

**AUTOMATIC HAY FEEDER FOR HORSES:
DESIGN AND MAKE A PROTOTYPE**



Bachelor's thesis

Automation Engineering

Valkeakoski, spring 2018

Quang Hung Trinh

Automation Engineering
Valkeakoski

Author	Quang Hung Trinh	Year 2018
Subject	Automatic hay feeder for horses: design and make a prototype	
Supervisor(s)	Juha Sarkula	

ABSTRACT

The objective of this thesis project was to design and build an automatic horse feeder which would be used at a family's farm. An additional objective of this project was to use Arduino platform to control the machine. A micro-controlled horse feeder is extremely useful in providing assistance to people who are usually busy at the feeding time to have their horses fed steadily and on time while keeping the horses healthy and safe. The thesis project was inspired by Mr. Markku Kippola and carried out under his supervision. This was also a major advantage for the author to gain more experiences about mechanical design and microcontrollers.

The main purpose of this design was the automation of a horse feeding device with an accurate and precise time set for the feeds to be released. The feeder was designed so that it was suitable for hay. It came with an implemented LED indicating when the container was empty, with an LCD to indicate steps of the process, and smart switches to control the solenoids of the gates. Furthermore, the mechanism of the prototype played an important role in order to achieve effectiveness and simplicity for the project.

The final prototype was completed and tested which met the arrangements between the commissioner and the author. Moreover, the outcome of this thesis project is that the design could be extended to implement at individual stalls. All cabinets are operated by setting the real-time and by assigning the desired time for the horses to be fed at the same time.

Keywords Arduino, hay feeder, pull-type solenoid

Pages 56 pages including appendices 6 pages

CONTENTS

1	INTRODUCTION	1
1.1	Introduction.....	1
1.2	Problem statement	1
1.3	Objectives.....	1
1.4	Scope of the project	2
1.5	Thesis roadmap	2
2	THEORETICAL BACKGROUND TO PROJECT.....	3
2.1	Product design and development	3
2.1.1	Characteristics of a successful product	3
2.1.2	Product development process flow	4
2.1.3	Challenges of designing a product	5
2.1.4	Concept development: The front-end process	6
2.2	Embedded system	8
2.2.1	History of microcontroller	8
2.2.2	RASBERRY PI 3	9
2.2.3	ARDUINO	11
2.2.4	PROFET™ – Smart protected high-side switches.....	14
2.3	Linear solenoid actuator	18
2.3.1	General actuators	18
2.3.2	Overview.....	19
2.3.3	Working principle	20
2.3.4	Solenoid duty cycle and voltage.....	21
2.3.5	Applications	21
2.4	Inter-integrated circuit (I ² C)	23
2.4.1	Introduction.....	23
2.4.2	Brief of history	23
2.4.3	Protocol	24
2.5	Real time clock (RTC).....	25
2.6	Existing hay feeder on current market	27
2.6.1	Heinätin Single (Hayer).....	27
2.6.2	DIY hay feeder for horses	28
3	PREREQUISITE PLANNING OF THE PROJECT.....	29
3.1	Characteristic of horses.....	29
3.2	Existing hay feeder	31
3.3	Inputs of the project.....	32
3.3.1	Identify customer needs.....	32
3.3.2	Target specifications.....	33
4	CONCEPT DESIGN	33
4.1	Suggested design version	35
4.2	Concept design version 1	35

4.3	Concept design version 2	37
4.4	Concept design version 3	38
4.5	Concept design version 4	39
5	FINAL PRODUCT.....	40
5.1	Product overview	42
5.2	Technical data	42
5.3	Arduino breadboard.....	45
5.4	Circuit architecture.....	45
5.5	Development of code.....	46
5.6	Advantages & features.....	52
5.6.1	Advantages	52
5.6.2	Features	52
6	LIMITATION AND EXPANSION POSSIBILITIES	53
7	CONCLUSION	54
	REFERENCES.....	55

Appendices

Appendix 1 ARUINO CODE

ACKNOWLEDGMENTS

Here, I would like to take the opportunity to express my deepest gratitude to people whom has played an important role in this project. Without them, this project would be a failure.

The first person I would like to thank is my parents. Their constant support throughout my whole life helps bring me here to the place I am today. Their moral and financial support helps me to complete this project. My older and my girlfriend were always side by side with me during the hardest time of project.

Secondly, I would like to personally thank my thesis commissioner Mr Markku Kippola, who gave an amazing thesis project work and lead me initial steps to let the project went well. Besides that, I would like to express my gratitude to my supervisor, Mr Juha Sarkula, and Mr Jan-Peter Nowak, who directly guided me and also gave me vast of useful advice. To Ms Niina Valtaranta, who gave lots of guidance for my thesis writings.

Thirdly, I would like to also express my gratitude to Mr Teppo Syrjäaho, mechanical design professor at Riihimäki HAMK campus, who really gave best advices in mechanical lessons, most important part in my thesis project.

Finally, I would like to thank all my friends and classmates whom lend me a helping hand and gave me advice, opinion and also criticism to complete this project. With their help, I can say that this project is a success.

Hung Trinh
Valkeakoski, 04.05.2018

1 INTRODUCTION

1.1 Introduction

An automatic horse feeder is a type of machine that allows the users to set the preferred time to activate the machine to automatically feed the horse at the specific time set. The goal of building the machine is reached by various methods for the users to set the preferred time such as using an external timer, a timing gate and many others.

The object of this thesis project was to design and build a prototype of an automated horse feeder for a small group of horses (two or three) which are taken care by a family. The basic concept of the machine is to feed the horse at a specific time per day such as early morning, mid-day and late night. For example, three hours per feeding, four feedings per cycle and 2 -3 kg per feeding. Furthermore, an additional feature of this feeding system can be to serve a large group of horses so that, the system comes in various sizes.

1.2 Problem statement

Mr Markku Kippola, the commissioner of the thesis, planned to build a feeding automation system for his own needs. His aim was to feed horses on a fixed schedule because the feeding time is highly labour intensive. Besides that, it is important to determine the variety and amount of food fed to the horses. This is because correct and balanced nutrition is a critical component of proper horse care. This means that the farmer has to wake up early in the morning and stay up late at night just to make sure that the feeding time is met.

1.3 Objectives

There were several objectives that were to be achieved in this project:

- To be creative and able to design an intelligent system of an automatic horse feeder using a microcontroller.
- To learn about product development, how to select the right objectives for the project.
- To learn about the mechanism of the actuator, mechanical building for the system's working principle.
- To learn about the art of programming in C++/C language.
- To combine all hardware skills, electronic knowledge with some software development in realising this project.

1.4 Scope of the project

The components for the machine can be divided into two types: automation parts and mechanical parts. The automation block components included: a push buttons, a monitor LCD, smart switch, and an Arduino microcontroller. Secondly, the mechanical part consists of linear solenoid and the mechanical structure. The main feature of this project is to automatically feed the horse at a certain time range which is set by the user. In addition, the mechanical part of the system should be carefully made to ensure that the amount of feed for each feeding can be set to prevent any waste. Furthermore, the machine can feed manually by the push button for horses when they were outside.

1.5 Thesis roadmap

This thesis consists of five main chapters as follows:

Chapter two, Theoretical background, presents the literature aspects for the problem which contain the embedded system, product design and development, the solenoid actuator, Inter-integrated circuit, and a real-time clock.

Chapter three, Scope of the project, describes related factors as well as the inputs and outputs of the project. The chapter also discusses the requirements of the commissioner and the author's targets.

Chapter four, Concept design, presents a set of concept ideas for the project which were issued by the author.

Chapter five, Final product, shows the latest product as designed by the author. The details of the product such as the materials, hardware, code, and features are also presented.

Chapter six, Limitations and expansions, discusses the current limitations and shows the future improvements of the product.

2 THEORETICAL BACKGROUND TO PROJECT

2.1 Product design and development

The mechanical design played an important role in this thesis project, so that the author mentioned the literature of product design and development in the following section. This section presents characteristic of successful product, a generic product development process flow, the context of designing product and activities of concept development.

In fact, the author had to learn about mechanical design beforehand. Because the mechanical theory is used as seeds for the creation of development methods, uniquely suited to the personality of designers, their talents. Practise without theory can easily result in thwarting and fails to cultivate the knowledge that successful product development professionals and projects have accumulated overtime. On the other hand, guidance without practise is unproductive because there are many nuances, exceptions, and subtleties to be learnt in practical settings and because some necessary tasks simply lack sufficient theatrical underpinning (T.Ulrich, Karl; D.Eppinger, Steven, 2016).

2.1.1 Characteristics of a successful product

Four specific dimensions below, all of which ultimately relate to profit, are commonly used to assess the performance of a product development effort. High performance, along these four dimensions, should ultimately cause economic success; however, there are many other performance criteria also important.

- **Demonstrable:** The products will be commercialized shapely and they should have some kind of feature or aspect that makes people admire and take note. When the products meet the desired requirements and the better visual demonstration, they will have more chances for success.
- **Practical and problem solving:** The product can be well explained to the customer the problem, solution and benefit. The product might increase the chances of success when they have more and better common problem solving.
- **Easy to explain how it works:** Easy-to-understand introduction of how the product works is an important way to please and seize the consumer.
- **Product pricing:** The price should be equal the value of the product, its function, manufacture cost and time. For example, same type of product, but lower cost in manufacture and less development time will be the better choice.

(Jeske, 2015)

2.1.2 Product development process flow

A common product development process consists of five phases, as presented in Figure 1. Beginning with planning phase, which is related to research activities about the subject. The output of the planning phase is the statement of project's mission, which will be the requirement to start the concept development phase. The outcome of the product development process is the debut of the product to the consumer which can please the consumer's need (T.Ulrich, Karl; D.Eppinger, Steven, 2016).

The spiral product development process is divided into five phases as following:

1. **Planning:** The planning stage is referred to as initial phase because it overreaches the project approval and launch of the actual product development process. Opportunity identification directed by corporate strategy is the first element of this phase, and this phase also comprises assessment of technology developments and market objectives. The output of the planning phase is mission statement and this phase is presented in chapter three.

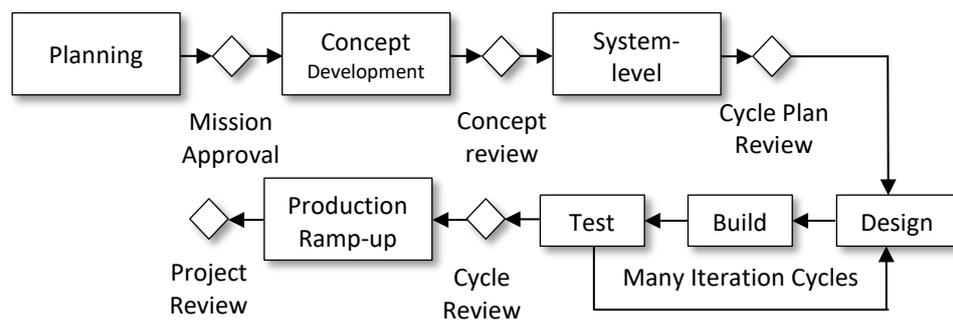


Figure 1. Spiral Product Development Process (T.Ulrich, Karl; D.Eppinger, Steven, 2016)

2. **Concept development:** In this phase, the needs of the products are identified, alternative product concepts are generated and evaluated, and one or more concepts are picked for further testing. A concept is a description of the form, function, economic justification and features of a product. In chapter four, four concept designs are presented.
3. **System-level Design:** This phase is known as product architecture, decomposition of the product into subsystems and components, preliminary design of key components, and allocation of detail design responsibility to both internal and external resources. The purpose of the product architecture is to describe the basic physical building blocks of the product in terms of what they do and what their interfaces are to the other devices. Architectural decisions allocate the detailed design and examination of these building blocks to be assigned, such that development of different portions of the product can be performed concurrently.

4. **Many iteration cycles:** The product development process normally follows of activity and information flow. Rapid-build products allow a spiral product development process whereby detail design, prototyping, and test activities are reiterated many times.
5. **Production ramp-up:** In the production ramp-up phase, the product is made using the planned production system. The reason the ramp-up is to work out any residual problems in the production process. The review which may occur shortly after the debut comprises an assessment of the project from both commercial and technical perspectives and is aimed to identify ways to advance the development process for future projects.

(T.Ulrich, Karl; D.Eppinger, Steven, 2016)

2.1.3 Challenges of designing a product

Creating and developing great products are complicated and hard. There are some of the characteristics that make product development challenging:

- **Trade-off:** One of the most difficult aspects of product development is recognizing, understanding, and managing such trade-offs in a way that maximizes the success of the product.
- **Dynamics:** Due to the development of technologies, customer preferences evolve, competitors introduce new products, the macroeconomic environment.
- **Time pressure:** The product development decisions must usually be made quickly and without complete information.
- **Economics:** Developing, producing, and marketing a new product requires a large investment. To find the reasonable pay-back on this investment, the resulting product must be both appealing to customers and relatively inexpensive to produce.
- **Creation:** The product development process starts with an idea and finishes with the production of a physical artefact. When viewed both in its entirety and at the level of individual activities, the product development process is intensely creative.
- **Satisfaction of societal and individual needs:** All kind of products which debuted are aimed at satisfying needs of consumers. Individuals interested in developing new products can almost always find institutional settings in which they can develop products satisfying what they consider to be important needs.
- **Sustainability:** Some designers have killer design ideas, but unfortunately those ideas are not sustainable either on an economic or environmental level. The product may have an amazing design, but it costs too large to produce in huge quantities. By taking this into account, the designers can ensure that the product design can be resumed far into future market.

(T.Ulrich, Karl; D.Eppinger, Steven, 2016)

2.1.4 Concept development: The front-end process

The concept development stage of the development process demands perhaps more coordination among functions than any other. In this section, the concept development is enlarged into what called front-end process. The front-end process basically consists many interdependent activities, ordered roughly as presented in Figure 2.

In practice, the front-end activities may be overlapped in time and iteration is often necessary. The uncertain nature of progress in product development is revealed by the dashed arrows in Figure 2. At almost any stage, new information may become available or results learned that can cause the team to step back to repeat an earlier activity before proceeding. This repetition of nominally complete activities is known as development iteration (T.Ulrich, Karl; D.Eppinger, Steven, 2016).

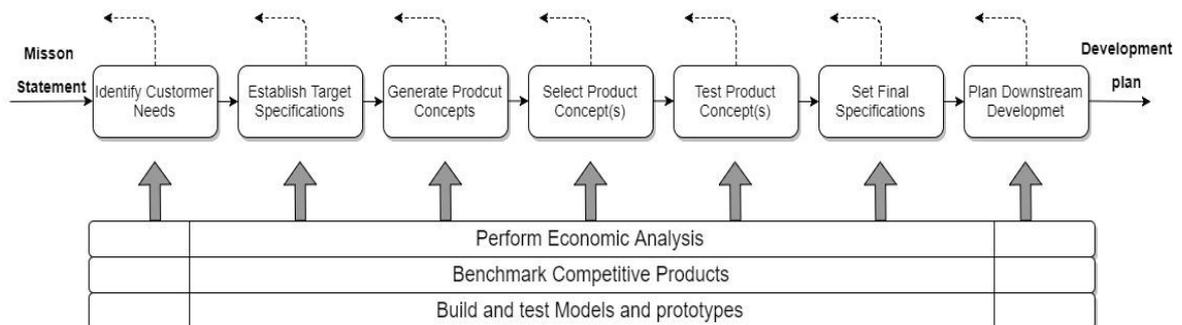


Figure 2. The many front-end activities comprising the concept development phase. (T.Ulrich, Karl; D.Eppinger, Steven, 2016)

The front-end process includes the following actions:

- **Identifying customer needs:** The goal of this activity is to understand customers' needs. The output of this step is a set of carefully constructed customer need statements, organized in a hierarchical list, with importance weightings for many or all the needs. A method for this activity is presented in Chapter 3.
- **Establishing target specifications:** Specifications provide a precise description of what a product has to do. They are the translation of the customer needs into technical terms. A method for the specification activity is given in Chapter 3.
- **Concept generation:** The goal of concept generation is to thoroughly explore the space of product concepts that may address the customer needs. Concept generation includes a mix of external search, creative problem solving within the team, and systematic exploration of the various solution fragments the team generates. The result of this activity is usually a set of concepts, each typically represented by a sketch and brief descriptive text.
- **Concept selection:** Concept selection is the activity in which various product concepts are analysed and sequentially eliminated to identify the most promising concept(s). The process usually requires several iterations and may initiate additional

concept generation and refinement. A method for this activity is described in Chapter 5.

