyse når bryteren har høyt signal, og være slukket ellers.

> HINT: Les gjennom eksempelkoden "BlinkWithoutDelay".

4.6 Data typer og størrelser

etellerannet

4.7 Ekstraoppgave

Lag en funksjon som konverterer verdien vi får fra **`analogRead()`**, til den faktiske spenningen som er på pinnen.

> HINT: For å kunne regne med desimaler, må vi bruke datatypen **`float`**.

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 5

Interrupts

Interrupts er en måte å varsle en mikrokontroller at noe har skjedd. I denne oppgaven skal vi fokusere på eksterne interrupts. Arduino mega 2560 har 5 pinner som støtter eksterne interrupts. disse er koblet til encoderen til dc-motoren og encoderen på frontpanelet og trykk-knappen på encoderen.

5.1 Encoder posisjon

Lag et program som bruker interrupts til å måle posisjonen til encoderen på frontpanelet. Dersom posisjonen endrer seg skal den nye posisjonen skrives ut ved hjelp av serieterminalen eller LCD-displayet.

5.2 Ekstra oppgave: Timer interrupts

Lag et program som bruker timer interrupts til å blinke LEDs

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 6

Bluetooth

Servolab-en har en innebygd bluetooth modul. I denne oppgaven skal vi komunisere trådløst mellom en annen blåtann enhet. Det letteste er å bruke en Android telefon. Iphone kan ikke brukes da den kun støtter lyd over bluetooth.

6.1 Tilkobling

Last ned en bluetooth terminal på android telefonen. Bluetooth Terminalfra Next Prototypesfunker greit. Åpne terminalen og koble til bluetooth-modulen. modulens enhetsnavn er Servolab_x, koden er 1234. Når enhetene har koblet sammen, vil den blå lysdioden på frontpanelet lyse.

6.2 komunikasjon

Bluetooth modulen komuniserer med arduinoen gjennom serieporten: Seraial2". Baudraten på denne denne porten settes til 9600kbs. Data som skrives til denne porten vil sendes over bluetooth til den andre enheten og vica versa.

Lag et program som gjør det mulig å "chatte"mellom telefonen og arduino serial monitor.

HINT: For å få til dette, må serie-data mottatt fra serial0 sendes ut igjen gjennom Serial2. Data mottatt fra Serial2, må sendes ut igjen gjennom Serial0. Husk å bruke Serial.write() i stedet for Serial.print(), siden vi skriver ett og ett ascii-tegn.

6.3 Ekstraoppgave?

styr lysdiodene, og dc-motor gjennom seriekommandoer sendt over bluetooth.

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 7

Stepmotor

I denne oppgaven skal vi lære å styre step-motoren modulen på servolaben. Driveren til step-motoren er av typen "chopper driver", som regulerer strømmen i hver motorfase. Hvor mye strøm som får lov til å gå gjennom motoren bestemmes av en DAC(digital to analog converter). Ved å variere den tillatte strømmen gjennom motoren kan vi mikro-steppe motoren.

7.1 initialisering

For å gjøre det lettere å skrive verdier til DAC-en er det laget et bibliotek for dette. Last ned biblioteket: **DAC_bibliotek** fra fronter? og legg det inn i mappen **libraries** i arduino.

Før vi kjører motoren i full- eller halv-step, må vi sende en verdi til DAC-en som styrer strømmen gjennom motoren. For å gjøre dette importer vi biblioteket DAC_bibliotek in i sketchen. I `void setup()` skriver vi linjen `dac_init();` og deretter `set_dac(4095, 4095);`. Dette vil få motordriveren til å tillate 1 ampere gjennom hver motorfase. Vi må også huske å sette styresignalene til motorfasene (D66-D69) til outputs.

7.2 fullstep

Lag funksjonen `step()` som får stepperen til å ta et step fremover hver gang den kjøres. sett denne funksjonen inn i `void loop()`. Bruk `delay(1000)` slik at funksjonen blir kjørt hvert sekund.

I `step()` funksjonen kan det lage en teller som teller fra 0 til 3, og deretter nullstiller seg. Denne telleren kan brukes i en `switch(teller)` for å sette rett step-sekvens.

