



NTNU – Trondheim
Norwegian University of
Science and Technology

Design of a Precision Spray Matrix

Valve Matrix Control for Microspraying in
Precision Agriculture

Frode Urdal

Master of Science in Cybernetics and Robotics

Submission date: May 2014

Supervisor: Jan Tommy Gravdahl, ITK

Co-supervisor: Trygve Utstumo, Adigo AS

Norwegian University of Science and Technology
Department of Engineering Cybernetics



Master project TTK4900

Student: Frode Urdal
Program: M.Sc. Engineering Cybernetics

Title: **Design of a Precision Spray Matrix**
Valve Matrix Control for Microspraying in Precision Agriculture

Adigo is developing a system for automatic detection and control of weeds in row crops, "Asterix". The system uses computer vision to detect and classify weed and crop and form a spray map. As the mobile robot moves forward each weed leaf is targeted by droplets of herbicide by a microsprayer. The system will reduce the herbicide use drastically from conventional spraying with an accompanied environmental benefit.

As a continuation of the specialization project the student will design the second version of the spray controller which is to be used in lab and field tests.

The focus of this thesis is to improve and integrate the spray matrix with the rest of the application resulting in a prototype to be functional in field trials this spring. A valve control algorithm, with a spray map and position log as inputs, have to be designed and implemented in software. In addition an interface between spray controller and computer will be developed.

The main parts of this project includes:

1. Test the remaining parts of the prototype PCB, improve the design, and produce the second version
2. Implement software for field tests on the microcontroller on the PCB
3. Prototype a microsprayer system with the required surrounding system for pressurized spray liquid handling, communication and power.
4. Perform laboratory experiments with High speed filming and tests of drop formation and droplet-plant interaction.
5. Based on the main findings in the project/thesis work, write, as 1st author, and submit a paper to ICARCV 2014
6. Implement the communication interface over RS485 with both the microcontroller firmware and PC-client.
7. Design and implement a valve control algorithm using python

Implement a valve/droplet control algorithm and consider the timing constraints and requirements of the system.

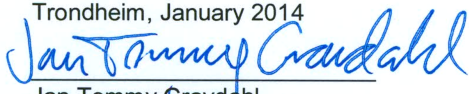
Advisor: Jan Tommy Gravdahl
Professor, Dept. of Engineering Cybernetics

External advisors: Trygve Utstumo
Ph.D. candidate, Dept. of Engineering Cybernetics
MSc. Engineering Cybernetics, Adigo AS

Jan Kåre Vatne
Siv.ing. Teknisk kybernetikk, Adigo AS

Project assigned: January 2014
To be handed in by: June 2, 2014

Trondheim, January 2014



Jan Tommy Gravdahl
Professor, Dept. Engineering Cybernetics

Preface

During the last year I have gotten familiar with the cutting edge of research within precision agriculture. A steep learning curve and exciting problem formulations made this a rewarding and interesting year.

I would like to mention Adigo AS for providing the interesting challenge and a motivating working environment. Especially Trygve Utstumo and Jan Kåre Vatne for guidance and advices, and how to best utilize the theory I have studied in practice.

I would like to thank my advisor Jan Tommy Gravdahl for his insight and helpful advice. In addition i would like to thank my fellow students at NTNU for all the good times, and the workshops in the Department of Engineering Cybernetics for providing equipment and helping with different tasks.

Abstract

Drop-on-demand weed control is a field of research within precision agriculture. The herbicide application is controlled down to individual droplets. In this master thesis the precision spray matrix for an autonomous field robot, Asterix, is designed. Asterix is currently under development by Adigo AS, and focuses on carrot and turnip cabbage. Such applications can reduce the herbicide usage drastically.

The overall system is presented with connections and timing concerns. This includes how the camera and the rest of the application is synchronized. With the use of a trigger from the camera, the time for each image can be logged on the printed circuit board and the inertial measurement unit can give each image an accurate time, position and orientation stamp. As a result the computer do not need to be synchronized with the rest of the system and may perform the calculations needed with relative times. An algorithm for valve control which generate spray commands from a spraymap and navigation log was designed and implemented.

A second revision of the printed circuit board was designed and tested. The design chosen for controlling the micro-dispensing valves is pulse-width modulation. A much used strategy for closing solenoid valves is by discharging the energy in the coil over two schottky diodes in reverse series. However the pulse-width modulation is used to close the valve by reversing the voltage in this design. Testing of the droplet tail and laboratory experiments verify this control strategy, and the reverse voltage time can be calculated for different valves.

Sammendrag

Ugresskontroll ved hjelp av dråpeskyter er et forskningsfelt innenfor presisjonsjordbruk. Påføring av ugress kontrolleres ned til individuelle dråper. I denne masteroppgaven er det utviklet en sprøytematrise for en autonom feltrobot, Asterix, som er under utvikling av Adigo AS. Fokuset er hovedsakelig gulrot og kålrot. Slike systemer vil redusere den brukte mengden av ugressmiddel drastisk.

Hele systemet er presentert med koplinger og tidsbegrensninger. Dette inkluderer hvordan kamera og resten av systemet er synkronisert. En trigger fra kameraet logges i mikrokontrolleren og navigasjonssensorene så eksakt tid for hvert bilde er kjent. Navigasjonssensorene gir så hvert bilde eksakt tid, posisjon og orientering. Slik trenger PC'en kun å regne på relative tider, og slipper å være synkronisert med resten av systemet. En algoritme for å generere sprøytekommandoer fra et sprøytekart og navigasjonsdata ble designet og implementert.

Revisjon nummer to av kretskortet er designet og testet. Puls-bredde modulasjon ble brukt for å styre ventilene. En mye brukt strategi for å lukke solenoider er å lade ut energien i spolen over to schottky dioder i revers serie. Her brukes puls-bredde modulasjon for å reversere spenningen over spolen istedet, og lader den ut på den måten. Testing for halen på dråpen verifiserer denne strategien, og tiden for den reverserte spenningen kan regnes ut for forskjellige ventiler.

Contents

Project Description	1
Preface	i
Abstract	iii
Sammendrag	v
1 Introduction	1
1.1 Background	1
1.2 Objectives	2
1.3 Structure of the Report	3
2 Literature Survey and Theory	5
2.1 Previous Applications	5
2.2 Valve and Nozzle Limitation	6
2.3 Droplet Efficacy	7
2.4 Modeling of Fluid Systems	8
2.5 Droplet Formation	11
3 Overall System	17
3.1 System Architecture	17
3.2 Valve Drivers	18
3.3 Time Synchronization	21
3.4 Valve Alternatives and Selection	23
4 PCB Design, Second Revision	27
4.1 Testing of Prototype Card	27
4.1.1 Results for Prototype Card	27
4.2 Improving the Design	28
4.3 Assembly	29

4.4	Tests and Debugging	29
4.4.1	Debugging and Fixes	29
4.4.2	Testing of Operational Functions	30
4.5	Results Second Revision	30
5	Software Design	33
5.1	Valve Mapper	33
5.1.1	Input	34
5.1.2	Output	34
5.1.3	Internal Functions	34
5.1.4	Python Skeleton	37
5.2	Communication Protocol	39
5.3	Microcontroller Code	40
6	Droplet Formation	41
6.1	Droplet Optimization	41
6.1.1	Test Setup	41
6.1.2	Results and Discussion	42
6.2	Parameter Testing	44
6.2.1	Test Configuration	44
6.2.2	Results	44
7	Application testing	45
7.1	Valve Test Setup	45
7.2	Valve Test Results	45
7.3	Laboratory Test Setup	46
7.4	Laboratory Test Results	47
8	Discussion	49
8.1	PCB	49
8.1.1	Valve Control Strategy	49
8.2	Overall system	50
8.3	Valve Control Algorithm	51
8.4	Microcontroller software	52
8.5	Liquid properties	52
8.6	Future Work	53

9 Conclusion	55
A Acronyms	57
B ICARCV Paper	59
C PCB layout	67
D Components	73
D.1 Components	73
D.2 Datasheets	75
E Source Code	83
E.1 Microcontroller	83
E.1.1 linkedList.h	83
E.1.2 likedList.c	84
E.1.3 msgHand.h	86
E.1.4 msgHand.c	86
E.1.5 pwm.h	88
E.1.6 pwm.c	89
E.1.7 Turn On and Off Valves	92
E.1.8 systemTask, checkMsg and sprayCommand	92
E.2 Valve Mapper Algorithm	94
E.2.1 setup.py	94
E.2.2 config.py	95
E.2.3 valveMapper.py	96
E.2.4 spraymap.py	100
E.2.5 navigation.py	101
E.2.6 com.py	103
E.2.7 valveMapper_tests.py	104
Bibliography	105

Chapter 1

Introduction

1.1 Background

Weed control is a crucial part of agriculture with potential for improvement. Today's most used methods are to control the weeds by applying herbicide or mechanically/by hand, which is very time consuming. When applying herbicide, usually the whole field is targeted, thus soil and crop are sprayed as well. Drop-on-demand (DOD) weed control is a field of research within precision agriculture where herbicide application is controlled down to individual droplets. If effective precision agriculture is achieved, the environmental loads and herbicide usage will be reduced dramatically, as only the weeds are targeted. Motivated by this, Adigo AS is currently developing an autonomous robot, Asterix, for weed control in row-crops. The goal is to only target the weeds, avoiding soil and crop. Inter-row weeds can effectively be controlled mechanically, so the focus for the DOD system is the intra-row weeds. A demo picture for the Asterix project is illustrated in Figure 1.1. The focus for this master's thesis is to design hardware and software for the DOD part of the application, including a printed circuit board (PCB) for control of the valves and an algorithm for generating spray commands. This master's thesis is a continuation of the project thesis Urdal (2013) where the first revision of the PCB was designed. A second revision of the PCB is to be designed and produced along with software for the real-time application. Tests will be conducted to examine the control strategy of the valve driver and droplet formation. For more information of the complete system and initial design consult Utstumo and Gravdahl (2013) and Utstumo (2011).



Figure 1.1: A demo picture of a spraymap for the Asterix project (Adigo AS)

1.2 Objectives

The main objectives for this thesis are:

- Test the remaining parts of the prototype PCB, improve the design and produce the second revision
- Complete a microsyringe system with the required surrounding system for pressurized spray liquid handling, communication and power
- Design and implement a valve control algorithm with a spray map and position log as inputs
- Implement software on the microcontroller for operating the valves
- Perform laboratory experiments with high speed filming and tests of drop formation, and prepare an IEEE style publication on droplet control and formation

- Implement the communication interface over RS-485 with both microcontroller firmware and PC-client

1.3 Structure of the Report

Chapter 2 covers a literature survey of the fields relevant for this thesis. Included are information and results from state of the art applications in precision agriculture and herbicide efficiency. Theory for modeling a fluid in a straight tube as an electrical equivalent and calculating the response is presented. In addition some theory regarding fluid properties for droplet formation and stability is presented as this is a crucial part for achieving good droplets.

Chapter 3 presents the overall system and some design challenges. This explains all the different parts of the application and concerns for the real time system and how this is handled to achieve a precise synchronization. In addition two valves of interest is compared before choosing the valve to be used in the final application.

Chapter 4 covers how the remaining tests of the prototype PCB was conducted, improvement of the design, assembly and testing of the second revision.

Chapter 5 presents the software for the microcontroller and the PC. The valve mapper algorithm have spraymaps and a navigation log as inputs and produces spray commands sent to the PCB. This includes some theory of how to make a python skeleton. The communication protocol used is described and software for handling the communication on both the computer and microcontroller is implemented. The valve driver software is implemented for the PCB to operate the valves from the commands received.

Chapter 6 includes testing of the initial valve and how the closing time of the PWM influences the tail of the droplet. Most of the test were done with the test valve, INKX0514300A. These tests were done to optimize the droplets. An IEEE paper was prepared on this topic for ICARCV 2014, notification of acceptance is 1st of July 2014.

Chapter 7 explains the laboratory experiments conducted and the results for each test. This include valve testing and a combined PCB and valve mapper test.

Chapter 8 discusses the results of the project and some future work than may improve the application. This includes control strategy, overall system, hardware, software and liquid properties.

Chapter 9 gives a conclusion of the master's thesis.

Appendix A explains the abbreviations used.

Appendix B includes the paper prepared for ICARCV 2014, the notification of acceptance is 1st July 2014.

Appendix C includes the layout of the PCB, the schematics is not included due to confidentiality.

Appendix D lists all the components used and includes some important datasheet pages of the most critical components.

Appendix E includes some of the source code implemented in this project.

Chapter 2

Literature Survey and Theory

The following paragraph and 2.1, 2.2 and 2.3 are from Urdal (2013):

Concerns for the human health and environment when using herbicides have led to legal regulations in several countries as described in Christensen et al. (2009). These regulations have resulted in new weed control technology emerging, as the Asterix project. For an autonomous system to work in the field, many different kinds of technologies are required. First of all a vision system is needed to detect the weeds. Secondly a navigation system is needed if it is not operated by a human. Finally a weed removal system, mechanical or chemical, is needed to treat the weeds. Asterix is an example of such a robot, and a picture of it in an early stage in the development process is shown in Figure 2.1.

2.1 Previous Applications

There are some similar systems to Asterix that have been developed in the past, for instance Lee et al. (1999) designed a robotic weed control system for tomatoes, Nieuwenhuizen et al. (2009) developed an automated detection and control system for volunteer potatoes in sugar beet fields and Midtiby et al. (2011) created a crop/weed discriminating microsprayer. Nieuwenhuizen et al. (2009) looked on the efficacy of a drop on demand/microspray application compared to flat fan nozzles. They decided on the microspray system, as did Midtiby et al. (2011) in their studies. Some advantages of the microsprayer is that off-target spraying can be avoided, which is not possible with the flat fan nozzles as described by Nieuwenhuizen et al. (2009). Thus spraying of the soil and crop can be avoided which is a major benefit for the environment and human health, as herbicide residue in the food will be avoided. The potential herbicide savings



Figure 2.1: An prototype of Asterix (Adigo AS)

of a drop on demand system compared to a conventional broadcast application could be $> 95\%$ according to Christensen et al. (2009). When using a drop on demand application, each nozzle can only hit a very limited area, and weighing of resolution versus complexity/price must be considered. Midtiby et al. (2011) did some calculations on the probability of hitting small plants. Six nozzles were used to cover a distance of 53 mm, which resulted in a 10.5 mm spacing between two adjacent nozzles. The hit probability of this system showed that it was suitable for targeting plants larger than $11 \text{ mm} \times 11 \text{ mm}$ but was unsatisfactory for smaller plants. Calculations like these are important in order to decide which stages in the growing period a system is effective.

2.2 Valve and Nozzle Limitation

Another important aspect in these kinds of applications are the specifications of the valves and nozzles. Nieuwenhuizen et al. (2010) observed some satellite droplets, that preferably should be avoided. Alexander Tallund Klungerbo, Klungerbo (2013), did research for Adigo AS on his master thesis to try to shoot droplets as far as possible without disintegrating, and avoid the presence of satellite droplets. Klungerbo (2013) used a VHS Dispensing valve (INKX0514300A) from The Lee Company with a nozzle with internal diameter of 0.254 mm. Klungerbo calibrated

the system to shoot with 4.0 ± 0.4 m/s. This was done by tuning the pressure on the fluid. Higher pressure does not only increase the velocity of the droplets, but the volume as well. The tests showed that he was able to shoot droplets of water a minimum of 39.5 cm before disintegrating, while Isopropanol only was able to shoot droplets up to 17.0 cm on the shortest. The frequency of which the valve can produce droplets also affects the driving speed of the application, given a required resolution. Nieuwenhuizen et al. (2010) had a required resolution of 100 mm^2 . The nozzles used could only operate at 80 Hz , thus limiting the driving speed 0.8 m/s . The droplet size and frequency are closely connected, as larger droplets require the valve to stay open longer, thus reducing the frequency. This is an important effect when deciding the number of valves and droplet size to be used.

2.3 Droplet Efficacy

Efficacy of droplet application of herbicide has previously been studied. The results from these reports are important for producing a successful system. Even if the application can hit all weeds with 100% accuracy, it could end up with not reducing the weed in the rows if the effect of single droplets is non-observable. Sogaard et al. (2006) did tests regarding efficacy of herbicide droplets on weeds. They used seeds of *Solanum nigrum* planted in pots under outdoor conditions, and a spray liquid consisting of water mixed with glyphosate (Roundup®, $360 \text{ g a.i. L}^{-1}$). The results showed that approximately $0.8 \mu\text{g}$ of glyphosate per plant reduced the biomass by 95%. Midtiby et al. (2011) used this result and a spray liquid with 5 g glyphosate per liter. Thus a droplet of only $0.2 \mu\text{L}$ should theoretically be required to efficiently control the growth of the weed seedlings. The droplets can thus be very small, and still be powerful enough to control the weeds. On the other hand, Sogaard et al. (2006) performed an outdoor field experiment as well. The micro-spray system had an efficacy of 82% when the average dose for each plant was $22.6 \mu\text{g}$, which is almost 30 times more than what they initially showed was needed. An exact amount of glyphosate necessary for controlling the weed is thus not straightforward to calculate, as the environment and precision of the placement of droplets seems to have a significant influence. However, each droplet in the field trial was $2.5 \mu\text{L}$ with $5 \mu\text{g}$ of glyphosate. Larger leaves were hit several times, causing the large amount of glyphosate. Leaf surfaces of less than 100 mm^2 had a low hit percentage, making the overall hit probability lower. 82% is the percentage of the weeds sprayed, and it seems like all those plants were controlled when they said the total efficacy was 82%. The amount of glyphosate could possibly be reduced and still give the same result. Thus making the conclusion of their field trial a bit misleading, as a lower amount

of glyphosate may have had the same efficacy. If it would be as low as $0.8\mu\text{g}$ as with the first test is hard to say, as the herbicide was applied manually, which results in better precision.

2.4 Modeling of Fluid Systems

When working with droplets, the time interval for the flow in the valve is limited to a few milliseconds. Thus a steady state approach for the liquid flow may provide a poor estimate of the droplet volume due to the response of the fluid. As Watton (1989) explains, a linear fluid system can be modeled with an electric circuit equivalent. The rest of this section is based on Watton (1989). Nonlinearities that are common in fluid power systems cannot be easily modeled, and fluid equations including pressure differential may lead to incorrect circuits according to Watton (1989). Watton (1989) says “*If the particular reference volume is a circular pipe of uniform cross-sectional area it follows from the flow continuity equation (3.6) that:*

$$Q_i - Q_0 = \frac{V}{\beta_e} \frac{dP}{dt} \quad (2.1)$$

where Q_i and Q_0 are the volume flow, $V = al$, $a = \frac{\pi d^2}{4}$ and β_e is the effective bulk modulus”. For the electric equivalent pressure equals voltage, while volume flow rate equals current. By following the deduction the result is, Watton (1989):

$$\begin{aligned} \text{fluid resistance} \quad R &= \frac{128\mu l}{\pi d^4} \\ \text{fluid capacitance} \quad C &= \frac{V}{\beta} \\ \text{fluid inductance} \quad L &= \frac{\rho l}{a} = \frac{M}{a^2} \end{aligned} \quad (2.2)$$

Furthermore two different circuit representations are presented, the π and T networks. The π network introduces an intermediate flow, while for the T network introduces an intermediate pressure. In a π network, the time derivatives is the pressures and the intermediate volume flow, while in the T network the flows and intermediate pressure is used as derivatives. Since the pressures is know and fixed, the volume flow can be calculated by choosing the T network.

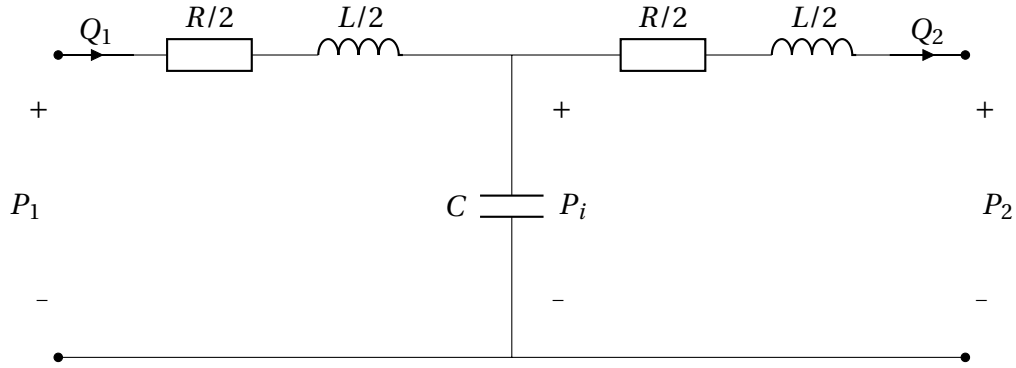


Figure 2.2: A modified version of the T network from Watton (1989).

The following equations then represents the system:

$$\begin{aligned}
 Q_1 - Q_2 &= C \frac{dP_i}{dt} \\
 P_1 - P_i &= \frac{R}{2} Q_1 + \frac{L}{2} \frac{dQ_1}{dt} \\
 P_i - P_2 &= \frac{R}{2} Q_2 + \frac{L}{2} \frac{dQ_2}{dt}
 \end{aligned} \tag{2.3}$$

where the electrical circuit equivalent is presented in Figure 2.2.

Using this theory, a response for the flow out of the nozzle can be calculated. If the time constant is in the millisecond range, the flow may be limited, as the valve is open only 8 ms at a time. Thus some knowledge about the time constant can be crucial to decide for the right valve and nozzle diameter.

The valves used in this project is provided by The Lee Company. They have another definition for calculating flow in a valve. They operate solely with fluid resistance, or Lohm as they call it. Every valve and nozzle have it's own Lohm value, and formulas for calculating serial and parallel resistance, pressure drop and flow are presented. Some of their definitions is included in the following section, as their theory combined with the modeling above can be used to calculate an estimate of the dynamic response of the liquid in the nozzle. As The Lee Company only includes resistance in their calculations, they are restricted for steady state solutions only. However the equations can be used to calculate the pressure drop over the valve and/or nozzle when only the total pressure drop is known. The following definitions are taken from The Lee Company (2014). The most important definition is Lohm, and the valve/nozzle is defined to have a value of 1 Lohm if it will flow 100 gallons of water, with a temperature of 80° F, per minute

when the pressure drop is 25 psi. Alternatively it can be written, The Lee Company (2014):

$$\text{Lohms} = \frac{100}{\text{flow (gal/min } H_2O @ 25 \text{ psi)}}$$

When combining restrictors, one formula for parallel and one for series flow is presented. In this project, only the series restrictor is relevant. The differential pressure is not linear, thus the series restrictor is not equal to the electrical one, but follows this formula:

$$L_T = \sqrt{L_1^2 + L_2^2 + L_3^2 + \dots + L_N^2} \quad (2.4)$$

In order to use these definitions, a relationship between Lohm and flow must be established, in this case Lohms law. It is defined as the following:

$$I = \frac{KV}{\text{Lohms}} \sqrt{\frac{H}{S}} \quad (2.5)$$

where I is flow rate, H differential pressure, V viscosity correction factor ($V = 1$ for water at 80° F), S specific gravity ($S = 1$ for water at 80° F), K is a constant to take care of units and measures, for instance, when using psi and gpm (galons per minute) $K = 20$, and for bar and L/min $K = 288$. V can be found for a number of different liquids at a wide temperature range in a table provided by The Lee Company (2014). Using this the pressure drop over the nozzle can be calculated to get an estimate of the volume flow. Calculations with the initial setup used in Urdal (2013) is presented in the following section. First calculations to find the pressure drop over the nozzle must be conducted. Instead of calculations by hand, a calculator provided by The Lee Company (2014) was used. It is assumed that the operational conditions is a total pressure drop of about 0.4 bar with water at room temperature as the liquid. The solenoid valve has a Lohm value of 4750, while the nozzle has a Lohm value of 7500. The total Lohm becomes:

$$L_T = \sqrt{L_v^2 + L_n^2} = \sqrt{4750^2 + 7500^2} = 8877.6 \text{ Lohms} \quad (2.6)$$

The resulting flow rate then becomes 20.5 mL/min. When solving for upstream pressure for the nozzle the result is 0.271 bar. Thus, the pressure over the nozzle is calculated and theory from Watton (1989) can be used to get the flow response. The resistance, capacitance and inductance

is calculated from equation 2.2:

$$\begin{aligned}
 R &= \frac{128\mu l}{\pi d^4} = \frac{128 \times 8.90 \cdot 10^{-4} * 8.89 * 10^{-3}}{\pi * (0.254 * 10^{-3})^4} = 7.75 * 10^{10} \\
 C &= \frac{V}{\beta} = \frac{\pi r^2 l}{\beta} = \frac{\pi * (0.127 * 10^{-3})^2 * 8.89 * 10^{-3}}{2.15 * 10^9} = 2.1 * 10^{-19} \\
 L &= \frac{\rho l}{a} = \frac{997.05 * 8.89 * 10^{-3}}{\pi * (0.127 * 10^{-3})^2} = 1.75 * 10^8
 \end{aligned} \tag{2.7}$$

Rearranging equation 2.3 results in:

$$\begin{aligned}
 \dot{P}_i &= \frac{Q_1 - Q_2}{C} \\
 \dot{Q}_1 &= 2 \frac{P_1 - P_i}{L} - \frac{R}{L} Q_1 \\
 \dot{Q}_2 &= 2 \frac{P_i - P_2}{L} - \frac{R}{L} Q_2
 \end{aligned} \tag{2.8}$$

where $P_1 = 0.271 \text{ bar} = 27100 \text{ Pa}$ and $P_2 = 0 \text{ bar} = 0 \text{ Pa}$. Using Simulink to run a simulation of this system results in the volume flow response, Q_2 , seen in Figure 2.3. Some assumptions for generating this plot is that P_1 is constant (in reality it would be a function of the valve opening, it takes about 0.3 ms to achieve a full opening), and that P_i starts at half the pressure of P_1 . In practice P_1 would have its own response during the 0.3 ms it takes to open the valve. As can be seen, the steady state solution is not reached when the valve is open for 8 ms. The steady state value is reached at 27.79 ms with a value of 20.9805 mL/min. This steady state solution is slightly larger than when using the calculator from The Lee Company (2014), which could be an result of rounding errors or model differences. Integration of the results shows that the total volume of the droplet is 72.6% compared to the steady state solution. This is a significant reduction in volume, that may have to be accounted for. However, if the valve is set to be open for 2 ms the value drops to 33.7%. Then the nozzle would have to be replaced by one with a larger diameter to get a faster response. Thus the response have a large impact on the total flow when the frequency of the valve increases, as this limits the time the valve is open. In practice the flow probably would be a bit lower than calculated as the pressure response in the valve is ignored.