- **Concept testing:** One or more concepts are then tested to verify that the customer needs have been met, assess the market potential of the product, and identify any shortcomings that must be remedied during further development. If the customer response is poor, the development project may be terminated or some earlier activities may be repeated as necessary.
- **Setting final specifications:** The target specifications set earlier in the process are revisited after a concept has been selected and tested. At this point, the team must commit to specific values of the metrics reflecting the constraints inherent in the product concept, limitations identified through technical modelling, and trade-offs between cost and performance.
- **Project planning:** In this final activity of concept development, the team creates a detailed development schedule, devises a strategy to minimize development time, and identifies the resources required to complete the project.
- **Economic analysis:** The team, often with the support of a financial analyst, builds an economic model for the new product. This model is used to justify continuation of the overall development program and to resolve specific trade-offs between, for example, development costs and manufacturing costs. Economic analysis is shown as one of the ongoing activities in the concept development phase. An early economic analysis will almost always be performed before the project even begins, and this analysis is updated as more information becomes available.
- **Benchmarking of competitive products:** An understanding of competitive products is critical to successful positioning of a new product and can provide a rich source of ideas for the product and production process design. Competitive benchmarking is performed in support of many of the front-end activities.
- **Modelling and prototyping:** Every stage of the concept development process involves various forms of models and prototypes. These may include, among others: early “proof of-concept” models, which help the development team to demonstrate feasibility; “form only” models, which can be shown to customers to evaluate ergonomics and style; spreadsheet models of technical trade-offs; and experimental test models, which can be used to set design parameters for robust performance.

(T.Ulrich, Karl; D.Eppinger, Steven, 2016)

2.2 Embedded system

2.2.1 History of microcontroller

A microcontroller is a small computer on a single integrated circuit. Figure 3 shows that a microcontroller consists of one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Most microcontrollers used these days are embedded in other machinery such as telephones, robots, automobiles, and segments of computer systems. Microcontrollers are assigned to one task and run one specific program. In addition, the program is saved in read-only memory (ROM) and generally does not change (Brain, 2000).

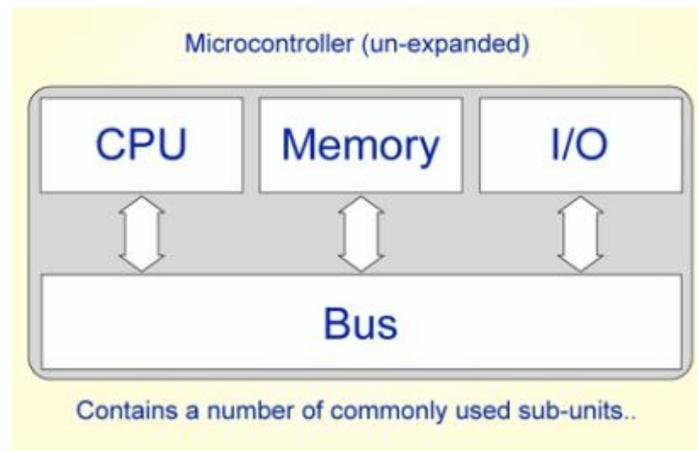


Figure 3. Main components of a microcontroller (Brain, 2000)

The timeline below illustrates the development history of microcontroller:

- The first microcontroller which was debuted in 1971 was the 4-bit Intel 4004, and in the next several years, Intel 8048, 8051 (Figure 4) and other more capable microprocessors became available.



Figure 4. Intel microcontrollers: Intel 8048 and 8051 (Maru, 2016)

- The TMS 1000 which created by TI engineers Gary Boone and Michael Cochran became commercially available in 1974. It consists read-only memory, read/write memory, processor and clock on one chip and was targeted at embedded system.
- The 8051 microcontrollers were introduced in 1980 and is one of the most popular microcontroller that days.
- In 1997, based on the existence of the single-chip TMS 1000, Intel developed a computer system on a chip which called Intel 8048 optimized for control applications. It combines RAM and ROM on the same chip.

- In 1993, Atmel introduced EEPROM memory which allowed microcontrollers to electrically erased quickly without an expensive package as required for EPROM, allowing both rapid prototyping, and in-system programming.
- In 1993, Atmel introduced the first microcontroller (Figure 5) using Flash memory, a special type of EEPROM.



Figure 5. Atmel microcontrollers (Maru, 2016)

- As times have changed, requirements have increased and the size of the controllers (devices that control processes) has decreased. Microcontrollers like AVR (Alf (Eigil Bogen) & Vegard (Wollan)'s Risc Processor), and PIC (Peripheral Interface Controller) have become smaller and sleeker yet more and more powerful. For example, in current market, there are so tiny microcontrollers available, small and cheap enough to be used in simple products like kid toys, electric toothbrushes, and smartphones. Moreover, microcontrollers become available widely for studying, researching and working on project, with large online communities around certain processors.

(Brain, 2000)

2.2.2 RASBERRY PI 3

Controlling the system by microprocessor Raspberry PI (the Pi) is an initial suggestion from commissioner. After the research about the Pi, the author has gathered some advantages and possibilities of the Pi when applied to the prototype.

Introduction

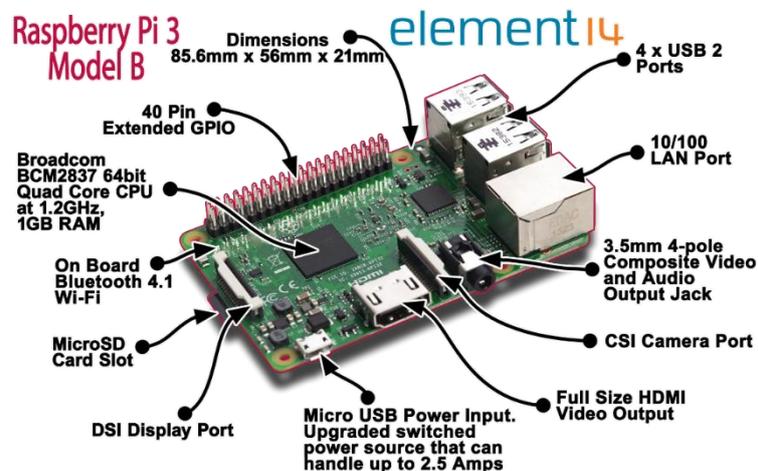


Figure 6. Raspberry Pi 3 model B (Joeman, Kellyhensen, 2016)

The Raspberry Pi is a credit-card-sized single board computer that connect with at least a combination of a screen, a keyboard, a mouse and a power supply. It is equipped a Quad-Core 64-bit CPU, WiFi & Bluetooth. Furthermore, the Pi has four USB ports to connect other devices, Ethernet socket to connect data transferring devices, and other specs

In practical, it is a capable little computer which can be used in electronics projects and for many of things that desktop PC does, like coding, word processing, browsing the internet, and playing games. With a reasonable price under 35 euros, a Raspberry Pi 3 can be purchased online.

Get started

Once all cables are plugged into the Pi, it can run a full Linux based operating system and has hardware support for SPI, I²C and Serial. The OS, which is constantly being improved, recently had a graphical overhaul, and includes an optimized web browser, an office suite, programming tools, educational games, and other software (HeathNick, 2017).

The Pi can operate as a budget PC, however it will lag loading heavier websites and, when browsing these demanding sites, having tabs open at once runs the risk of overloading the Pi's memory that causing a lengthy freeze (Heath, 2017).

Features

Until the end of 2017, raspberry Pi 3 model B was the latest version (Figure 6). Currently, Pi 3 model B+ is the newest version which has much more modifications but that version is not used widely these days. The features of model B are listed below:

- Now 10x Faster - Broadcom BCM2387 ARM Cortex-A53 Quad Core Processor powered Single Board Computer running at 1.2GHz!
- 1GB RAM so you can now run bigger and more powerful applications
- Fully HAT compatible
- 40pin extended GPIO to enhance your "real world" projects.
- Connect a Raspberry Pi camera and touch screen display (each sold separately)
- Stream and watch Hi-definition video output at 1080
- Micro SD slot for storing information and loading your operating systems.
- 10/100 BaseT Ethernet socket to quickly connect the Raspberry Pi to the Internet

(HeathNick, 2017)

Advantages

- It is easy to connect to internet, connect with wireless devices.
- Can be programmed using variety of programming languages such as Python, C++/C, ...

- Driving a more complicated robot, performing multiple tasks, doing intense calculations.

Limitations

- Complicated to use than another microcontroller such as Arduino.
- Approaching hardware is not real-time. If the CPU is busy, then interfacing with hardware can be delayed.
- Not open source and taking times to learn and use well with the Pi.

2.2.3 ARDUINO

Arduino platform is a secondary option which selected from the author. Because of its usage and beneficial which listed in this section, Arduino is chosen to be a controller for this project, a prototype. The problems solving and simple programming system are two main purposes when using this platform although there are some limitations

Introduction

In a modern world, embedded systems are everywhere — in cars, buildings, factories and roads. There are numerous fascinating applications and interesting companies in the market. The global embedded systems market is worth over 200 billion USD, and rapidly growing.

(Viitala, 2018)

Arduino is an open-source microcontroller used for building electronics projects which can be easily programmed, debugging and reprogrammed at any instant of time. Arduino contains of both physical programmable circuit board (often referred to as a microcontroller) and a piece of software for developing the code known as the Arduino IDE (Integrated Development Environment) that runs on user's computer, used to write and upload computer code to physical board through USB connection. Built up with the 8-bit Atmel AVR microcontroller's that are manufactured by Atmel or a 32-bit Atmel ARM, these microcontrollers can be programmed easily using the C or C++ language in the Arduino IDE (b_e_n, 2012).

Moreover, Arduino has I²C bus which reducing the components' wires into two (SDA – data and SCL – clock). As there are literally thousands of components that use the I²C interface. And Arduino boards can control them all. Currently, many applications use I²C bus connection, such as real-time clocks, digital potentiometers, memory chips, FM radio circuits, I/O expanders, LCD controllers, amplifiers, temperature sensors, and so on. The maximum number of I²C devices can operate at one time is 112 (Boxall, 2014).

Since it reached a wider community, the Arduino board started changing to get used to new needs and challenges, it offers a vast range of

applications from 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

Arduino type

In the market these days, there are several types of Arduino, and these types presented in Table 1.

Table 1. Arduino types (b_e_n, 2012)

Arduino type	Microcontroller	Clock speed
Arduino Uno	ATmega328	16 MHz with auto-reset
Arduino Duemilanove / ATmega328	ATmega328	16 MHz with auto-reset
Arduino Nano	ATmega328	16 MHz with auto-reset
Arduino Mega 2560 or Mega ADK	ATmega2560	16 MHz with auto-reset
Arduino Leonardo	ATmega32u4	16 MHz with auto-reset
Arduino Mini w/ ATmega328	ATmega328	16 MHz with auto-reset
Arduino Ethernet	Equivalent to Arduino UNO with an Ethernet shield	
Arduino Fio	ATmega328	8 MHz with auto-reset
Arduino BT w/ ATmega328	ATmega328	16 MHz with auto-reset
Lily Pad Arduino w/ ATmega328	ATmega328	8 MHz (3.3V) with auto-reset
Arduino Pro or Pro Mini	ATmega328	16 MHz with auto-reset
Arduino NG	ATmega8	16 MHz with auto-reset

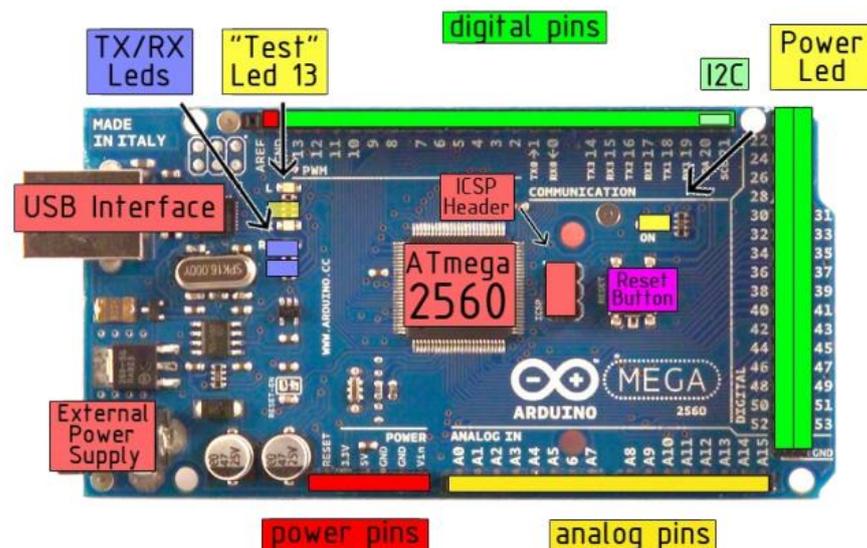


Figure 7. Arduino MEGA 2560 (b_e_n, 2012)

Arduino MEGA board (Figure 7) was chosen for this project cause of its huge extension. The Arduino Mega is like a big brother of Arduino Uno. It owns 54 of digital input/output pins which consist of 14 pins as Pulse-width Modulation (PWM) outputs, 16 Analog inputs, a USB connection, a power jack, and a reset button. With this board, it contains everything needed to operate a microcontroller; just a simply connect the board to a computer through a USB cable or with a Ac to DC adapter. With the vast of pins makes this board very handy for complicated projects that require a bunch of digitals inputs or outputs (such as lots of LEDs or buttons or sensors).

Advantages

Arduino also shortens the process of working with microcontroller, but it proposals some advantages for users over other systems:

- Inexpensive hardware: Since Arduino is an open source platform the software is not purchased and only the cost of buying the board or its parts is incurred, thus making it very inexpensive.
- Multi -platform environment: The Arduino programming software IDE can run on many platforms including Windows, Linux and Mac OSX. This makes the user community even larger.
- Simple and clear programming language: The Arduino Software (IDE) is friendly and easy-to-use for beginners, adaptable enough for processed users to take advantages as well.
- Growth of Arduino: Arduino was growing with intent to provide an economical and trouble-free way for users to build devices that interact with their situation using sensors and actuators. Because of the development, this makes it perfect for new users to get started quickly.
- Open source and extensible software: the Arduino software is published as open source tools, the extensive can be provided by advanced programmers. Additionally, the Arduino IDE uses a simplified version of C++ which makes it easier to learn to program.

(Louis, 2016)

- Can run one program at a time (real time), over and over again.

Limitations

- Not very powerful when compared with Raspberry Pi (Microcontroller vs Microprocessor). Cannot run a lot of heavy algorithms, or interface with a touchscreen, and without external shields. Clearly, Arduino cannot run OpenCV.
- It is difficult to connect to internet (the user should have internet shields and libraries, but is not straight forward).
- Short range of temperature in use
- Memory limitations (extremely low when compared to the Pi)

2.2.4 PROFET™ – Smart protected high-side switches

Other option

To drive actuators, at first, the author used simple relay motor driver such as ULN2803 which is indicated in Figure 8.

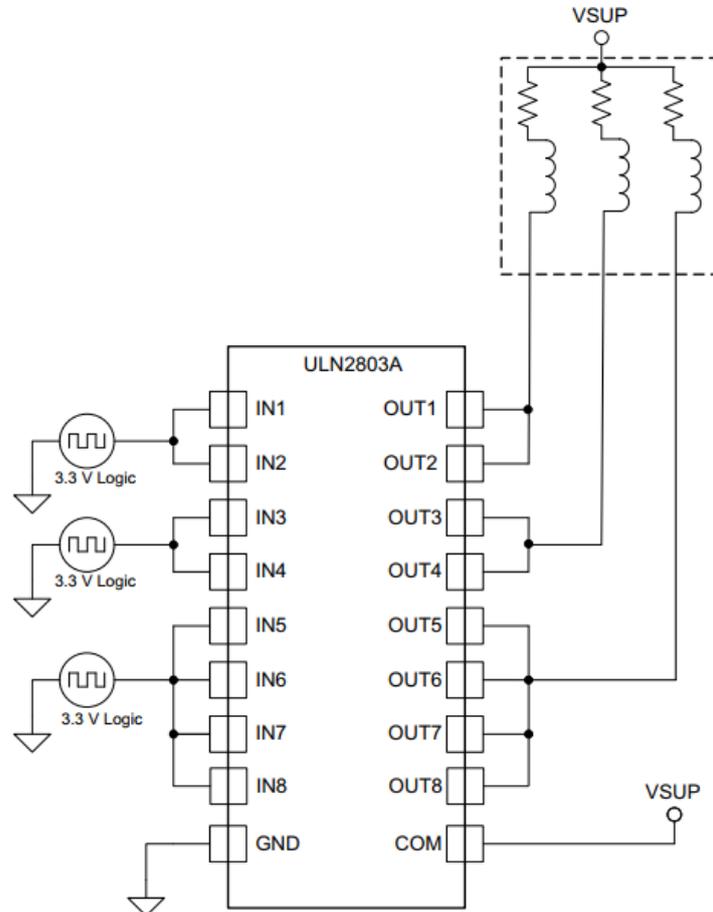


Figure 8. ULN2803A as Inductive load driver

ULN2803A device functions as a relay driver which consists of eight NPN Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads (ULN2803A Darlington Transistor Arrays, 2017). The output current is 300 mA per channel with 500 mA is a peak collector current.

Because the prototype needed high current solenoids (such as higher than 1A) to run so that the high current motor driver required. Unfortunately, ULN2803A model is not suitable because of its short range of output current. The author decided to use PROFET instead of relay driver and its features are presented in following section.

PROFET overview

PROFET intelligent power switches which consist of Double-Diffused MOS (DMOS) power transistor and Complementary metal-oxide-semiconductor (CMOS) logic circuitry for complete built-in protection offers protection against overload, overvoltage, short-circuit, loss of ground, power supply loss. The vast range of PROFET brings protection

and diagnosis features to drive high current loads in automotive (12V, 24V), industrial application and commercial, construction & agricultural vehicles (CAV).