7.3 turtallstyring

bytt linjen med `delay(1000);` med `delayMicroseconds(delay(analogRead(A0)/4+1);`. nå kan tiden mellom step-ene til motoren justeres ved hjelp av potentiometeret.

7.4 halvstep

Når stepmotoren går med fullstep får den en veldig hakkete gange, spesielt ved lave turtall. Dette kan gi problemer i form av støy, vibrasjoner og resonans. I forrige oppgave la dere kansje merke til hvilke turtallsområder som ga resonans og fikk stepperen til å miste step. Ved å benytte halvstep, vil motoren gå jevnere, og det blir mindre problemer med resonans.

Gjør de nødvendige endringene i programmet slik at vi kan kjøre motoren med halvstep.

7.5 Posisjonskontroll

En av de fordelene med step-motoren er vi kan kontrollere motorens posisjon og hastighet uten noe form for tilbakemeldinig.

Lag funksjonen **`Stepper_move(int steps, int delaytime)`**. **`int steps`** skal bestemme hvor mange step motoren skal gå. Dersom **`steps`** er positiv, skal motoren rotere med klokka. Dersom den er negativ skal den rotere mot klokka. **`int delaytime`** skal bestemme tiden mellom hvert step i millisekunder. Bruk denne funksjonen til å lage et program som roterer motoren en runde framover og deretter en runde bakover.

7.6 Ekstraoppgave mikrostep

Ved å variere strømmen gjennom hver motor-spole ved hjelp av **`set_dac(fase a, fase b)`**, kan vi mikrosteppe motoren med den oppløsningen vi vil. Lag et program som kjører step-motoren ved hjelp av mikrostep.

7.7 Ekstraoppgave Timer interrupt

Timing er viktig når vi kjører en stepper. I de tidligere oppgavene har vi brukt **`delay()`** for å lage en pause mellom hvert step. Ved å bruke timer interrupts i stedet får vi et mer nøyaktig tidsintervall mellom hvert step. Samtidig kan prosessoren jobbe med andre ting i mellomtiden. Lag et program som bruker timer interrupts til å steppe motoren.

7.8 Ekstraoppgave rampe

I posisjonskontroll-oppgaven må motoren akselerere fra stillestående til maksfart i løpet av et step. Ved å lage en opp- og ned-rampe i begge endeposisjoner, kan vi kjøre motoren mye fortere.

Skriv om programmet slik at motorens hastighet ramper opp i begynnelsen og ramper ned når den nærmer seg endeposisjon.

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 8

Styring av dc-motor

I denne oppgaven skal vi kontrollere dc-motoren vi skal kontrollere rotasjonsretningen og hastigheten ved hjelp av PWM.

8.1 enveiskjøring

Til å begynne med skal vi lage et enkelt program som styrer hastigheten til motoren. Lag et program som måler det analoge signalet fra potensiometeret, og konverterer det til et 8-bits PWM signal til å styre pådraget til motoren motoren.

PWM signalet skal sendes til motordriverens enable-pin. Det er også viktig at driverens phase-pin er enten høy eller lav.

8.2 frem og tilbake

Lag et program som gjør det mulig å kjøre motoren begge retninger ved hjelp av joysticken.

Midtpunktet til joysticken er på ca 511. Her skal pådraget være 0. Når verdien fra joysticken øker skal phase-pinen skrives høy og pådraget øke opp mot 255. Når verdien fra joysticken synker til under 511, skal phase-pinen skrives lav, og pådraget igjen økes mot 255. Det kan også være lurt å skrive ut alle analoge verdier til seriemonitoren. Da er det mye enklere å "debugge" koden

8.3 måling av posisjon og hastighet

Måling av motorens posisjon og hastighet kan gjøres ved hjelp av encoderen. Utvid programmet fra forrige oppgave slik at den hvert sekund skriver motorens posisjon(radianer) og hastighet(rad/s) til seriemonitoren.

En metode for å måle hastighet ved hjelp av encoderen, er å finne motorens posisjonsendring i løpet av et tidsintervall. deretter finner man hastigheten $\omega = \frac{d\theta}{dt}$.