2.5 Droplet Formation

A droplet produced by a DOD system consists of two or three sections, the main droplet, the filament and a tail. The filament is a cylindrical stream of flow following the main droplet, while

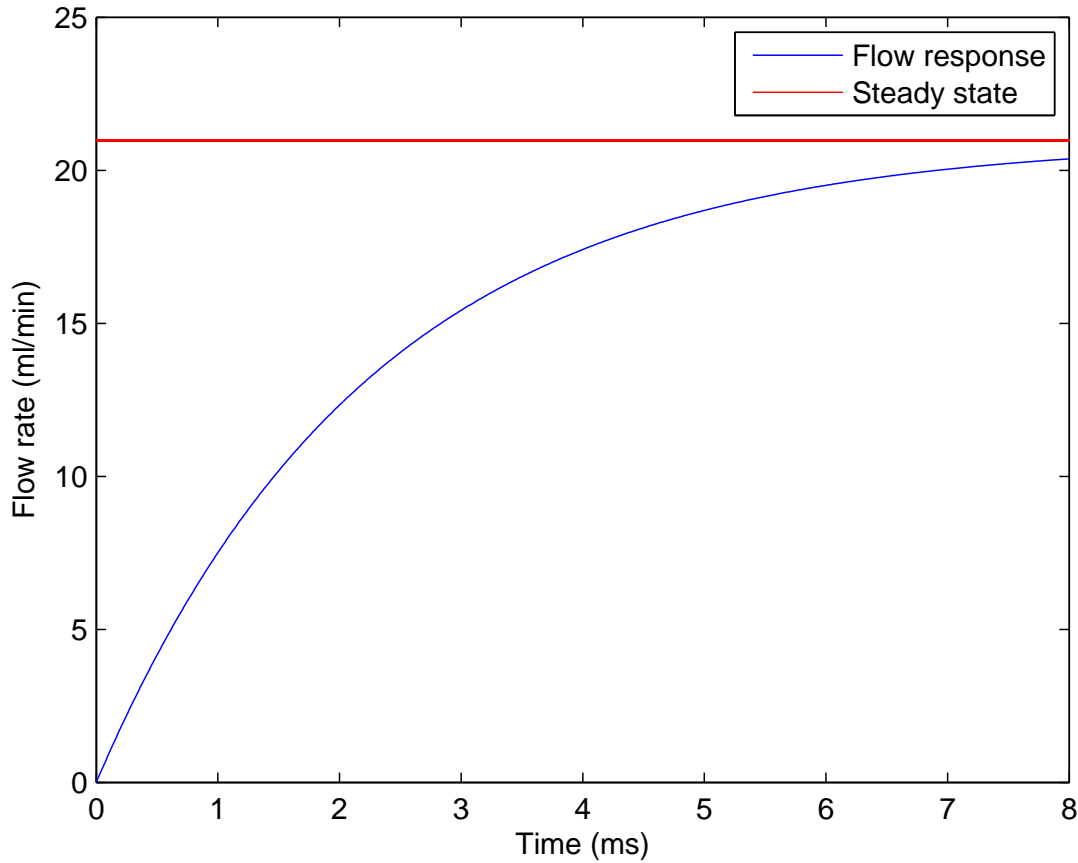


Figure 2.3: Flow response for the nozzle ($d = 0.254$ mm and $l = 8.89$ mm) with a pressure drop of 0.271 bar.

the tail is a thin flow behind the filament. The different parts are illustrated in Figure 2.4. For more information consult Vadillo et al. (2010).

The relative importance to the filament stability from surface friction and viscosity can be expressed through the Ohnesorge number, Hoath et al. (2013):

$$Oh = \frac{\eta}{\sqrt{\rho\sigma R}} \quad (2.9)$$

where η , ρ and σ are the viscosity, density and surface tension of the liquid, respectively, while R denotes the radius of the cylindrical filament. Furthermore, the initial filament aspect ratio, $\Lambda = L/2R$, will decide if the filament breaks up or not, L is the length of the filament. The critical value for filament breakup, Λ_c , increases with Oh , Hoath et al. (2013).

When the droplet is falling, a number of scenarios may occur: the filament may be absorbed into the main droplet, it may break at the main droplet thus creating a single satellite droplet or

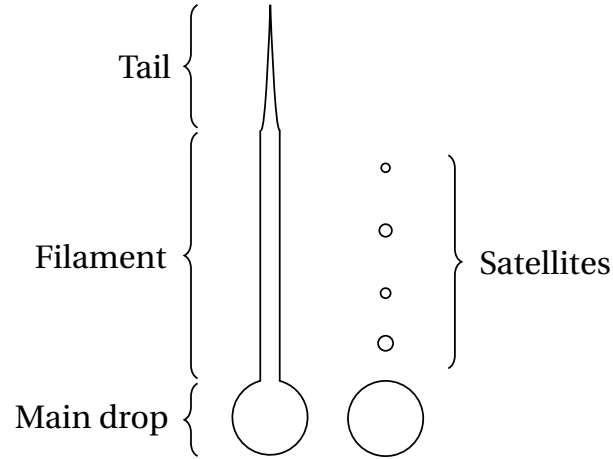


Figure 2.4: Droplet definitions

a Rayleigh-Platou instability may occur, creating multiple satellite droplets, Vadillo et al. (2010).

Satellite droplets are small droplets lagging behind the main droplet, often caused by the disintegration of the tail or filament. Without wind and other disturbances that could be present for a DOD application in movement, the satellite droplets will typically catch up with the main droplet and coalesce with it, and the dispensed fluid volume reaches the target as a single droplet, Dong et al. (2006). This is a result of less drag on the satellites as they are smaller and travel in the wake of the main droplet under ideal conditions. However, how the satellites will behave in the field is not certain. Most of the research described above are results from ink jet printers with droplets much smaller than what is needed for herbicide applications. However, Castrejón-Pita et al. (2008) verifies that the theory applies for larger droplets as well, which is more relevant for this project.

The filament break up if radial pinch off occurs before it has fully collapsed lengthwise. By treating the pinch off and lengthwise collapse as two independent processes an estimate of the critical aspect ratio Λ_c can be calculated, Hoath et al. (2013). Keller (1983) shows that the axial contraction speed is constant and equal to the Taylor speed given by:

$$V_T = \sqrt{\frac{2\sigma}{\rho R}} \quad (2.10)$$

Thus the axial contraction is completed at time:

$$t_T = l/V_T = \sqrt{\frac{L^2 \rho R}{8\sigma}} \quad (2.11)$$

By simplifications the radial collapse time is found to be, Hoath et al. (2013):

$$t_R = \alpha \left(\frac{\eta R}{\sigma} \right) \quad (2.12)$$

where α is a numerical factor. Assuming that $t_T = t_R$ results in $\Lambda_c = \sqrt{2\alpha} \text{ Oh}$.

When studying the droplets in air the Weber number is of importance, and is defined as, Eggers and Villermaux (2008):

$$We = \frac{\rho u^2 d}{\sigma} \quad (2.13)$$

Where ρ is the density of air, u the droplet velocity, d the droplet diameter and σ the surface tension. With the assumption of spherical droplets, the droplet is stable if its Weber number is below the critical Weber number, which lies between 10 and 40, Wierzba (1990).

The ideal solution for this DOD application is to have the filament merge with the main droplet without disintegrating while maintaining a stable droplet in flight. As can be seen from the theory above this is not straight forward. To minimize the filament breakup the surface tension of the liquid have to remain low, while the viscosity should be high. However, a low surface tension decreases the Weber number, and results in a less stable droplet in flight. Thus carefully choosing the liquid properties is of importance for generating a droplet with as little filament breakup as possible, while maintaining a stable droplet for about 15 cm in flight for this application.

Another aspect to consider is how the liquid properties influences the droplet-plant interaction. Rioboo et al. (2002) describe this process by five different phases: kinematic, spreading, relaxation, wetting and equilibrium. In addition to the liquid properties the surface properties have a significant influence on how the droplet will behave. The surface tension of the liquid is reported to have no influence on the kinematic, spreading or relaxation phases, but a significant impact in the wetting phase, Jung and Hutchings (2012). Higher surface tension results in a smaller final diameter in the equilibrium phase. However, the viscosity of the liquid did not influence any of the phases according to Jung and Hutchings (2012). In addition Jung and Hutchings (2012) report that the wettability of the surface only influences the wetting phase. All these tests were performed on different kinds of glass surfaces. The velocity had no effect on the wetting phase, but influenced the diameter of the droplet in the spreading phase. The weeds to be sprayed in the Asterix project have different surfaces, some have a wax like surface while others have a more hairy surface. Hence one liquid may work well for one kind of weed, but not for another. Common for all weeds is that a low surface tension is desirable for covering as large

an area as possible in the equilibrium phase. However, this will lead to a more unstable droplet as discussed above. Consequently some more research on how to achieve effective weed control with DOD systems should be conducted to achieve the best solution possible.

Chapter 3

Overall System

In this chapter the architecture of the overall system is represented along with challenges and solutions for getting the real time application to perform as specified.

3.1 System Architecture

First of all, a list of necessary elements for this application is presented.

- A vehicle for transportation
- Navigation sensors consisting of GPS (Global positioning system), IMU (Inertial measurement unit) and wheel odometry
- Camera for both weed detection and navigation purposes
- Computer for autonomous operation of the vehicle, processing navigational data, image processing and generating valve commands.
- A PCB with valve controller hardware and software
- Microsprayer system, including pressurized liquid tank, for both herbicide and water, and solenoids with tubings and nozzles.

How these modules is connected and interacting is presented in Figure 3.1. This illustration includes some of the timing requirements and delays. These times are to illustrate how the system will perform, and may be changed in the future.

Some requirements for the application crucial for this master's thesis are:

- Driving speed of up to 3 km/h, testing with 1 km/h

- Control resolution of about $6\text{ mm} \times 6\text{ mm}$ or better
- Accuracy better than the resolution of control so each droplet hit inside its target
- Droplet volumes of minimum $1\ \mu\text{L}$

One of the main problems for this application to work as specified is timing and delays. So far, the resolution of control is set to about $6\text{ mm} \times 6\text{ mm}$, while driving speed is up to about 1 m/s . If the accuracy of the timing lies between $\pm 1\text{ ms}$, the deviation for the droplet accuracy will be $\pm 1\text{ mm}$ or $\pm 16.7\%$. Thus the timing has to be very accurate, especially if the velocity is required to increase in the future.

The computer processes the images along with the navigational input. The spray map generation and valve mapper algorithm is implemented on the computer as well. The computer does not satisfy the strict real time constraints, microseconds accuracy, as complete control of the timing is not achieved. This inaccuracy includes varying delays on the interfaces. This is a major concern as the timing is crucial in this application, effectively requiring the computer to be outside the most critical part of the system. The resulting design was achieved by connecting the valve controller to the camera, receiving a trigger for each image taken. This solution simplifies the design significantly, while it is able to follow the strict timing requirements.

The trigger from the camera must also be sent to the IMU to get an accurate position stamp for each image. More on this synchronization is described in Chapter 3.3. Another aspect that has to be carefully measured/calculated is the propagation delay from the valve control signal initiated by the microcontroller, until the valve is open. This not only includes the propagation delay in the electronic control signal, but the time constant of the solenoid circuit as well as the plunger response time.

3.2 Valve Drivers

The design chosen for driving 26 Valves with one microcontroller is quite elegant. Since the valves are operated by a pulse-width modulated signal (PWM) it would usually take several microcontrollers, since most do not have that many PWM outputs. This is circumvented by only using two PWM signals in addition to 26 enable signals. That way an AND-gate with the one PWM and enable signal as inputs achieves the desired effect. The other PWM signal is used as a common signal for all valves. Thus the enable signal must be high for the valve to be operated. This solution limits the operations of different valves to be performed at the same time, but this is something that would be desirable anyway in order to keep the system architecture as simple

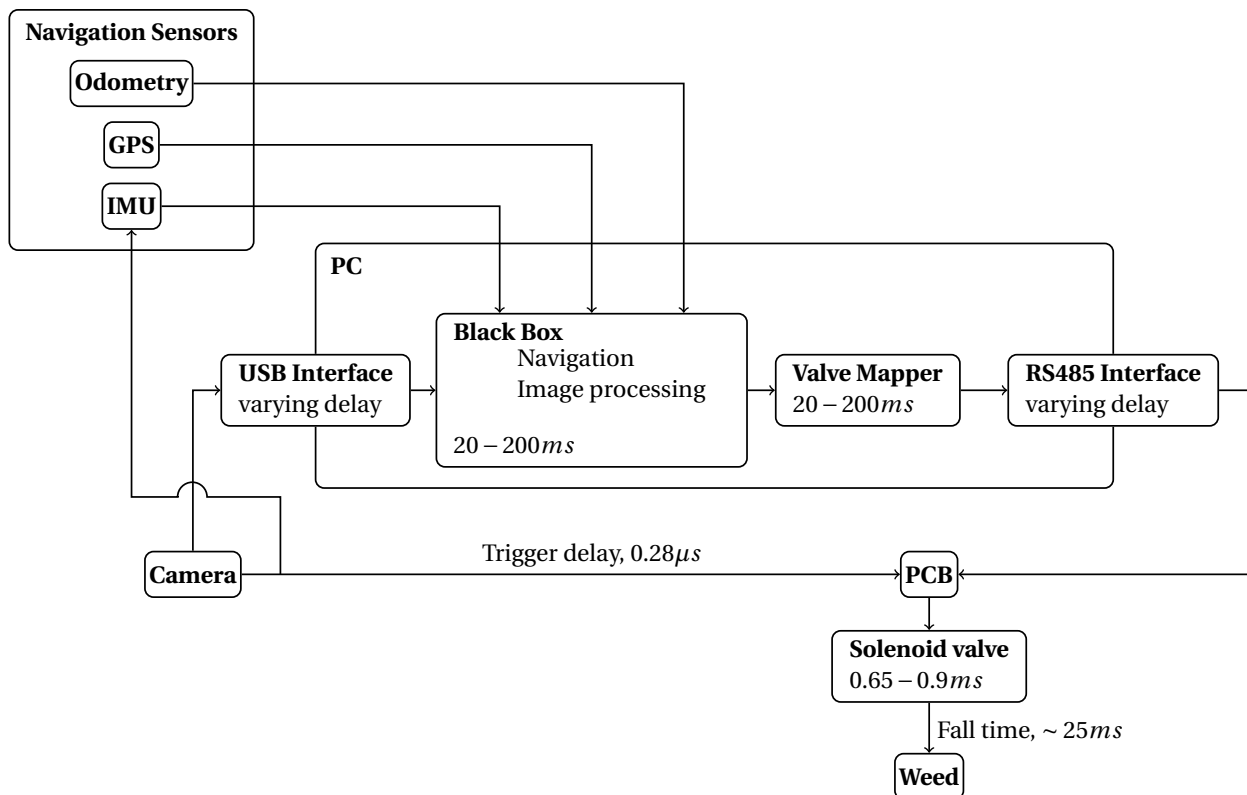


Figure 3.1: Representation of all the components for the overall system, with information flow as presented by the arrows. Crucial delays and timing requirements are included

as possible while still achieving the desired resolution of control. If the system should try to take advantage of having the valves operating at different times, the spraymap could not consist of rows and columns, probably making the control strategy too complex and time consuming for a real time system. In addition the amount of messages sent between the computer and the microcontroller could be almost 26 times as many in a worst case scenario. In total the code needed to run on the microcontroller is thus reduced to a minimum, resulting in a system with less probability for exceeding time limits. In practice all the valves are controlled by a full H-bridge, however as half the bridge is common for all valves only 27 half bridges are used instead of 52.

For a precise and fast operation of the valves, a spike of 24 V for about 0.3 ms is required for opening the valves. This is to increase the current in the coil, resulting in a larger magnetic field, which in turn overcome the spring force faster. When the solenoid is open, a voltage level of minimum 3.5 V is required to hold it open. This limits the current, reducing the probability for damaging the coil because of excessive heating. Another advantage of a lower current is that the energy stored in the coil is reduced, so closing the valve will be faster as the time for reducing the energy in the coil will be shorter. When closing the valves, the solution used here is to invert the voltage to reduce the current to zero. Another solution would be to have two schottky diodes in reverse series. The diodes could be chosen to 50 V, thus closing the solenoid faster than with 24 V. A more detailed description of the tests and pros and cons can be viewed in the paper prepared for the ICARCV 2014. This paper is included in Appendix B. The solenoids used in this application is non polarized 2-way normally closed. This means that the current can flow both ways through the coil. This way a negative current will not keep it closed, but actually open it as for a positive current. When closing the valve, a fast reduction in the current is necessary to make the tail as little as possible. However, if the voltage applied is for a too large duration, the solenoid will open again/stay partially open. Some calculations can be done to get an estimate of the duration needed. The general solution of the first order response for an RL circuit is given here:

$$I = I_0 + (I_1 - I_0)(1 - e^{-t/\tau}) \quad (3.1)$$

By inserting the various time steps and voltages chosen for this design, the resulting currents is presented in Table 3.1 and a simulation of the current in the coil under optimal conditions can be viewed in Figure 3.2. The valve used in these experiments was the INKX0514300A with the INZA4710975H nozzle. The inductance in the coil is 12 mH while the resistance is 40Ω, this results in a time constant $\tau = 0.3$ ms. Similar calculations for the schottky diode closing results

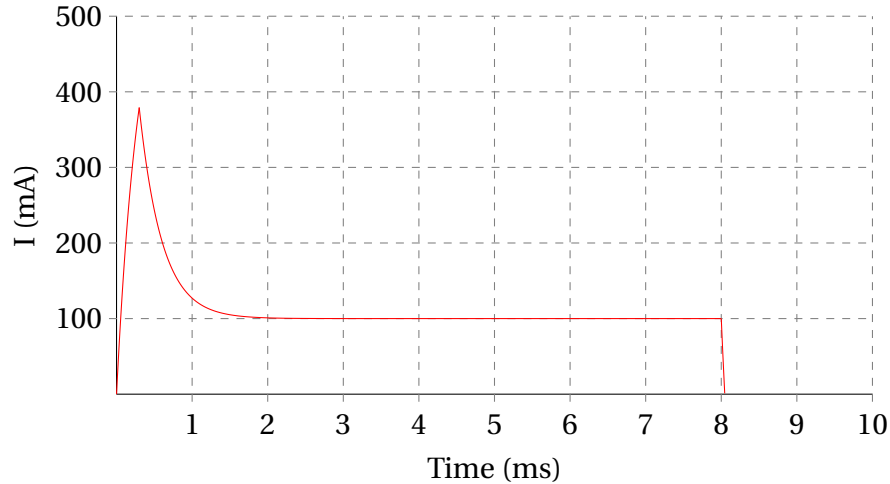


Figure 3.2: Simulation of the current in the solenoid during operation

in a closing time of 0.0231 ms, where the time for the PWM solution is 0.0463 ms. This is about half the time, however the time with the PWM solution is still short enough that it should yield a good result. The PWM solution is also a much more flexible design allowing for new valves by minor adjustments in software. In addition the diode solution can be applied by just adding two diodes to the PWM circuit, while this is not the case the other way around.

Table 3.1: Current in the coil for given points in time.

I_0 (mA)	I_1 (mA)	t (ms)	$I(t)$ (mA)
0	600	0.3	379.3
379.3	99	8	99
99	-600	8.0458	0

3.3 Time Synchronization

One of the largest concern when designing the PCB for this system was to achieve a precise enough synchronization between all the different parts of the application. The images must have an exact time and position stamp for the valve controller to be able to execute the commands at the correct time. In addition the position estimate must be very accurate and all modules must be synchronized. The USB interface from the camera to the computer have a delay that can vary due to operations executed on the computer while the data is sent. The same problem will appear on the RS-485 interface as complete control over the delay is not achievable. In addition the microcontroller should be synchronized with the rest of the system with

microseconds accuracy as this would influence the timing of the valve driver. Even if the synchronization between the computer and the microcontroller was perfect, the unknown delay on the interfaces may lead to droplets missing their targets as a little delay can influence the system too much.

The timing uncertainty would be a major concern for the application as it is unknown. So even if the system was synchronized quite well, the practical implications would require a reduction in the vehicle's velocity to compensate for the uncertainty. In addition, clock drifting would require the system to be synchronized with a given time interval. The solution in this case was to connect a trigger signal from the camera to the IMU and microcontroller so the exact time of the images is known to the crucial modules. This is an elegant solution that keeps the computer out of the strict real-time loop, avoiding the synchronization issue. The solution can be seen in Figure 3.1. Thus the computer does not need to be synchronized with the rest of the system and the interface delays do not influence the timing of the commands. The computer then calculates the delay from the image was taken to when the valves will be at the locations of the weeds. This time difference can then be sent to the valve controller with a bitmap of the valves to be operated at that time.

With the solution chosen for this application the computer only processes the data without concerns of the absolute times. Although it is not synchronized with the rest of the system the algorithms implemented must have a time limit. This limit is a function of the distance between the camera and the valve array and the velocity of the vehicle. The images must be analyzed for generating a spraymap before an algorithm processes the spraymap to spray commands. In addition the commands have to be sent to the valve controller a given time before they should be executed so a delay on the communication interface does not break the timing constraints. Although this solution requires some constraints with regards to timing, the accuracy of the system will be much better than without the trigger between camera and valve controller as the synchronization problem with the computer is avoided. This solution is also less complex as keeping track of the trigger signals and times can be done easily and efficiently on the microcontroller.

A simple timing diagram representing some key aspects of the system solution is represented in Figure 3.3. The trigger is active low, and the delay from the camera starts shooting to the trigger is important for compensating the time on the microcontroller. The delays are represented, but the time may vary. The time given for the black box is the requirement set for the algorithm to process the image taken and produce a spraymap. The valve mapper is represented as always high after the initial image as it will process and produce commands whenever a new image is present. The only element from the navigation part that is included is the response of the trig-

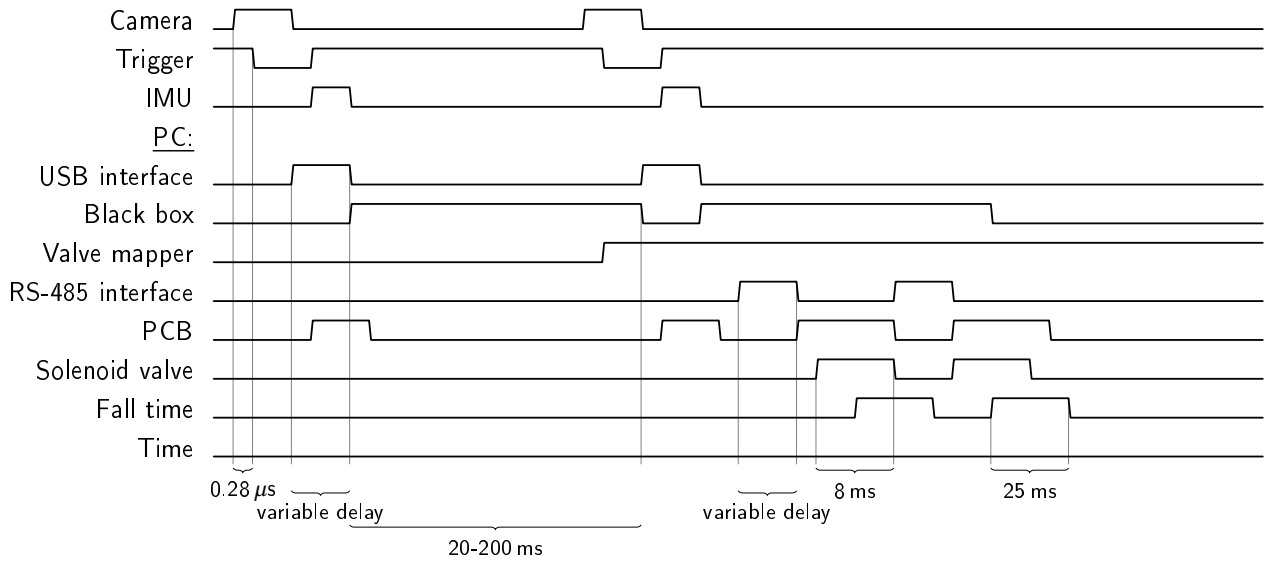


Figure 3.3: A timing diagram for representing some key features and how they work along side each other. The PCB reacts on the Trigger and input from RS-485. The time of the solenoid includes response time for both closing and opening the valve.

ger for the IMU. The rest of the navigation algorithms and messages will run continuously in the background as well, but are not represented in the figure as the goal is to show how the synchronization is achieved. The IMU will log the position of each image, thus the computer knows the exact time and position. This is crucial for the synchronization of the system. This way the algorithm needed for valve control only deal with relative times, leaving the camera, IMU and PCB to handle the strict timing requirements.

3.4 Valve Alternatives and Selection

When deciding on a valve for the final application, two valves from The Lee Company were relevant, the VHS-series used for earlier testing, and the INK-series. Both series are 2-way normally closed valves. Thus a current must be applied for the valve to open, while it will close when the current source is removed. Several factors were weighted before landing on the final setup, this includes response time, Lohm, material, size, filtering and price. Both valves side by side is presented in Figure 3.4.

First of all, the pricing is a significant aspect. The VHS valve used for testing in Urdal (2013) was a so called 062 MINSTAC inlet / 062 MINSTAC outlet valve. The MINSTAC is a hinged connection mechanism, allowing for easily changing the nozzle or connect the inlet tubing. This valve costs about 400\$. If the MINSTAC connectors are removed the price drops to about 200\$

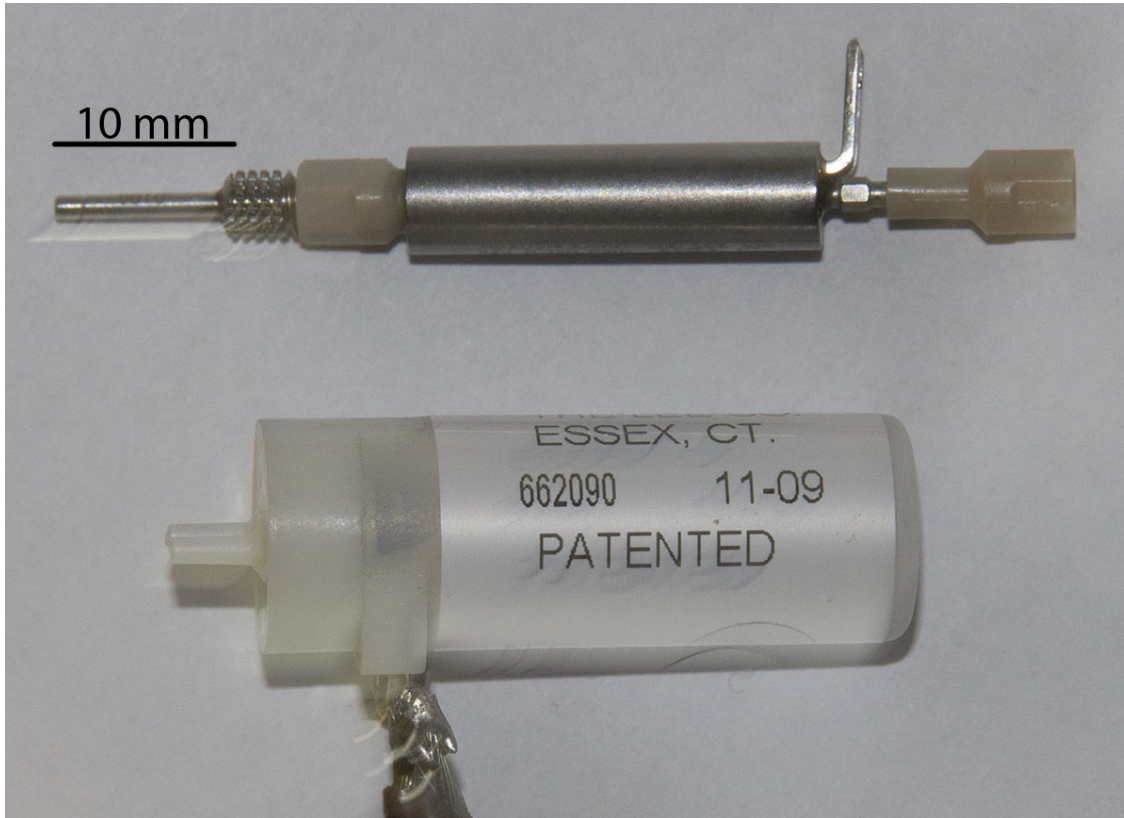


Figure 3.4: The INKX0514300A valve with INZA4710975H nozzle (upper) and the INKA2402160D valve (bottom).

for the VHS-series. The INK-series valves on the other hand, is about 100\$. If only one valve should be used the price may not have been a huge concern, however when at least 26 valves is to be mounted for each system the price difference becomes an important factor.