The PROFET diagnostics make the option between status or current sense features, or a combination of both. Moreover, the status feature highlights the diagnosis of overtemperature or open-load. Diagnostic features also provide the user with precise data of switch and load. Knocking out the need for further discrete circuitry and assembly causes reducing risks for diagnostic feedback and load current sensing.

There are some common features of all PROFET models:

- Overload protection
- Current limitation
- Short circuit protection
- Thermal shutdown
- Overvoltage protection (including load dump)
- Fast demagnetization of inductive loads
- Under-voltage and overvoltage shutdown with auto-restart and hysteresis.
- Open drain diagnostic output
- Open load detection in ON-state
- CMOS compatible input
- Loss of ground and loss of V_{bb} protection
- Electrostatic discharge (ESD) protection
- Green Product (RoHS compliant)
- AEC Qualified

(Infineon Technologies AG, 2013)



Figure 9. Building intelligence into Automotive switches (Protected and Intelligent Power Switches, 2016)

With the evolution of automotive applications, the performance required of switching devices is also changing as can be seen in Figure 9. Replacing from the traditional automotive switches - relay drivers to intelligent power switch. With this evolution, it helps to resolve various issue such

as low value resistance, longevity, noise, high temperature, size (Protected and Intelligent Power Switches, 2016).

PROFET model: BTS410E2

Intelligent high side switches BTS410E2 are designed to control a wide variety of loads in automotive and industrial systems, especially perfect for automotive load switching applications. These switches are intended to protect loads from transients by isolating the load from the transient energy rather than absorbing it. The layout of this model is presented in Figure 10. This model is compact in size and has high reliability. The size is completely reduced to an equivalent mechanical relay product.

Application examples:

- Lighting
- Heating
- Power distribution
- Motor control

Load:

- Capacitive, such as lamps and glow plugs
- Resistive, such as seat heating
- Inductive, such as solenoids
- Electronic, such as ECU

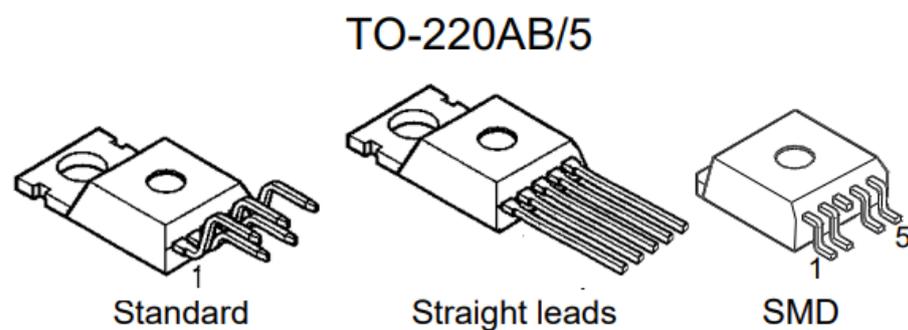


Figure 10. BTS410E2 model (Smart High-Side Power Switch, 2013)

Table 2. BTS410E2 pins description (Infineon Technologies AG, 2013)

Pin	Symbol		Function
1	GND	-	Logic ground
2	IN		Input, activates the power switch ion case of logical high signal
3	V _{bb}	+	Positive power supply voltage, the tab is shorted to this pin
4	ST	S	Diagnostic feedback, low on failure
5	OUT (Load, L)	O	Output to the load

The pin details information is listed in Table 2. The pin number from 1 to 5 are presented from left to right of the module as can be seen in Figure 5. Pin 1 (GND) relates to all GND from Arduino, power source and actuator. Pin 2(IN) of switch will connect digital pin in Arduino board to control the motor statement. Pin 3(V_{bb}) is attached to positive of power source. When the output of motor needed to be measured, and diagnosed feedback, then Pin 4(ST) will be used. Finally, the positive of motor is connected to Pin 5(load out) (Infineon Technologies AG, 2013).

With N-channel vertical power FET with charge pump, ground referenced CMOS compatible input and diagnostic feedback, monolithically integrated in Smart SiPMOS[®] technology. This product is fully protected by embedded protection functions. Based on the product summary in Table 3, this can be illustrated in these applications:

- μ C compatible power switch with diagnostic feedback for 12V and 24V DC grounded loads.
- All types of resistive, inductive and capacitive loads.
- Replaces electromechanical relays, fuses and discrete circuits.

(Infineon Technologies AG, 2013)

Table 3. Product summary

Overvoltage protection V_{bb} (AZ)	Operating voltage V_{bb} (on)	Load current I_L (ISO)	Current limitation I_L (SCr)	On-state resistance R_{ON}
5V	4.7 ... 42V	5A	1.8A	220 mOhm

According to the block diagram in Figure 11 of model BTS410E2, we can assume the following features:

- Basic features:
 - Using an Intelligent power device (IPD) with protection/self-diagnostic functions built in.
 - The overcurrent protection function discontinues current to the load when overcurrent detection value is reached. Moreover, they also protect related components through wire harness.
 - RoHS compliant & AEC qualified
 - ESD protection, optimized EMC
 - 3.3V and 5V compatible logic inputs
 - Very low power DMOS leakage current in OFF
- Protection features
 - Load dump
 - Current limitation
 - Loss of ground/battery
 - Efficiently drives large current load with low on-resistance, thus contributing to reduced power consumption of the ECU.
 - Overvoltage protection.
- Diagnostic features

- Proportional load current sense
- Open-load in ON- and OFF-state
- Short-circuit to battery and ground
- Overtemperature sense

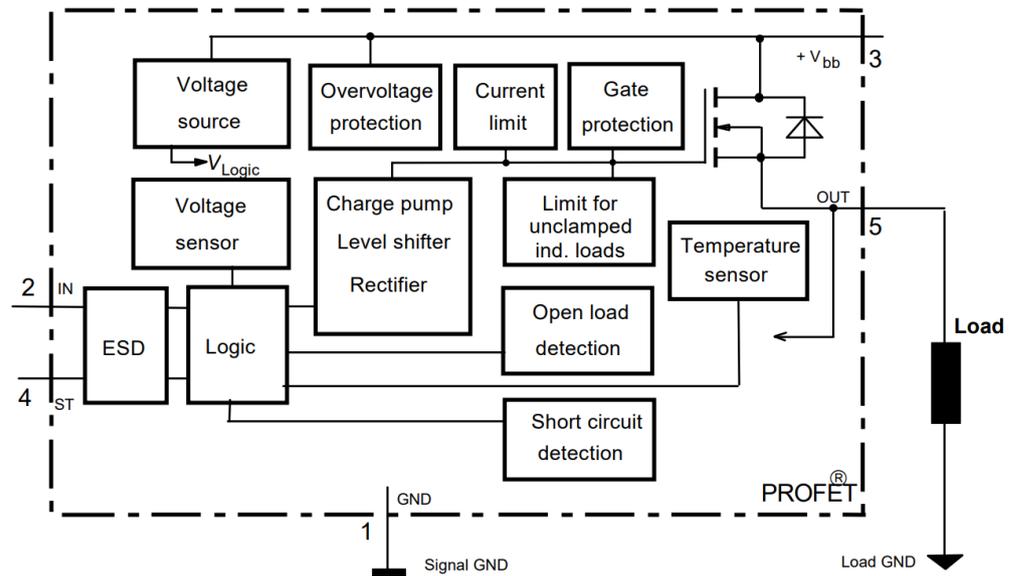


Figure 11. BTS410E2 Block diagram (Infineon Technologies AG, 2013)

2.3 Linear solenoid actuator

2.3.1 General actuators

Introduction

An actuator typically is a mechanical component that takes energy (is normally created by air, electricity or liquid) and converts it into motion/force. That motion can be in virtually any form, such as blocking, clamping or ejecting (Poddar, 2015). Actuators are used in many kinds of projects, manufacturing or industrial applications.

The architecture of actuator is illustrated in Figure 12. When energy is applied to actuator, motion/force is generated. Moreover, digital signal is also an input for actuator which is sent by controllers.

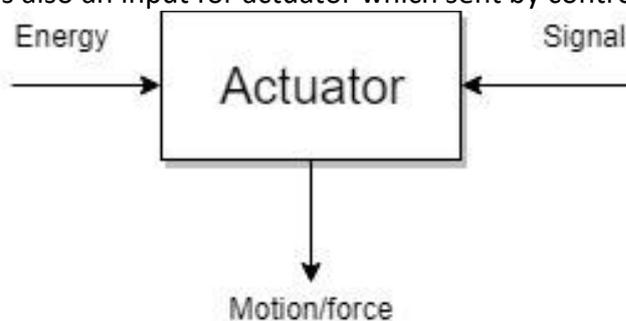


Figure 12. Actuator's architecture

Energy source

The most common source applying to actuator is air and that type is called a pneumatic cylinder or air cylinder. This method is that uses the stored energy of compressed air to move a piston when the air is released or uncompressed. These kinds of actuators are commonly used in manufacturing, grippers from robotics and assembly processes.

Electricity or hydraulics is another option of power source to supply to actuators. Pretty much like there are air cylinders, there are also electric cylinders and hydraulic cylinders in which the cylinders convert electricity or hydraulics into motion. (Poddar, 2015)

Actually, many actuators can operate by more than one type of power source. For instance, solenoid valves can be powered by both air and electricity or hydraulics and electricity.

Type of motion

Actuators can produce a linear motion, rotary motion or oscillatory motion that means in one direction, in a circular motion or in opposite directions at regular intervals. There are two actions that an actuator can present. Single-acting means that the energy source causes movement in one direction and a spring is used for the other direction. Double-acting cylinders mean that the energy is used in two directions.

In this thesis project, the author has selected the linear solenoid actuator to run the mechanism. And using the single-acting to perform the unique working principle which was created by the author.

2.3.2 Overview

A linear solenoid, an example of which is seen in Figure 13, is another type of an electromagnetic actuator that transforms an electrical signal into a magnetic field producing a linear motion (Storr, 2014.).

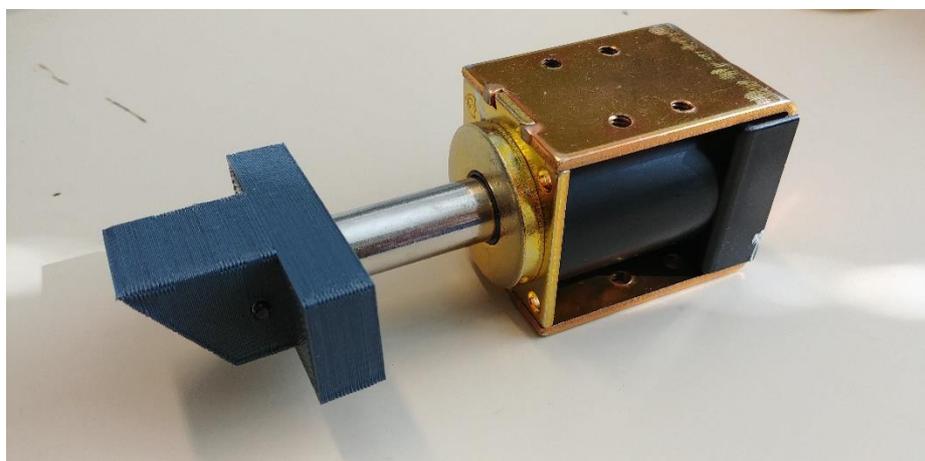


Figure 13. Pull- type of solenoid

The device has the name as “Linear solenoid” due to the linear directional movement and the action of the plunger. Linear solenoids are divided into two basic types called the “pull-type” as the plunger is normally outside the solenoid because the spring naturally forces the plunger out, the force pulls the plunger into the solenoid when energized; and the “push-type” is the opposite, in that the spring forces the plunger into the solenoid, but when energized the plunger is pushed out.

2.3.3 Working principle

Basically, the linear solenoid includes an electrical coil wound around a cylindrical tube with a ferro-magnetic actuator or “plunge” that is free to move or slide “IN” and “OUT” of the coils body. When sending an electric current through the coil, a magnetic field (as can be seen in Figure 14) is created. This coil of wire becomes an “Electromagnet” with its own north and south poles exactly the same as that for a permanent type magnet. The tenacity of this magnetic field can be increased or decreased by either controlling the amount of current flowing through the coil or by changing the number of turns or loops that the coil has (Storr, 2014.).

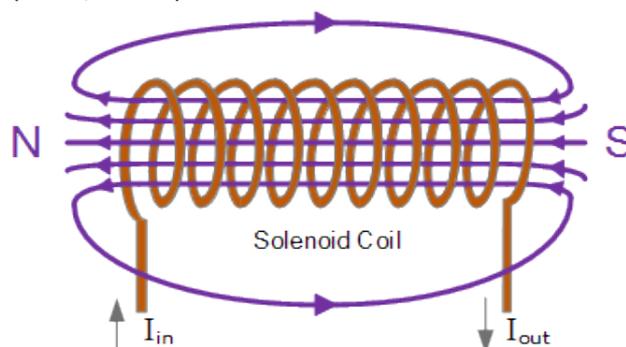


Figure 14. Electromagnetic field due to the flow of. (Storr, 2014)

This solenoid does not need a H-bridge or an expensive motor driver. Instead, a single MOSFET/ PROFET or relay will switch and control the solenoid. The inner shaft of a solenoid is a piston like cylinder made of iron and steel, called the plunger or slug (pertain to an armature). This plunger is handled a force causing by a the magnetic field, either attracting or repeling it. When the power is turned off, the electromagnetic field which generated previously by the coil collapses and the energy stored in the compressed spring forces the plunger back to its original rest position. This back on forth movement of the plunger is called “stroke” (as can be seen in Figure 15) , in other word the distance the plunger can travel in either an “IN” of an “OUT” direction (Storr, 2014.).

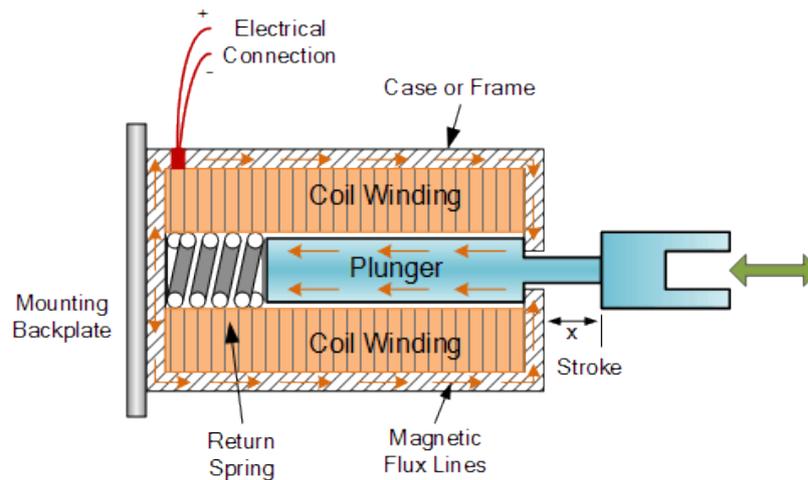


Figure 15. Inside the pull-linear solenoid (Storr, 2014)

2.3.4 Solenoid duty cycle and voltage

Solenoid duty cycle

The duty is known as a percentage and represents the proportion of time that solenoid is energized: For example:

- If a solenoid is powered for 15 seconds and switched “off” for 45 seconds before being energised again. Then the total on/off cycle time is 60 seconds which means 25% duty cycle. This called intermittent duty solenoid.
- If the energised period is continuous then the solenoid will need to be rated at 100%. Continuous duty solenoids are developed for operating conditions with continuous, heavy-duty uses. They are more durable than intermittent solenoids and physically larger in size.

Many recent solenoids come with a standard 100% duty cycle which is ideal for most applications (The glossary of solenoid, n.d.).

Voltage

The user should specify the operating voltage of solenoids. Most solenoids are DC and can be wound for any voltage. A solenoid can be “over-voltage” to achieve a greater force than the spec design. However, the over voltage will create additional heat, thus reducing the duty cycle.

2.3.5 Applications

With the features of linear solenoids, they can be applied to many applications which are presented from Figure 16 to Figure 19.

There is an announcement that an electromechanical solenoid can melt if the user power it too long. When selecting linear solenoid, as with any other solenoid style, it is important to consider the effects of heat, since an increase in coil temperature reduces the work output and the life of the unit. Life rating extend to 5 million cycles depending on the product

size and application. When determining an application's force requirement, the user must apply a 1.3 to 1.5 safety factor. For example: when a 2 kg pull force is required, recommendation is selecting a model with a safety factor of 1.3 to 1.5 times (2.6 to 3 kg).

In this project, the author selected the pull-type of liner solenoid to control the position of the plates. Moreover, the box frame type of linear solenoid has been chosen. This solenoid has a four-sided closed box frame and solid plunger. The closed, box frame also provides improved mechanical strength.

Kick

With kick application, a solenoid takes responsibility to immediately kick out the rejection part with the combination of motion electronics and detection sensors.

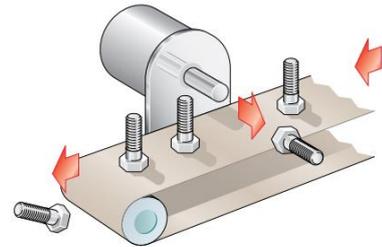


Figure 16. Kick application (Solenoids, 2012)

Divert

To divert one line into two separate lines, linear solenoids can also do it. The gate diverters, can be used continuously or very infrequently. Prefer long life rating of solenoid up to 100+ million actuations.