Document C.2: Laboratory Assignments for the All in one Servo Lab

Dersom begge interrupt-pinnene blir brukt og vi får interrupt på både stigende og synkende flanke, vil vi få 64 "ticks" per omdreining. Det vil si at hvert "tick" er $\frac{2\pi}{64} = 0,0982rad$

8.4 måling av motorstrøm og momentstyring

Ofte er det ønskelig å vite moment som ligger over motoren. dette kan finnes ved å måle strømmen gjennom motoren, og multiplisere denne med motorens momentkonstant(kt). Lag et program som kjører motoren den ene veien. Hver gang momentet på svinghjulet overstiger 1 Nm, skal motoren stoppe og deretter gå andre veien.

Når motoren skifter retning, er det lurt å rampe opp pådraget gradvis. Ellers vil motoren overstige momentgrensen under akselerasjonen.

8.5 Ekstraoppgave P-regulator

Lag en enkel P-regulator for posisjonsstyring av dc-motoren.

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 9

Lead og lag kontroller

I denne oppgaven lage en Lead og en Lag kontroller. Vi skal benytte oss av et ferdig skrevet arduino-program, og bruker et javaScript skrevet i Processing på pc-en til å styre arduinoen.

9.1 P-regulator

Last ned, og åpne arduino sketchen Servolab_Lead_Lag.ino. Last opp programmet til arduinoen. Åpne deretter filen Servolab_Lead_lag_HMI.exe. Java må være installert på maskinen for at programmet skal virke. Det er også viktig at arduinoen kjører riktig program og at den er tilkoblet pc-en. Hvis alt virker skal du få opp dette vinduet

Her kan vi styre motorens posisjon samtidig som vi får opp "real-time" grafer av settpunkt, motorens faktiske posisjon, og motorens pådrag(ved å trykke på "paadrag" knappen). Settpunktet kan settes manuelt, eller det kan settes til å følge en sinus, trekant, eller firkantpuls. Nede til høyre er det tre felter som kan brukes for å lage en Lag eller Lead kompansator for systemet. Hvis disse feltene ikke er benyttet, vil en enkel P-regulator gjelde for systemet. Se etter at servolabens strømforsyning er 12V og test systemet. Hvordan oppfører systemet seg? Ved å velge forskjellige verdier av "K" endrer vi systemets forsterkning. Finn en grei k-verdi. Hva skjer når K er lav, og hva skjer hvis den blir for høy?

9.2 Lag kontroller

For å forbedre systemets stasjonære avvik, og samtidig gjøre systemet mer stabilt. Vi vil derfor lage en Lag-kompansator til systemet. Vi ønsker at systemet skal ha et oversving på mindre enn 5% og kunne følge en rampe med stigning X rad/sek med et avik på mindre enn x rad. Alle verdier er referert til motors aksel. Transferfunksjonen til en Lag-kompansator er $h_{LAG} = K\frac{a \cdot Ts+1}{Ts+1} a < 1$ Finn verdien til K, a og T og sett inn i de tre tekstfeltene og trykk "send". Kontroller at systemet er innenfor spec.

9.3 Lead kontroller

Vi ønsker å forbedre systemets båndbredde. For å gjøre dette benytter vi oss av en Lead-kompensator. Vi ønsker samme oversving, og rampe-avik som på forrige oppgave. Finn verdien til K, a og T og kontroller at systemet er i henhold til spec. Hva er oppfører systemet seg med en Lead-kompansator sammenlignet med en Lag-kompansator?

Document C.2: Laboratory Assignments for the All in one Servo Lab

9.4 Ekstraoppgave Lead/Lag regulator

Document C.2: Laboratory Assignments for the All in one Servo Lab

LAB OPPGAVE 10

Motordata

10.1 Dc-motor

	Value	Unit	
Type	GB37Y3530-131EN	-	
Nominal voltage	12	V	
No load speed	11500	rpm	
No load current	242	mA	
Stall current (@12V)	5,06	A	
Terminal resistance	2,37	Ohm	
Terminal inductance	1,79	mH	
Ke	9.407×10^{-3}	Vs/rad	
Kt	9.407×10^{-3}	Nm/A	
Mechanical time constant(with gear and fly wheel)	50	mS	
Electrical time constant	0,757	mS	
Rotor inertia	1.85×10^{-6}	kgm^2	
Load inertia (fly wheel)	1.324×10^{-4}	kgm^2	
Gear ratio	131:1	-	