The response time of the valve is according to Buck (2013) the delay between the electrical signal and actuation of the plunger. The most important aspect to consider here is that the on time of the vale must be larger than the response time, if this is not the case the process becomes unstable and it will not be possible to produce droplets with repeatable volumes, Buck (2013). For all relevant valves, the response time lies between 0.25 ms - 0.9 ms. In order to get large enough droplets, the on-time for the valves is set to be around 8 ms, but may in the future be reduced to a minimum of 2 ms. Thus the response time is not a large concern for this application, as the minimum on time is at least 2 times the response time.

The material used could limit the possible choices as some of the fluids to be used are chemically strong. The VHS valves are available with higher chemical and thermal resistance than the INK-series, in addition to being superior with regards to mechanical strength. The INK-series should be able to withstand the fluids initially selected, but this may change after some testing

or introduction of other fluids. Thus both valves fulfill the mechanical and chemical resistance required for the application in its current state. Another downside with the INK-series valves is that they are about twice as big. This makes mounting on a PCB more cumbersome. A solution for this is to mount 13 on each side, a bit displaced horizontally so that they do not block the ones on the other side when soldering.

Maybe the most important parameter to consider is the Lohm value. The VHS series have a value of 4750 Lohm while the INK-series have a value of 1800 Lohm. Simplified Calculations were made to see how this will affect the flow. All calculations are done with water as liquid and on time of 8 ms. The flow response is not taken into account when the volume of the droplets are set, so the Lohm values have to be a bit lower to produce the actual volume. L_t is the total Lohm, while L_n is the Lohm of the nozzle, the values are presented in Table 3.2. The largest difference is for $5\mu\text{L}$, here it can be seen that the Lohm values for the nozzles can be much higher with the INK-series. The maybe most important conclusions from these calculations is that the INK-series provides more flexibility when it comes to producing larger droplets, or increasing the frequency. The on-time of the INK-series valves can be decreased compared to the VHS-series while maintaining the same volumes. Increased frequency can lead to an increase in speed of the vehicle while the resolution of control remains the same, or increase the control resolution if the velocity stays unchanged. Alternatively a combination of the two can be achieved.

The last aspect considered is the filtering of the fluids. The INK-series valves tolerate larger particles than the VHS-series. The recommended filtration for the INK-series valves is 35 microns, while it is 12 microns for the VHS-series valves. Thus the probability of clogging the valve is much lower with the INK-series. Altogether the price and Lohm value were considered much more important than the rest, so the decision so far is to use the INK-series. This allows for more flexibility of the overall system.

Table 3.2: Lohm values required for the nozzles for different pressures and volumes when on time is 8 ms.

description	L_t (Lohm)	L_n with INK series (Lohm)	L_n with VHS series (Lohm)
$2.5\mu\text{L}, 0.2\text{bar}$	6880	9552	8480
$2.5\mu\text{L}, 0.4\text{bar}$	9720	6640	4977
$2.5\mu\text{L}, 0.6\text{bar}$	11900	11763	10911
$5\mu\text{L}, 0.2\text{bar}$	3440	2932	N/A
$5\mu\text{L}, 0.4\text{bar}$	4860	4514	1028
$5\mu\text{L}, 0.6\text{bar}$	5950	5671	3583

Chapter 4

PCB Design, Second Revision

4.1 Testing of Prototype Card

Before a second revision was designed, all the parts of the prototype card had to be tested. The remaining tests to be done was the RS-485 interface and trigger circuit. The microcontroller was programmed with an ICD 3 unit using MPLAB X. A template for the microcontroller was provided by Adigo AS. For the RS-485 interface a little code snippet was programmed to light a LED (light emitting diode) if the received message was identical to the one sent from the computer. In addition the microcontroller was programmed to send "hello" over the serial link. Testing of the trigger circuit was only done using a battery as input signal as a camera was not available. The trigger output was not tested as the camera were needed.

4.1.1 Results for Prototype Card

Both tests for the RS-485 interface were successful. The LED was lit as expected, and the computer read "hello" from the serial link. The trigger test with battery was functional. A result summary is presented in Table 4.1.

Table 4.1: Results Summary

Function	Description	Status
RS-485 interface	Tested both sending and receiving	Tested OK
Trigger signal input	Only tested with battery, Camera not available	Test with Battery OK
Trigger signal output		Not tested
Daisy chain for trigger		Not tested

4.2 Improving the Design

The main parts of improving the design were to make a better card and correct some small errors that were found in the testing. In addition some minor design changes were made in order to ease the assembly process. One part that is important to notice is the trigger circuit. In order to allow a flexible application, the trigger circuit was adapted to work with most of the Point Grey cameras. That way the application is independent of what other hardware is connected, making it more flexible. It was decided that the camera should generate the trigger, making the design more elegant. This resulted in removing the input optocoupler, as the cameras need to have an external pull-up on the opto-isolated output. If this external pull-up was not to be supplied by the PCB, an external battery and resistor had to be used. The result was to use the card that receives the trigger to power the cable, while still obtaining the galvanic isolation as it is achieved by the output circuit of the camera. That way the trigger output from the camera and the PCB can be made identical, the PCB can then be triggered by another card or the camera itself without making any changes. Another advantage from this change is that all the cards will get an input trigger, making the software identical on every card, as no master is needed.

Another fix was to correct the PWM signals. On the prototype card every second PWM signal was inverted, thus only allowing for half of the valves to be operated at the same time. The following list includes all the changes made to the second revision.

- Removed 24V from pin 5 on the LM2671 circuits (internal pull-up makes the need for an external signal obsolete)
- Fixed the PWM signals that were inverted
- Marked the positive side of the capacitors in the LM2671 circuit to avoid confusion in the assembly process
- Some diodes were marked at the anode, this was changed to mark the cathode for consistency
- More space was added for a support mechanism for the valves
- Test points for ground were added
- Modified the trigger circuit
- Adding connections for the INK-series valves, while keeping the VHS series connections, that way the card can be used with both valves if a change of valve is required in the future

After the changes were made a quality assurance process were done on the modified parts to verify the changes. This verification was conducted by Jan Kåre Vatne, Trygve Utstumo and myself. The main parts inspected were the trigger circuit and PWM signals as this were the most crucial parts modified. The same procedure was followed this autumn for the project thesis and can be viewed in Urdal (2013). The PCB design last autumn was the first revision, therefore the test was more comprehensive as all the parts were inspected. This process is important for avoiding errors in the design as it much more time consuming and expensive to debug for faults and create new revisions.

4.3 Assembly

The assembly of a full card was done as described in Urdal (2013). This include applying the soldering paste by using a stencil and carpenter cutter blade. Placing of the components was done with a pincer, and a microscope was used where needed. Then the PCB was inserted in the reflow oven for soldering. How the temperatures and times were chosen for the different phases, preheat, reflow, and cool-down, are described in detail in Urdal (2013). The same components were used for the second revision, so there was no need of adjusting these parameters. When the soldering was complete some short circuits was removed and a capacitor had to be resoldered as it was out of placement before the soldering was initiated.

4.4 Tests and Debugging

4.4.1 Debugging and Fixes

First of all the PCB was tested to see if the voltages were correct. This was done by increasing the voltage and measuring the output of the regulators. The 3.3 V regulator did not output a high enough voltage. The resistance from 3.3 V to GND was about 6Ω , and should be higher. Examination of the 3.3 V part of the circuit was done to see if/where the partial short circuit was. The fault was found to be that the microcontroller was rotated 90 degrees, so 3.3 V and GND was connected through the microcontroller. This was fixed by removing and soldering a new microcontroller with the help of a microscope. The PCB was then tested again the same way. The voltage regulators, both 3.3 V and 12 V were functioning as the should. The rest of the PCB debugging is described in the next section.

4.4.2 Testing of Operational Functions

This sections include some of the tests performed to verify that the PCB was functional. When the power supplies were correct the first test was to see if programming the microcontroller with the ICD 3 circuit was achieved. This is because the microcontroller must be programmed to test the rest of the PCB. Next the LED's, both microcontroller status LED and valve enable LED's, were tested. Then a command was sent from the computer to initiate a purge process of valve 1-4. This test includes the receiver part of the RS-485 circuit as well as the AND-gates, bridge driver and transistors. The transmit part of the communication interface was done by sending a message to the computer with a one second interval. This message was read on the computer using a terminal window. The test for the RS-485 interface was done with a USB to RS-485 FTDI cable. All the above testing was done without finding any errors, so no fixes were required.

The initial camera was replaced by an uEye UI-2280SE camera. The flash output of the camera have an optocoupler output with a voltage limit of 30 V. This was connected to the trigger circuit, which has a 12 V pullup. Software for the camera was installed on the computer an the camera was programmed to produce an active low output for the flash when the camera captures images. The flash output was then logged on the microcontroller.

4.5 Results Second Revision

The finished board can be seen in Figure 4.1. Programming of the microcontroller was achieved and all the code executed as expected. All the LED's and driver circuits were functioning and communication was working as it should. The trigger circuit was able to log the triggers and functioned as specified. A result summary for the different parts of the PCB with comments is presented in Table 4.2.

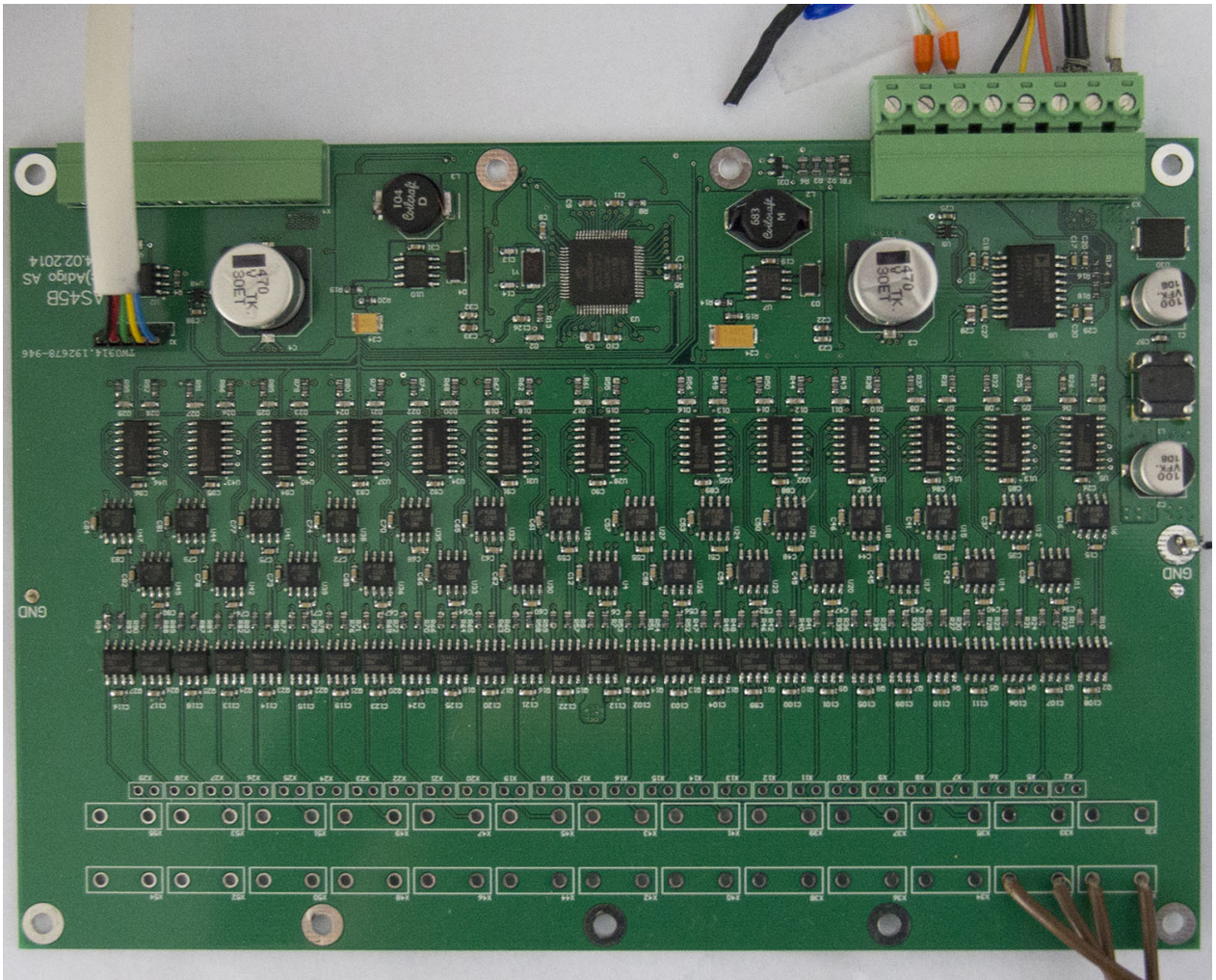


Figure 4.1: The assembled PCB with connections and wires to the valves.

Table 4.2: Results Summary

Function	Description	Status
Line Filter		Tested OK
3.3 V DC/DC supply		Tested OK
12 V DC/DC supply		Tested OK
Programming with ICD3		Tested OK
Microcontroller		Tested OK
Microcontroller stats LED		Tested OK
Valve indicator LED's		Tested OK
AND-gates		Tested OK
Valve control output		Tested OK
PWM driver		Tested OK
Valve spraying		Tested with some valves, OK
RS-485 interface		Tested OK
Trigger signal input		Tested OK
Daisy chain for trigger	Only one card was produced, so could not test this function	Not tested
Daisy chain for RS-485	Only one card was produced, so could not test this function. Direct connection, so should be no problems.	Not tested

Chapter 5

Software Design

In this chapter some assumptions for the software are presented, some to minimize the code needed on the microcontroller. This is in order to get a faster response and more predictive delays with regards to the spraying. First of all the knowledge of timing and content that is to be sent on the serial link is known. This allows for some simplifications, for instance the check to see if a new message has arrived do not need to consider all scenarios, allowing for a faster and less complex test. The assumption is that there will be no continuous stream of data, as each message will be followed by a pause. Another assumption is that you cannot get commands to spray two drops with an interval less than the shutter time, and that every command is sent in order. Thus no sorting is done on the microcontroller, and data from image one can be deleted when the first command with connections to image two has been executed. A requirement is that the shutter time can be altered during operation. This actually requiring a bit more code, but makes the application more flexible, as changes in the weather could require a different droplet size to maximize the effect of the herbicide. First the algorithm implemented on the computer is described, then communication and last the valve driver code on the microcontroller. Some of the source code is attached in Appendix E.

5.1 Valve Mapper

Before every scenario is accounted for, some simplifications are done. This includes that the following assumptions are true:

- The height from the nozzles to the weeds is known and fixed
- Pitch and roll are zero

- The yaw-rate is constant, that is $\dot{\psi} = 0$
- Time stamp for each spray map and orientation estimate are included.

Under these conditions the following subsections will describe the system and tasks that needs to be executed in order for the system to work properly.

5.1.1 Input

First of all, a navigation input have to be included for calculating the position of the valves. This includes position in NED, x , y and z (z is constant), in addition to the yaw angle, $\psi \in [0, 2\pi]$ and velocity in x and y direction, that is \dot{x} and \dot{y} . A demo picture for showing the orientation of the spraymap and valve array is illustrated in Figure 5.1. The spraymap have to include orientation, scale (pixels/m) and time of capture. The scale is needed in order to know the size of the area in meters. Although the figure have a large angel between image frame and valve array this angle is almost constant and the difference is about 90 degrees. This is due to both camera and array have a fixed position on the vehicle, which is driving straight forward.

5.1.2 Output

The output of the algorithm have to consist of the image number with a delay from the image is captured to the valve array is above the weed. In addition, the output will consist of a bitmap for the valves. That way all valves will operate at the same time. The output have to be ready $x \mu s$ before the command should be executed as unknown delays, including the RS-485 interface, must be considered. This is to make sure the command does not arrive after it should have been executed.

5.1.3 Internal Functions

For the algorithm to work properly a number of internal functions should be programmed to get a modulated program and more readable code. Some of the functions needed is to execute the following tasks:

- Rotate image coordinates to valve coordinates
- Estimate the area the valve array will cover

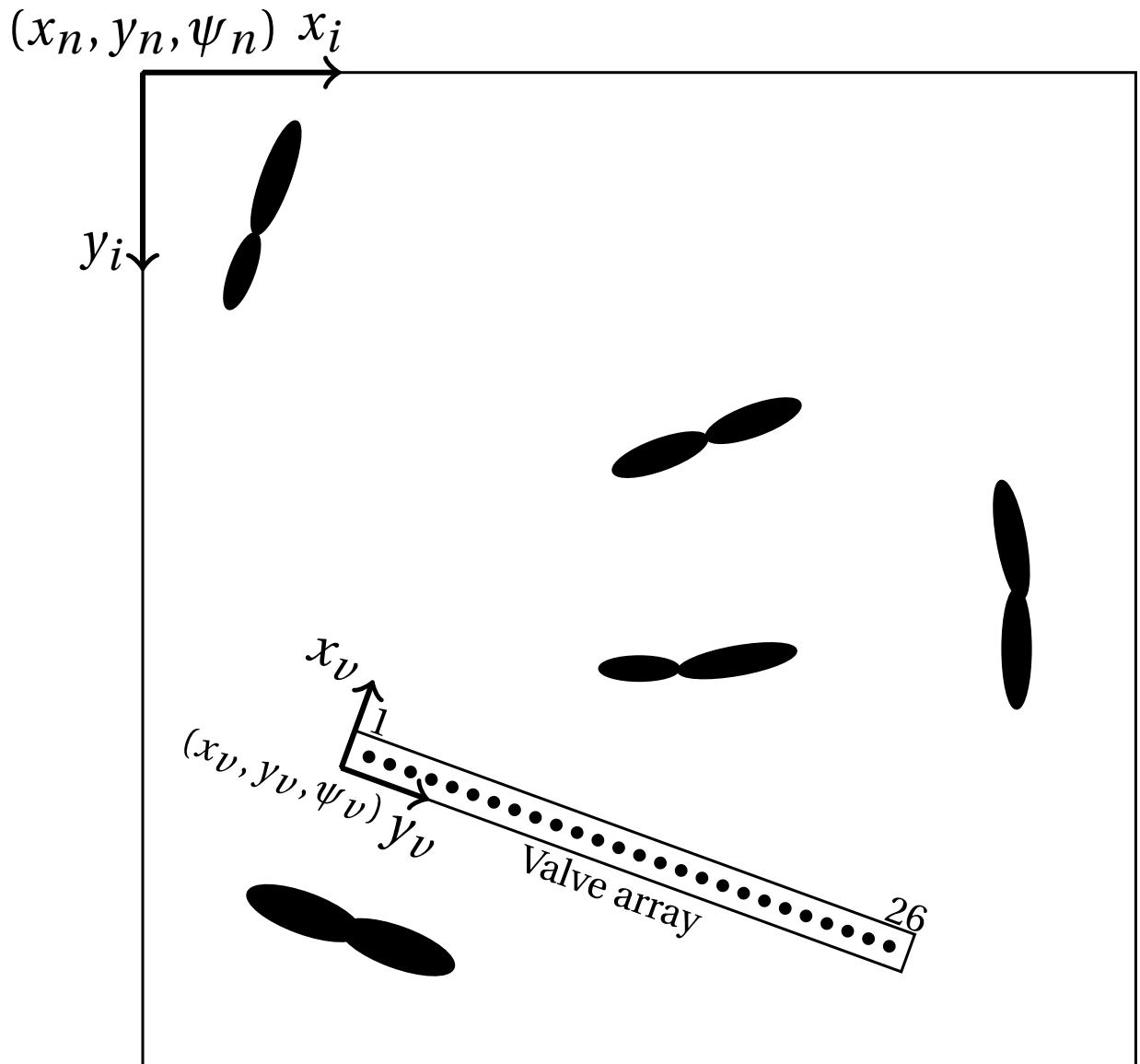


Figure 5.1: Demo of a spraymap and valve array with coordinates and orientations.

- Function for deciding what part of the image that can be ignored. For instance everything closer to the valve array than a given time, as the commands would not reach the valve controller in time to be executed
- Decide which image to use, calculate the distance from the valve to the middle of the images
- Divide the image into sections for each valve
- A spray decision for each valve, have to consider size of weeds, position and so on
- Communication
- An algorithm to decide that the distance between two spray commands is larger than the shutter time of the valve times the velocity of the vehicle
- A function for the maximum working distance, that is the maximum distance the algorithm should produce commands for. This is to minimize the effect of not knowing how the vehicle will behave in the future.

All the internal functions have to be programmed so that every case is accounted for. Initially this will be a simplified version with constant height and yaw-angle. Some important cases in the simplified model is:

- Ignore out of reach weeds
- Ignore weeds between valves
- Do not spray if the weed is too small
- Handle rotation, even when the yaw rate is set to zero
- Stop spraying if the robot is turning, for instance in the headland

Although this initially is a simplified program more or less every function can be used as it is, or just by modifying the core of the function, if a more complex algorithm is to be implemented in the future. It should not be necessary to include different input and output for the functions as all the data needed is already included. This was done on purpose as the goal is to include more features in the future. Testing of the accuracy of the system and how well the simplifications work in the first laboratory and field tests will influence what parameters that needs to be considered more carefully in later revisions.

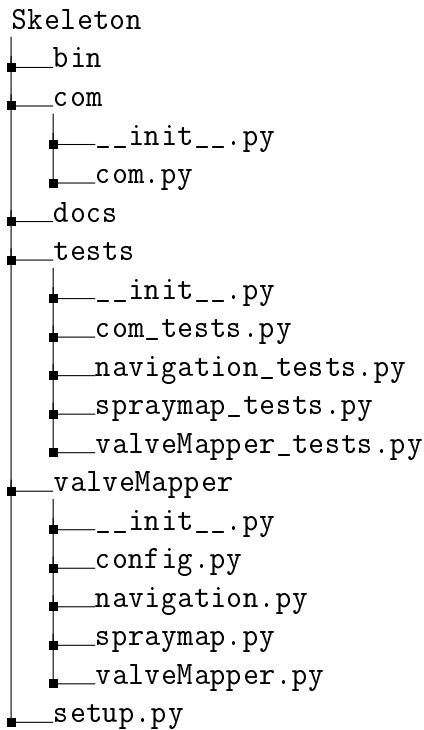
5.1.4 Python Skeleton

The algorithm was specified to be programmed in Python, and that a Python skeleton¹ should be used so the project easily can be implemented in other projects. This requires some set-up and installing of packages not needed for a single Python script. Packages and modules are created along with test files to check if there are some runtime errors. This set-up will be described in this section. For generating the folder structure some important libraries are needed. These packages are listed below and were installed using *pip*, a program for easily installing Python packages:

- distribute
- nose
- virtualenv

The next step was to create the folder structure and files needed for the different modules. The main folders needed are bin, docs, tests and package folders like *valveMapper*. In addition a setup script is needed in the parent folder. The result for this project was two packages, a *valveMapper* package with modules that are somewhat connected, and a *com* package used for communication. The *com.py* script was programmed for this special case and is essentially a wrapper for the messages to be sent and not a driver for the serial communication, as this was done using the *pyserial* package. The *com* package includes generating messages and checking the checksums. This could also have been a module in the *valveMapper* package as the serial communication was done using the *pyserial* package. The following list represents the final folder structure for this project:

¹for information and a guide to setup a skeleton: <http://learnpythonthehardway.org/book/ex46.html>



Every folder containing an `__init__.py` file is a package, and all python files in the package is a module. The *nose* package installed enables a `nosetests` command that runs through every file in the tests package and notifies if there are errors in the files. Here, each file have it's own test file. The *com* package is added as a separate package as the functionality is different from the rest of the modules. Global constants like the number of valves, length of the array and much more is specified in the `config.py` file. In addition some parameters as the PCB address, number and direction is specified. Thus the operation of several PCB's can easily be achieved by just adjusting the config file. Other important parameters specified in the config file are maximum percentage of crop and minimum percentage of weeds in the control square for spraying, frequency of the valve, maximum and minimum work distance for the algorithm and so on. The navigation module includes all the functions required for reading the navigation log and calculating rotation matrices, position and timing. The *spraymap* module handles everything with regards to the spraymap, including a function for finding the position of a given pixel in the NED-frame. The *valveMapper* module runs the main loop and handle the commands for the PCB. The source-code is attached in Appendix E.

5.2 Communication Protocol

For communication with the PCB, RS-485 protocol was chosen. Two different messages have to be sent over this interface, an initialization message and a command message. The operation is specified for this PCB only, so simplifications for achieving a less complex protocol can be used. The most important assumptions are listed next, including some requirements for the software sending the messages:

- Only two different messages, init and command
- There will always be a pause after each message
- All commands sent must be in the order they should be executed
- The init message is individual for each PCB
- One command message for all PCB's. Increasing the bitmap is all that is needed as every PCB knows what valves to operate.

The assumptions described above require just an easy protocol to be implemented. As the first field trials do not use multiple PCB's at the same time the functionality regarding variable bitmap size is not included in the source code. This can easily be adjusted for later revisions as only some minor changes are needed. The result for the initialization and command messages is detailed in Table 5.1 and 5.2.

Table 5.1: Init message from computer to microcontroller.

Byte	Description	Value
1	Init	'i' (0x69)
2	Length of message, from byte 3 to 10	0x08
3	Address	'A'-'Z' (0x41-0x5A)
4	PCB number	0x00-0xFF
5	PCB direction	0x00-0xFF
6	Valve shutter time in milliseconds, byte 0	0x00-0xFF
7	Valve shutter time in milliseconds, byte 1	0x00-0xFF
8	Valve shutter time in milliseconds, byte 2	0x00-0xFF
9	Valve shutter time in milliseconds, byte 3	0x00-0xFF
10	Purge valve	0x00 or 0x01
11	Checksum, sum byte 3-10, modulus 256	0x00-0xFF
12	End	'e' (0x65)
13	Newline	0x0A

Table 5.2: Command message sent from computer to the microcontroller.