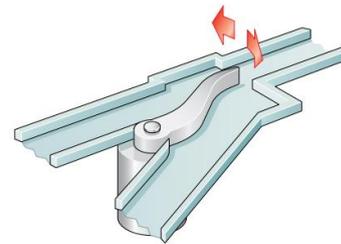


Figure 17. Divert application (Solenoids, 2012)

Lock / latch

Solenoids are used as locking applications including vault doors, cash registers, disk drives and missile systems.

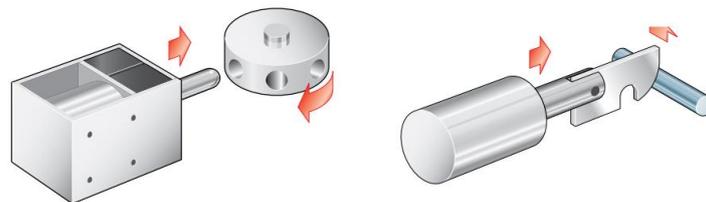


Figure 18. Lock/latch application (Solenoids, 2012)

Position

Positioning applications can range from a simple ratcheting device, such as the one illustrated, to precise variable positioning sub-assemblies.

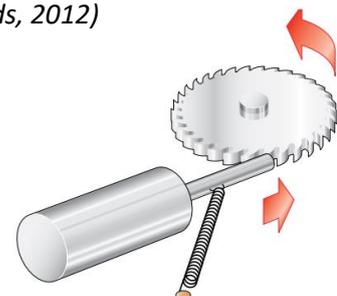


Figure 19. Position application (Solenoids, 2012)

2.4 Inter-integrated circuit (I²C)

2.4.1 Introduction

The inter-integrated circuit (I²C) protocol is the official procedure aimed to allow multiple “slave” digital integrated circuits (“chips”) to communicated with one or more “master” chips. It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance (SFUPTOWNMAKER, 2013).

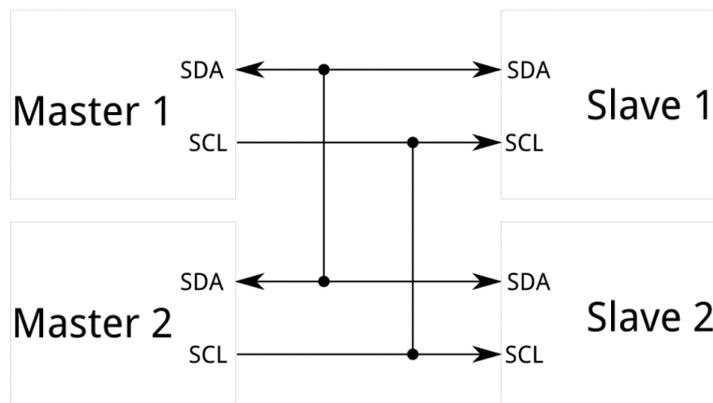


Figure 20. I²C master and slave relationship (SFUPTOWNMAKER, 2013)

I²C needs a mere two wires, as well as asynchronous serial, but those two wires can support up to 1008 slave devices. I²C can support a multi-master system, all devices on the bus are communicated more than one master (although the interaction between master devices cannot reach over the bus and they have to take turns using the bus lines).

2.4.2 Brief of history

- In 1982, I²C was invented and developed by Philips for vast of Philips chips. The spec allowed for only 100kHz communications and 7-bit addresses are provided, limiting the number of devices on the bus to 112 (there are several reserved addresses, which will never be used for valid I²C addresses).
- In 1992, a 400kHz fast-mode was added as well as an expanded 10-bit address space. There are three additional modes specified: fast-mode plus at 1MHz; high speed mode at 3.4MHz; and ultra-fast mode at 5 MHz.
- In 1995, Intel introduced a variant called “System Management Bus” (SMBus). SMBus intended to maximize predictability of communications between support ICs on PC motherboards. SMBus contents a clock timeout mode which makes low-speed operations illegal, although many SMBus devices will support it anyway to maximize interoperability with embedded I²C systems
- Today, the I²C bus is used in many difference application fields than just audio and video equipment. The I²C bus has been

accepted by several leading chip manufactures like Xicor, ST Microelectronics, Infineon Technologies, Intel, Texas Instrument, Maxim, Atmel, Analog Devices and others.

(SFUPTOWNMAKER, 2013)

2.4.3 Protocol

The signalling must stick fast to a certain protocol for the devices on the bus to admit it as valid I2C communications.

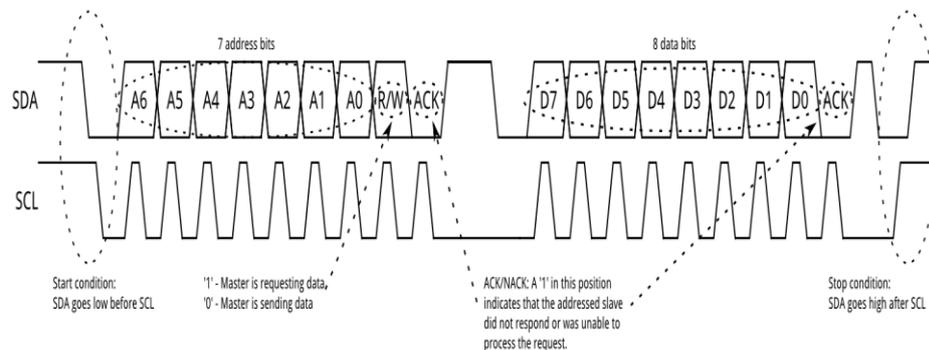


Figure 21. Bit frame of I2C (SFUPTOWNMAKER, 2013)

The signal messages are divided into two types of frame: an address frame, where the slave is indicated by the master to which the message is being sent, and one or more data frames, which are 8-bit data messages passed from master to slave or vice versa. Data is located on the SDA line after SCL goes low, and is sampled after the SCL line goes high. The devices on the bus define the time between clock edge and data read/write.

Start condition

To start the address frame, the master device leaves SCL and pulls SDA low. With this action, all slave devices are put on notice that a transmission is to initiate. If two master devices intend to take ownership of the bus at one-time, whichever device pulls SDA low first wins the race and gains control of the bus.

Address frames

The address frame is always appeared first in any new communication sequence. For a 7-bit address, the address is clocked out most significant bit (MSB) first, followed by a read/write bit indicating whether this a read (1) or write (0) operation.

From the Figure 21, for every 8 bits of data to be sent, one extra bit of meta data appears (the "ACK/NACK" bit). This is the case for all frames (data and address). Once the first 8 bits of the frame are sent, the receiving device is given control over SDA. If the receiving device does not pull the SDA line low before the 9th clock pulse, it can be inferred that

the receiving device either did not receive the data or did not know how to parse the message.

Data frames

After the address frame has been sent, data can start to be conducted. The master will continue developing clock pulses at a regular interval, the data will be located on SDA by either the master or the slave, depending on whether the R/W bit indicated a read or write operation.

Stop condition

Since all data frames have been transmitted, the master will generate a stop condition. These conditions are defined by $0 \rightarrow 1$ (from low to high) transition on SDA after a $0 \rightarrow 1$ transition on SCL, with SCL remaining high. During normal data writing operation, the value on SDA should not change when SCL is high, to avoid false stop conditions.

(SFUPTOWNMAKER, 2013)

2.5 Real time clock (RTC)

When working with Arduino platform, Arduino itself has built-in timekeeper, but we still need a separate RTC for Arduino project (Figure 22). The point is that the RTC module operates with a battery and can keep track of the time even if the users reprogram the microcontroller or turn off the main power.



Figure 22. Real time clock DS3231 model

The version used in this project is DS3231. The DS3231 is an inexpensive and highly accurate I²C real-time clock with an integrated temperature-compensated crystal oscillator (TCXO). The main feature is retaining hours, minutes and seconds, moreover the RTC can cover day, month and year information as well. In addition, it has automatic compensation for leap-years and for months with fewer than 31 days. The module DS3231 operates with voltage from 3.3 to 5V and needs a typical CR2032 3V battery to power up the module and maintain the information. Furthermore, the module uses the I2C Communication Protocol which makes the connection to the Arduino board easy (Nedelkovski, 2016).

Pinouts:

- Power pins
 - VCC – this is the power pin. To power the module, just give it the same power as the logic level of microcontroller such as Arduino 5V.
 - GND – common ground for power and logic
- I2C logic pins
 - SCL – I²C clock pin, connect to microcontrollers I²C clock line. This pin has a 10K pull-up resistor to V_{in}
 - SDA – I²C data pin, connect to microcontrollers I²C data line. This pin has a 10K pull-up resistor to V_{in}
- Other pins:
 - 32K – 32KHz oscillator output. Open drain, you need to attach a pullup to read this signal from a microcontroller pin
 - SQW - optional square wave or interrupt output. Open drain, you need to attach a pullup to read this signal from a microcontroller pin

Connecting module to Arduino (Figure 23):

- Connect V_{in} to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off.
- Connect GND to common power/data ground.
- The module uses the I²C bus, which makes the connection becomes very easy.
- Identifying which pins on Arduino or compatible boards are used for the I²C bus, these will be known as SDA (data) and SCL (clock). On Arduino Mega 2560 or compatible boards, these pins are D20 and D21 respectively.
- The module was tested by the author and can run in good precision.

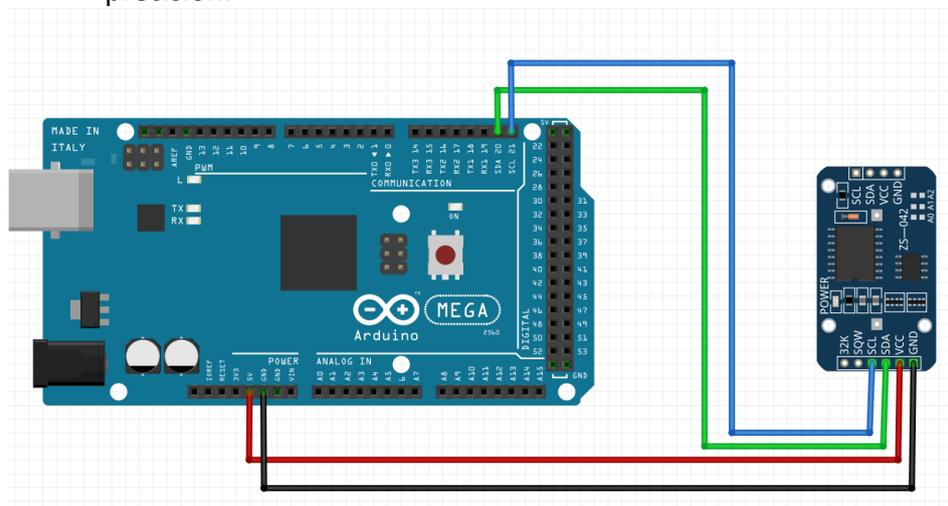


Figure 23. Connection between RTC module and Arduino Mega

2.6 Existing hay feeder on current market

2.6.1 Heinätin Single (Hayer)



Figure 24. Hayer from Oy Equine Innovations Ltd (Oy Equine Innovations Ltd, 2018)

The Figure 24 presents the existing hay feeder which manufactured and developed by Oy Equine Innovations Ltd. It has a triangle shape which made of powder-coated steel and it can locate on the corner of rented stall. The dimensions of the Hayer cabinet are 120x60x60cm. It has three layers which can contain feeding food (hay), and three actuators which placed on the left side control positions of those layers such as holding the layers and releasing the layers to fall downward. The Hayer can serve three portions per cycle with a maximum weight of 6kg per feeding. When the timer is triggered, hay will fall by gravity. The Figure 25 shows that the horse was fed by the hay feeder at winter time. Then the owner will fill the Hayer unit manually when empty. (Oy Equine Innovations Ltd, 2018).

Strength of the Hayer:

- Comes in two models, Multi and Single. The Multi models are suitable for stables with 1-5 Hayer units and the battery-driven Single models are suitable for individual horses in rented stalls, for example.
- It looks stylish, no corner sharp.
- Long battery life: up to 10 days outdoors at -15 degrees Celsius.
- The battery-powered Hayer Single does not require any wiring, and it is easy to move to a new stall or stable with your horse, if necessary.
- Less electricity consumption.



Figure 25. Hayer is located outdoor at winter time (Oy Equine Innovations Ltd, 2018)

2.6.2 DIY hay feeder for horses



Figure 26. DIY hay feeder (the left steel feeder) (Hukka, 2017)

The Figure 26 illustrates the DIY hay feeder which made by a Youtuber named Sari Hukka. The mechanism of this machine is pretty much same with the previous hay feeder from Oy Equine Innovations Ltd that using one layer and one simple actuator which controlled by a switch. So that on this model, there is only one portion on one cycle, and it can store up to 4 kg of hay. (Hukka, 2017)

Limitation of this model:

- No automation module, less productivity.
- Can only serve one portion on one cycle.

3 PREREQUISITE PLANNING OF THE PROJECT

3.1 Characteristic of horses

Horse is a herb animal (herbivore) and feeds on primarily grasses (hay), herbs, vegetables, and water. The Figure 28 and 29 present hay, oat and grain which are the mainly food supply for domestic horses in Finland. Lacking of protection of the herb, it is difficult for the horse to survive in the wild. Being as a group, horses become safer, more securities, protections, peaces and a greater chance of survival.

Domestication of the horse has influenced the characteristics of the horse, and this period took place around 5000 to 6000 years ago. To help the horse, the human gave the horse food and shelter. The horse is dependent on humans instead of the behavior of finding. Therefore the horse nowadays evolve through centuries from being hunted animal to become riding animal, working horse, beast of burden and draught-horse.

In the Figure 27 we can see that is the stable of a horse which currently located in Valkeakoski, Finland and the owner was taking care of the horse and feeding hay manually by laying hay on the ground. This picture was captured the last feed for her horse at 8:00 pm, after having a ride with it. Not only hay, oats and grain (Figure 29) are also fed to supply enough nutrition for horses.



Figure 27. A stable for horse



Figure 28. Bale of hay



Figure 29. Grain (left) and Oat (right)

In the wild, horses forage and graze for up to 18-hours every day. However, in captivity, most horses are fed only twice per day. This intelligent, sensitive, and curious animal may become bored while waiting several hours for his next feeding. Destructive and unhealthy behaviours may ensue resulting in aggressiveness cribbing, and even foraging & eating dirt, sand, & rocks. And free-feeding won't work for most horses, because they will gorge, which may lead to fatal colic. So according to current research of the eating behaviour of the horses, feeding small amounts of forage throughout the day helps promote a healthier gastrointestinal system, and also helps alleviate boredom. The result is a healthier and happier horse.

3.2 Existing hay feeder

Currently the job of feeding hay for domestic horses is the responsibility of humans. The owner of horse must spend a lot of time to feed the horse at a specific time in a day. However, when the owners cannot build the stalls for their horses, they must send horses to rented stalls illustrated in Figure 30 and hire employees to take care of their horses at night and during busy times.



Figure 30. Stalls for horses



Figure 31. Bunch of hay on the ground (left) and a hanging package of hay (right)

In Figures 31 and 32, there are two ways of feeding hay. On the left picture of Figure 31, the hay was placed in one corner or on the hay racks. On Figures and 31 and 32, the horses were fed by hay bags with long trips that allow the horse to lower his head during rest stops so its nasal passages can drain. These methods are mainly provided manually by humans if the cabinet is empty.



Figure 32. Hanging package of hay outdoor

3.3 Inputs of the project

After picking up knowledge about control system, phases of product development and scope of the project, the author wants to highlight to the reader here how the final product was debuted. Firstly, the planning phase known as the input for the project is presented in this chapter.

3.3.1 Identify customer needs

The goal of this activity is to understand the needs of the commissioner as well as other customers, and to effectively connect it with the other phases. The output of this step is a set of thoroughly constructed customer need statements, organized in a hierarchical list, with importance weightings for many or all the needs. After collecting requirements from the commissioner, the highlights for this project were:

- Build a prototype for an automatic feeder controlled by a Raspberry Pi or Arduino.
- Can provide 3-4 portions per cycle, each portion containing 2-3 kg of hay, three hours between portions
- A simple prototype, low cost in material and the installation.
- There are some feeding time options to select by the users.

3.3.2 Target specifications

The target specifications are responsible for description of what a product has to do. They are the translation of the customer need into technical terms. Targets for the specifications are set early in the process and represent the desires of the author. The output of this stage is a list of target specifications. Each specification comprises of a metric, and marginal and ideal values for that metric (T.Ulrich, Karl; D.Eppinger, Steven, 2016). The author established a set of targets in following part:

- Ergonomics: The system makes the user comfortable, safety in use and improves the productivity.
- The system can be controller and programmed to a precise schedule using real-time clock and its timer. For example, there are some options such as option 1: 2 hours; option 2: 2 hours 30 minutes; option 3: 3 hours; ...
- Friendly user interaction: There are simple inputs like two buttons which programmed by Arduino. The users can easily learn to use.
- There is a LCD to display recent time and next time for following portion.
- The system can supply enough portions for horses when the user left.