Tabell 10.1: Teknisk data. Dc-motor

10.2 Step-motor

Document C.2: Laboratory Assignments for the All in one Servo Lab

	Value	Unit	
Type	42HS40-1704JA05-A20	-	
Terminal voltage	3,4	V	
Terminal resistance	2	Ohm	
Terminal inductance	3	mH	
Rotor inertia	5.7×10^{-6}	kgm^2	
Load inertia (fly wheel)	1.32×10^{-4}	kgm^2	
step angle	1,8	degrees	
holding torque (@1,7A)	0,335	Nm	

Tabell 10.2: Teknisk data. Step-motor

Document C.2: Laboratory Assignments for the All in one Servo Lab

Bibliografi

[1] M. D. licker et al., *Dictionary of Engineering*. McGraw-Hill, 2003.

Document C.2: Laboratory Assignments for the All in one Servo Lab

Nomenklatur

Hz $1/s$

mekatronikk m

Document C.2: Laboratory Assignments for the All in one Servo Lab

Ordliste

Mechatronics *A branch of engineering that incorporates the ideas of mechanical and electronic engineering as a whole, and, in particular, covers those areas of engineering concerned with the increasing integration of mechanical, electronic and sofware engineering into a production process.[1]*

Servo(motor) *The electric, hydraulic, or other type of motor that serves as the final control element in a servomechanism. It receives power from the amplifier element and drives the load with a linear or rotary motion.[1]*

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

TILLEGG A

Tegninger

A.1 Her er det

Foreløpig ingenting

Document C.2: Laboratory Assignments for the All in one Servo Lab

Document C.2: Laboratory Assignments for the All in one Servo Lab

TILLEGG B

Løsningsforslag

Det er viktig å poengtere at innen programmering, så er det bortimot uendelig mange måter å løse et problem på. Så langt det er mulig, holdes løsningsforslagene innen "vanlig" Arduino-kode. Det finnes likevel unntak, slik som å skrive og lese hele porter i stedet for enkeltpinner.

B.1 Laboppgave 1

B.1.3 Modifisere blink

Vi begynner med å legge til variabelen led2, som skal være 48 (pinnenummeret)

Programkode B.1.3-1: Blink2.ino

```
11 int led2 = 48;
```

Deretter må vi definere pinMode for led2 som **OUTPUT** i **setup**-delen

Programkode B.1.3-2: Blink2.ino

```
17   pinMode(led2, OUTPUT);
```

Når den ene lyser, så skal den andre være av, og frekvensen skal være 5Hz. Det vil si at løkka skal repeteres 5 ganger i sekundet. Det er to pauser (delay), som er gitt ved:

$$delay = \frac{1}{2} * 5Hz = \frac{1}{2} * \frac{1}{5}s * 1000\frac{ms}{s} = \frac{1*1000}{2*5}ms = 100ms \tag{B.1}$$

altså må vi bruke **delay(100);**

Det komplette programmet ser nå slik ut:

Programkode B.1.3-3: Blink2.ino

```
1  /*
2    Blink
3    Turns on an LED on for one second, then off for one second,
        repeatedly.
4
5    This example code is in the public domain.
6   */
7
```

Document C.2: Laboratory Assignments for the All in one Servo Lab


```
8  // Pin 13 has an LED connected on most Arduino boards.
9  // give it a name:
10 int led = 49;
11 int led2 = 48;
12
13 // the setup routine runs once when you press reset:
14 void setup() {
15   // initialize the digital pin as an output.
16   pinMode(led, OUTPUT);
17   pinMode(led2, OUTPUT);
18 }
19
20 // the loop routine runs over and over again forever:
21 void loop() {
22   digitalWrite(led, HIGH);   // turn the LED on (HIGH is the
        voltage level)
23   digitalWrite(led2, LOW);   // turn the LED on (HIGH is the
        voltage level)
24   delay(100);                // wait for a second
25   digitalWrite(led, LOW);    // turn the LED off by making the
        voltage LOW
26   digitalWrite(led2, HIGH);  // turn the LED on (HIGH is the
        voltage level)
27   delay(100);                // wait for a second
28 }
```

This page is intentionally left blank.

"The knack of flying is learning how to throw yourself at the ground and miss."

–Douglas Adams