Byte	Description	Value
1	Start	's' (0x73)
2	Length of message, from byte 3 to 11+n	0x00-0xFF
3	Address	'A'-'Z' (0x41-0x5A)
4	Image number, byte 0	0x00-0xFF
5	Image number, byte 1	0x00-0xFF
6	Image number, byte 2	0x00-0xFF
7	Image number, byte 3	0x00-0xFF
8	Time delay in microseconds, byte 0	0x00-0xFF
9	Time delay in microseconds, byte 1	0x00-0xFF
10	Time delay in microseconds, byte 2	0x00-0xFF
11	Time delay in microseconds, byte 3	0x00-0xFF
12 to 11+n	Valve bitmap for 8*n valves	0x00-0xFF
12+n+1	Checksum, sum byte 3 to 11+n, modulus 256	0x00-0xFF
12+n+2	End	'e' (0x65)
12+n+3	Newline	0x08

5.3 Microcontroller Code

This sections explains some of the functions implemented on the microcontroller. A template for the microcontroller was provided by Adigo AS, consequently, some drivers needed were not necessary to implement. That leaves the control software and all modules needed for controlling the valves. This includes message handling, valve drivers using PWM, trigger logging and linked lists containing image number, time and commands.

As it is assumed that all commands will arrive in order the trigger list is to delete the information about the previous image when references to the next one is executed. The list containing triggers and commands will always delete the first entry while appending new items last. The valve driver is programmed according to the requirements and explanations in Chapter 3.2. All the microcontroller code is implemented using C, and the MPLAB X IDE. MPLAB ICD 3 was used to program the microcontroller. Some of the source code is attached in Appendix E, but as the template provided is confidential not everything is included.

Chapter 6

Droplet Formation

This chapter focuses on how the solenoids should be operated for generating optimal droplets, and how different parameters affect the velocity and accuracy of the droplets. First the optimization for generating the droplets without satellite droplets and with a smooth tail is addressed in 6.1. Then the focus is shifted to how the droplets are affected by different pressures and shutter times in 6.2. The paper prepared for the ICARCV 2014 covers some of this data and is attached in Appendix B.

6.1 Droplet Optimization

Although the optimal solution for the timing according to the data sheet was found in chapter 3.2, a time resolution limit has to be set for the microcontroller. Weighing of the resolution versus the result was done in a series of tests.

6.1.1 Test Setup

For testing a high speed camera, PROMON 501 from AOS Technologies, was rigged for getting close-up pictures of the nozzle when shooting droplets. The camera was set to take 1000 pictures per second with a resolution of 800×128 pixels. A LED panel was set up behind the nozzle, pointing directly towards the camera to give enough light. The duration of the negative spike was set to 0.1 ms and 0.05 ms and images were taken in order to study how the spike duration influences the tail of the droplet. The tests done in this initial optimization can be seen in Table 6.1.

Table 6.1: Initial tests to optimize the droplet control.

Test number	Hold voltage (U_{hold} (V))	Spike period (t_{spike} (ms))
1	3.6	0.1
2	3.96	0.1
3	3.96	0.05

6.1.2 Results and Discussion

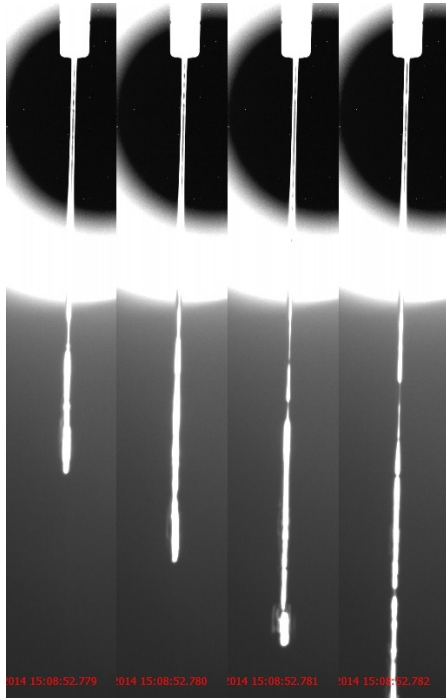
Some important images from the tests are presented in Figure 6.1a - 6.1d. ¹

First a comparison of Figure 6.1a and 6.1b can be conducted. The main difference can be seen close to the nozzle. The liquid jet is completely smooth for test 2, while test 1 shows sign of some variation, most significant in the second picture from the left. As the hold voltage for the solenoid is specified between 3.5 V and 4.5 V, the variation when using PWM can be enough to have a small oscillation in the dynamics, resulting in small variations in the valve opening. The PWM is much faster than the dynamics of the electric circuit, but when lying near the limit of the hold voltage, the variations may influence the system. This effect is not observed in test 2. If this is due to the hold voltage or other factors is not certain, but it seemed consistent when the tests were conducted. This little variation should be avoided, and increasing the hold voltage to about 4 V removed this effect.

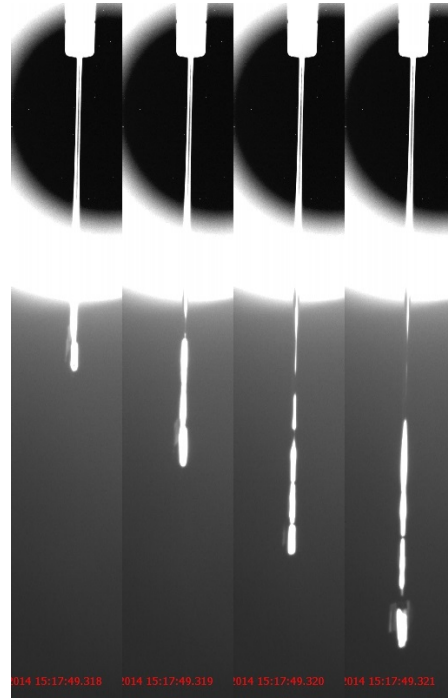
When looking at Figure 6.1c and 6.1d, the difference in the negative spike is clear. Instead of cutting the fluid jet instantly, a long secondary tail, that disintegrates into many small satellite droplets are formed. Some satellite droplets are observed in test 3 as well, but there are less of them, and the secondary tail is not observed. If calculating for the ideal circuit using Equation (3.1), the resulting current after a 0.1 ms spike is -99.15 mA. Thus the current is strong enough to open the solenoid again or stop it from closing completely as the response of the plunger lags behind the electric signal of the coil. The same calculations using 0.05 ms results in -8.31 mA, which is not enough to open the solenoid. This is reflected well by the images as described. Although the perfect length for the spike was calculated in Chapter 3.2, some adjustments have to be made for the real system. Checking the time to often may affect different parts of the software, as less time will be free for other tasks. Thus a desirable spike time will result in no secondary tail and be large enough to minimize the time spent executing that task in software.

Another method of adjusting the current after the spike is by increasing/decreasing the hold voltage, a little difference in the final current level after two spikes with the same duration can be achieved by adjusting the hold voltage. That way it may be possible to work with a larger

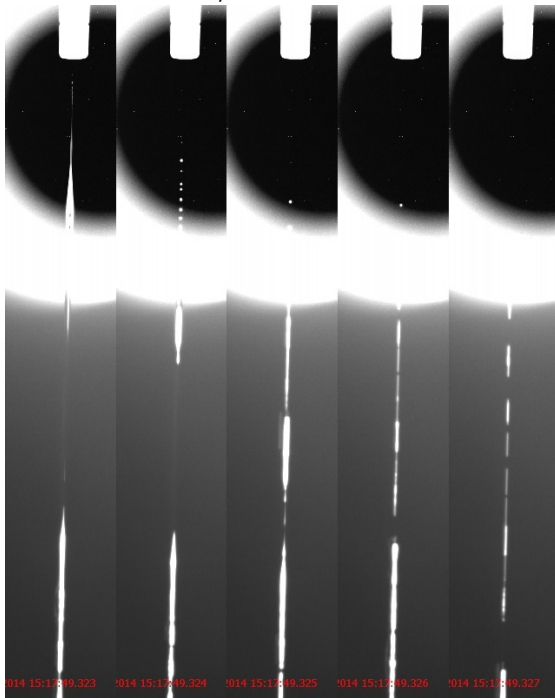
¹Two slow motion movies from the tests can be viewed on <http://folk.ntnu.no/frodeu/Movies/>



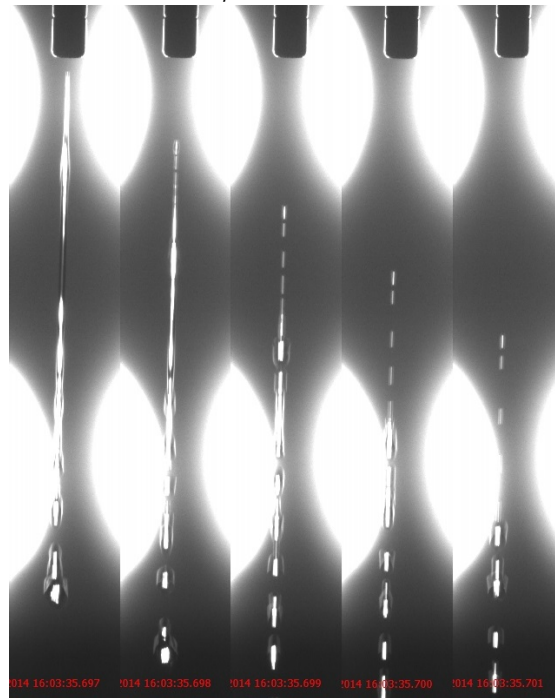
(a) Test 1, hold period, $U_{hold} = 3.6$ V,
 $t_{spike} = 0.1$ ms.



(b) Test 2, hold period, $U_{hold} = 3.96$ V,
 $t_{spike} = 0.1$ ms.



(c) Test 2, tail, $U_{hold} = 3.96$ V, $t_{spike} = 0.1$ ms.



(d) Test 3, tail, $U_{hold} = 3.96$ V, $t_{spike} = 0.05$ ms.

Figure 6.1: Three different tests for deciding hold voltage and spike time

resolution in the time, and tune the hold voltage to get the desired effect. If the hold voltage is raised to 4.5 V the resulting current will be -0.0031 mA instead of -8.31 mA, as calculated above.

This is not that significant, and a downside is that the larger current generate more heat, thus increases the chance for the solenoid to be burned off. As a result it is desirable to keep the hold voltage at a minimum.

6.2 Parameter Testing

6.2.1 Test Configuration

Some testing of how the pressure influences the initial velocity of the droplet was conducted. The valve was mounted with a ruler beside it. The same camera used for the tail test was used. This time the LED panel was placed above the camera and a white paper sheet was used in order to reflect the light back into the camera. This was done in order to be able to see the ruler on the image sequence.

6.2.2 Results

The results from the tests is presented in Table 6.2. The velocity of the droplet is of importance when hitting the weeds. A too large velocity may lead to the droplet bouncing off the weed or splatter may hit the soil or crop. On the other hand a too low velocity may reduce the precision of the droplets. The result shows that the relationship between pressure and velocity is not linear.

Table 6.2: Initial velocity for different pressures.

Relative pressure (Bar)	Initial velocity (m/s)
0.2	3
0.4	4
0.6	7.5

Chapter 7

Application testing

Testing of the valves and valve mapper algorithm was conducted to verify the design and code programmed. The field trials of the complete application is not included in this thesis as the tests are to be conducted after delivery of the project.

7.1 Valve Test Setup

The valve used in Urdal (2013) was replaced by another for the final system. The new valve used is an INK-series valve from The Lee Company. This valve had to be tested before a full set was ordered. The test setup consisted of a pressure container with a regulator on the output. This was connected to a bottle with water and further to the valve. The setup is illustrated in Figure 7.1. The pressure was set to 0.4 bar as this have proven to be a relevant pressure for this application. The system with the PCB was used for controlling the valve. The only test done was to verify that the valve was operating as expected.

7.2 Valve Test Results

The valve produced droplets as desired. This was as expected, but should be tried out before ordering a full set. The original liquid container was not available, therefore a bottle was used as a replacement. This result verified that the new valve can be used as the specifications suggested.



Figure 7.1: Test setup with pressurized liquid, valve and nozzle.

7.3 Laboratory Test Setup

For testing the PCB part of the overall system including the valve mapper algorithm a simple laboratory test was performed. This was done before the camera and all valves were available. The triggers were added in code instead of using the camera, and test spraymaps were used. The valve mapper algorithm ran on the computer, selected valves and sent the commands. Although the valves were not used, a status LED for each valve shows what valves were operated. This allows for testing without the valves. Timing of the commands were done using a stopwatch to see if the LED were actuated at the correct time. Later testing of the trigger was confirmed, therefore using the camera or adding the triggers in code would produce the same result.

7.4 Laboratory Test Results

The laboratory test described above was working as expected. The correct LED's light up at the correct time. Thus the code and hardware for the PCB and microcontroller have been verified. The only difference when conduction the real time testing is that all valves will be connected and the application will be moving. The algorithm have to include real spraymaps as well as a navigation log to produce the commands. The valves have been proven to operate as they should earlier, consequently the only change needed for the microcontroller code is to tune the duration of the negative spike to close the valves and the delay from the command is produced to the droplet should be generated. For tuning the delay the complete application have to be tested.

Chapter 8

Discussion

8.1 PCB

The remaining tests of the PCB were conducted before a second revision was made. The second revision mostly corrected some small design errors. The most significant change was the trigger circuit that was modified to work for a larger range of cameras. For the valve control strategy PWM have been chosen, another possibility is to use two shcotty diodes in reverse series to discharge the energy in the coil. The PWM solution can be modified to the diode solution by just adding the diodes, this is not possible the other way around. Although the camera initially specified was replaced the trigger circuit worked as expected. The optoisolated output of the flash control signal was used as trigger. The trigger circuit can be used for all cameras that have a optoisolated output that works with a 12 V pullup. This is quite common for industrial cameras, as a result the trigger design is very flexible with regards to the type of camera. The valve control strategy and how it affects the droplet is described more in the following section.

8.1.1 Valve Control Strategy

For the autonomous weed control application to work, the DOD system must be very accurate. The accuracy does not solely depend on target precision, but the presence of satellite droplets and their behavior. It is crucial that the satellites coalesce with the main droplet, or that they both hit the same spot. Therefore the droplets tail should be minimized, as the tail will split up in much smaller droplets than the filament.

The usual method for driving solenoid valves in this kind of application is by a spike and hold driver circuit, with two diodes in reverse series to discharge the energy in the coil. However, the described method is based on a full H-bridge, for PWM control. The maximum voltage is

chosen as the spike voltage, thus a long spike will open the valve before the PWM control limits the voltage to the hold value. When closing the valve, the energy in the coil is discharged with a significant negative spike. The spike time has been calculated using the first order response of a RL-circuit when the inductance and resistance of the valve is known.

Tests of this control strategy confirms the theory, as a long spike time resulted in a thin secondary tail, while a spike time of appropriate length avoided this. The long spike time started to actuate the plunger when it should be closed, by contrast the small undershoot for the appropriate time did not actuate the plunger at all.

Thus the spike time of 0.05 ms is close enough to the theoretical time for closing the valve. The main disadvantage of PWM control is the increase in time for discharging the energy, as the diodes can be chosen with a higher voltage level. This is not as easy with the PWM solution, as the higher the voltage, the more robust the components must be. This is due to the increase in voltage when using only one source will influence the robustness of the components, especially the transistors, used in the PWM control.

However there have not been observed any negative effects of the slower closing of the PWM solution compared to the Schottky closing. As a result an increase in the voltage is not necessary.

Breakup of the filament was observed in the tests, but this will occur regardless of the control strategy, and might be decreased with increased nozzle diameter. The theory and calculations regarding the valve control was confirmed by the experimental setup. The tail was avoided although the closing time is increased compared to the diode solution, thus the advantages of the PWM control strategy may be exploited. This includes a more flexible design with regards to the valves and fewer components are needed for the circuit. The main focus is to make sure that the negative spike time does not undershoot more than a certain level, as this will create a tail that should be minimized.

8.2 Overall system

For the overall system the main parts of the project have been to make sure the interfaces between camera, PC and PCB are functioning, as well as the PCB itself. Some discussion around the nozzle matrix is done, although the design and production are done at Adigo. The initial design have 6 mm between the nozzles, and the valves are connected directly in the nozzle array. Thus wires are needed between the PCB and the valves. The nozzle matrix have four inlets, two for water and two for herbicide. This minimizes tubing as the rest is done in the nozzle array itself. The design makes it easy to modify the nozzle diameter since it is only required to replace

a nozzle plate. Another solution would have been to mount the valves on the PCB, however that way extra tubing and an individual nozzle for each valve would be needed. That way it would require more work in order to tune the position for each nozzle. Additionally, if a different nozzle diameter is found preferable in the future, each and every one of the nozzles would have to be replaced, instead of just replacing one plate. Thus both modification of the nozzles and tubing is much less complex, while the result is identical.

It is important to notice the trigger circuit, as this circumvents a significant problem for the synchronization needs. This operation allows the PC to calculate with relative times, thus, there is no need of synchronization with the computer is needed. Without this trigger circuit the PC would have to be synchronized with the rest of the system, and the timing would have much larger uncertainties. The trigger connection is maybe the most critical part of the system for achieving precise dispensing of the droplets. Although the modules may have been connected in another way, the timing uncertainties would lead to a slower moving vehicle if the resolution of control should be constant. A faster moving vehicle is crucial for achieving an effective application. In addition the precise synchronizations allows for finer resolution of control, thus the system can be effective earlier in the season as it can target smaller weeds.

RS-485 was chosen for communication between the PC and PCB. This was done for easy and robust daisy-chain possibilities. That way several PCB's can be connected to get a wider nozzle array or finer resolution of control. This was to achieve a system that can easily be modified for other crop as no new designs are necessary. To verify that the overall system performs as specified and all components work well together some final field trials must be conducted.

8.3 Valve Control Algorithm

The valve control algorithm can in a way be compared to a printer. The problem in this application is that the future vehicle movement is unknown and the velocity and direction may differ. If every case should be accounted for, the algorithm would probably become too complex to work in a real time system. Some simplifications are done to achieve a faster algorithm, for instance that the height is constant, as well as the yaw rate. This is because the spraying will be conducted on straight rows of crops, in addition, pitch and roll are set to zero. When a new picture is available the algorithm loops through to find where the weeds are. Then calculations are done in order to estimate when the nozzles will be above the weeds, before it decides which valves which should be operated. This is decided by the coverage of weed and crop in the area covered by the given nozzle. The commands is then sent to the PCB including image number,

delay and valve bitmap. In the future some of the simplifications may have to be modified for the system to achieve precise enough droplet dispensing. Most of the functions used in this algorithm have included everything needed for a full design without simplifications, as a result, only the core of the functions should have to be modified as the input and output should already be specified as needed. In addition, some constants like weed percentage needed and max crop percentage may have to be tuned in order to achieve the desired effect. Full scale tests in the field have to be conducted to gather data for evaluating the algorithm.

8.4 Microcontroller software

The microcontroller software implemented includes message handling, valve drivers using PWM control, linked lists for trigger and spray commands, and some functions for controlling the main application and enabling of valves from a bitmap. The microcontroller code is implemented in C with a basis provided by Adigo. This made the use of RS-485 and setup of important functions straightforward. The trigger and commands are stored in linked lists that are compared too see if the delay of the command matches the given time. If this is the case the bitmap is enabled and the PWM signals generated. All commands received on the microcontroller is assumed to be sorted, therefore no code for sorting the lists are needed. The communication over RS-485 required a protocol for how the data is to be sent. Some simplifications were used for generating this protocol. For instance, ut is assumed that there will always be a pause after each message and that only two different messages is to be sent, an init message and a command message. Although this may not be a complete protocol, it should be adequate for this application. The focus when implementing the code on the microcontroller was to minimize the code needed while maintaining all the functionality. For instance sorting of the commands can be done/avoided on the PC instead of on the microcontroller.

8.5 Liquid properties

A possible solution for decreasing the filament breakup in this application is to use a larger nozzle and shorter on-time. That way the circumference-to-length ratio is increased for droplets with the same volume. The time before the filament breaks up is thus increased, and the filament will break up in larger and fewer droplets. Manipulation of the liquid to increase the Ohnesorge number is another solution for decreasing filament breakup. However, this is likely to influence the Weber number and stability of the droplet.

Theory regarding the Ohnesorge number and filament breakup finds that a liquid with low surface tension and high viscosity reduces the filament breakup as the Ohnesorge number increases. However when the droplets have to travel a significant distance before hitting their targets it is important that the droplets do not disintegrate. The increased stability of the droplet leads to more satellites due to increased filament breakup. Under ideal conditions the satellites from filament breakup will merge with the main droplet, this may not be the case in the field. Clearly the best solution would be to avoid filament breakup while maintaining a stable droplet in air. A compromise between the Ohnesorge number and the Weber number is of importance when shooting droplets a significant distance. This is due to the requirement of a stable droplet throughout the whole flight, while trying to minimize the filament breakup.

8.6 Future Work

When the system is up and running, a number of tests have to be conducted to improve the design further. This includes testing of the precision of the droplets to decide on a minimum resolution of control and tuning of the delays of the valve driver. The precision of the droplets may also influence the design of the nozzle matrix in the future as this may have to be altered to achieve a more precise system. In addition testing of liquid properties and different liquids for herbicide control should be conducted to optimize the effect on the weeds. This is crucial for achieving an effective system, as the precision is irrelevant if the droplets do not affect the weeds as they should. In addition the simplifications used in the valve mapper algorithm may turn out to be inaccurate, requiring the algorithm to be modified. On the other hand, the simplifications may prove to be accurate enough and no modification is required.

This kind of DOD system may be modified for different kind of crops as well. The application is already designed to easily be modified to several PCB's in daisy-chain, allowing for a wider nozzle array if the crop have wider rows. In addition, the system may be used to apply nutrition on the crop if desirable. The only modifications needed is to select the crop instead of the weed when creating the spraymap.

Chapter 9

Conclusion

The second revision of the PCB designed in this thesis proved to work as specified and no modifications are needed so far. The solenoid control for the DOD application is designed with a full H-bridge and PWM voltage regulation to generate the spike and hold voltages. Usually solenoids are discharged by the use of two diodes in reverse series, however for this application a negative spike is used instead. The spike time have to be calculated from the resistance and inductance of the coil. A too large or short spike time will cause the droplet to get a secondary tail that should be avoided. The experiments also illustrate the filament breakup and its connection to the Ohnesorge number, while the Weber number is essential to the stability of the droplet in flight. An IEEE paper was prepared for ICARCV 2014 on this topic and the notification of acceptance is 1st July 2014.

For the synchronization of the modules in the overall application the trigger circuit circumvents a huge problem. This way the exact time of capture for each image is know for the microcontroller and IMU. That way the PC can be kept out of the strict real-time loop, and only operate with relative times instead of having to be synchronized with the other modules. The synchronization with the PC would have been cumbersome and it may have been too inaccurate for producing an effective system. Thus, this way of synchronizing the modules is crucial for an effective system. As all parts of the total application the PCB and trigger circuit is made as flexible as possible. This is to allow for changes in different hardware without the need of modifying the rest of the system.

A valve control algorithm with spray map and navigation log as inputs was designed and implemented. Some simplifications were made may be sufficient for the real time operation, and as a result the algorithm will be faster. This includes that the height is constant and known and the yaw-rate is set to zero. In addition, the nozzle array and image orientation is assumed to be slowly varying. These simplifications should be precise enough as the vehicle is set to drive

in a straight line, and the height of the nozzle array is fixed. Most of the functions used in the algorithm can be extended to include more complex scenarios and less simplifications without altering the input and output as this is assumed to be done in the future. Before these changes are to be made, the system should be tested to see how the application performs and what needs to be addressed in later revisions. It is not necessarily required to modify the algorithm if the simplifications prove to produce the accuracy specified.

The valve control algorithm and PCB were tested in the laboratory and performed as specified together. To test if some changes are needed it have to be tested in the field. The only difference from the laboratory to the field is that real spraymaps and navigation data is used. Thus the system should perform as specified in the field, but the timing of the droplets may have to be tuned for achieving a good result. Inspection of the accuracy of the droplets in the field have to be conducted to optimize the spray liquid and nozzle matrix as well, as disturbances may influence the system when moving.

Appendix A

Acronyms

DOD Drop-on-demand

GPS Global positioning system

IMU Inertial measurement unit

PCB Printed circuit board

PWM Pulse-width modulation

ICSP In circuit serial programming

LED Light emitting diode

EMC Electromagnetic compatibility

DC Direct current

IDE Integrated development environment

N/A Not applicable

LED light emitting diode

Appendix B

ICARCV Paper

This appendix includes the paper prepared for the ICARCV December 2014. The notification of acceptance is 1st July 2014.

Design and control of precision drop-on-demand herbicide application in agricultural robotics

Frode Urdal^{†*}, Trygve Utstumo^{†*}, Simen Andreas Ådnøy Ellingsen[‡], Jan Tommy Gravdahl[†]

[†] Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491 Trondheim

[‡] Department of Energy and Process Engineering, Norwegian University of Science and Technology, NO-7491 Trondheim

* Adigo AS, NO-1405 Langhus, frode@adigo.no

Abstract—Drop-on-demand weed control is a field of research within Precision Agriculture, where the herbicide application is controlled down to individual droplets. This paper focuses on the fluid dynamics and electronics design of the droplet dispensing. The droplets are formed through an array of nozzles, controlled by two-way solenoid valves.

A much used control circuit for opening and closing a solenoid valve is a spike and hold circuit, where the solenoid current finally is discharged over a Schottky diode on closing. This paper presents a PWM design, where the discharge is done by reversing the polarity of the voltage. This demands an accurate timing of the reverse spike not to recharge and reopen the valve. The PWM design gives flexibility in choosing the spike and hold voltage arbitrarily, and may use fewer components. Calculations combined with laboratory experiments verify this valve control strategy.

In early flight the stability of the tail, or filament, is described in theory by the Ohnesorge number. In later flight, when a droplet shape has formed, the droplet stability is governed by the Weber number. These two considerations have opposite implications on the desired surface tension of the fluid. The Weber number is more important for longer distances, as the filament satellites normally catch up and join the main droplet in flight.

I. INTRODUCTION

In this paper we study the use of a H-bridge, PWM, as the valve control strategy for a drop-on-demand(DOD) herbicide application in precision agricultural robotics. The design and control strategy has been guided by experiments with droplet dynamics, and the effect of reverse voltage overshoot has been illustrated.

Weed control is a vital part of agriculture, and herbicide application is the most efficient and common control strategy. Environmental and health concerns lead to restrictions and regulations on the use of herbicides, which stimulate initiatives for other weed control strategies [1]. Precision agriculture is an active area of research and methods in agriculture which focuses on adapting the field treatment to the spatial and temporal heterogeneity of a field. Weed control in row crops, such as carrots, can be separated into controlling weeds within, and in between the crop rows: Intra- and inter-row weed control.

DOD herbicide application for intra-row weeding has been investigated by several research groups: [2] designed a robotic weed control system for tomatoes, [3] developed an automated detection and control system for volunteer potatoes in sugar

beet fields and [4] created a crop/weed discriminating microsprayer. Common for all tree applications is the use of a valve array to only target the weeds, thus avoiding crop and soil. The literature displays promising results, and experiments indicate that the herbicide use can be reduced by more than 95 % [1]. The literature also illustrates that there are remaining challenges with precisely targeting droplets, classifying weeds by machine vision and maintaining a precise motion estimate for the robotic platform and nozzle array. The review article [5] presents a good overview of the field.