4 CONCEPT DESIGN

According to commissioner's needs, many concepts have been generated by the author. Product concept is an approximate description of the technology, working principle, and the form of the product. It is a brief description of how the product will satisfy the customer needs. A sketch usually indicates a normal concept design as well as a rough three-dimensional model and is often accompanied by a brief textual description.

The concept generation process starts with set of customer needs and target specifications and results in a group of product concepts from which the designer will make a final choice. The method of creating concepts, which outlined in Figure 33, consists five steps. The complicated problem splits into simpler subproblems. The external and internal search process determines solution concepts for the subproblems. Then classification trees and concept combination tables are used to systematically examine the space of solution concepts and to integrate the subproblem solutions into a total solution. Afterwards, the designer reflects on the validity and applicability of the results, as well as on the process used (T.Ulrich, Karl; D.Eppinger, Steven, 2016).

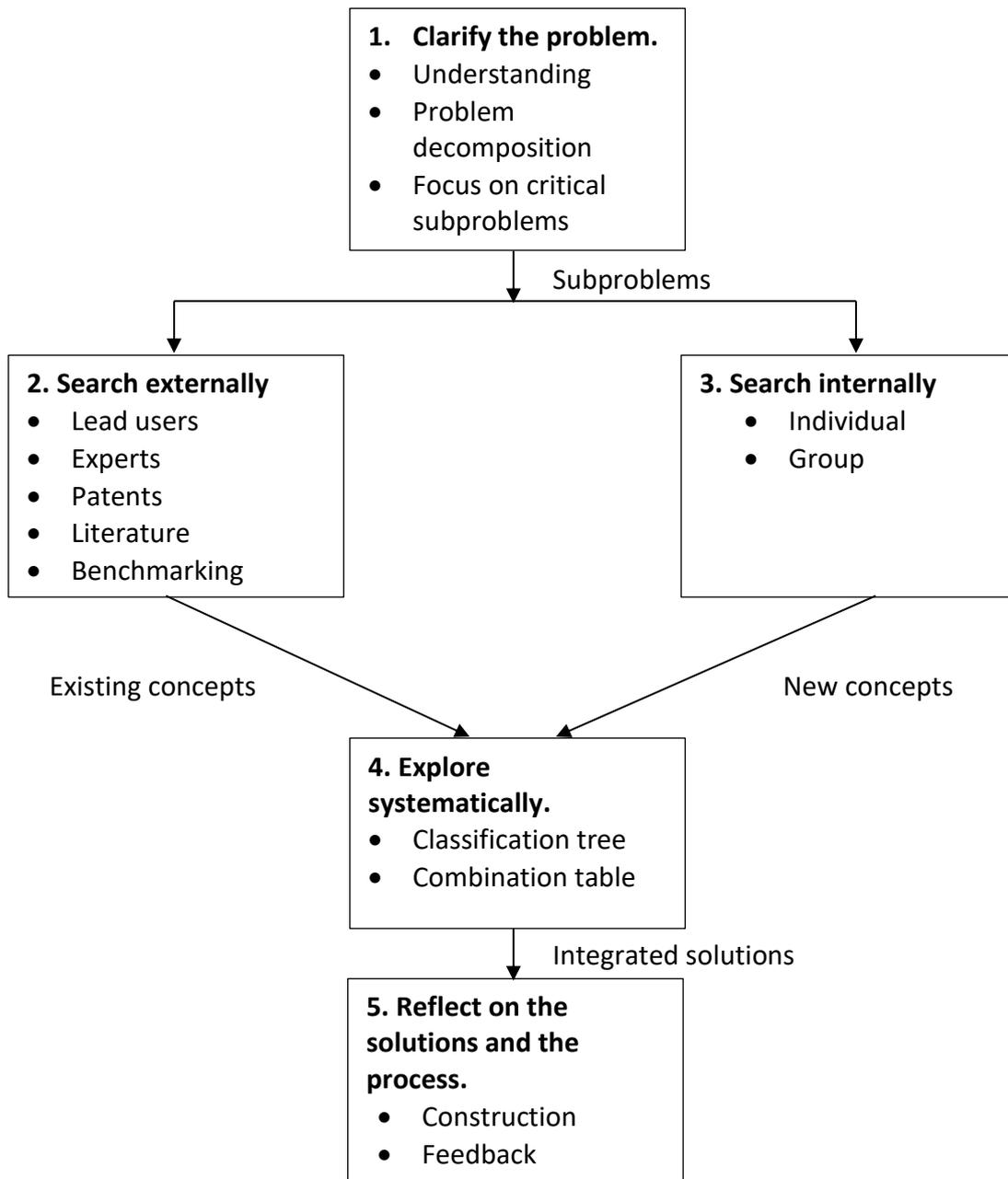


Figure 33. Concept generation method (T.Ulrich, Karl; D.Eppinger, Steven, 2016)

The purpose of designing layout for the feeder is intending to please commissioner's needs, improve quality and ergonomics and increase automation factor. At the end of concept design period, four different layouts were realized, the prototypes, the functional description, benefits and limitations would be highlighted in this chapter.

The factors which were used during the concept generation period is listed below:

- Customer needs
- Quality
- Price
- Ergonomics

- Friendly with both humans and horses
- Reliability and easy to maintenance
- Productivity

In this project, I chose the way to express the development process is as the initial creation of package of alternative product concepts. And then after testing time, the alternatives can be tightened and the final product which inherited the advantages from other alternatives will be attained. In opinion of the author, this design function plays the important role in describing the physical form of the product to best meet customer needs. The design function consists engineering design (mechanical, electrical, software, etc.) and industrial design (aesthetics, ergonomics, user interfaces). A suggested idea from commissioner and four idea designs which issued by the author are presented below.

4.1 Suggested design version

Figure 34 shows a proposed idea from the commissioner. The design has two layers which have joints with the left column. The feeding food – hay is stored above the layers and there will be two portions per one cycle. There are two lock-type actuators to hold the layers and release the layers when energised based on the timer.

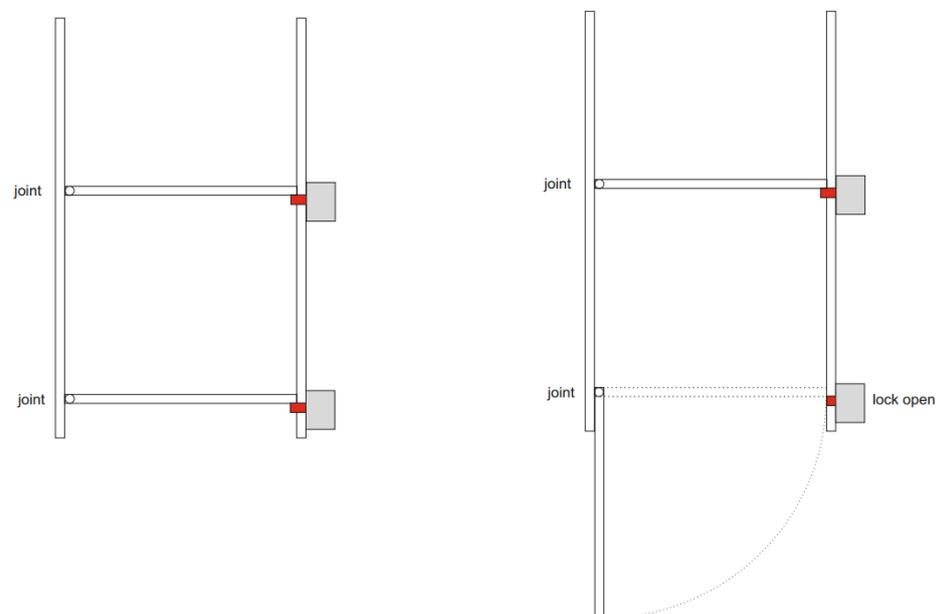


Figure 34. Suggested design version from commissioner

4.2 Concept design version 1

Figure 35 shows a completely different design from the suggestion layout. This concept uses the rotating working principle. There is a “grey” top tank which storing feeding food, in this case is hay. It can store up to 2 days of hay (which means 30kg of hay) in use depends on the

dimensions of the tank. There is a “black” cyclinder wheel is attached to the tank by the (orange) hard steel tube through their middle holes. In addition, there are (red) saw-toothes on the edge of the wheel, they will make the hay will follow smoothly. The wheel has a quarter-cut, it means there is a space for hay, and the wheel will rotate 60 degrees in order to transfer the hay downward to the tray.

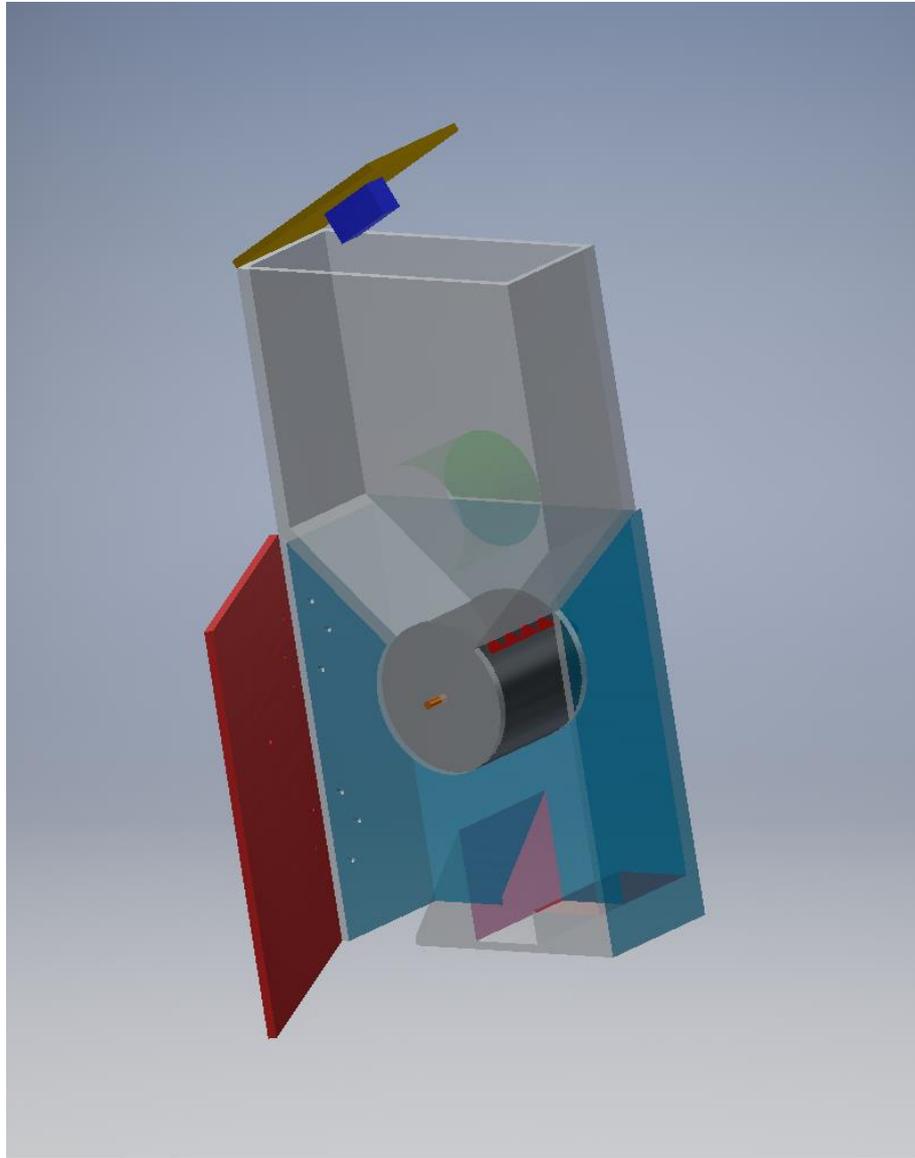


Figure 35. Concept design version 1

Advantages of concept design version 1:

- System can be fully automated for whole day and up to two days, eight portions per day, 3 hours for per portion, 2-3 kg per portion.
- Hay capacity is improved.
- Work load of human reduces.

Disadvantages of concept design version 1:

- Need a high-power motor to rotate the wheel.
- Cost of design could be expensive.

- There will be a fault between the wheel and the tank because of the hay stuck.
- Maintenance work is complex.

4.3 Concept design version 2

In this concept (Figure 36), the author used a linear actuator to operate the prototype. Basically, there are four layers carrying 2-3kg of hay and attached to the middle column. By experience, 2kg of hay equal to 30 dm³. The space between two layers is 30 cm, the length of the layers is also 30 cm and the depth of the layers is 35 cm. Furthermore, the layers are tilted at about 60 degrees to the middle column.

When the timer is triggered, the linear actuator will be energised and pull the layer down. Then all the hay on top of that layer will fall by gravity. The rest layers will be pulled down in turn from bottom to top. When all layers are released, there will be a push-button in order to pull those layers up to original positions and they hay will be provided manually by human.

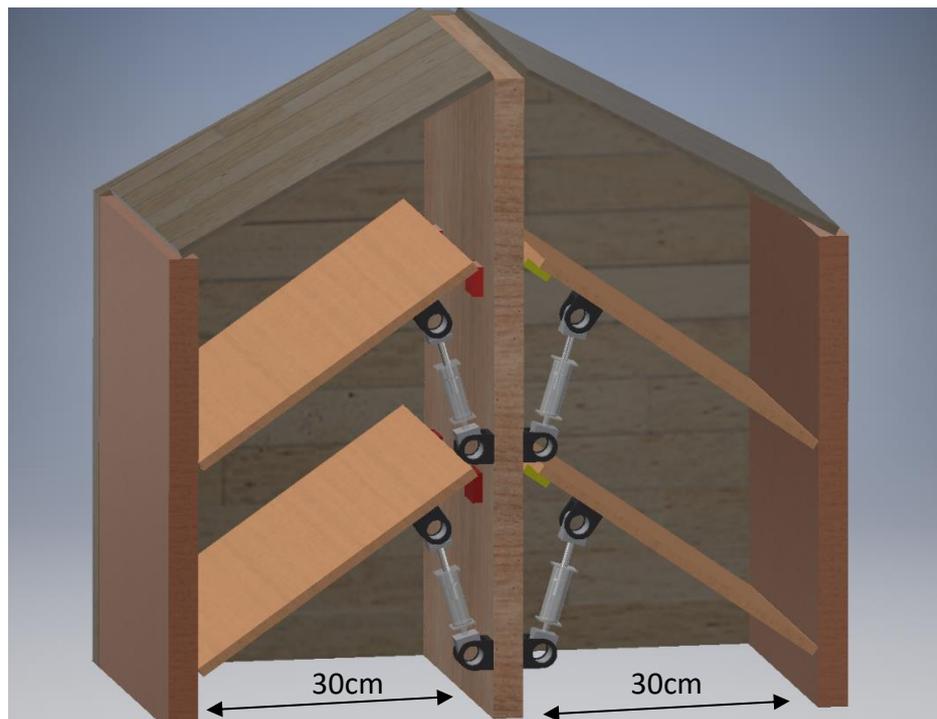


Figure 36. Concept design version 2

Advantages of concept design version 2:

- Fully automated controlling layers.
- Safety and ergonomics are improved.

Disadvantages of concept design version 2:

- High cost in actuators.
- Complex in working principle and position of linear actuator.

- Power consumption is increased.

4.4 Concept design version 3

Figure 37 illustrates the operating principles of push/pull linear solenoid actuators. There are eight layers which are attached to two side plates by hinges, and they carry 2 kg of hay which equals to 30 dm³. The space between two layers is 30 cm, the length of the layers is also 30 cm and the depth of the layers is 35 cm. Furthermore, the layers are tilted at about 60 degrees to the left and right hand side columns.

Four two-way-pull solenoids which are located on the middle plate take responsibility for controlling positions of the layers. Once the left side of the solenoid is pulled-in, then the left layer will fall. Vice versa, the right layer will fall.

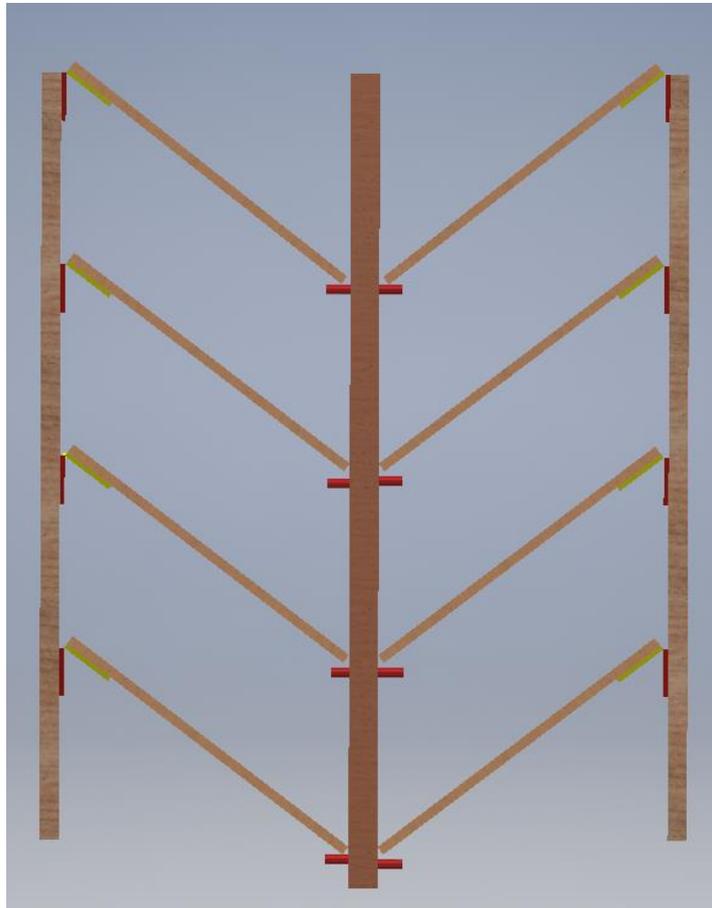


Figure 37. Concept design version 3

Advantages of concept design version 3:

- The productivity is increased.
- Safety and ergonomics are improved.
- Take less space to place actuators.