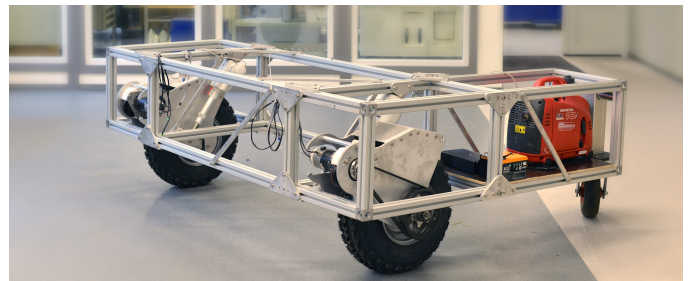


Fig. 1. The Asterix robot platform design for operation in row crops. The platform has two driven wheels and a passive caster wheel. The Asterix modules with the DOD system and machine vision will be mounted in the open area between the two wheels.

The work presented in this paper has been done in the framework of the Asterix project, which works towards a functional robot for DOD intra-weed control in carrots and other row crops. The robotic prototype platform for Asterix is shown in Figure 1 and the localization and attitude estimation for this robot has been described in [6]. In the following sections we will focus on the design and control of the DOD array of nozzles and the control strategy, while we also present our experimental results accompanied with some of the fluid dynamic theory of droplet stability. Droplet stability for at least 15 cm is necessary for this application.

A. Valve and nozzle limitations

The valve and nozzle used are of type INKX0514300A and INZA4710975H respectively, from The Lee Company, as illustrated in Figure 2. The requirement on resolution of control decides what time of the season a system is effective. The control resolution will have a practical lower limit depending on the droplet accuracy. If the droplets have an accuracy

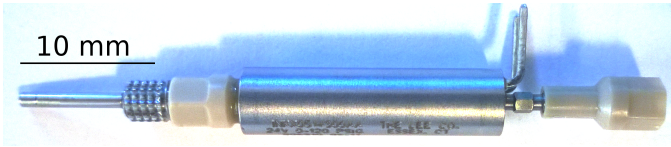


Fig. 2. A VHS valve, INKX0514300A, with minstack mountings and nozzle, INZA4710975H, from The Lee Company.

of ± 2.5 mm there is no need to have finer resolution than $5 \text{ mm} \times 5 \text{ mm}$ as it would result in many droplets missing the target. The sideways resolution is only a function of nozzle placement, ref. [4] used one row with a spacing of 10.5 mm. Their results and calculations showed that the system was not suitable for targeting weeds smaller than $11 \text{ mm} \times 11 \text{ mm}$. Ref. [3] used a similar system with resolution of control about 100 mm^2 . Thus neither system will be efficient in the early stages of the season when the weeds are still smaller than 100 mm^2 .

The resolution in driving direction can be controlled by the frequency of the valves and the velocity of the vehicle. For instance, ref. [3] used a valve limited to a maximum frequency of 80 Hz, and the demand for control resolution was 100 mm^2 , thus limiting the velocity of the vehicle to 0.8 m/s.

Flat fan nozzles are an alternative that allows for smaller weeds to be targeted by spraying a small patch. Recent work has investigated the efficiency of patch spraying with flat nozzles [7]. These tests showed promising results for spraying of $100 \text{ mm} \times 100 \text{ mm}$ patches. When working with row crops, especially carrots, a DOD application with finer resolution is interesting, as the seeds are placed close to each other and weed in between should be controlled. The use of flat fan nozzles in row crops was also examined in ref. [3], where DOD was found beneficial.

Solenoid valves have an upper limit for droplet frequency, and for some microdispensing valves this limit may be hundreds of hertz. However, due to the required droplet volume, the real upper limit may end up around 100 Hz, as a higher frequency would further reduce the volume. Relevant volumes per droplet for a DOD herbicide application lies between $1 \mu\text{L}$ and $5 \mu\text{L}$, and on-times of about 10 ms.

One aspect that needs to be considered when dealing with valve opening time intervals of a few milliseconds, is the fluid dynamics. The fluid in a straight tube can be modelled as an equivalent electrical circuit [8]. This can then be applied to simulate the fluid response in the nozzle under ideal conditions. Increasing the diameter or decreasing the length of the nozzle will result in increased volume rate deposition, but may alter the properties for the droplet in flight.

B. Droplet formation

A droplet produced by a DOD system consists of two or three sections, the main droplet, the filament and a tail. The filament is a cylindrical stream of flow following the main droplet, while the tail is a thin flow behind the filament. The different parts are illustrated in Figure 3. For more information consult [9].

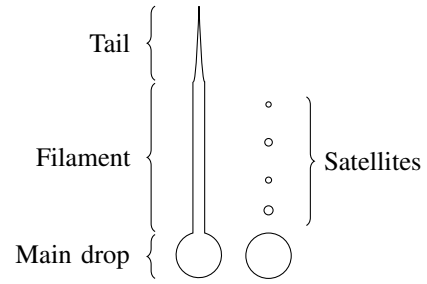


Fig. 3. Droplet definitions

The relative importance to the filament stability from surface friction and viscosity can be expressed through the Ohnesorge number:

$$Oh = \frac{\eta}{\sqrt{\rho\sigma R}} \quad (1)$$

where η , ρ and σ are the viscosity, density and surface tension of the liquid, respectively, while R denotes the radius of the cylindrical filament [10]. Furthermore, the initial filament aspect ratio, $\Lambda = L/2R$, will decide if the filament breaks up or not, L is the length of the filament. The critical value for filament breakup, Λ_c , increases with Oh [10].

When the droplet is falling, a number of scenarios may occur: the filament may be absorbed into the main droplet, it may break at the main droplet thus creating a single satellite droplet or a Rayleigh-Platou instability may occur, creating multiple satellite droplets [9].

Satellite droplets are small droplets lagging behind the main droplet, often caused by the disintegration of the tail or filament. Without wind and other disturbances that could be present for a DOD application in movement, the satellite droplets will typically catch up with the main droplet and coalesce with it, and the dispensed fluid volume reaches the target as a single droplet [11]. An image sequence illustrating this is presented in Figure 8. This is a result of less drag on the satellites as they are smaller and travel in the wake of the main droplet. This will happen under ideal conditions, but how the satellites will behave in the field is not certain. Most of the research described above are results from ink jet printers with droplets much smaller than what is needed for herbicide applications. However, ref. [12] verifies that the theory applies for larger droplets as well, which is more relevant for this project.

The Weber number is of importance when studying the droplets in air, and is defined as, [13]:

$$We = \frac{\rho u^2 d}{\sigma} \quad (2)$$

Where ρ is the density of air, u the droplet velocity, d the droplet diameter and σ the surface tension. With the assumption of spherical droplets, the droplet is stable if its Weber number is below the critical Weber number, which lies between 10 and 40 [14].

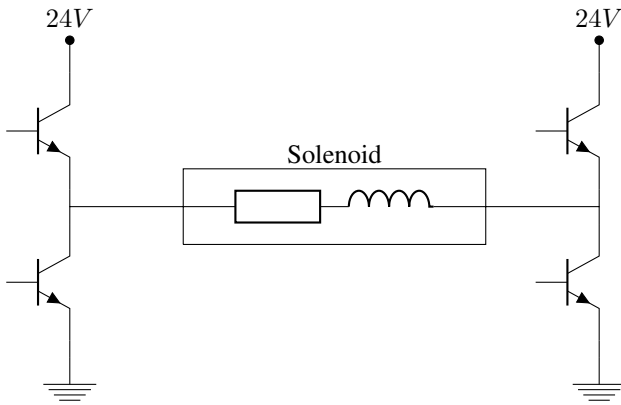


Fig. 4. PWM H-bridge valve driver for a single solenoid

C. Herbicide efficiency with DOD

By far the most common herbicide in use today is glyphosate. It has been widely used through the past 40 years [15], and a water solution of glyphosate is a natural and common choice for DOD weed control [5].

Tests on the efficiency of single droplets of herbicide is presented in [16]. The tests were done with seeds of *Solanum nigrum* planted in pots under outdoor conditions. Results showed that approximately $0.8 \mu\text{g}$ of glyphosate per plant reduced the biomass by 95% when applied by hand.

In field trials with a DOD system, the microspray system was set to dispense droplets of $2.5 \mu\text{L}$ with $5 \mu\text{g}$ glyphosate each. The system achieved 82% efficiency when the average dose per plant was $22.6 \mu\text{g}$. This is only about 4% of the recommended application [16].

II. VALVE CONTROL

In DOD applications the ideal solenoid valve would open and close instantaneously, and the droplet size would be directly proportional with the opening time of the valve. Any physical solenoid valve has a response time τ , which allows for the solenoid coil to charge and the plunger to open. In selecting a valve for DOD applications, one should focus on achieving a response time significantly smaller than the open time, $\tau < T_{open}$.

Several methods are in use for valve control. Typical configurations are: **Single voltage source** controlled by a transistor. This is a simple driver, but it takes longer to open the valve, as the voltage cannot be higher than the hold voltage as it may burn off the coil. Thus charging the coil takes longer than using a higher voltage source. **Spike and hold** drivers with two different voltage sources, one for the spike and another for the hold voltage. They are more complex, but achieve a much faster response. Common for both configurations when closing the valve is that the energy in the coil is burnt off over two diodes in reverse series parallel to the valve. Another solution is to use **PWM** control to create a spike and hold driver equivalent, with diodes to discharge the coil. However, if the PWM is extended to a full H-bridge, it can be used to discharge the energy in the coil.

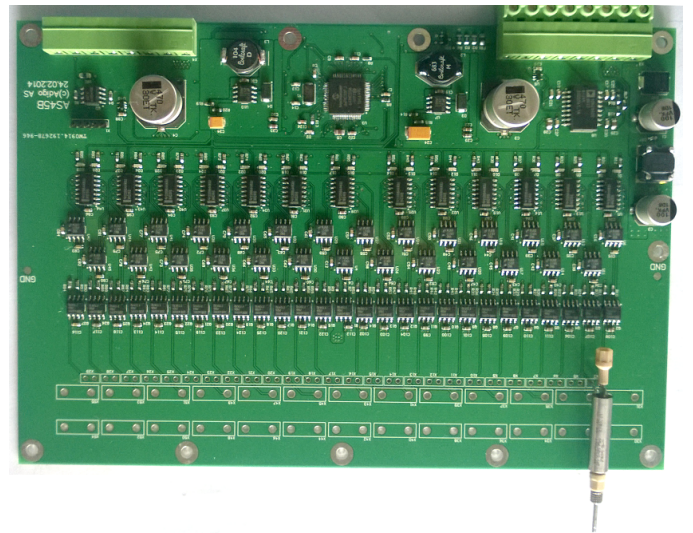


Fig. 5. The DoD demand control unit with one valve and nozzle mounted for an experimental setup.

A. Comparison of PWM and Schottky for solenoid discharge

The main idea for controlling the valve with PWM is that only one voltage source is needed, in a traditional spike and hold driver, two sources are needed, as the spike voltage will overheat the valve if applied for too long. When using PWM the voltage source can be adjusted to fit the spike voltage, that way a large spike followed by a PWM signal to reduce the voltage to the hold value will simulate a spike and hold driver circuit. The PWM control can discharge the solenoid by reversing the voltage over the diode for a significant time, so the current in the coil reaches zero. It is important that the current avoids excessive undershoot as this may open the valve for a short duration before it is closed. This solution will be detailed below. The schematic principle for one single valve driver is presented in Figure 4. When closing the valve, the voltage is reversed over the valve, thus discharging the energy in the coil. The discharge time is reduced with increased voltage, just as the opening time is reduced by increased spike voltage. The Schottky diode solution discharges the coil by burning off the energy in the coil over two schottky diodes in reverse series.

When using the PWM method, the voltage across the valve is limited to the spike voltage, but when using diodes the voltage can be increased further. 50 V reverse voltage is quite common for the schottly diodes for small solenoids with a hold value of about 3.5-4.5 V. The time to close the valve with an internal resistance of 40Ω , inductance of 12 mH, and hold voltage of 4 V can be calculated for the different solutions. The current response of a resistor in series with an inductor follows the first order response:

$$I(t) = I_0 + (I_1 - I_0)(1 - e^{-t/\tau}) \quad (3)$$

where $I(t)$ is the current at time t , I_0 is the initial current, I_1 is the steady state current for the final solution and τ is

TABLE I
ELECTRICAL CHARACTERISTICS FOR THE LEE INKX0514300A VALVE
FROM DATASHEET

Description	Value
Resistance	40 Ω
Inductance	12 mH
Hold voltage	4 V

TABLE II
EXPERIMENTAL AND CALCULATED REVERSE VOLTAGE SPIKE TIMES,
WHERE THE FINAL CURRENT IS THE EXPECTED OVERSHOOT OR RESIDUE
CURRENT IN THE SOLENOID COIL.

Description	Negative spike duration (ms)		Final current (mA)
	Theoretical	Experimental	
Scottky	0.0231	-	0
Ideal PWM 24V	0.0463	0.05	-8.31
Control PWM 24V	-	0.10	-99.15

the time constant. For the PWM solution with 24 V the time to reduce the current level to zero is 0.0463 ms while using 50 V diodes results in a time of 0.0231 ms. This is about half the time, but represent a very small portion of the time which the valve is open. A typical open time interval for the solenoid is $T_{open} = 8$ ms. The response time of the valve is about $\tau \approx 0.3$ ms. The effect on the tail will be examined by experiments to ascertain whether this control strategy works or not.

The complexity of the control configurations is another aspect that needs to be inspected. For a microdosing system it is important to have a fast response circuit as a spike and hold circuit. There are many solutions for such a driver, but the main difference discussed here is how to discharge the energy in the coil. Regardless of the solution chosen the PWM approach will result in fewer components than the schottky diode solution for a valve matrix. This is achieved by using a half H-bridge for all the valves, in addition to a half H-bridge that is common for all valves. That way two diodes for each valve is avoided and only two more transistors are needed. This makes the circuit less complex and easier to control.

Another advantage for the PWM control is the need of just one voltage source. A solution to remove one voltage source for the spike and hold driver is to use a voltage regulator to produce the hold voltage. The problem here is that when the number of valves increases, several regulators are needed as the current becomes larger. The reduction in components influences the cost of the final PCB as well. Another significant advantage is that the PWM solution is more flexible. If the valve is replaced, the only requirement for the new valve is that the spike voltage needed does not exceed the initial design specifications. Thus only software adjustment is required instead of modifying the circuit.

B. Negative spike time

The negative spike time of the PWM circuit must be carefully chosen. The electrical characteristics of the valve are

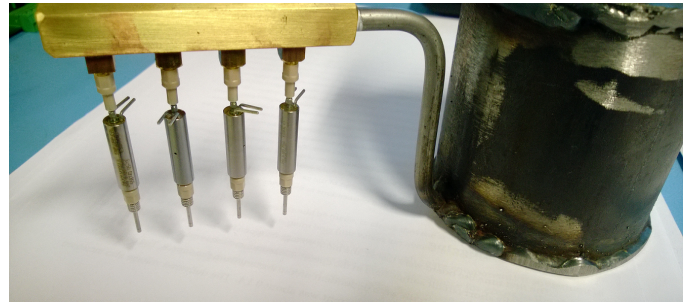


Fig. 7. Experimental setup with valves and pressurized liquid container, for early experiments with droplet formation, as shown in Figure 6 and 8.

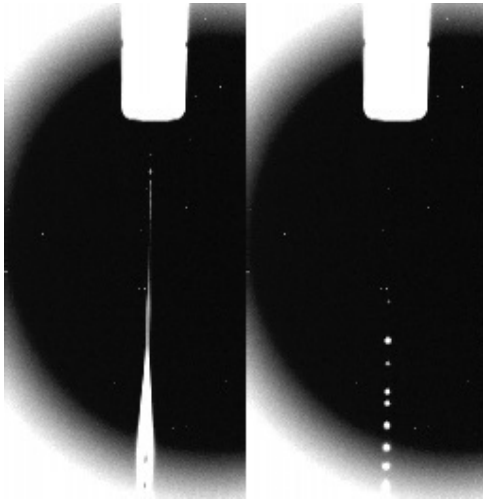
presented in Table I. Using the first order response of the RL circuit as in Equation 3 the exact time can be derived. The timing and currents are presented in Table II. As calculated before the time for closing the valve under ideal conditions is 0.0463 ms. In practice the resolution in time may have to be limited. The important part is to have the current close to zero so the plunger is not activated again. The residue current will be burnt off over the diodes in the transistors.

To test how the closing time influences the droplet, a test rig was set up. This was done with a black and white high speed camera, PROMON 501 from AOS Technologies. To provide sufficient light for shooting with 1000 fps a LED panel was placed behind the nozzle pointing directly at the camera. The valve was operated by the PCB controlled from a computer. The rest of the setup consisted of a pressurized liquid container with water and tubing, as shown in Figure 7. The pressure was set to 0.4 bar, which produces droplets with an initial velocity of about 4 m/s. In this experiment regular tap water was used.

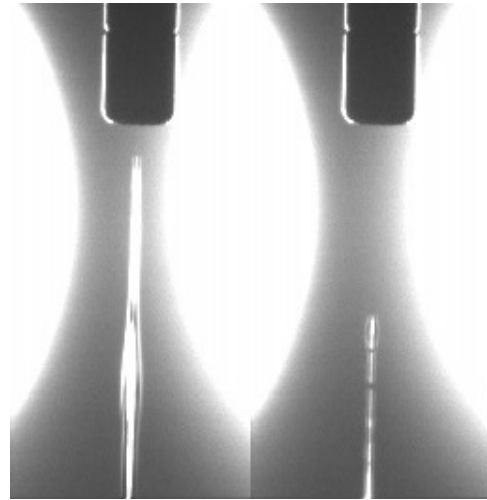
The main test was to see how the time resolution affected the droplet, initially two spike times were chosen. The requirement was that both times should be realistic regarding how the system will be programmed for the field. For the first test the spike duration was set to 0.1 ms, while for the second test it was 0.5 ms. The calculations represented in Table II shows the theoretical times for discharging the energy in the coil and the theoretical residue current for the experimental times. A spike duration of 0.1 ms should result in a current of -99.15 mA, while 0.05 ms result in an undershoot of -8.31 mA. A current larger than 87.5 mA is enough to hold the valve open. Thus the larger spike duration may cause the valve to start opening again. In this test it was of interest to see how such an undershoot affects the droplets properties. An undershoot of -8.31 mA should not be enough to actuate the valve at all, thus the difference should be observable.

C. Results

The experimental setup was designed with one valve with a pressure of 0.4 bar. The spike voltage was set to 24 V and the hold voltage to 3.96 V. The only difference in the two tests was the negative spike duration. Figure 6a shows the end of the droplet using a spike duration of 0.1 ms, while Figure 6b shows the end of the droplet when using 0.05 ms for the spike duration. For the first test a thin secondary tail is observable



(a) Test 1, tail, $U_{hold} = 3.96$ V, $t_{spike} = 0.1$ ms



(b) Test 2, tail, $U_{hold} = 3.96$ V, $t_{spike} = 0.05$ ms

Fig. 6. High speed footage of the droplet tail with a pressure of 0.4 bar, 1000 fps. Figure (a) show the extra tail resulting from the reverse spike overshoot.

before it breaks into many small satellite droplets. This is however avoided in the second test. Common for both tests is that the filament is beginning to break up. The length of the filament makes it unstable as described previously. The breakup of the tail in Figure 6a is similar to filament breakup, but because it is so much thinner than the filament it breaks up faster and to smaller droplets.

Under ideal conditions the satellite droplets will overtake the main droplet, but in practice, the robot will be moving and the presence of wind may affect the satellites differently than the main droplet. Thus the filament breakup and the tail breakup should be minimized. This is to reduce the possibility of satellite droplets not merging with the main droplet and missing the target.

III. DISCUSSION

For the autonomous weed control application to work, the DOD system must be very accurate. The accuracy does not solely depend on target precision, but the presence of satellite droplets and their behavior. It is crucial that the satellites coalesce with the main droplet, or that they both hit the same spot. Therefore the droplets tail should be minimized, as the tail will split up in much smaller droplets than the filament.

The usual method for driving solenoid valves in this kind of application is by a spike and hold driver circuit, with two diodes in reverse series to discharge the energy in the coil. However the described method is based on a full H-bridge, for PWM control. The maximum voltage is chosen as the spike voltage, thus a long spike will open the valve before the PWM control limits the voltage to the hold value. When closing the valve the energy in the coil is discharged with a significant negative spike. The spike time has been calculated using the first order response of a RL-circuit when the inductance and resistance of the valve is known.

Tests of this control strategy performs confirms the theory, as a long spike time resulted in a thin secondary tail, while a

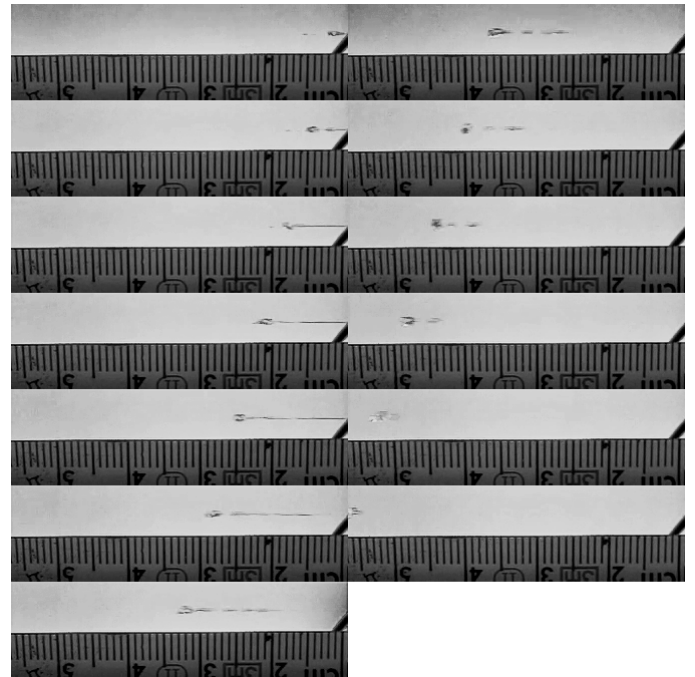


Fig. 8. 1200 fps image sequence of a water droplet with initial velocity of 4 m/s (illustration taken from ref. [17]). The filament first break up to satellites, which then drift in the wake of the main droplet, and join the droplet.

spike time of appropriate length avoided this. The long spike time started to actuate the plunger when it should be closed, but the small undershoot for the appropriate time did not actuate the plunger at all.

Thus the spike time of 0.05 ms is close enough to the theoretical time for closing the valve. A possible solution for decreasing the filament break up in this application is to use a larger nozzle and shorter on-time. That way the circumference-to-length ratio is increased for droplets with the same volume. The time before the filament breaks up is thus increased,

and the filament will break up in larger and fewer droplets. Manipulation of the liquid to increase the Ohnesorge number is another solution for decreasing filament breakup. However, this is likely to influence the Weber number and stability of the droplet.

The main disadvantage of PWM control is the increase in time for discharging the energy, as the diodes can be chosen with a higher voltage level. This is not as easy with the PWM solution, as the higher the voltage, the more robust the components must be. This is due to the increase in voltage when using only one source will influence the robustness of the components, especially the transistors, used in the PWM control.

However there have not been observed any negative effects of the slower closing of the PWM solution compared to the Schottky closing. Thus an increase in the voltage is not necessary.

Breakup of the filament was observed in the tests, but this will occur regardless of the control strategy, and might be decreased with increased nozzle diameter. The theory and calculations regarding the valve control was confirmed by the experimental setup. The tail was avoided although the closing time is increased compared to the diode solution, thus the advantages of the PWM control strategy may be exploited. This includes a more flexible design with regards to the valves and fewer components are needed for the circuit. The main focus is to make sure that the negative spike time does not undershoot too much as this will create a tail that should be minimized.

Theory regarding the Ohnesorge number and filament breakup fines that a liquid with low surface tension and high viscosity reduces the filament breakup as the Ohnesorge number increases. However when the droplets have to travel a significant distance before hitting their targets it is important that the droplets do not disintegrate. The increased stability of the droplet leads to more satellites due to increased filament breakup. Under ideal conditions the satellites from filament breakup will merge with the main droplet, but this may not be the case in the field. Clearly the best solution would be to avoid filament breakup while maintaining a stable droplet in air. A compromise between the Ohnesorge number and the Weber number is of importance when shooting droplets a significant distance. This is due to the requirement of a stable droplet throughout the whole flight, while trying to minimize the filament breakup.

IV. CONCLUSION

A valve controller has been developed for drop on demand weed control, using a full H-bridge design and PWM voltage regulation to generate the spike and hold voltages. In contrast with common design practices with solenoid drives, we have not included the discharge diodes. The solenoid discharge is instead done by applying a reverse voltage to the solenoid.

The timing of the reverse voltage has to be calculated using the solenoid inductance given from the datasheet. If the reverse spike is held longer the solenoid may reopen and dispense a secondary tail, which will create additional satellite droplets.

If the reverse spike is not long enough the residue current will discharge over the protective diodes in the H-bridge drivers.

The design results in fewer components per solenoid, but demands accurate timing of the reverse voltage spike. The PWM allows for arbitrary spike and hold voltages up to the supply voltage, which for this project has been 24 V. The experiments also illustrate the filament breakup and its connection with the Ohnesorge number, while the Weber number is essential to the stability of droplets in flight.

ACKNOWLEDGMENT

The authors would like to thank Alexander Klungerbo for conducting experiments to conclude what parameters affect the distance traveled for droplets. Jan Kåre Vatne for the assistance and guidance in electronics design.

REFERENCES

- [1] S. Christensen, H. T. Sogaard, P. Kudsk, M. Nørremark, I. Lund, E. S. Nadimi, and R. Jørgensen, "Site-specific weed control technologies," *Weed Research*, vol. 49, no. 3, 2009.
- [2] W. Lee, D. Slaughter, and D. Giles, "Robotic weed control system for tomatoes," *Precision Agriculture*, vol. 1, no. 1, 1999.
- [3] A. T. Nieuwenhuizen, "Automated detection and control of volunteer potato plants," Ph.D. dissertation, Wageningen University, 6 2009.
- [4] H. Midtby, S. K. Mathiassen, K. Andersson, and R. Jørgensen, "Performance evaluation of a crop / weed discriminating microsprayer," *Computers and Electronics in Agriculture*, vol. 77, no. 1, pp. 35–40, 2011.
- [5] D. Slaughter, D. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63–78, 2008.
- [6] T. Utstumo and J. T. Gravdahl, "Implementation and comparison of attitude estimation methods for agricultural robotics," in *proceedings of the 4th IFAC Conference on Modelling and Control in Agriculture, Horticulture and Post Harvest Industry (AgriControl 2013)*, Aalto University, School of Electrical Engineering, Espoo, Finland, August 28-30, 2013.
- [7] P. K. Jensen, I. Lund, and D. Nuyttens, "Spray liquid distribution and biological efficacy of commercially available nozzles used for precision weed control," *Biosystems engineering*, vol. 116, no. 4, pp. 316–325, 2013.
- [8] J. Watton, *FLUID POWER SYSTEMS: Modeling, simulation, analog and microcomputer controll*. Prentice Hall International (UK) Ltd., 1989.
- [9] D. Vadillo, T. Tuladhar, A. Mulji, S. Jung, S. Hoath, and M. Mackley, "Evaluation of the inkjet fluid's performance using the "cambridge trimaster" filament stretch and break-up device," *Journal of Rheology*, vol. 54, no. 2, pp. 261–282, 2010.
- [10] S. D. Hoath, S. Jung, and I. M. Hutchings, "A simple criterion for filament break-up in drop-on-demand inkjet printing," *Physics of Fluids*, vol. 25, no. 2, 2013.
- [11] H. Dong, W. W. Carr, and J. F. Morris, "An experimental study of drop-on-demand drop formation," *Physics of Fluids*, vol. 18, no. 7, p. 072102, 2006.
- [12] J. Castrejón-Pita, G. Martin, S. Hoath, and I. Hutchings, "A simple large-scale droplet generator for studies of inkjet printing," *Review of Scientific Instruments*, vol. 79, no. 7, p. 075108, 2008.
- [13] J. Eggers and E. Villermaux, "Physics of liquid jets," *Reports on progress in physics*, vol. 71, no. 3, p. 036601, 2008.
- [14] A. Wierzba, "Deformation and breakup of liquid drops in a gas stream at nearly critical weber numbers," *Experiments in Fluids*, vol. 9, no. 1-2, pp. 59–64, 1990.
- [15] A. D. Baylis, "Why glyphosate is a global herbicide: strengths, weaknesses and prospects," *Pest Management Science*, vol. 56, no. 4, pp. 299–308, 2000.
- [16] H. T. Sogaard, I. Lund, and E. Graglia, "Real-time application of herbicides in seed lines by computer vision and micro-spray system," in *American Society of Agricultural and Biological Engineers, ASABE Annual International Meeting*, 2006.
- [17] A. T. Klungerbo, "Drop-on-demand i presisjonsjordbruk," Master's thesis, NTNU, 2013.

Appendix C

PCB layout

In this appendix a picture of each layer of the 4-layer PCB is shown. The images included are the files from Altium Designer. Due to confidentiality the schematics is not included.

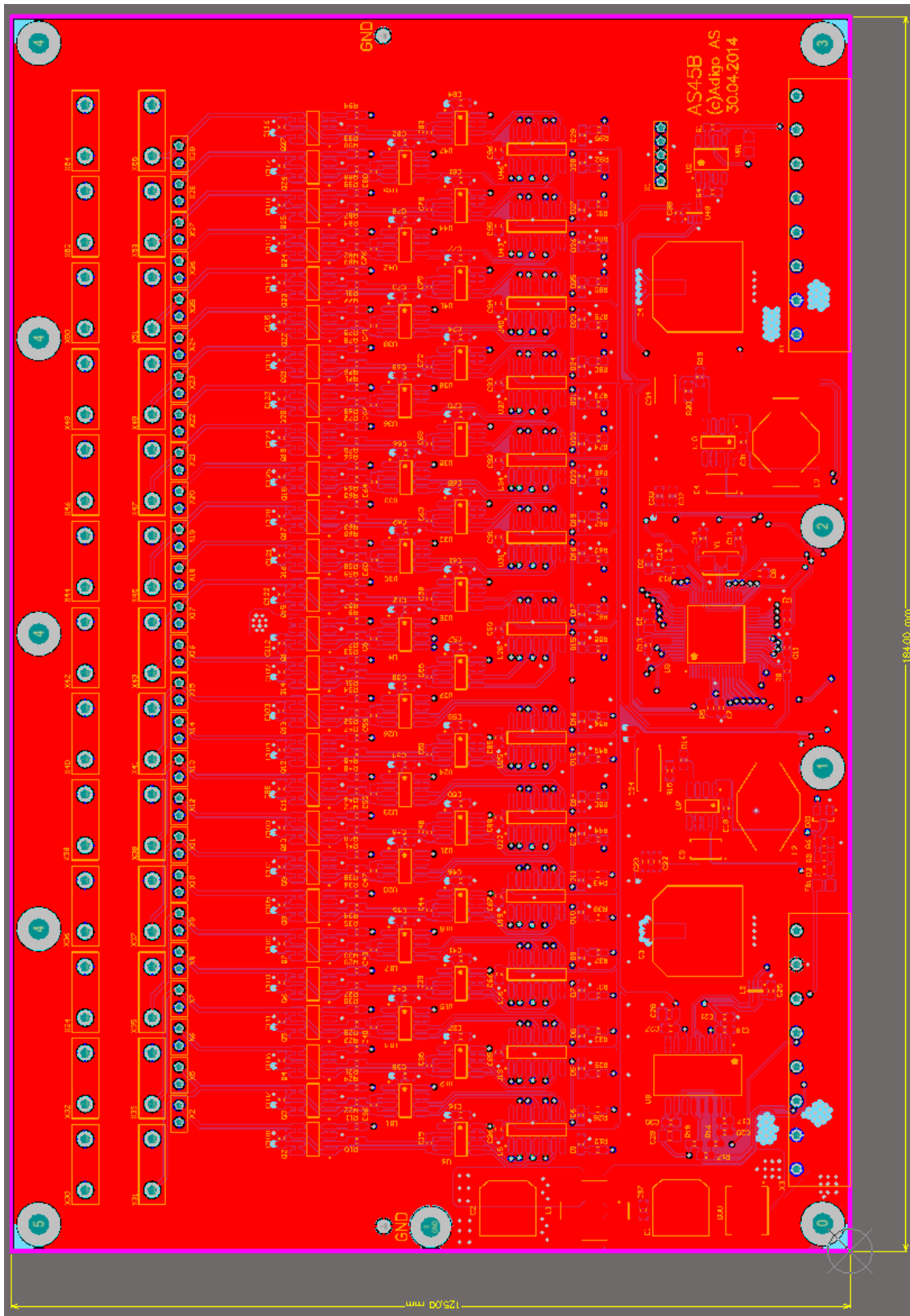


Figure C.1: Top layer

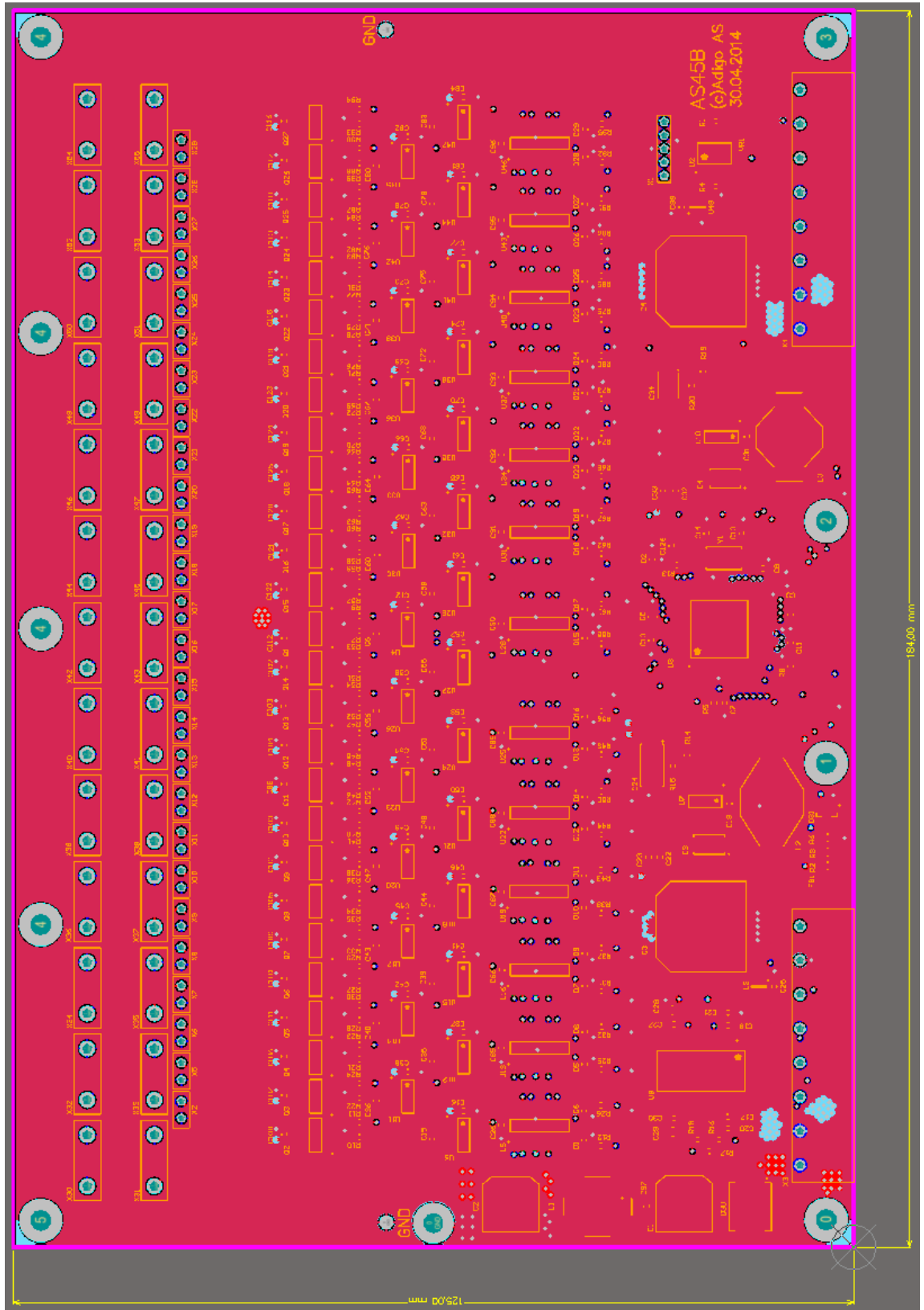


Figure C.2: Mid layer 1

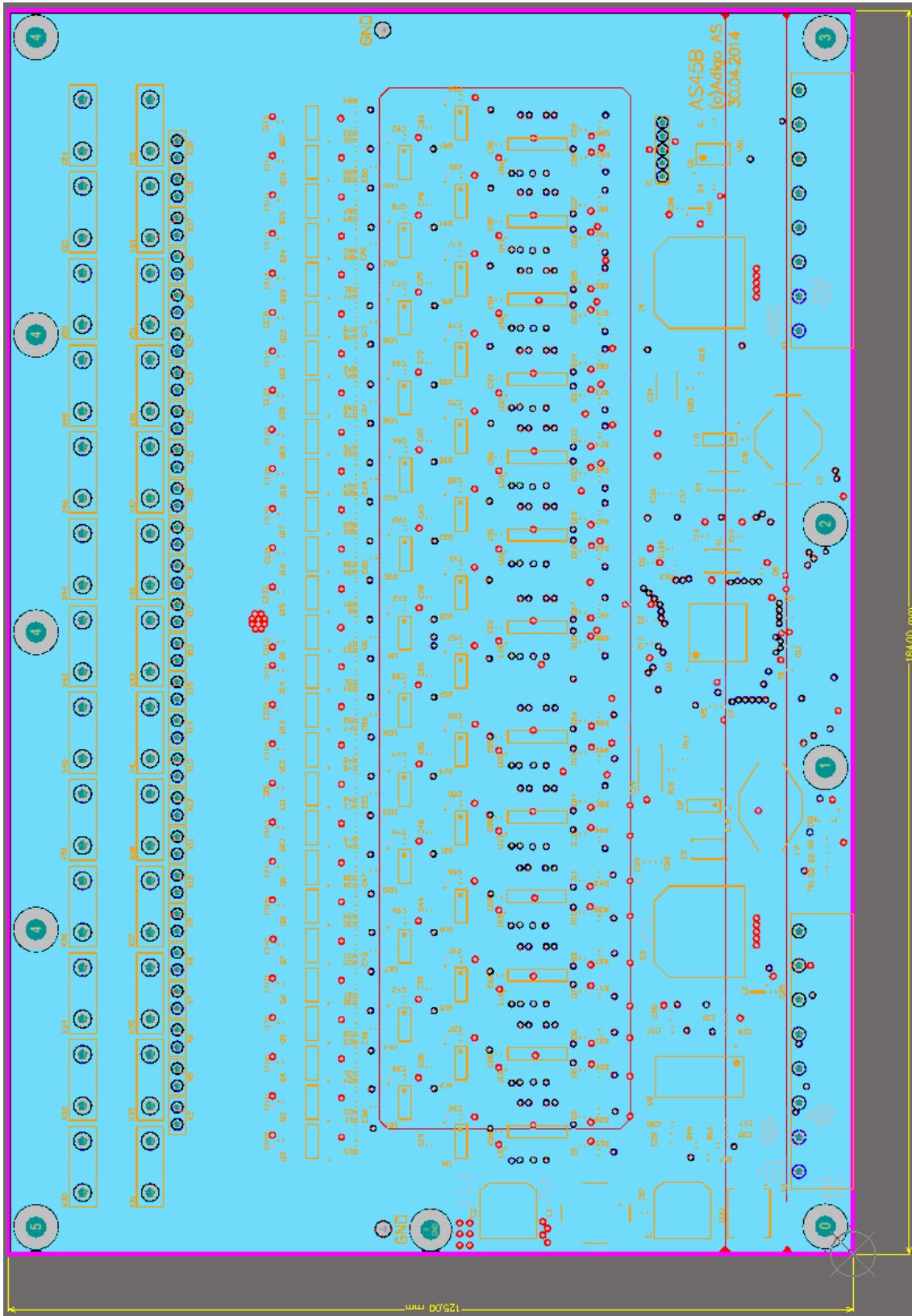


Figure C.3: Mid layer 2

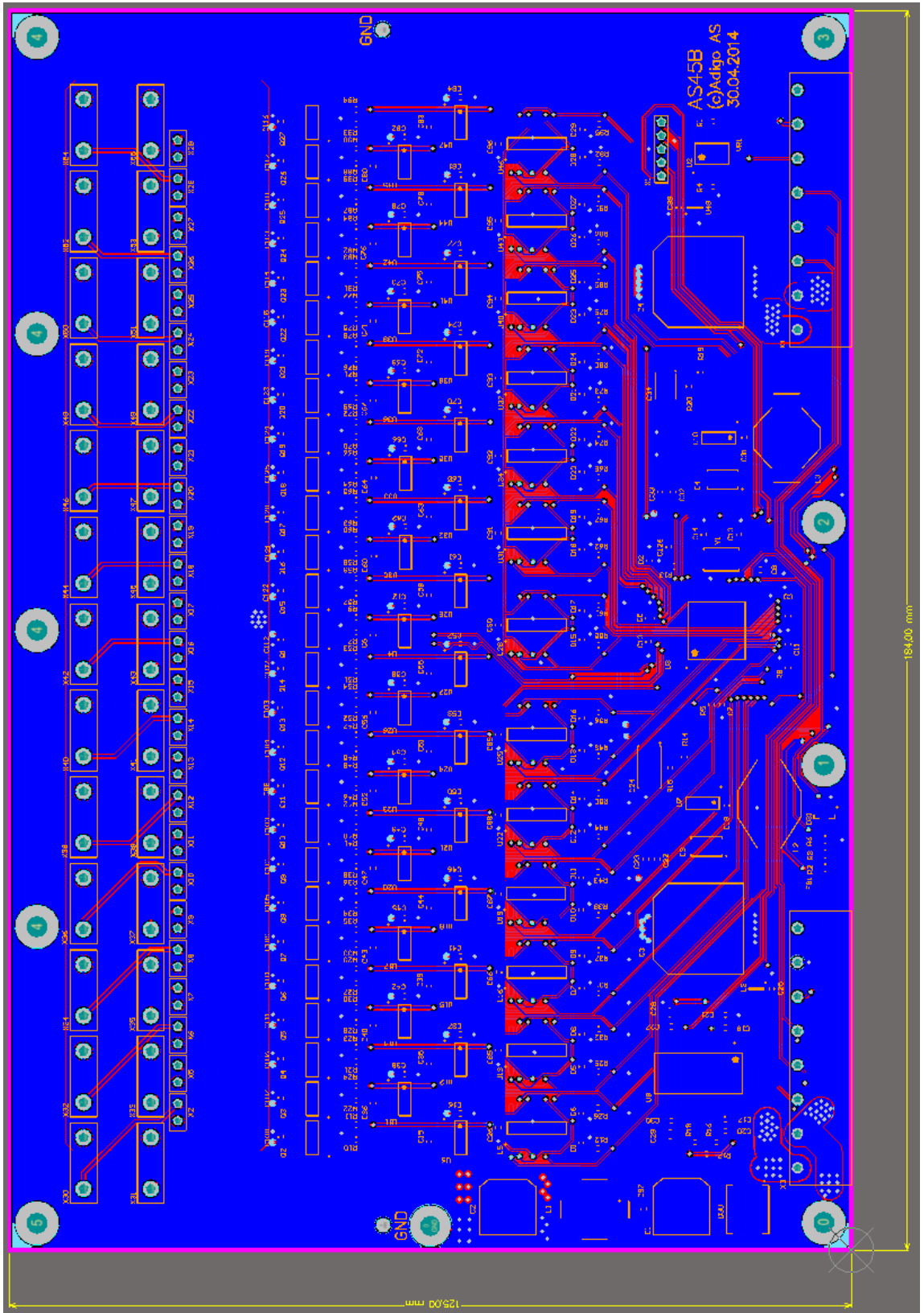


Figure C.4: Bottom layer

Appendix D

Components

This appendix includes a list of all components used for the design and some important datasheet pages.

D.1 Components

VR24V/0805 VARISTOR, 0805, 25VAC

SP8K32TB1 MOSFET, NN CH, 60V, 4.5A, SOP8

SN74HC08D Quadruple 2-Input Positive-AND Gate

M5x1_2mm Header, 2mm, straight, 5 pin

LM5101CMA/NOPB 1A High Voltage High-Side and Low-Side Gate Driver, 8-pin Narrow SOIC, Pb-Free

LM2671M-ADJ SIMPLE SWITCHER® Power Converter High Efficiency 500mA Step-Down Voltage Regulator with Features, 8-pin Narrow SOIC

LED orange/0603 VISHAY - TLMO1000-GS08 - LED, 0603, ORANGE

IL207At OPTOCOUPLER, TRANSISTOR O/P

FB/0805 FERRITE BEAD, 0.1OHM, 300MA, 0805

DSPIC33FJ64MC506A-I/PT High-Performance 16-bit Digital Signal Controller, 64 KB Flash, 8 KB RAM , 64-Pin TQFP, Industrial Temperature

BAT54S RECTIFIER, ULTRAFAST, 0.2A, 30V, SOT-23

ADM2682EBRIZ Analog devices RS485 interface with galvanic isolation

744272251 SMD Common Mode Line Filter WE-SL5, L = 250 μ H

470uF/35V CAPACITOR, 470UF, 35V, H13 CASE

470R/0603 RESISTOR, 470R, 0.063W, 1%, 0603

470K/0603 RESISTOR, 470K, 0.063W, 1%, 0603

220R/0603 RESISTOR, CERAMIC, 220R, 0.1W, 1%, 0603

100uH 0.82A INDUCTOR, PWR, 100UH, 1.3A, 20%,9MHZ

100uF/35V CAP, ALU ELECT, 100UF, 35V, 20%, SMD

100uF/16V/2312 CAPACITOR, CASE F, 100 UF, 16V

100uF/10V/1210 CAPACITOR TANT, 100UF, 6.3V, 10%, 1210

100pF/10V/0603 CAP, MLCC, X7R, 100PF, 10V, 0603

100nF/10V/0603 CAPACITOR, 0.1UF, 10V, X7R, 0603

100n/50V/0603 CAP, CERAMIC, 0.1UF, 50V, X7R, 0603

100N/16V/0603 CAP, CERAMIC, 0.1UF, 16V, X7R, 0603

74AHC1G04 74AHC SINGLE GATE, SMD, 74AHC1G04

68uH 0.99A INDUCTOR, PWR, 68UH, 1.5A, 20%,10MHZ

33K/0805 RESISTOR, POWER, 0805, 1%, 33K0

22pF/10V/0603 CAPACITOR, 0603, NP0, 10V, 22PF

12K/0805 RESISTOR, POWER, 10K, 0.33W, 1%, 0805

12K/0603 RESISTOR, 0603, 12K0, 1%, 0.1W

10uF/16V/0805 CAP, MLCC, 10UF, 16V, X5R, 0805

10R/0603 RESISTOR, 0603, 1% 50PPM 10R 100mW

10nF/16V/0603 CAP, CERAMIC, 0.01UF, 16V, X7R, 0603

10K/0603 RESISTOR, 10K, 0.1W, 1%, 0603, SMD

8P 5.08mm 8 Pol 5.08mm male

8MHz CRYSTAL, 8MHZ, 18PF, SMD

8K2/0603 RESISTOR, 0603, 8K20, 1%, 0.1W

6K8/0603 RESISTOR, 0603, 6K80, 1%, 0.1W

3A 40V schottky DIODE, SCHOTTKY, 3A, 40V, SMA

1.5KW 30V DIODE, TVS, 30V, 1.5KW, UNI, 5%, SMC

1uF/35V/0603 CAPACITOR, 0603, X7R, 35V, 1UF

1K/0603 RESISTOR, THICK FILM, 1K, 0.1W, 1%

D.2 Datasheets

In this section some informative pages of the most important datasheets are listed.

The following pages are included in this appendix:

- Microcontroller page tree and four
- LM5101 front page
- LM2671 front page
- RS-485 transceiver front page
- And-Gate front page
- Optocoupler front page

The following website lists the trigger modes for the camera. All are active low.

<http://www.ptgrey.com/support/kb/index.asp?a=4&q=239>



MICROCHIP dsPIC33FJXXMXX06A/X08A/X10A

High-Performance, 16-Bit Digital Signal Controllers

Operating Range:

- Up to 40 MIPS operation (@ 3.0-3.6V):
 - Industrial temperature range (-40°C to +85°C)
 - Extended temperature range (-40°C to +125°C)
- Up to 20 MIPS operation (@ 3.0-3.6V):
 - High temperature range (-40°C to +140°C)

High-Performance DSC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set
- 16-bit wide data path
- 24-bit wide instructions
- Linear program memory addressing up to 4M instruction words
- Linear data memory addressing up to 64 Kbytes
- 83 base instructions: mostly 1 word/1 cycle
- Two 40-bit accumulators:
 - With rounding and saturation options
- Flexible and powerful addressing modes:
 - Indirect, Modulo and Bit-Reversed
- Software stack
- 16 x 16 fractional/integer multiply operations
- 32/16 and 16/16 divide operations
- Single-cycle multiply and accumulate:
 - Accumulator write back for DSP operations
 - Dual data fetch
- Up to ± 16 -bit shifts for up to 40-bit data

Direct Memory Access (DMA):

- 8-channel hardware DMA
- 2 Kbytes dual ported DMA buffer area (DMA RAM) to store data transferred via DMA:
 - Allows data transfer between RAM and a peripheral while CPU is executing code (no cycle stealing)
- Most peripherals support DMA

Interrupt Controller:

- 5-cycle latency
- Up to 67 available interrupt sources
- Up to five external interrupts
- Seven programmable priority levels
- Five processor exceptions

Digital I/O:

- Up to 85 programmable digital I/O pins
- Wake-up/Interrupt-on-Change on up to 24 pins
- Output pins can drive from 3.0V to 3.6V
- All digital input pins are 5V tolerant
- 4 mA sink on all I/O pins

On-Chip Flash and SRAM:

- Flash program memory, up to 256 Kbytes
- Data SRAM, up to 30 Kbytes (includes 2 Kbytes of DMA RAM)

System Management:

- Flexible clock options:
 - External, crystal, resonator, internal RC
 - Fully integrated PLL
 - Extremely low jitter PLL
- Power-up Timer
- Oscillator Start-up Timer/Stabilizer
- Watchdog Timer with its own RC oscillator
- Fail-Safe Clock Monitor (FSCM)
- Reset by multiple sources

Power Management:

- On-chip 2.5V voltage regulator
- Switch between clock sources in real time
- Idle, Sleep and Doze modes with fast wake-up

Timers/Capture/Compare/PWM:

- Timer/Counters, up to nine 16-bit timers:
 - Can pair up to make four 32-bit timers
 - 1 timer runs as Real-Time Clock (RTC) with external 32.768 kHz oscillator
 - Programmable prescaler
- Input Capture (up to eight channels):
 - Capture on up, down or both edges
 - 16-bit capture input functions
 - 4-deep FIFO on each capture
- Output Compare (up to eight channels):
 - Single or Dual 16-Bit Compare mode
 - 16-Bit Glitchless PWM mode

dsPIC33FJXXMCX06A/X08A/X10A

Communication Modules:

- 3-wire SPI (up to two modules):
 - Framing supports I/O interface to simple codecs
 - Supports 8-bit and 16-bit data
 - Supports all serial clock formats and sampling modes
- I²C™ (up to 2 modules):
 - Full Multi-Master Slave mode support
 - 7-bit and 10-bit addressing
 - Bus collision detection and arbitration
 - Integrated signal conditioning
 - Slave address masking
- UART (up to 2 modules):
 - Interrupt on address bit detect
 - Interrupt on UART error
 - Wake-up on Start bit from Sleep mode
 - 4-character TX and RX FIFO buffers
 - LIN/J2602 support
 - IrDA® encoding and decoding in hardware
 - High-Speed Baud mode
 - Hardware flow control with CTS and RTS
- Enhanced CAN (ECAN™ technology) 2.0B active (up to 2 modules):
 - Up to 8 transmit and up to 32 receive buffers
 - 16 receive filters and three masks
 - Loopback, Listen Only and Listen All Messages modes for diagnostics and bus monitoring
 - Wake-up on CAN message
 - Automatic processing of Remote Transmission Requests
 - FIFO mode using DMA
 - DeviceNet™ addressing support

Motor Control Peripherals:

- Motor Control PWM (up to eight channels):
 - Four duty cycle generators
 - Independent or Complementary mode
 - Programmable dead time and output polarity
 - Edge or center-aligned
 - Manual output override control
 - Up to two Fault inputs
 - Trigger for ADC conversions
 - PWM frequency for 16-bit resolution (@ 40 MIPS) = 1220 Hz for Edge-Aligned mode, 610 Hz for Center-Aligned mode
 - PWM frequency for 11-bit resolution (@ 40 MIPS) = 39.1 kHz for Edge-Aligned mode, 19.55 kHz for Center-Aligned mode
- Quadrature Encoder Interface (QEI) module:
 - Phase A, Phase B and index pulse input
 - 16-bit up/down position counter
 - Count direction status
 - Position Measurement (x2 and x4) mode
 - Programmable digital noise filters on inputs
 - Alternate 16-Bit Timer/Counter mode
 - Interrupt on position counter rollover/underflow

Analog-to-Digital Converters (ADCs):

- Up to two ADC modules in a device
- 10-bit, 1.1 Msps or 12-bit, 500 Ksps conversion:
 - Two, four or eight simultaneous samples
 - Up to 32 input channels with auto-scanning
 - Conversion start can be manual or synchronized with one of four trigger sources
 - Conversion possible in Sleep mode
 - ±1 LSb max integral nonlinearity
 - ±1 LSb max differential nonlinearity

CMOS Flash Technology:

- Low-power, high-speed Flash technology
- Fully static design
- 3.3V (±10%) operating voltage
- Industrial and extended temperature
- Low-power consumption

Packaging:

- 100-pin TQFP (14x14x1 mm and 12x12x1 mm)
- 80-pin TQFP (12x12x1 mm)
- 64-pin TQFP (10x10x1 mm)
- 64-pin QFN (9x9x0.9 mm)

Note: See the device variant tables for exact peripheral features per device.