Disadvantages of concept design version 3:

- Hard to find suitable solenoids.

- It is difficult to put the layers to the original position.

4.5 Concept design version 4

In concept design version 4 illustrated in Figure 38, the author took his inspiration from the suggested design. There are four layers which can contain 2 kg of hay and they are connected to the middle plate. Now, four pull-type solenoids are placed on two side plates and control four layer 'positions.

Once the solenoids are energised, their slugs pull in, hence the layers will fall and hay will drop down. The layers can be placed to their original positions by hand, then the cycle will reset.



Figure 38. Concept design version 4

Advantages of concept design version 4:

- System cost is lower
- Installation time is lower
- Control system design is simple
- No extra training for operator personnel
- Easy of maintenance.

Difficulties of concept design version 4:

- Need to find the right force for solenoids to hold the layers.
- Need to cover the solenoids in order not to break them.

5 FINAL PRODUCT

The systematic design process of concept design typically follows these steps:

1. The identification and analysis of the problem
2. Definition of the goal of the task
3. Searching for solutions
4. Evaluation of solutions
5. Choosing the most promising variants for follow-up development
(T.Ulrich, Karl; D.Eppinger, Steven, 2016)

After following above steps, integrated with benefits of all versions, the author and the commissioner have decided to choose and design the final model as can be seen in Figure 39. Besides, the real size product is conducted (Figure 40 and 41). Because there was lack of tools to build the machine, so that the real size model is a bit different when comparing with final concept design. However, the mechanism of this product is still achieved.

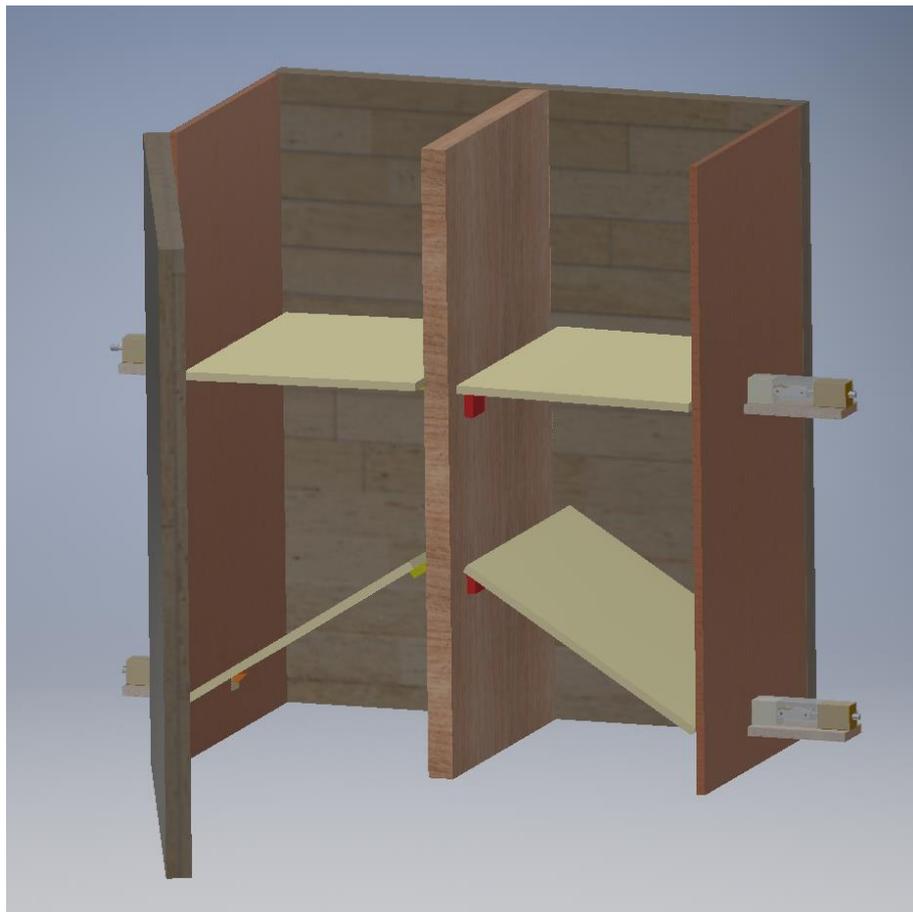


Figure 39. Final concept design

Figure 37 shows the front view of the real size product and clearly see that there are four white layers made of polycarbonate with the wood frame and the control box on the left side.



Figure 40. Real size product

Figure 38 presents two sides of the product which obviously see four solenoids' places.



Figure 41. Left and right side of real size product

5.1 Product overview

The hay feeder is an out-of-the-box solution for feeding individual horses in rented stalls. The automatic feeder provides pre-set amounts of forage (hay) for the horse, up to four portions per one cycle and a set of waiting timer for each portion. To serve four portions, the system has four layers to carry the amount of hay (in this thesis project, 2kg is enough). There are four pull-type solenoids take responsibility for holding and releasing the layers at the edge of them. When a solenoid is energised, the related layer will move downward, then hay will fall by gravity.

The hay is loaded directly into the feeder unit in the stable by humans. There are decent spaces between layers to store enough 2kg of hay. After finishing a cycle, the layers can be placed back to the original positions with simple action and this action will be mention in feature chapter.

The system is operated by microcontroller Arduino with simple programming code in C+ language. In addition, with the real-time clock module, the current time and date are always updated in the system. The program is designed to generate three pages which consist recent time for users to keep on track. Besides that, there are five timer options to select for portion delay such as:

- Option 1: 2 hours / portion
- Option 2: 2 hours and 30 mins / portion
- Option 3: 3 hours / portion
- Option 4: 3 hours and 30 mins / portion
- Option 5: 4 hours / portion

An automatic hay feeder is a good solution for feeding hay to horses more regularly around the clock. The hay feeder is an optimal solution for horses, their owners and caretakers alike.

5.2 Technical data

The hay feeder contains four feeding of hay with a maximum weight of 2.5kg per feeding. The dimensions of the feeder cabinet are 75x70x35cm. The hay feeder is made of wood for frame and polycarbonate for layers. It will be mounted on the wall of individual stall and there is a front cover to avoid the interaction form horses.

There is a control box located on the left of feeder for the user to interact with the system. The user can enter eight times to the feeder to perform the best solution for horses per day. The system only operates indoor because of type of materials.

The components of this thesis project included:

Microcontroller: Arduino MEGA 2560

Real time clock module: DS3231

Motor switches:

Model BTS410E2 of PROFET is chosen to drive the solenoids. The size of this model is compact and simple to connect pins.



Figure 42. PROFET model BTS410E2

Liquid Crystal Display (LCD) has two rows with 16 characters each.



Figure 43. LCD 16x2

Solenoid:

Pull-type solenoids from Guardian electric Manufacturing (Figure 44). According to the theoretical part, this model of solenoid is chosen to run the mechanism with the adding 3d printing part. The following table illustrates the technical meter of solenoid:

Table 4. Solenoid overview

Model no.	Duty Cycle	Voltage (V)	Resistance	Power (W)	Current (A)
4HD-I-24D	Intermittent	24VDC	18.9	32	1.27A



Figure 44. Pull-type solenoid

Continuous Duty: 100% 'On' time

Intermittent Duty: 25% 'On' time, (100 seconds 'On' maximum, followed by 300 seconds 'Off' minimum)

The Table 6 shows the range of force for the solenoid. Because of the working principle which selected by the author, Intermittent duty is chosen in order to present higher force for the application. Furthermore, there is an instruction when using intermittent duty solenoid, that a solenoid can be energised only 100s. And after that the solenoid must rest at least 300 seconds. Hence, the solenoid can operate in perfect force.

Table 5. Force of solenoid

Stroke (cm)	Pull force (N)								Holding force (N)
	0.127	0.318	0.635	0.953	1.27	1.588	1.905	2.54	-
Continuous Duty	34.8	25.6	13.3	9.2	5.6	2.8	2.2	1.4	56.7
Intermittent Duty	45.9	36.1	23.6	16.7	11.1	8.3	6.1	5.6	65.3

Pushbutton

There are two momentary switches which closing a circuit when pressed to control the system. Menu pushbutton has function to change three pages in turn. Select pushbutton has function to select portion options in option page.

Control box

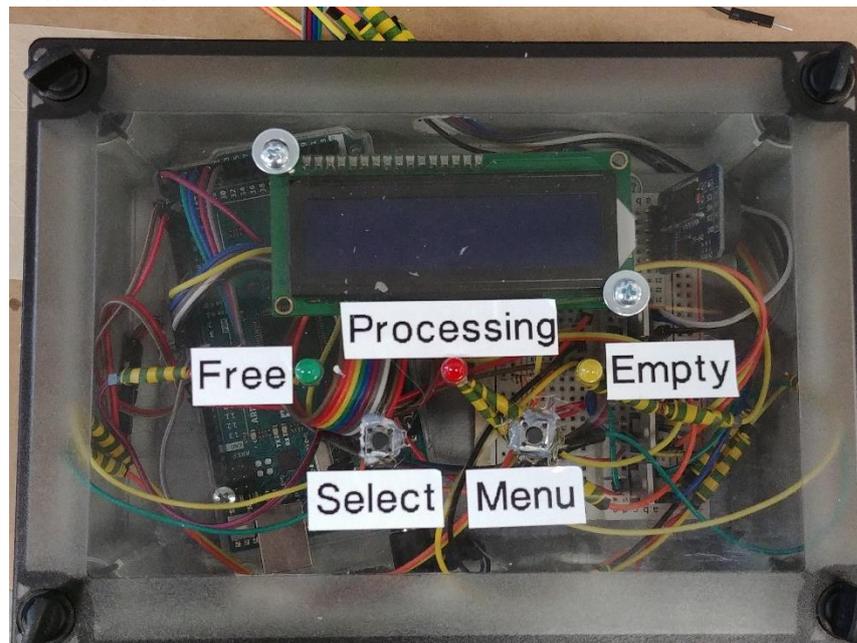


Figure 45. Control box

The Figure 45 presents the Arduino control box which has four outputs such as LEDs and LCD, and two simple inputs as pushbuttons. The green LED presents the free status of the system, the red LED turns on when the system is in process, and the yellow LED illustrated the cabinet is empty. There are three holes on the bottom of the box which can plug in USB cable to debug the program and power source cables to supply power to the board and the solenoids.

5.3 Arduino breadboard

Figure 46 - Arduino breadboard was designed by the author which regarding to hay feeder project. The breadboard shows the wiring and number of component which appear on the system.

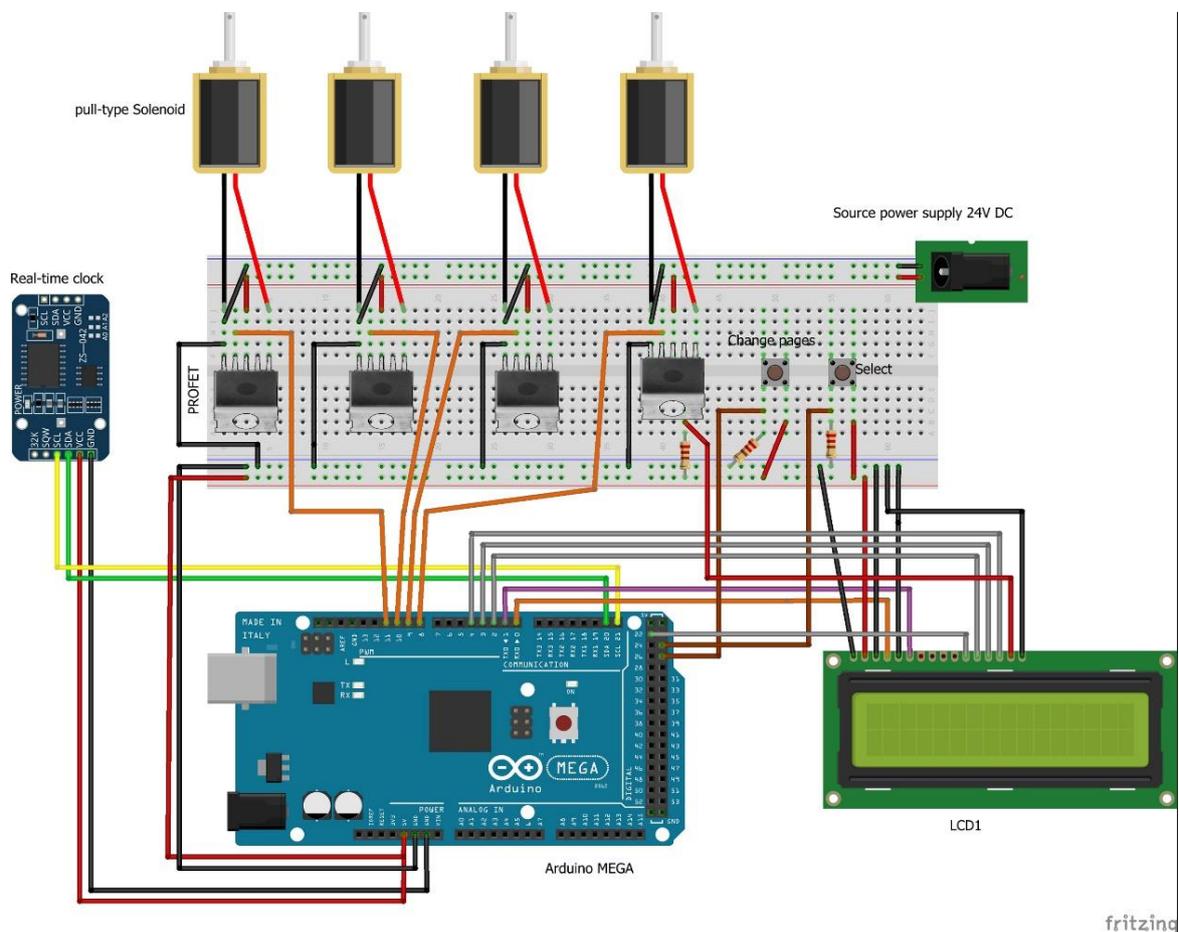


Figure 46. Arduino breadboard

5.4 Circuit architecture

The figure 47 illustrates the circuit architecture diagram which is a type of flowchart that shows the relationships between components. For system developers, they need the circuit architecture diagram to understand, clarify, and communicate ideas about the system structure and the user requirements that the system has to support.

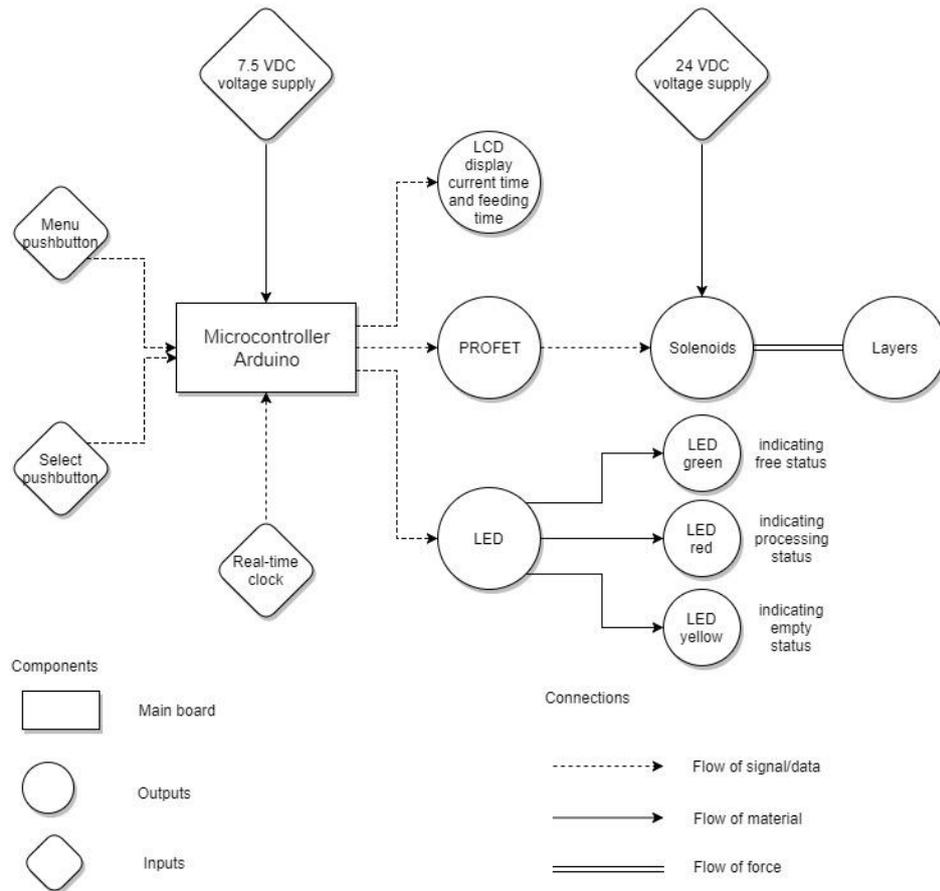


Figure 47. Circuit architecture

5.5 Development of code

In this section, brief of code development is presented. Full version is listed on Appendix 1.