LM5100/LM5101

High Voltage High Side and Low Side Gate Driver

General Description

The LM5100/LM5101 High Voltage Gate Drivers are designed to drive both the high side and the low side N-Channel MOSFETs in a synchronous buck or a half bridge configuration. The floating high-side driver is capable of operating with supply voltages up to 100V. The outputs are independently controlled with CMOS input thresholds (LM5100) or TTL input thresholds (LM5101). An integrated high voltage diode is provided to charge the high side gate drive bootstrap capacitor. A robust level shifter operates at high speed while consuming low power and providing clean level transitions from the control logic to the high side gate driver. Under-voltage lockout is provided on both the low side and the high side power rails. This device is available in the standard SOIC-8 pin and the LLP-10 pin packages.

Features

- Drives both a high side and low side N-Channel MOSFET
- Independent high and low driver logic inputs (TTL for LM5101 or CMOS for LM5100)

- Bootstrap supply voltage range up to 118V DC
- Fast propagation times (25 ns typical)
- Drives 1000 pF load with 15 ns rise and fall times
- Excellent propagation delay matching (3 ns typical)
- Supply rail under-voltage lockouts
- Low power consumption
- Pin compatible with HIP2100/HIP2101

Typical Applications

- Current Fed push-pull converters
- Half and Full Bridge power converters
- Synchronous buck converters
- Two switch forward power converters
- Forward with Active Clamp converters

Package

- SOIC-8
- LLP-10 (4 mm x 4 mm)

Simplified Block Diagram

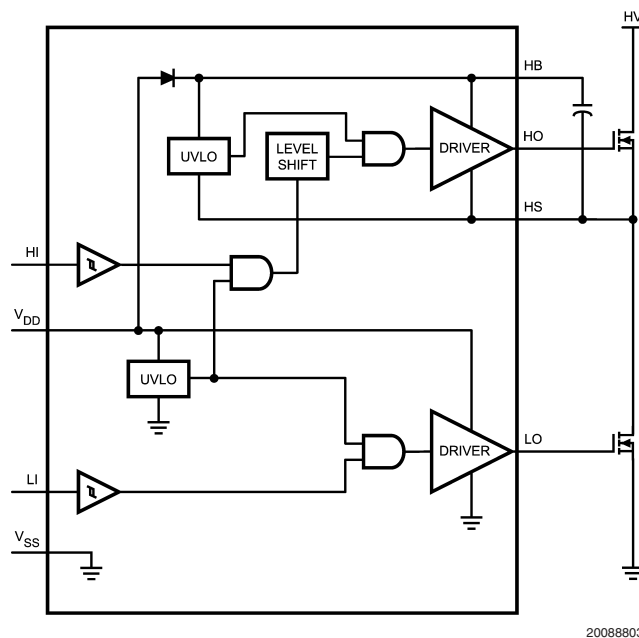


FIGURE 1.

LM2671 SIMPLE SWITCHER® Power Converter High Efficiency 500mA Step-Down Voltage Regulator with Features

General Description

The LM2671 series of regulators are monolithic integrated circuits built with a LDMOS process. These regulators provide all the active functions for a step-down (buck) switching regulator, capable of driving a 500mA load current with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5.0V, 12V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include patented internal frequency compensation (Patent Nos. 5,382,918 and 5,514,947), fixed frequency oscillator, external shutdown, soft-start, and frequency synchronization.

The LM2671 series operates at a switching frequency of 260 kHz, thus allowing smaller sized filter components than what would be needed with lower frequency switching regulators. Because of its very high efficiency (>90%), the copper traces on the printed circuit board are the only heat sinking needed.

A family of standard inductors for use with the LM2671 are available from several different manufacturers. This feature greatly simplifies the design of switch-mode power supplies using these advanced ICs. Also included in the datasheet are selector guides for diodes and capacitors designed to work in switch-mode power supplies.

Other features include a guaranteed $\pm 1.5\%$ tolerance on output voltage within specified input voltages and output load conditions, and $\pm 10\%$ on the oscillator frequency. External shutdown is included, featuring typically 50 μA stand-by current. The output switch includes current limiting, as well as thermal shutdown for full protection under fault conditions.

To simplify the LM2671 buck regulator design procedure, there exists computer design software, *LM267X Made Simple* (version 6.0).

Features

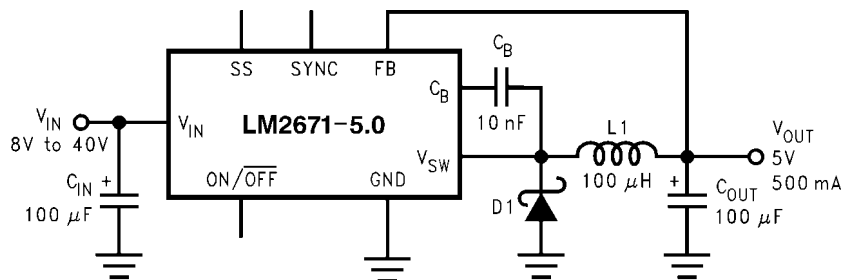
- Efficiency up to 96%
- Available in SO-8, 8-pin DIP and LLP packages
- Computer Design Software *LM267X Made Simple* (version 6.0)
- Simple and easy to design with
- Requires only 5 external components
- Uses readily available standard inductors
- 3.3V, 5.0V, 12V, and adjustable output versions
- Adjustable version output voltage range: 1.21V to 37V
- $\pm 1.5\%$ max output voltage tolerance over line and load conditions
- Guaranteed 500mA output load current
- 0.25 Ω DMOS Output Switch
- Wide input voltage range: 8V to 40V
- 260 kHz fixed frequency internal oscillator
- TTL shutdown capability, low power standby mode
- Soft-start and frequency synchronization
- Thermal shutdown and current limit protection

Applications

- Simple High Efficiency (>90%) Step-Down (Buck) Regulator
- Efficient Pre-Regulator for Linear Regulators

Typical Application

(Fixed Output Voltage Versions)



10004201

ADM2682E/ADM2687E

FEATURES

- 5 kV rms isolated RS-485/RS-422 transceiver, configurable as half or full duplex
- isoPower* integrated isolated dc-to-dc converter
- ± 15 kV ESD protection on RS-485 input/output pins
- Complies with ANSI/TIA/EIA-485-A-98 and ISO 8482:1987(E)
- Data rate: 16 Mbps (ADM2682E), 500 kbps (ADM2687E)
- 5 V or 3.3 V operation
- Connect up to 256 nodes on one bus
- Open- and short-circuit, fail-safe receiver inputs
- High common-mode transient immunity: >25 kV/ μ s
- Thermal shutdown protection
- Safety and regulatory approvals
 - UL recognition (pending)
 - 5000 V rms for 1 minute per UL 1577
 - CSA Component Acceptance Notice #5A (pending)
 - IEC 60601-1: 400 V rms (basic), 250 V rms (reinforced)
 - IEC 60950-1: 600 V rms (basic), 380 V rms (reinforced)
 - VDE Certificates of Conformity (pending)
 - DIN EN 60747-5-2 (VDE 0884 Part 2): 2003-01
 - $V_{IORM} = 846$ V peak
- Operating temperature range: -40°C to $+85^{\circ}\text{C}$
- 16-lead wide-body SOIC with >8 mm creepage and clearance

APPLICATIONS

- Isolated RS-485/RS-422 interfaces
- Industrial field networks
- Multipoint data transmission systems

GENERAL DESCRIPTION

The ADM2682E/ADM2687E are fully integrated 5 kV rms signal and power isolated data transceivers with ± 15 kV ESD protection and are suitable for high speed communication on multipoint transmission lines. The ADM2682E/ADM2687E include an integrated 5 kV rms isolated dc-to-dc power supply that eliminates the need for an external dc-to-dc isolation block.

They are designed for balanced transmission lines and comply with ANSI/TIA/EIA-485-A-98 and ISO 8482:1987(E).

The devices integrate Analog Devices, Inc., *iCoupler*[®] technology to combine a 3-channel isolator, a three-state differential line driver, a differential input receiver, and Analog Devices *isoPower*[®] dc-to-dc converter into a single package. The devices are powered by a single 5 V or 3.3 V supply, realizing a fully integrated signal and power isolated RS-485 solution.

FUNCTIONAL BLOCK DIAGRAM

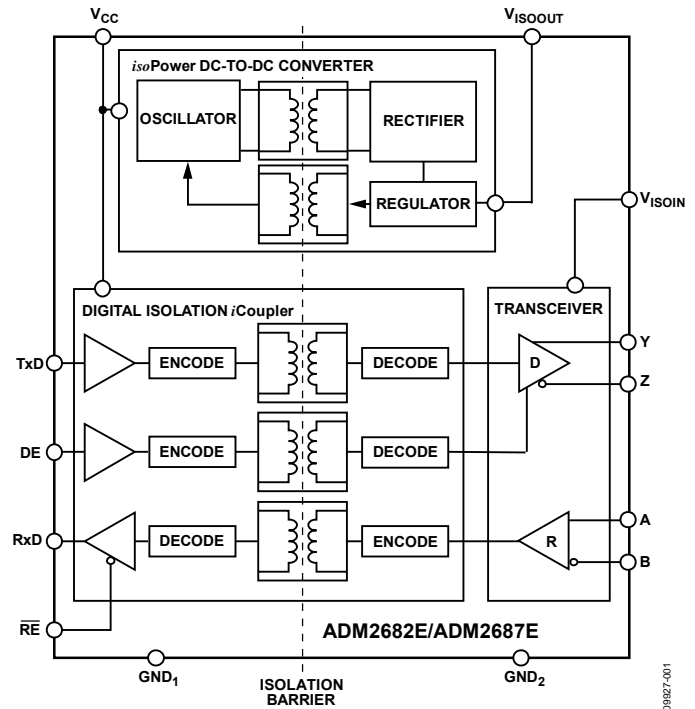


Figure 1.

The ADM2682E/ADM2687E drivers have an active high enable. An active low receiver enable is also provided, which causes the receiver output to enter a high impedance state when disabled.

The devices have current limiting and thermal shutdown features to protect against output short circuits and situations where bus contention may cause excessive power dissipation. The parts are fully specified over the industrial temperature range and are available in a highly integrated, 16-lead, wide-body SOIC package with >8 mm creepage and clearance.

The ADM2682E/ADM2687E contain *isoPower* technology that uses high frequency switching elements to transfer power through the transformer. Special care must be taken during printed circuit board (PCB) layout to meet emissions standards. Refer to AN-0971 Application Note, *Recommendations for Control of Radiated Emissions with isoPower Devices*, for details on board layout considerations.

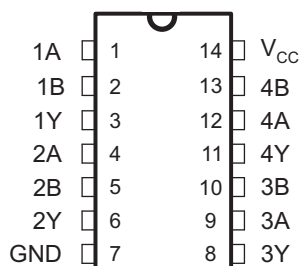
Rev. 0

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners.

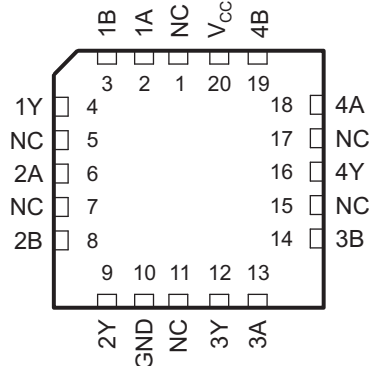
FEATURES

- Wide Operating Voltage Range of 2 V to 6 V
- Outputs Can Drive Up To 10 LSTTL Loads
- Low Power Consumption, 20- μ A Max I_{CC}
- Typical $t_{pd} = 8$ ns
- ± 4 -mA Output Drive at 5 V
- Low Input Current of 1 μ A Max

SN54HC04...J OR W PACKAGE
SN74HC04...D, DB, N, NS, OR PW PACKAGE
(TOP VIEW)



SN54HC04...FK PACKAGE
(TOP VIEW)



NC – No internal connection

DESCRIPTION/ORDERING INFORMATION

The 'HC08 devices contain four independent 2-input AND gates. They perform the Boolean function $Y = A \cdot B$ or $Y = \overline{A + B}$ in positive logic.

ORDERING INFORMATION

T_A	PACKAGE ⁽¹⁾		ORDERABLE PART NUMBER	TOP-SIDE MARKING
-40°C to 85°C	PDIP – N	Reel of 1000	SN74HC08N	SN74HC08N
		Reel of 1000	SN74HC08DE4	HC08
	SOIC – D	Reel of 2500	SN74HC08DR	
		Tube of 250	SN74HC08DT	
	SOP – NS	Reel of 2000	SN74HC08NSR	
			SN74HC08NSRG4	
	SSOP – DB	Reel of 2000	SN74HC08DBR	HC08
			SN74HC08DBRE4	
	TSSOP – PW	Tube of 90	SN74HC08PW	HC08
		Reel of 2000	SN74HC08PWR	
Tube of 250		SN74HC08PWT		
-55°C to 125°C	CDIP – J	Reel of 1000	SNJ54HC08J	SNJ54HC08J
	CFP – W	Reel of 900	SNJ54HC08W	SNJ54HC08W
	LCCC –FK	Reel of 2200	SNJ54HC08FK	SNJ54HC08JFK

(1) Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.



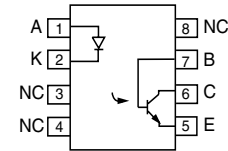
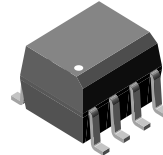
Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.



Optocoupler, Phototransistor Output, With Base Connection in SOIC-8 package

Features

- High BV_{CEO} , 70 V
- Isolation Test Voltage, 3000 V_{RMS}
- Industry Standard SOIC-8A Surface Mountable Package
- Compatible with Dual Wave, Vapor Phase and IR Reflow Soldering



Agency Approvals

- UL - File No. E52744 System Code Y
- DIN EN 60747-5-2(VDE0884)
DIN EN 60747-5-5 pending
Available with Option 1

A specified minimum and maximum CTR allows a narrow tolerance in the electrical design of the adjacent circuits. The high BV_{CEO} of 70 V gives a higher safety margin compared to the industry standard 30 V.

Description

The IL205AT/ IL206AT/ IL207AT/ IL208AT are optically coupled pairs with a Gallium Arsenide infrared LED and a silicon NPN phototransistor. Signal information, including a DC level, can be transmitted by the device while maintaining a high degree of electrical isolation between input and output. This family comes in a standard SOIC-8A small outline package for surface mounting which makes them ideally suited for high density application with limited space. In addition to eliminating through-hole requirements, this package conforms to standards for surface mounted devices.

Order Information

Part	Remarks
IL205AT	CTR 40 - 80 %, SOIC-8
IL206AT	CTR 63 - 125 %, SOIC-8
IL207AT	CTR 100 - 200 %, SOIC-8
IL208AT	CTR 160 - 320 %, SOIC-8

Available on Tape and Reel only.

For additional information on the available options refer to Option Information.

Absolute Maximum Ratings

$T_{amb} = 25\text{ }^{\circ}\text{C}$, unless otherwise specified

Stresses in excess of the absolute Maximum Ratings can cause permanent damage to the device. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operational sections of this document. Exposure to absolute Maximum Rating for extended periods of the time can adversely affect reliability.

Input

Parameter	Test condition	Symbol	Value	Unit
Peak reverse voltage		V_R	6.0	V
Forward continuous current		I_F	60	mA
Power dissipation		P_{diss}	90	mW
Derate linearly from 25 $^{\circ}\text{C}$			1.2	mW/ $^{\circ}\text{C}$

Appendix E

Source Code

E.1 Microcontroller

In this section some of the code implemented for the microcontroller is added. This includes the `linkedList` files, message handling files, `pwm` files, some functions for turning on and off valves, `systemTask`, `checkMsg`, and `sprayCommand`.

E.1.1 `linkedList.h`

```
1  /*****  
2  /**@file linkedList.h  
3  *  
4  * @author Frode Urdal  
5  *****/  
6  
7  #ifndef __LINKEDLIST  
8  #define __LINKEDLIST  
9  
10 struct linkedListNode  
11 {  
12     INTEGER32 imageNum;  
13     UNSIGNED32 time;  
14     UNSIGNED32 bitmap;  
15     struct linkedListNode *next;  
16 };  
17  
18 struct linkedList  
19 {  
20     struct linkedListNode *head;  
21     struct linkedListNode *tail;  
22 };  
23  
24 void linkedListCreateNode(struct linkedList** list , INTEGER32 num, UNSIGNED32 timer ,  
    INTEGER32 map);
```

```

25 void linkedListAddLast(struct linkedList** list , INTEGER32 num, UNSIGNED32 timer ,
    INTEGER32 map);
26 void linkedListDeleteFirst(struct linkedList** list);
27 void linkedListCreateList(struct linkedList** list);
28
29 #endif

```

E.1.2 likedList.c

```

1  /* ***** */
2  /**@file likedList.c
3  *
4  * @author Frode Urdal
5  * ***** */
6
7  #include "sysutil.h"
8  //#include "p33FJ64MC506A.h"
9  //#include "sysdef.h"
10 //#include "iodef.h"
11 #include "linkedList.h"
12 #include <stdlib.h>
13 #include <stdio.h>
14
15 //Function for creating the list
16 void linkedListCreateList(struct linkedList **list)
17 {
18     struct linkedList *ptr = (struct linkedList*)malloc(sizeof(struct linkedList));
19     ptr->head = NULL;
20     ptr->tail = NULL;
21     *list = ptr;
22 }
23
24 //Function for creating the the first node
25 void linkedListCreateNode(struct linkedList** list , INTEGER32 num, UNSIGNED32 timer ,
    INTEGER32 map)
26 {
27     struct linkedListNode *ptr = (struct linkedListNode*)malloc(sizeof(struct
        linkedListNode));
28     if(ptr == NULL)
29         //do something
30         return;
31     ptr->imageNum = num;
32     ptr->time = timer;
33     ptr->bitmap = map;
34     ptr->next = NULL;
35     (*list)->head = ptr;
36     (*list)->tail = ptr;
37 }
38
39 //Function for adding an element at the end
40 void linkedListAddLast(struct linkedList** list , INTEGER32 num, UNSIGNED32 timer ,
    INTEGER32 map)

```

```

41 {
42     struct linkedListNode *ptr = (struct linkedListNode*) malloc(sizeof(struct
        linkedListNode));
43     if((*list)->head == NULL)
44     {
45         free(ptr);
46         linkedListCreateNode(list, num, timer, map);
47     }
48
49     if(ptr == NULL)
50         //do something
51         return;
52     ptr->imageNum = num;
53     ptr->time = timer;
54     ptr->bitmap = map;
55     ptr->next = NULL;
56     (*list)->tail->next = ptr;
57     (*list)->tail = ptr;
58 }
59
60 //Function for deleting the first element
61 void linkedListDeleteFirst(struct linkedList** list)
62 {
63     struct linkedListNode* temp = (*list)->head;
64     if((*list)->head == (*list)->tail)
65     {
66         (*list)->head = NULL;
67         (*list)->tail = NULL;
68     }
69     else
70     {
71         (*list)->head = (*list)->head->next;
72     }
73     free(temp);
74 }
75 /*
76 //Main for testing on computer
77 #ifndef __main
78 #define __main
79
80 int main(void)
81 {
82     int i = 0;
83     struct linkedList *myList;
84     linkedListCreateList(&myList);
85     while(TRUE)
86     {
87         for(i = 0; i<10; i++)
88             linkedListAddLast(&myList, i, i, i);
89         for(i = 0; i<10; i++)
90             linkedListDeleteFirst(&myList);
91     }
92     return 0;
93 }

```

```

94
95 #endif
96 */

```

E.1.3 msgHand.h

```

1  /*****
2  /**@file msgHand.h
3  *
4  * @author Frode Urdal
5  *****/
6
7  #include "linkedList.h"
8
9  BYTE generateChecksum(BYTE* msg);
10 BOOL validateChecksum(BYTE* msg);
11 INTEGER16 readMsg(BYTE* msg, INTEGER16 *pcbNum, INTEGER16 *pcbDir, INTEGER32 *shutterTime
    , BOOL *purge, struct linkedList** list);
12 BOOL validateMsg(BYTE* msg, INTEGER16 count);

```

E.1.4 msgHand.c

```

1  /*****
2  /**@file msgHand.c
3  *
4  * @author Frode Urdal
5  *****/
6
7  #include "sysutil.h"
8  #include "p33FJ64MC506.h"
9  #include "sysdef.h"
10 #include "iodef.h"
11 #include "msgHand.h"
12 #include "linkedList.h"
13
14 //Function for generating the checksum, returns the checksum
15 BYTE generateChecksum(BYTE* msg)
16 {
17     BYTE check1 = 0;
18     INTEGER32 check2 = 0;
19     INTEGER16 i = 0;
20     INTEGER16 length = msg[1];
21     for(i = 0; i < length; i++)
22     {
23         check1 += msg[2+i];
24         check2 += msg[2+i];
25     }
26     check2 = check2%256;
27     if(check1 == check2)
28         return check1;

```

```

29     return 0x00;
30 }
31
32 //Function for verifying the checksum, returns true if it's a match
33 BOOL validateChecksum (BYTE* msg)
34 {
35     BYTE check1 = 0;
36     INTEGER32 check2 = 0;
37     INTEGER16 i = 0;
38     INTEGER16 length = msg[1];
39     INTEGER16 temp = msg[2+length];
40     temp = temp;
41     for(i = 0; i < length; i++)
42     {
43         check1 += msg[2+i];
44         check2 += msg[2+i];
45     }
46     check2 = check2%256;
47     if(check1 == check2 && check1 == msg[2+length])
48         return TRUE;
49     return FALSE;
50 }
51
52 //Function for reading the init message, returns true if successful
53 static BOOL readInit (BYTE* msg, INTEGER16 *pcbNum, INTEGER16 *pcbDir, INTEGER32 *
shutterTime, BOOL *purge)
54 {
55     if (validateChecksum (msg))
56     {
57         *pcbNum = msg[3];
58         *pcbDir = msg[4];
59         *shutterTime = (0xFFFF&msg[5]) + ((0xFFFF&msg[6]) <<8) + ((0xFFFF&msg[7]) <<16) + ((0
xFFFFFFFF&msg[8]) <<24);
60         if (msg[9] == 0x00)
61             *purge = FALSE;
62         else if (msg[9] == 0x01)
63             *purge = TRUE;
64         return TRUE;
65     }
66     return FALSE;
67 }
68
69 //Function for reading the command message, returns true if successful
70 static BOOL readCommand (BYTE* msg, struct linkedList** list)
71 {
72     INTEGER32 tempImg = 0;
73     UNSIGNED32 tempTime = 0;
74     UNSIGNED32 tempBitmap = 0;
75     INTEGER16 i = 0;
76     if (validateChecksum (msg))
77     {
78         for (i = 0; i < 4; i++)
79         {
80             tempImg += (0xFFFFFFFF&msg[3+i]) <<(i*8);

```



```

81         tempTime += (0xFFFFFFFF&msg[7+i])<<(i*8);
82         tempBitmap += (0xFFFFFFFF&msg[11+i])<<(i*8);
83     }
84     if ((*list)->head == NULL)
85         linkedListCreateNode(list, tempImg, tempTime, tempBitmap);
86     else
87         linkedListAddLast(list, tempImg, tempTime, tempBitmap);
88     return TRUE;
89 }
90 return FALSE;
91 }
92
93 //Function for deciding if it's a command message or init message that is sent
94 INTEGER16 readMsg(BYTE* msg, INTEGER16 *pcbNum, INTEGER16 *pcbDir, INTEGER32 *shutterTime
95 , BOOL *purge, struct linkedList** list)
96 {
97     if (msg[0] == 0x69) // i
98     {
99         if (readInit(msg, pcbNum, pcbDir, shutterTime, purge))
100             return 1;
101     }
102     else if (msg[0] == 0x73) // s
103     {
104         if (readCommand(msg, list))
105             return 2;
106     }
107     return -1;
108 }
109
110 BOOL validateMsg(BYTE* msg, INTEGER16 count)
111 {
112     if (msg[count-2] != 0x65)
113         return FALSE;
114     if (msg[1] != count-0x05)
115         return FALSE;
116     return TRUE;
117 }

```

E.1.5 pwm.h

```

1  /*****
2  ** \file PWM.h
3  *
4  * @author Frode Urdal
5  *****/
6
7  void initPWM(INTEGER32 shutterTime);
8  BOOL runPWM(void);
9  BOOL purgeValves(void);
10 BOOL staticPurge(void);
11 BOOL dynamicPurge(INTEGER16 freq);

```

E.1.6 pwm.c

```

1  /*****
2  /** @file pwm.c
3  * Controls PWM for valve control.
4  *
5  * @author Frode Urdal
6  *****/
7  #include "sysutil.h"
8  #include "p33FJ64MC506.h"
9  #include "iodef.h"
10 #include "clock.h"
11 #include "sysdef.h"
12 #include "pwm.h"
13
14 /*Do not modify the foloving constants*/
15 #define SPIKE_TIME      650          //0.3ms = 300, 650, 900
16 #define SHUTDOWN_TIME  150          //0.05ms = 50, 150, 200
17 #define TIC             50          //0.05ms
18 #define HOLD_VOLTAGE   PTPER*0.64  //3.96V = 0.33, 0.64 = 7.68V, 0.75 = 9V
19 #define SPIKE_VOLTAGE  PTPER*2      //24V
20
21 /*Do not modify the following variables*/
22 static INTEGER16 START = 0;
23 static INTEGER16 HOLD = SPIKE_TIME/TIC;
24 static INTEGER16 END;           // (SPIKE_TIME+holdTime)/TIC
25 static INTEGER16 SHUTDOWN;     // (SPIKE_TIME+holdTime+SHUTDOWN_TIME)/TIC
26 static INTEGER16 RESET;        //period/TIC-1
27
28 static INTEGER32 holdTime = 7100; //7.7ms = 7700
29 static UNSIGNED32 pwmTimer;
30 static INTEGER32 pwmCounter;
31 static INTEGER16 initMode;
32 static INTEGER16 frequency;
33 static INTEGER16 purgeTime;
34
35 void initPWM(INTEGER32 shutterTime)
36 {
37     SysGetTimer(&pwmTimer);
38     pwmCounter = -1;
39     initMode = 0;
40     frequency = 100;
41     purgeTime = 0;
42     holdTime = shutterTime - SPIKE_TIME;
43     END = (SPIKE_TIME+holdTime)/TIC;
44     SHUTDOWN = ((SPIKE_TIME+holdTime)/TIC)+SHUTDOWN_TIME/TIC;
45     RESET = SHUTDOWN+1;        //USE this when using the real time system
46 }
47
48 //Return true when done
49 BOOL runPWM(void)
50 {
51     if (SysDelay(&pwmTimer, TIC))
52     {

```

```

53     pwmCounter++;
54     if (pwmCounter == START)
55         PDC2 = SPIKE_VOLTAGE;
56     else if (pwmCounter == HOLD)
57         PDC2 = HOLD_VOLTAGE;
58     else if (pwmCounter == END)
59     {
60         PDC2 = 0;
61         PDC1 = PTPER;
62     }
63     else if (pwmCounter == SHUTDOWN)
64         PDC1 = 0;
65     else if (pwmCounter == RESET)
66     {
67         pwmCounter = -1;
68         return TRUE;
69     }
70 }
71 return FALSE;
72 }
73
74 /* Returns true when the purging is done
75 * Purging process:
76 * 1. The Valve should be held open for 10 sec.
77 * 2. Run the following sequence of different frequencies 2 times:
78 *    100 Hz, 150Hz, 200Hz, 250Hz, 300Hz, 350Hz, 400Hz, 450Hz and 500Hz
79 *    Each frequency should be held for 5 seconds*/
80 BOOL purgeValves(void)
81 {
82     if (SysDelay(&pwmTimer, TIC))
83     {
84         pwmCounter++;
85         if (initMode == 0)
86         {
87             if (staticPurge())
88                 initMode++;
89         }
90         if (initMode == 1 || initMode == 2)
91         {
92             if (dynamicPurge(frequency))
93             {
94                 frequency = frequency + 50;
95                 if (frequency == 550)
96                 {
97                     frequency = 100;
98                     initMode++;
99                 }
100            }
101        }
102        if (initMode == 3)
103        {
104            pwmCounter = -1;
105            return TRUE;
106        }

```

```
107     }
108     return FALSE;
109 }
110
111 BOOL staticPurge(void)
112 {
113     if(pwmCounter == START)
114         PDC2 = SPIKE_VOLTAGE;
115     else if(pwmCounter == HOLD)
116         PDC2 = HOLD_VOLTAGE;
117     else if(pwmCounter == 100000*100/TIC)
118     {
119         PDC2 = 0;
120         PDC1 = PTPER;
121     }
122     else if(pwmCounter == 100000*100/TIC+1)
123         PDC1 = 0;
124     else if(pwmCounter == 100000*100/TIC+41)
125     {
126         pwmCounter = -1;
127         return TRUE;
128     }
129     return FALSE;
130 }
131
132 BOOL dynamicPurge(INTEGER16 freq)
133 {
134     if(pwmCounter == 0)
135         PDC2 = SPIKE_VOLTAGE;
136     else if(pwmCounter == HOLD)
137         PDC2 = HOLD_VOLTAGE;
138     else if(pwmCounter == (INTEGER32)((8000/TIC)*(100.0/freq)))
139     {
140         PDC2 = 0;
141         PDC1 = PTPER;
142     }
143     else if(pwmCounter == (INTEGER32)((8000/TIC)*(100.0/freq))+1)
144         PDC1 = 0;
145     else if(pwmCounter == (INTEGER32)((8000/TIC)*(100.0/freq))+10)
146     {
147         pwmCounter = -1;
148         purgeTime++;
149     }
150     if(purgeTime == 5*freq)
151     {
152         pwmCounter = -1;
153         purgeTime = 0;
154         return TRUE;
155     }
156     return FALSE;
157 }
```

E.1.7 Turn On and Off Valves

```

1 void turnOffAll(void)
2 {
3     UNSIGNED16 temp = LATD;
4     LATB = 0x0000;           //Turn off EN1-16
5     temp = temp>>8;        //Clear the eight last bits of LATD, Turn off EN17-24
6     temp = temp<<8;
7     LATD = temp;
8     EN25 = LED_OFF;
9     EN26 = LED_OFF;
10 }
11
12 //Function for enabeling all LEDs
13 void turnOnAll(void)
14 {
15     UNSIGNED16 temp = LATD;
16     LATB = 0xFFFF;         //Turn on EN1-16
17     temp = temp>>8;        //Clear the eight last bits of LATD and set them to 1,
18     turns on EN17-24
19     temp = temp<<8;
20     LATD = temp+0x00FF;
21     EN25 = LED_ON;
22     EN26 = LED_ON;
23 }
24 /*Function for enableing valve outputs, lsb = EN1
25 * EN1 - EN16 = LATB0-LATB15
26 * EN17 - EN24 = LATD0-LATD7
27 * EN25 and EN26 = LATG0 and LATG1
28 */
29 void enableValves(UNSIGNED32 bitmap)
30 {
31     UNSIGNED16 temp = LATD;
32     LATB = 0x0000FFFF&bitmap; //Assign the last 16 bit to LATB
33     temp = temp>>8;          //Clear the eight last bits of LATD
34     temp = temp<<8;
35     temp = temp + (0xFF&(bitmap>>16)); //Shift the bitmap 16 places to the right and add
36     the eight last bits to LATD
37     LATD = temp;
38     EN25 = (0x01000000&bitmap)>>24; //Set EN25
39     EN26 = (0x02000000&bitmap)>>25; //Set EN26
40 }

```

E.1.8 systemTask, checkMsg and sprayCommand

```

1 void SystemTask(void)
2 {
3     UNSIGNED32 tempTime = 0;
4
5     LedUpdate();
6

```

```

7   if (NewMess)
8       checkMsg();
9   if (pastTrig < TRIG)
10      pastTrig = TRIG;
11  if (pastTrig > TRIG)
12  {
13      pastTrig = TRIG;
14      SysGetTimer(&tempTime);
15      linkedListAddLast(&triggers, imageNum, tempTime, 0);
16      imageNum++;
17  }
18
19  if (purge)
20  {
21      turnOnAll();
22      if (purgeValves())
23      {
24          purge = FALSE;
25          turnOffAll();
26      }
27  }
28  else if (!purge)
29  {
30      if (!dispense)
31          sprayCommand();
32      if (dispense)
33          if (runPWM())
34          {
35              dispense = FALSE;
36              turnOffAll();
37          }
38  }
39  asm (" CLRWDT");
40 }
41
42 void checkMsg(void)
43 {
44     if (validateMsg(RxBuf, RxCnt))
45     {
46         memcpy(msg, RxBuf, RxCnt);
47         lastMsg = readMsg(msg, &pcbNum, &pcbDir, &shutterTime, &purge, &commands);
48         if (lastMsg == 1)
49         {
50             initPWM(shutterTime);
51             clearMsg();
52         }
53         else if (lastMsg == 2)
54         {
55             clearMsg();
56             //Test sequence
57             /*      turnOffAll();
58                enableValves(commands->head->bitmap);
59                linkedListDeleteFirst(&commands);
60                dispense = TRUE;*/

```

```

61     }
62     else
63     {
64         clearMsg(); //Something went wrong
65     }
66 }
67 else
68 {
69     NewMess = FALSE;
70     clearMsg();
71 }
72 }
73
74 void sprayCommand(void)
75 {
76     UNSIGNED32 Temp=0;
77
78     if(commands->head != NULL && triggers->head != NULL)
79     {
80         if(commands->head->imageNum > triggers->head->imageNum)
81             linkedListDeleteFirst(&triggers);
82         if(triggers->head != NULL)
83         {
84             SysGetTimer(&Temp);
85             if(Temp > triggers->head->time + responseTime && SysDelay(&(triggers->head->
86                 time), commands->head->time - responseTime))
87             {
88                 enableValves(commands->head->bitmap);
89                 dispense = TRUE;
90                 linkedListDeleteFirst(&commands);
91             }
92         }
93     }

```

E.2 Valve Mapper Algorithm

In this section the important files of the valve mapper algorithm are added. In addition the setup file and one test file is included to show how they are implemented. All `__init__.py` files are blank.

E.2.1 setup.py

```

1 try:
2     from setuptools import setup
3 except ImportError:
4     from distutils.core import setup
5

```

```

6 config = {
7     'description': 'Valve mapper project',
8     'author': 'Frode Urdal',
9     'url': 'URL to get it at.',
10    'download_url': 'Where to download it.',
11    'author_email': 'frode@adigo.no',
12    'version': '0.1',
13    'install_requires': ['nose'],
14    'packages': ['valveMapper'],
15    'scripts': [],
16    'name': 'ValveMapper'
17 }
18
19 setup(**config)

```

E.2.2 config.py

```

1 __author__ = 'frodeu'
2
3 """ This config file is used to define constants and includes
4     Length of the valve array, roation from vehicle to valveArray etc.
5 """
6 import numpy as np
7
8 # Robot configurations
9
10 # Length of the valve array in m
11 valveLength = 0.156
12
13 # Width of the nozzle in m
14 nozzleWidth = 0.000254
15
16 # position of center relative to corner
17 nozzleArrayCenter = np.matrix([[nozzleWidth/2],[valveLength/2]])
18
19 # Distance (c/c) for the valves in m
20 valveDist = 0.006
21
22 # Rotation from vehicle to valve array
23 rotBtoV = np.matrix([[1,1,1],
24                      [1,1,1],
25                      [1,1,1]])
26
27 # Valve operation
28
29 # Frequency of the valve in Hz
30 valveFrequency = 100.0

```



```
31
32 # Shuttertime of valves in micro seconds
33 shutterTime = 8000
34
35 # Number of valves
36 numberOfValves = 26
37
38 # PCB number for when several cards are connected
39 pcbNum = b'\x00'
40
41 # PCB direction, 0x00 of valve 1 is first, 0x01 if valve 26 is first
42 pcbDir = b'\x00'
43
44 # PCB address
45 pcbAddr = b'\x41'
46
47 # ValveMapper parameters
48
49 # Minimum time needed from command sent to actuation in s
50 minTime = 0.01
51
52 # Maximum time for generating a command before it should be executed in s
53 maxOperationTime = 100
54
55 # Percentage of valve area needed to be covered by weeds to spray
56 weed = 0.25
57
58 # Percentage of valve area that can be covered by crop and still be spray
59 crop = 0.1
60
61 # RS 485 parameters
62
63 # Baudrate
64 baudrate = 19200
65
66 # Com port
67 comPort = 'COM4'
```

E.2.3 valveMapper.py

```
1 __author__ = 'frodeu'
2
3 from com import com as com
4 import config as con
5 import spraymap as sm
6 import navigation as nav
7 import math
```

```

8 import numpy as np
9 import time
10 import struct
11 import bitstring as bit
12
13
14 newSprayMap = False
15 newImagePos = 0
16 newImageTime = 0
17 newImage = 0
18 newPos = True
19 newPosV = 0
20 newVelocity = 0
21 newTime = 0
22 lastSprayTime = 0.0
23
24 #Function for initializing the PCB
25 def initPCB(msg):
26     shutter = con.shutterTime
27     pcbNum = con.pcbNum
28     pcbDir = con.pcbDir
29     addr = con.pcbAddr
30     purge = False
31     return com.generateInit(addr, shutter, purge, pcbNum, pcbDir, msg)
32
33 # Function for calculating the minimum distance between to spray commands
34 def minInterval():
35     xDot = 0
36     yDot = 0
37     deltaD = con.shutterTime * math.sqrt(math.pow(xDot,2) + math.pow(yDot,2))
38     return deltaD
39
40 # calculates what image is closest by using the center pixle fo each image
41 def whatPicture(pos1, rot1, img1, pos2, rot2, img2, scale, posV, rotV):
42     valve = posV + rotV*con.nozzleArrayCenter
43     pixel1 = sm.posFromPixel(pos1, np.matrix([[int(len(img1)/2), int(len(img1)
44         [0])/2]]), scale, rot1)
45     pixel2 = sm.posFromPixel(pos2, np.matrix([[int(len(img2)/2), int(len(img2)
46         [0])/2]]), scale, rot2)
47     if(np.linalg.norm(valve-pixel1) < np.linalg.norm(valve-pixel2)):
48         return 1
49     else:
50         return 2
51
52 # Calculate the minimum distance that is needed from valve array to weed
53 def minDistance(vel):
54     return vel*con.minTime

```

```

54 # Calculates the maximum working distance
55 def maxWorkDistance(vel):
56     return vel*con.maxOperationTime
57
58 # Convert the bitarray to an integer
59 def generateBitmap(bits):
60     bits.reverse(0,len(bits))
61     bitmap = bits.uint
62     return bitmap
63
64 def main():
65     # Scale for image, pix/m
66     scale = 2000.0
67
68     lastImage = -1
69     currentImage = 0
70     nextImage = 0
71     imageNumber = 0
72     sprayTime = 0
73     init = bytearray(13)
74     weeds = np.matrix([[0, 0]])
75     images = np.array([[image1],[image2]])
76     imagePos = np.array([[image1pos], [image2pos]])
77     imageTime = np.array([[image1Time], [image2Time]])
78     command = bytearray(18)
79     rot = np.array([[nav.getRot(np.pi/2)], [nav.getRot(np.pi/2)]])
80     posV = np.matrix([[0.0], [-0.05]])
81     rotV = nav.getRot(0)
82     lastSprayTime = -1.0
83
84     #Test variables
85     count = 0
86     commands = np.zeros(1000)
87
88     init = initPCB(init)
89     com.openPort()
90     com.sendMessage(init)
91
92     while(True):
93         # Read position log and new spray maps
94         if(newImage):
95             images = np.concatenate((images,newImage))
96             imagePos = np.concatenate((imagePos, newImagePos))
97             imageTime = np.concatenate((imageTime, newImageTime))
98         if(newPos):
99             posV += np.matrix([[0.002],[0.0]]) # = newPosV
100             #velocity = newVelocity
101             currentTime += 0.005 # = newTime

```

```

102     #Find the weed centers of new images
103     if(currentImage != nextImage and len(images) >= 1):
104         nextImage = currentImage
105         lastImage -= 1
106         imageNumber += 1
107         imagePos = np.delete(imagePos, 0, 0)
108         imageTime = np.delete(imageTime, 0, 0)
109         images = np.delete(images, 0, 0)
110         lastSprayTime = -1.0
111     elif(len(images) <= 1 and len(weeds) is 1):
112         break
113     if(len(images) > 1 and whatPicture(imagePos[currentImage][0], rot[
114         currentImage][0], images[currentImage][0], imagePos[
115         currentImage+1][0], rot[currentImage+1][0], images[
116         currentImage+1][0], scale, posV, rotV) is 2):
117         nextImage += 1
118     if(lastImage != currentImage):
119         temp = sm.findWeedCenters(images[currentImage][0],
120             imagePos[currentImage][0], scale, nav.getRot(math.pi))
121         weeds = np.concatenate((weeds, temp))
122         lastImage = currentImage
123     # Find out when the nozzles are at the weeds and where to spray
124     if(len(weeds) > 1):
125         (sprayTime, intersection) = nav.findTime(posV, currentTime
126             , velocity, sm.posFromPixel(imagePos[currentImage][0],
127             weeds[1, :], scale, rot[currentImage][0]))
128     if((sprayTime) > con.minTime and (sprayTime) < con.
129         maxOperationTime and currentTime + sprayTime - imageTime[
130         currentImage][0] - lastSprayTime >= 1/con.valveFrequency):
131         sprayBitmap = sm.valveCoverage(images[currentImage][0],
132             intersection, imagePos[currentImage][0], scale, rot[
133             currentImage][0], velocity)
134         bitmapAsInt = generateBitmap(sprayBitmap)
135         weeds = np.delete(weeds, 1, 0)
136         #Generate and send the message
137         if bitmapAsInt != 0:
138             lastSprayTime = currentTime + sprayTime -
139                 imageTime[currentImage][0]
140             command = com.generateCommand(imageNumber,
141                 lastSprayTime, bitmapAsInt, command)
142             print(bitmapAsInt, sprayBitmap, (currentTime +
143                 sprayTime - imageTime[currentImage][0]),
144                 imageNumber)
145             com.sendMsg(command)
146             sprayTime = 0
147     elif((sprayTime < con.minTime) and len(weeds) > 1 and currentTime
148         + sprayTime - imageTime[currentImage][0] - lastSprayTime < 1/
149         con.valveFrequency):

```

```

134         weeds = np.delete(weeds, 1, 0)
135     com.closePort()
136
137 if __name__ == "__main__":
138     main()

```

E.2.4 spraymap.py

```

1 __author__ = 'frodeu'
2
3 import numpy as np
4 import config as con
5 import bitstring as bit
6 import time
7
8 # Calculate the position of a pixel
9 def posFromPixel(imagePos, pixel, scale, rot):
10     pixelDist = pixel/scale
11     pixelPos = imagePos + rot*np.transpose(pixelDist)
12     return pixelPos
13
14 # Calculate the pixle from a given position
15 def pixelfromPos(imagePos, pixelPos, scale, rot):
16     pixelDist = np.linalg.inv(rot)*(pixelPos-imagePos)
17     pixel = pixelDist*scale
18     return (int(pixel[0]), int(pixel[1]))
19
20
21 # Function for reading the spraymap
22 def readSprayMap():
23     map = 1
24
25 # Function for deciding if an area should be sprayed
26 def spray(map):
27     weeds = 0
28     crops = 0
29     total = 0
30     for i in range(len(map)):
31         for j in range(len(map[0])):
32             if(map[i][j] == 1): #np.equal(map[i][j], np.matrix([[1]]))
33                 weeds += 1
34             elif(map[i][j] == 2):# np.equal(map[i][j], np.matrix
35                 (([2])))):
36                 crops += 1
37     total += 1

```

```

37     if(total is not 0 and weeds/total >= con.weed and crops/total <= con.crop
38         ):
39         return 1
40     return 0
41 # Function for finding the rows containing weeds, only one pixle for each row is
    used
42 def findWeedCenters(map, mapPos, scale, rot):
43     start = time.clock()
44     weeds = np.zeros(shape = (len(map), 2))
45     count = 0
46     weedLen = 0
47     lastWeed = False
48     length = len(map)
49     width = len(map[0])
50     for i in range(length):
51         for j in range(width):
52             if(map[width-1-j][length-1-i] == 1 and lastWeed is False)
                : #if (np.equal(map[width-1-j][length-1-i], np.matrix
                    ([[1]])) and lastWeed is False):
53                 weeds[count][0] = width-1-j
54                 weeds[count][1] = length-1-i
55                 lastWeed = True
56                 count += 1
57                 break
58             lastWeed = False
59     print(time.clock()-start)
60     return weeds[weeds[:, 1] != 0]
61
62 # Function for calculating wherer the nozzles will be and if they should spray or
    not
63 def valveCoverage(image, intersection, imagePos, scale, rot, vel):
64     (startX, startY) = pixelfromPos(imagePos, intersection, scale, rot)
65     ArrayLength = int(con.valveLength*scale)
66     valveWidth = int(con.valveDist*scale)
67     valveHeighth = int(np.linalg.norm(vel)/con.valveFrequency*scale)
68     sprayBitmap = bit.BitArray('0x0000000')
69     for i in range(con.numberOfValves):
70         map = image[startX+(valveWidth*i):startX+(valveWidth*(i+1)):1,
                startY:startY+valveHeighth:1]
71         sprayBitmap[i] = spray(map)
72     return sprayBitmap

```

E.2.5 navigation.py

```

1 __author__ = 'frodeu'
2

```

```

3 import math
4
5 import numpy as np
6
7 # Get the yaw difference between image and valve
8 def getYaw(imageYaw, valveYaw):
9     yaw = imageYaw - valveYaw
10    return yaw
11
12 # Get the rotation matrix for yaw to NED frame
13 def getRot(yaw):
14     rot = np.matrix([[math.cos(yaw), -math.sin(yaw)],
15                     [math.sin(yaw), math.cos(yaw)]])
16    return rot
17
18 # Function for finding a point, given the relative distance and rotation
19 def translation(origin, point, rot):
20    return origin + rot*point
21
22 # Function for estimating the position at a given time
23 def estimatePos(pos, vel, time):
24    myPos = pos + vel * time
25    return myPos
26
27 # Function for finding the time which the array are at a given position
28 def findTime(pos1, time1, vel, finalPos):
29    # First find a vector orthogonal to the velocity
30    orth = np.matrix([[0], vel[0]])
31    orth[0] = vel[1]
32    orth[1] = -vel[0]
33    # Calculate the coordinate at which the spray array will be when it is on
34    # a line with the weed
35    if(vel[0] == 0.0):# np.equal(vel[0], np.matrix([[0.0]])):
36        xIntersect = pos1[0]
37        yIntersect = finalPos[1]
38    elif(vel[1] == 0.0):#np.equal(vel[1], np.matrix([[0.0]])):
39        xIntersect = finalPos[0]
40        yIntersect = pos1[1]
41    else:
42        c1 = pos1[0]/vel[0]
43        b1 = pos1[1] - c1 * vel[1]
44        a1 = vel[1]/vel[0]
45        c2 = finalPos[0]/orth[0]
46        b2 = finalPos[1] - c2 * orth[1]
47        a2 = orth[1]/orth[0]
48        xIntersect = (b2 - b1)/(a1 - a2)
49        yIntersect = a1 * xIntersect + b1
50    # Calculate distance and time to reach the point

```

```

50     temp = pos1
51     temp[0] = xIntersect
52     temp[1] = yIntersect
53     distance = np.linalg.norm(temp)
54     speed = np.linalg.norm(vel)
55     return (distance/speed, temp)

```

E.2.6 com.py

```

1  __author__ = 'frodeu'
2
3  import struct
4  import serial
5  import config
6
7  global port
8
9  def calculateChecksum(msg):
10     checksum = 0
11     for i in range(len(msg)-5):
12         checksum += msg[2+i]
13     checksum = checksum%256
14     return checksum
15
16  def generateChecksum(msg):
17     checksum = calculateChecksum(msg)
18     return checksum
19
20  def validateChecksum(msg):
21     checksum = calculateChecksum(msg)
22     if(checksum is msg[2 + msg[1]]):
23         return True
24     return False
25
26  def generateInit(address, shutterTime, purge, pcbNum, pcbDir, msg):
27     msg[0] = 0x69
28     msg[1] = 0x08
29     msg[2] = address
30     msg[3] = pcbNum
31     msg[4] = pcbDir
32     msg[5], msg[6], msg[7], msg[8] = struct.pack('i',shutterTime)
33     if(purge is True):
34         msg[9] = 0x01
35     msg[10] = generateChecksum(msg)
36     msg[11] = 0x65
37     msg[12] = 0x0A
38     return msg

```



```

39
40 # Generate a command from imageNr, timedelay in s(convert to us) an bitmap
41 def generateCommand(imageNr, timeDelay, bitmap, msg):
42     msg[0] = 0x73
43     msg[1] = 0x0D
44     msg[2] = 0x41
45     msg[3], msg[4], msg[5], msg[6] = struct.pack('i',imageNr)
46     msg[7], msg[8], msg[9], msg[10] = struct.pack('i',(timeDelay*1000000))
47     msg[11], msg[12], msg[13], msg[14] = struct.pack('I',bitmap)
48     msg[15] = generateChecksum(msg)
49     msg[16] = 0x65
50     msg[17] = 0x0A
51     return msg
52
53 def sendMsg(msg):
54     port.write(msg)
55     return True
56
57 def openPort():
58     global port
59     port = serial.Serial(config.comPort, config.baudrate, timeout=1)
60
61 def closePort():
62     port.close()
63
64 def readMsg():
65     msg = 1
66     return True

```

E.2.7 valveMapper_tests.py

```

1 from nose.tools import *
2 from valveMapper import valveMapper
3
4 def setup():
5     print "SETUP!"
6
7 def teardown():
8     print "TEAR DOWN!"
9
10 def test_basic():
11     print "I RAN!"

```

Bibliography

- Buck, R. (2013). http://digital.mdtmag.com/medicaldesigntechnology/november_december_2013#pg8. Accessed: 2010-03-06.
- Castrejón-Pita, J., Martin, G., Hoath, S., and Hutchings, I. (2008). A simple large-scale droplet generator for studies of inkjet printing. *Review of Scientific Instruments*, 79(7):075108.
- Christensen, S., Søgaaard, H. T., Kudsk, P., Nørremark, M., Lund, I., Nadimi, E. S., and Jørgensen, R. (2009). Site-specific weed control technologies. *Weed Research*, 49(3):233–241.
- Dong, H., Carr, W. W., and Morris, J. F. (2006). An experimental study of drop-on-demand drop formation. *Physics of Fluids (1994-present)*, 18(7):072102.
- Eggers, J. and Villermaux, E. (2008). Physics of liquid jets. *Reports on progress in physics*, 71(3):036601.
- Hoath, S. D., Jung, S., and Hutchings, I. M. (2013). A simple criterion for filament break-up in drop-on-demand inkjet printing. *Physics of Fluids (1994-present)*, 25(2):–.
- Jung, S. and Hutchings, I. M. (2012). The impact and spreading of a small liquid drop on a non-porous substrate over an extended time scale. *Soft Matter*, 8(9):2686–2696.
- Keller, J. B. (1983). Breaking of liquid films and threads. *Physics of Fluids (1958-1988)*, 26(12):3451–3453.
- Klungerbo, A. T. (2013). Drop-on-demand i presisjonsjordbruk. Master's thesis, NTNU.
- Lee, W., Slaughter, D., and Giles, D. (1999). Robotic weed control system for tomatoes. *Precision Agriculture*, 1(1):95–113.
- Midtby, H., K. Mathiassen, S., Andersson, K., and Jørgensen, R. (2011). Performance evaluation of a crop / weed discriminating microsprayer. *Computers and Electronics in Agriculture*, 77(1):35–40.

- Nieuwenhuizen, A., Hofstee, J., and van Henten, E. (2010). Performance evaluation of an automated detection and control system for volunteer potatoes in sugar beet fields. *Biosystems Engineering*, 107(1):46 – 53.
- Nieuwenhuizen, A. T. et al. (2009). *Automated detection and control of volunteer potato plants*. PhD thesis, Wageningen University.
- Rioboo, R., Marengo, M., and Tropea, C. (2002). Time evolution of liquid drop impact onto solid, dry surfaces. *Experiments in Fluids*, 33(1):112–124.
- Søgaard, H. T., Lund, I., and Graglia, E. (2006). Real-time application of herbicides in seed lines by computer vision and micro-spray system. In *American Society of Agricultural and Biological Engineers, ASABE Annual International Meeting*.
- The Lee Company (2014). Engineering data. <http://www.theleeco.com/engineering/engineering.cfm?topics=1>. Accessed: 2010-03-03.
- Urdal, F. (2013). Design of a precision spray matrix: Valve matrix control for microspraying in precision agriculture. Project thesis, NTNU.
- Utstumo, T. (2011). Attitude estimation in agricultural robotics: Design and implementation. Master's thesis, NTNU.
- Utstumo, T. and Gravdahl, J. T. (2013). Implementation and comparison of attitude estimation methods for agricultural robotics. In *Agricontrol*, volume 4, pages 52–57.
- Vadillo, D., Tuladhar, T., Mulji, A., Jung, S., Hoath, S., and Mackley, M. (2010). Evaluation of the inkjet fluids performance using the cambridge trimaster filament stretch and breakup device. *Journal of Rheology (1978-present)*, 54(2):261–282.
- Watton, J. (1989). *FLUID POWER SYSTEMS: Modeling, simulation, analog and microcomputer control*. Prentice Hall International (UK) Ltd.
- Wierzba, A. (1990). Deformation and breakup of liquid drops in a gas stream at nearly critical weber numbers. *Experiments in Fluids*, 9(1-2):59–64.