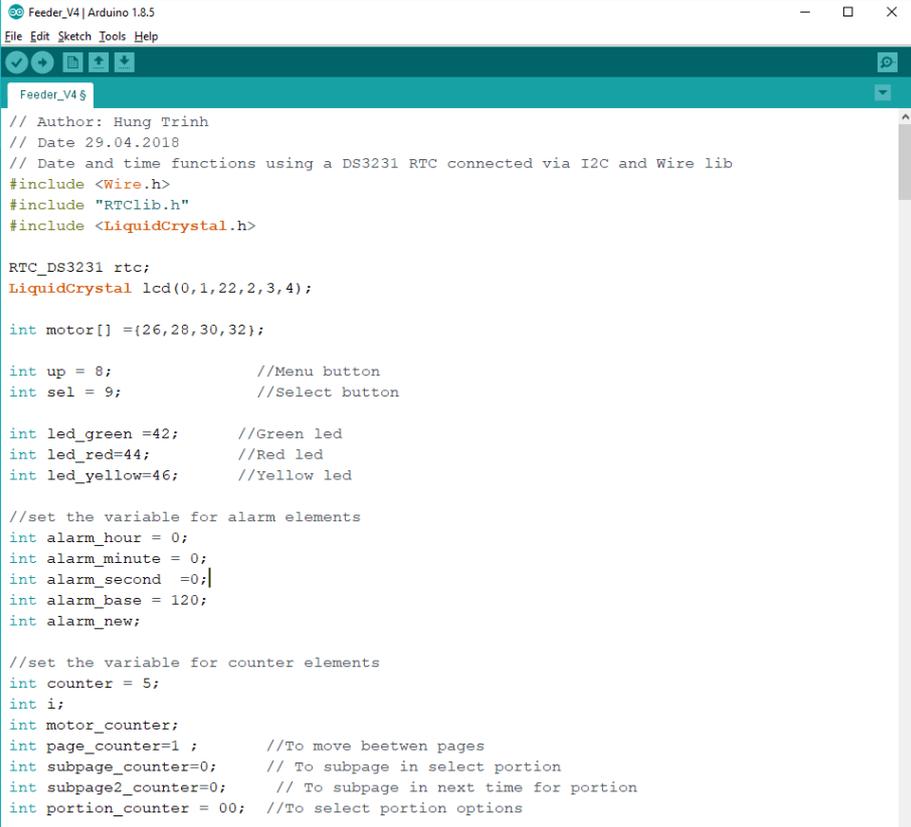
Software layout

The layout of Arduino software IDE is shown in Figure 48. The code was wrote and developed individually by the author. The code was used RTC library and some code lines to debug the real-time clock module.

RTC library and I²C library

The RTC library is retrieved online source on Adafruit (ada). The following code declares the RTC library and I²C library:

```
#include <Wire.h>
#include "RTClib.h"
```



```

Feeder_V4|Arduino 1.8.5
File Edit Sketch Tools Help
Feeder_V4$
// Author: Hung Trinh
// Date 29.04.2018
// Date and time functions using a DS3231 RTC connected via I2C and Wire lib
#include <Wire.h>
#include "RTCLib.h"
#include <LiquidCrystal.h>

RTC_DS3231 rtc;
LiquidCrystal lcd(0,1,22,2,3,4);

int motor[] = {26,28,30,32};

int up = 8;           //Menu button
int sel = 9;         //Select button

int led_green =42;   //Green led
int led_red=44;     //Red led
int led_yellow=46;  //Yellow led

//set the variable for alarm elements
int alarm_hour = 0;
int alarm_minute = 0;
int alarm_second =0;
int alarm_base = 120;
int alarm_new;

//set the variable for counter elements
int counter = 5;
int i;
int motor_counter;
int page_counter=1 ; //To move beetwen pages
int subpage_counter=0; // To subpage in select portion
int subpage2_counter=0; // To subpage in next time for portion
int portion_counter = 00; //To select portion options

```

Figure 48. Arduino software

There is a way to get the time using RTC lib, which is to call `now()`, a function that returns a `DateTime` object that describes the year, month, day, hour, minute and second when you called `now()`. The following code shows the current date and time on LCD display.

```

void loop () {
    DateTime now = rtc.now(); //declare now() to call RTCLib
    lcd.print("Time:");
    lcd.setCursor(6,0);
    if (now.hour() <10){ lcd.print("0");}
    lcd.print(now.hour(), DEC );
    lcd.print(':');
    if (now.minute() <10){ lcd.print("0");}
    lcd.print(now.minute(), DEC );
    lcd.print(':');
    if (now.second() <10){lcd.print("0");}
    lcd.print(now.second(), DEC);
    lcd.setCursor(0,1);
    lcd.print("Date:");
    lcd.setCursor(6,1);
    lcd.print(now.day(), DEC);
    lcd.print('/');
    lcd.print(now.month(), DEC);
    lcd.print('/');
    lcd.print(now.year(), DEC);
}

```

Menu pushbutton and select pushbutton

```

//-----If menu button pushed-----
if (subpage_counter ==0){
if (last_up== LOW && current_up == HIGH){
//When menu button is pressed
  lcd.clear();
  if(page_counter <3){
//Page counter never higher than 3(total of pages)
    page_counter= page_counter +1; //Page up
  }
  else{
    page_counter= 1;
  }
}
last_up = current_up;
}

```

The above code is designed to control the menu pushbutton. 'Page_counter' takes responsibility for storing order number of pages. Then using 'switch-case' function to show page's contents.

First page - home page

The following code shows the first page of the program (case 1) which has current time and date. Moreover, the Figure 49 illustrates the LCD and what is going on in first page. The author used RTCLib and function now.() to call time and date.

```

switch (page_counter) {

case 1:{ //Design of home page 1
  DateTime now = rtc.now();
  currentTime();
  lcd.setCursor(0,1);
  lcd.print("Date:");
  lcd.setCursor(6,1);
  lcd.print(now.day(), DEC);
  lcd.print('/');
  lcd.print(now.month(), DEC);
  lcd.print('/');
  lcd.print(now.year(), DEC);
}
break;

```



Figure 49. First page

Second page – option page



Figure 50. Second page

```

case 2: { //Design of option page
  lcd.setCursor(0,0);
  lcd.print("Select portion:");
  lcd.setCursor(2,1);
  if(portion_counter <10){ //To avoid "0" of number 10
    lcd.setCursor(2,1);
    lcd.print("0");
  }
  lcd.print(portion_counter);

  //Control portion_counter parameter
  if (last_up== LOW && current_up == HIGH){
  //When Select button is pushed
    if(portion_counter < 5){
  //portion_count never higher than 5 (max value)
      portion_counter ++ ;
    }
    else{
      portion_counter = 1;
  //If portion_count higher than 5, return to 0
    }
  }
  last_up=current_up;

```

The above code is programmed to generate the second page (case 2) which contains 'select portion:'. And the second page is illustrated in Figure 50. After that, the user need to press the select pushbutton to change '00' to different options such as '01','02','03','04','05'. Each

option is stored under 'portion_counter' parameter. With simple code of 'switch-case' function again, the user can have a look and select a desired waiting time. For example, the next code show option '01':

```
switch(portion_counter){
  case 1:{
    lcd.setCursor(6,1);
    lcd.print("2h");
    lcd.setCursor(8,1);
    lcd.print("  ");
    counter =4;
    i = 0;
  }
  break;
```

In Figure 51 shows five different waiting time options for controlling layers.

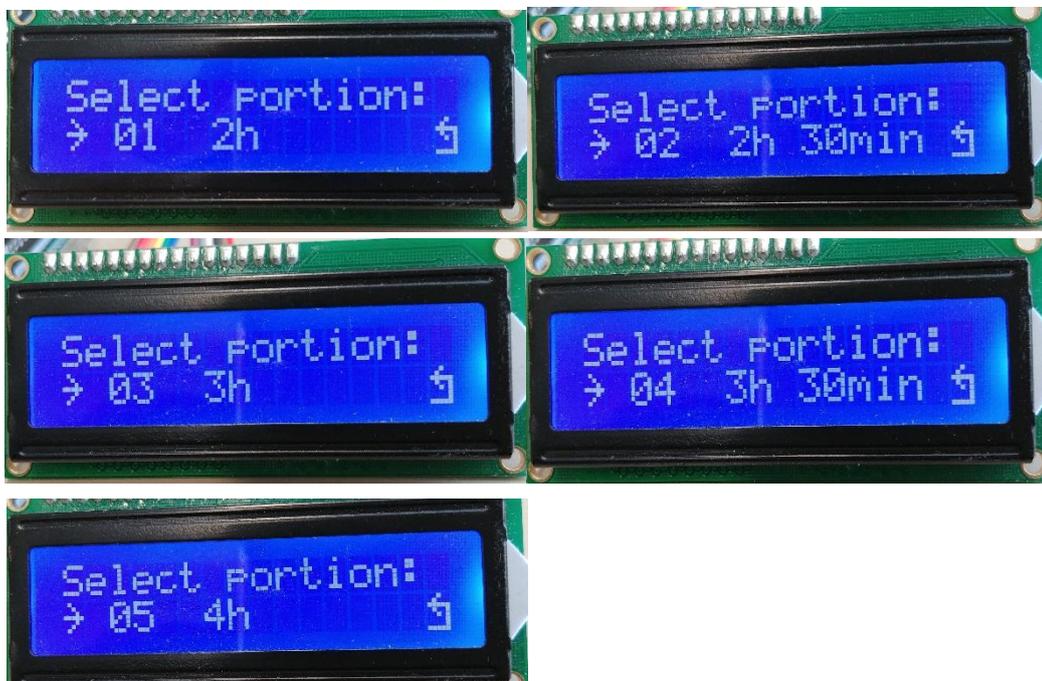


Figure 51. Five options of portion

As a result, after finding the best option, with simple calculation set by the author, the new time for next feeding to stored in 'alarm_hour', and 'alarm_minute'. These parameters will be timers for current time in page three.

```
DateTime now = rtc.now();
alarm_new = 90 + 30*portion_counter;
alarm_hour = now.hour() + (round(alarm_new/60));
if (alarm_hour >=24){ alarm_hour = alarm_hour-24;}
alarm_minute = now.minute() + (alarm_new-
(60*round(alarm_new/60)));
if (alarm_minute >= 60){
  alarm_minute = alarm_minute-60;
  alarm_hour = alarm_hour +1;
```

```
}

```

Third page – next portion time

After selecting a portion option, the user will change to page three to have a combination of current time and the next feeding time. The next code shows trouble-free comparison between current time and next feeding time. When the current hour equals to 'alarm_hour', current minute equals to 'alarm_minute', and current second equals to 10, the solenoid will be energised. After 5 seconds, the solenoid will be turned off.

```

    if (now.hour() == alarm_hour && now.minute() ==
alarm_minute && now.second() == 10){

        digitalWrite(motor[i],HIGH);
    }
    else if (now.hour() == alarm_hour && now.minute() ==
alarm_minute && now.second() == 15){
        digitalWrite(motor[i],LOW);
        i++;
        counter = counter -1;
        alarm_hour = now.hour() + (round(alarm_new/60));
        if (alarm_hour >=24){ alarm_hour = alarm_hour-24;}
        alarm_minute = now.minute() + (alarm_new-
(60*round(alarm_new/60)));
        if (alarm_minute >= 60){
            alarm_minute = alarm_minute-60;
            alarm_hour = alarm_hour +1;
        }
    }
}

```

LED – indicating machine status

```

//indicate for green led - free
if (counter == 5){
    digitalWrite(led_green, HIGH);
    digitalWrite(led_red, LOW);
    digitalWrite(led_yellow, LOW);
}
// indicate for red led - processing
if (counter < 5 && counter > 0){
    digitalWrite(led_green, LOW);
    digitalWrite(led_red, HIGH);
    digitalWrite(led_yellow, LOW);
}
//indicate for yellow led - empty
if (counter == 0){
    digitalWrite(led_green, HIGH);
    digitalWrite(led_red, LOW);
    digitalWrite(led_yellow, HIGH);
}

```

```
}

```

With the short code above, three LEDs are controlled to designate statuses of the machine such as free, processing, and empty.

5.6 Advantages & features

5.6.1 Advantages

- With the automatic hay feeders, you can upgrade your service range to a whole new level.
- Helps you to optimise the time and effort spent on the daily chores.
- The horses are fed at regular intervals around the clock, which improves their well-being and condition which leads to a deduced need for visits to the stable, thus saving money.
- A practical solution for feeding your horse in a rented stall at night.
- The price is reasonable, about 300 euros of materials and cost less than recent same products on market like 50% less.

5.6.2 Features

The solenoid head part is designed ergonomic in order to optimize the work that putting the layers back to their original position.

The user just needs to push the layers upwards to the slide part of the “lock-type” shape (as can be seen in Figure 52), the slugger will be pulled in. So that there will be spaces for layers to move on and then lay on the edges of “lock-type” shape. The step by step is illustrated in Figure 53.

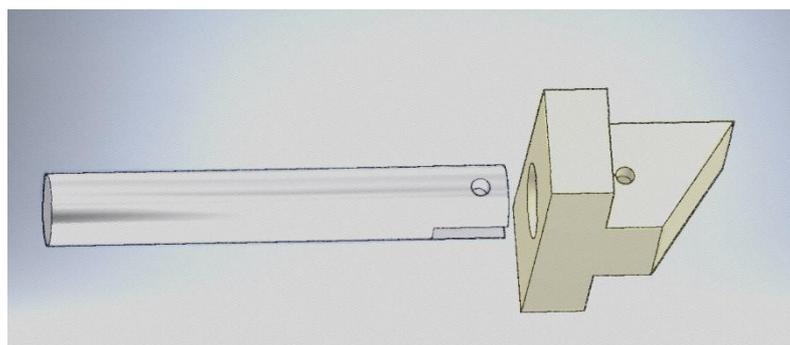


Figure 52. Lock-type shape (yellow part) which attached to slugger

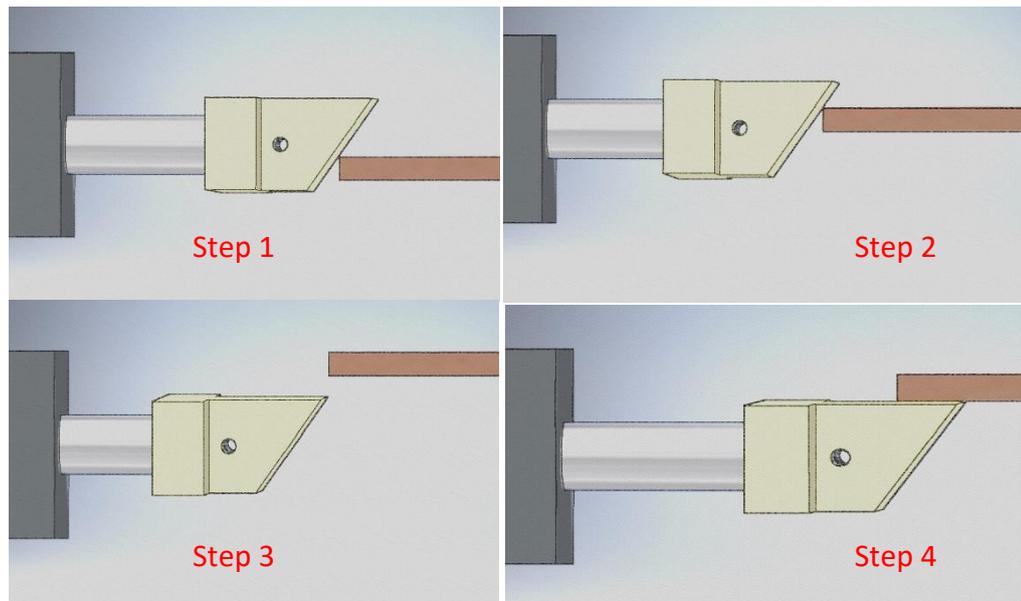


Figure 53. Step by step of mechanism

6 LIMITATION AND EXPANSION POSSIBILITIES

In the current design, the system has some limitations which are presented below:

- Only serves four portions in one cycle
- Does not really meet the user interface experience like sending statuses of machine to owner's smartphones.
- Because the feeder's frame is made of wood, so this model is only available indoor.
- The Arduino only operate in short range of temperature, cannot work well in really cold weather or very hot temperature.
- No sensors installed to detect the capacity of each feeding.

Further development of the final product was requested by the commissioner, he wants to expand the capacity of the Arduino to control up to four feeders with the same pre-set time for all the plates. Not only that, but also sending a signal to the owner's smartphone to warn him/her that the hay feeder is empty could be an option to add further automation.

For commercial products, this prototype needs lots of modifications. Firstly, the materials, the frame could be made of powder-coated steel and also the layers. Because with steel, the machine can handle in harsh environment such as lower temperature below 20C. Secondly, the smaller and stronger pull-type solenoids should be found to replace the recent solenoids. Thirdly, a set of sensors can be added to notice and warning the status of feeding stage. In addition, a bigger display should

be installed in order to highlight more details of the system. Finally, finding another microcontroller which can operate outdoor conditions.

7 CONCLUSION

After several months, the project was completed and it met all the requirements set by the commissioner. The author has gathered a lot of experiences in mechanical aspects in building an effective product in this project, and has learnt about automation logic to make an effective program.

The economic success of this project depended on the ability to identify the needs of commissioner and other customers. There was a need to quickly create products that would meet these needs and could be produced at low cost. For this prototype, the cost of manufacture is not so high when compared with market product, about 50% less. Moreover, this prototype is not difficult to assembly. Developing the working mechanism was the most challenging part of this project. The author designed a unique mechanism for the contact between the layers and the actuators to optimize the working principle of the feeder.

The user interaction with the hay feeder also played an important role in this project. Through practical experience, at a normal rented stall in Valkeakoski area, the author got to know the requirements how to feed horses and what the demands were for the user interface of the hay feeder. According to the factors, the author designed the programming for the system in order to keep it as simple as possible and easy to operate for the owner.

Overall, the satisfaction from the commissioner and the targets of this thesis project were achieved by the author. Further expansion and modification will be attained in the future and the author believes that this product can thrive.

REFERENCES

- ACTUATORS - SOLENOIDS*. (n.d.). Retrieved Mar 16, 2018, from society of robots: http://www.societyofrobots.com/actuators_solenoids.shtml
- ada, l. (n.d.). Adafruit DS3231 Precision RTC Breakout. [*web log message*]. Retrieved Mar 11, 2018, from learn adafruit: <https://learn.adafruit.com/adafruit-ds3231-precision-rtc-breakout?view=all>
- Arduino.cc. (2015, Feb 25). Introduction of Arduino. Retrieved Apr 2, 2018, from arduino: <https://www.arduino.cc/en/Guide/Introduction>
- b_e_n. (2012). What is an Arduino? [*web log message*]. Retrieved Apr 2, 2018, from sparkfun: <https://learn.sparkfun.com>
- Boxall, J. (2014, 11 28). Tutorial: Arduino and the I2C bus – Part One. Retrieved Apr 2, 2018, from tronixstuff: <http://tronixstuff.com/2010/10/20/tutorial-arduino-and-the-i2c-bus/>
- Brain, M. (2000, April 1). How Microcontrollers Work. [*web log message*]. Retrieved Mar 2, 2018, from howstuffworks: <https://electronics.howstuffworks.com/microcontroller.htm>
- Heath, N. (2017, Nov 30). What is the Raspberry Pi 3? Everything you need to know about the tiny, low-cost computer. Retrieved Apr 20, 2018, from zdnet: <https://www.zdnet.com/article/what-is-the-raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/>
- Hukka, S. (2017, July 26). Hunting machine for horses. Retrieved Apr 20, 2018, from Youtube: <https://www.youtube.com/watch?v=-8BJRqPPAvI>
- infineon. (2016, 03). *PROFET*. Retrieved from infineon: https://www.infineon.com/dgdl/Infineon-PROFET_Smart_high-side_switches_ProductOverview-SG-v01_00-EN.pdf?fileId=db3a304343a131180143b50598a525ab
- Infineon Technologies AG. (2013). *Smart High-Side Power Switch*. Retrieved from infineon: https://www.infineon.com/dgdl/Infineon-BTS410E2-DS-v01_01-en.pdf?fileId=db3a304331c8f8560131dcc9fc520e21
- Jeske, C. (2015, 10 05). Characteristic of successful products. [*web log message*]. Retrieved Mar 10, 2018, from willitlaunch: <http://www.willitlaunch.com/Characteristics-of-Successful-Products>
- Joeman, Kellyhensen. (2016, Feb 22). *Raspberry Pi 3 Model B Technical Specifications*. Retrieved Apr 15, 2018, from element14:

<https://www.element14.com/community/docs/DOC-80899/l/raspberry-pi-3-model-b-technical-specifications>

John. (2011, Feb 9). *Double-Diffused MOS (DMOS)*. Retrieved Mar 10, 2018, from circuitstoday: <http://www.circuitstoday.com/double-diffused-mos-dmos>

Louis, L. (2016). *WORKING PRINCIPLE OF ARDUINO AND USING IT*. India: IJCACS.

Maru, A. (2016). *Presentation on History of Microcontroller*.

Nedelkovski, D. (2016). Arduino and DS3231 Real Time Clock Tutorial. Retrieved Mar 10, 2018, from how to mechatronics: <https://howtomechatronics.com/tutorials/arduino/arduino-ds3231-real-time-clock-tutorial/>

Oy Equine Innovations Ltd. (2018, Apr 21). Retrieved from hevoskeksinnot: <http://www.hevoskeksinnot.fi/ota-yhteytta/>

Poddar, S. (2015, Nov 9). What is an actuator? Retrieved Apr 20, 2018, from quora: <https://www.quora.com/What-is-an-actuator>

Protected and Intelligent Power Switches. (2016, Mar 15). Retrieved Mar 24, 2018, from renesas: <https://www.renesas.com/en-in/products/intelligent-power-devices/high-side-driver-low-side-driver.html#productInfo>

Renesas. (2016). Protected and Intelligent Power Switches. Retrieved Mar 10, 2018, from renesas: <https://www.renesas.com/en-in/products/intelligent-power-devices/high-side-driver-low-side-driver.html#productInfo>

SFUPTOWNMAKER. (2013, JULY 8). I2C. Retrieved Mar 20, 2018, from learn sparkfun: https://learn.sparkfun.com/tutorials/i2c?_ga=2.3592880.125437095.1523299783-1098816997.1517222129

Smart High-Side Power Switch. (2013, Oct 15). Retrieved Mar 11, 2018, from Infineon Technologies AG: https://www.infineon.com/dgdl/Infineon-BTS410E2-DS-v01_01-en.pdf?fileId=db3a304331c8f8560131dcc9fc520e21

Solenoids. (2012). Retrieved Apr 1, 2018, from Johnson Electric: <http://www.johnsonelectric.com/en/resources-for-engineers/solenoids/~media/005A9A8652994C549B34B5C88C8284AC.ashx>

Storr, W. (2014, April). Linear Solenoid Actuator. Retrieved Apr 1, 2018, from electronics-tutorials: https://www.electronics-tutorials.ws/io/io_6.html

T.Ulrich, Karl; D.Eppinger, Steven. (2016). *PRODUCT DESIGN AND DEVELOPMENT 6TH EDITION*. New York: McGraw-Hill Education.

The glossary of solenoid. (n.d.). Retrieved Mar 15, 2018, from thesolenoidcompany:
<http://www.thesolenoidcompany.com/glossary>

ULN2803A Darlington Transistor Arrays. (2017, Feb). Retrieved from texas instrumentas:
<http://www.ti.com/lit/ds/symlink/uln2803a.pdf>

Viitala, A.-M. (2018). *Embedded Systems. [web log message]*. Retrieved Mar 15, 2018,
from tut.fi: <http://www.tut.fi/en/admissions/masters-studies-in-english/embedded-systems/index.htm>

ARDUINO CODE

```

// Date and time functions using a DS3231 RTC connected via I2C and
Wire lib
#include <Wire.h>
#include "RTClib.h"
#include <LiquidCrystal.h>

RTC_DS3231 rtc;
LiquidCrystal lcd(0,1,22,2,3,4);

int motor[] ={26,28,30,32};

int up = 8;      //Menu button
int sel = 9;     //Select button

int led_green =42;  //Green led
int led_red=44;    //Red led
int led_yellow=46; //Yellow led

//set the variable for alarm elements
int alarm_hour = 0;
int alarm_minute = 0;
int alarm_second =0;
int alarm_base = 120;
int alarm_new;

//set the variable for counter elements
int counter = 5;
int i;
int motor_counter;
int page_counter=1 ; //To move beetwen pages
int subpage_counter=0; // To subpage in select portion
int subpage2_counter=0; // To subpage in next time for portion
int portion_counter = 00; //To select portion options

//-----Storage debounce function-----//
boolean current_up = LOW;
boolean last_up=LOW;
boolean last_sel= LOW;
boolean current_sel = LOW;

//Custom return char
byte back[8] = {
  0b00100,
  0b01000,

```

```

0b11111,
0b01001,
0b00101,
0b00001,
0b00001,
0b11111
};
//Custom arrow char
byte arrow[8] = {
  0b01000,
  0b00100,
  0b00010,
  0b11111,
  0b00010,
  0b00100,
  0b01000,
  0b00000
};
//-----
void currentTime(){
  DateTime now = rtc.now();
  lcd.setCursor(0,0);
  lcd.print("Time:");
  lcd.setCursor(6,0);
  if (now.hour() <10){ lcd.print("0");}
  lcd.print(now.hour(), DEC );
  lcd.print(':');
  if (now.minute() <10){ lcd.print("0");}
  lcd.print(now.minute(), DEC );
  lcd.print(':');
  if (now.second() <10){lcd.print("0");}
  lcd.print(now.second(), DEC);
}
//-----
void setup () {
  lcd.begin(16,2);
  pinMode(motor[0], OUTPUT);
  pinMode(motor[1], OUTPUT);
  pinMode(motor[2], OUTPUT);
  pinMode(motor[3], OUTPUT);
  pinMode(led_green, OUTPUT);
  pinMode(led_red, OUTPUT);
  pinMode(led_yellow, OUTPUT);

  digitalWrite(motor[0], LOW);
  digitalWrite(motor[1], LOW);
  digitalWrite(motor[2], LOW);
  digitalWrite(motor[3], LOW);
}

```

```

digitalWrite(led_green, LOW);
digitalWrite(led_red, LOW);
digitalWrite(led_yellow, LOW);

delay(3000); // wait for console opening

if (! rtc.begin()) {
  lcd.println("Couldn't find RTC");
  while (1);
}

if (rtc.lostPower()) {
  lcd.println("RTC lost power");
  // following line sets the RTC to the date & time this sketch was
  compiled
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to
  set
  // January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
}
lcd.createChar(1, back);
lcd.createChar(2, arrow);
}
//---- De-bouncing function for all buttons----//
boolean debounce(boolean last, int pin)
{
  boolean current = digitalRead(pin);
  if (last != current)
  {
    delay(5);
    current = digitalRead(pin);
  }
  return current;
}
//-----
void loop () {

current_up = debounce(last_up, up);    //Debounce for Menu button
current_sel = debounce(last_sel, sel); //Debounce for Select button

//----Page counter function to move pages----//
//-----If menu button pushed-----
if (subpage_counter == 0){
  //Page Up
  if (last_up == LOW && current_up == HIGH){ //When up button is
  pressed
    lcd.clear();

```

```

    if(page_counter <3){          //Page counter never higher than 3(total
of pages)
        page_counter= page_counter +1; //Page up
    }
    else{
        page_counter= 1;
    }
}
last_up = current_up;
}

```

```
//----- Switch function to write and show what you want---//
```

```
switch (page_counter) {
```

```
case 1:{ //Design of home page 1
```

```
    DateTime now = rtc.now();
```

```
    currentTime();
```

```
    lcd.setCursor(0,1);
```

```
    lcd.print("Date:");
```

```
    lcd.setCursor(6,1);
```

```
    lcd.print(now.day(), DEC);
```

```
    lcd.print('/');
```

```
    lcd.print(now.month(), DEC);
```

```
    lcd.print('/');
```

```
    lcd.print(now.year(), DEC);
```

```
}
```

```
break;
```

```
case 2: { //Design of page 2
```

```
    lcd.setCursor(0,0);
```

```
    lcd.print("Select portion:");
```

```
    lcd.setCursor(2,1);
```

```
    if(portion_counter <10){ //To avoid "0" of number 10
```

```
        lcd.setCursor(2,1);
```

```
        lcd.print("0");
```

```
    }
```

```
    lcd.print(portion_counter);
```

```
    lcd.setCursor(15,1);
```

```
    lcd.write(byte(1)); //Return custom char
```

```
// Sub counter 2 control
```

```
if (last_sel== LOW && current_sel == HIGH){ //select button pressed
```

```
    if(subpage_counter <2){          // subpage counter never higher
than 2(total of items)
```

```
        subpage_counter ++;          //subcounter to move beetwen
```

```
submenu
```

```
    }
```

```

else{ //If subpage higher than 2 (total of items)
return to first item
    subpage_counter=1;
}
}
last_sel=current_sel; //Save last state of select button

//select portion number
if(subpage_counter==1){
    lcd.setCursor(14,1);
    lcd.print(" "); //Delete last arrow position
    lcd.setCursor(0,1);
    lcd.write(byte(2));

    //Control portion counter variable
    if (last_up== LOW && current_up == HIGH){ //Select button is
pushed
        if(portion_counter < 5){ //portion_count never higher
than 5 (max value)
            portion_counter ++ ;
        }
        else{
            portion_counter = 1; //If portion_count higher
than 5, return to 0
        }
    }
    last_up=current_up;
}
switch(portion_counter){
case 1:{
    lcd.setCursor(6,1);
    lcd.print("2h");
    lcd.setCursor(8,1);
    lcd.print(" ");
    counter =4;
    i = 0;
}
break;

case 2:{
    lcd.setCursor(6,1);
    lcd.print("2h 30min");
    counter = 4;
    i = 0;
}
break;

case 3:{

```

```

        lcd.setCursor(6,1);
        lcd.print("3h");
        lcd.setCursor(8,1);
        lcd.print("  ");
        counter =4;
        i = 0;
    }
    break;

    case 4:{
        lcd.setCursor(6,1);
        lcd.print("3h 30min");
        counter =4;
        i = 0;
    }
    break;

    case 5:{
        lcd.setCursor(6,1);
        lcd.print("4h");
        lcd.setCursor(8,1);
        lcd.print("  ");
        counter =4;
        i = 0;
    }
    break;

}
//Second item control (subpage_counter==2) back
if(subpage_counter==2){
    lcd.setCursor(0,1);
    lcd.print(" ");           //Delete last arrow position
    lcd.setCursor(14,1);     //Place the arrow
    lcd.write(byte(2));
    if (last_up== LOW && current_up == HIGH){
        subpage_counter=0;   //Exit submenu, up/down pages
        enabled
        lcd.setCursor(14,1);
        lcd.print(" ");
    }
    last_up=current_up;
}
DateTime now = rtc.now();
alarm_new = 90 + 30*portion_counter;
alarm_hour = now.hour() + (round(alarm_new/60));
if (alarm_hour >=24){ alarm_hour = alarm_hour-24;}
alarm_minute = now.minute() + (alarm_new-
(60*round(alarm_new/60)));

```

```

    if (alarm_minute >= 60){
        alarm_minute = alarm_minute-60;
        alarm_hour = alarm_hour +1;
    }
}
break;

case 3: { //Design of page 3

    //set the layout for Page 3
    DateTime now = rtc.now();
    currentTime();
    lcd.print(" ");
    lcd.print(counter);

    // Print the next time for next portion
    lcd.setCursor(0,1);
    lcd.print("Next:");
    lcd.setCursor(6,1);
    if (alarm_hour <10){ lcd.print("0");}
    lcd.print(alarm_hour);
    lcd.print(':');
    if (alarm_minute <10){ lcd.print("0");}
    lcd.print(alarm_minute);
    lcd.print(':');
    lcd.print("10");

    if (now.hour() == alarm_hour && now.minute() == alarm_minute &&
now.second() == 10){

        digitalWrite(motor[i],HIGH);
    }
    else if (now.hour() == alarm_hour && now.minute() == alarm_minute
&& now.second() == 15){
        digitalWrite(motor[i],LOW);
        i++;
        counter = counter -1;
        alarm_hour = now.hour() + (round(alarm_new/60));
        if (alarm_hour >=24){ alarm_hour = alarm_hour-24;}
        alarm_minute = now.minute() + (alarm_new-
(60*round(alarm_new/60)));
        if (alarm_minute >= 60){
            alarm_minute = alarm_minute-60;
            alarm_hour = alarm_hour +1;
        }
    }
}
if (counter ==0){
    lcd.clear();
}

```

```
    lcd.setCursor(0,0);
    lcd.print("The process is ");
    lcd.setCursor(0,1);
    lcd.print("finished");
    counter = 4;
    delay(4000);
    page_counter = 1;
    lcd.clear();
  }
}
break; // Break for case 3
} // end switch
//indicate for green led - free
if (counter == 5){
  digitalWrite(led_green, HIGH);
  digitalWrite(led_red, LOW);
  digitalWrite(led_yellow, LOW);
}
// indicate for red led - processing
if (counter < 5 && counter > 0){
  digitalWrite(led_green, LOW);
  digitalWrite(led_red, HIGH);
  digitalWrite(led_yellow, LOW);
}
//indicate for yellow led - empty
if (counter == 0){
  digitalWrite(led_green, HIGH);
  digitalWrite(led_red, LOW);
  digitalWrite(led_yellow, HIGH);
}
} // end Loop
```