INTRODUCTION

The scientific and technological revolution is becoming increasingly widespread and has achieved significant advancements in various fields, particularly in industrial automation. As a result, many industrial factories and even households are incorporating scientific principles to produce machinery and equipment. This integration helps improve productivity, reduce human resources, and enhance economic efficiency throughout the production process.

In light of this reality, our group has chosen the topic "Research and manufacture an automatic egg fertility detection system" to enhance student's awareness and knowledge, establish a solid foundation in their professional endeavors, and contribute to improving productivity for small-scale businesses with optimized production costs.

The "Research and manufacture an automatic egg fertility detection system" is designed to rapidly differentiate between fertilized and non-fertilized eggs after four days of incubation, at a rate of 100 eggs per 15 seconds with an accuracy of up to 90%. This system significantly reduces the need for manual labor, enhances work efficiency, and contributes to overall economic effectiveness, fostering a scientific and technological working and production environment.

Designing such a system requires a deep understanding of complex principles and demands high precision and aesthetics. Building upon this, our group conducted research and applied knowledge in mechanics, electronics, and programming to design the "Research and manufacture an automatic egg fertility detection system" that meets the requirements of the project topic.

COMMITMENT

Along with the rapid development of science and technology, machinery and equipment are increasingly produced to serve humans. Depending on the purpose of use and the needs of consumers, there are different ways to solve various problems. Therefore, there will be many ideas with different creativity.

With that spirit, our author team, consisting of Le Minh Nhat, Nguyen Ngoc Khoi and Le Duc Anh, carried out the topic "Research and manufacture an automatic egg fertility detection system" based on real surveys in small and medium-sized egg businesses.

In this topic, we fully declare that we have worked independently under the guidance and support of professors in the faculty of mechanical engineering.

We are committed that the data and research results presented in this thesis are entirely honest and not copied from any scientific report.

Danang, June 2023

Students perform

Le Minh Nhat Nguyen Ngoc Khoi

ACKNOWLEDGMENTS

The topic " Research and manufacture an automatic egg fertility detection system" is the content that our group has chosen to study and carry out our graduation thesis after 4 years of studying in the "Mechatronics Engineering" program at the Danang University of Science and Technology

To complete the thesis, first of all, we would like to express our sincere thanks to Dr. Vo Nhu Thanh for his comments and guidance during the process of completing this thesis. At the same time, we would like to thank the faculty members and the university who have conveyed to us a lot of useful knowledge during our learning process, which we can apply to our current thesis and future work.

During the process of completing the graduation thesis, due to limited practical experience, there may be some errors in this thesis and the report. We hope to receive feedback and contributions from our teachers to learn from our mistakes, which will be a solid foundation for us to stand firm on our future path.

We would like to express our sincere thanks to everyone!

TABLE OF CONTENTS

LIST OF TABLES, FIGURES

LIST OF SYMBOLS, ABBREVIATIONS

CHAPTER 1: INTRODUCTION OF THE RESEARCH SITUATION IN THE FIELD OF THE TOPIC

1.1. Overview of research situation in the field of the topic:

1.1.1. Field, related categories and international documents:

Currently, there are several successful companies worldwide that have designed and manufactured automatic egg classification models that can determine whether eggs are hatching or not, such as Sanovo, Viscon, etc., but they can only detect and classify old eggs (from 12 days old). In addition, there have been some studies on classifying eggs from 0 to 4 days old, but complete classification systems have not yet been developed. [1]

1.1.2. Field, related categories and domestic documents:

Currently, there is no research or similar machinery in Vietnam.

1.2. Reason for choosing the topic:

Poultry egg incubation models have been around in Vietnam for quite some time, but most are manually done by hand. Only in recent years have scientific and technical methods been applied to automate the incubation process. In a poultry egg incubation model, an important process that cannot be overlooked is the egg classification process, which determines whether eggs can hatch. This is a very important process because if not checked and removed from the incubation process, eggs that cannot hatch (infertile eggs) will decay and produce harmful bacteria in the incubator, affecting the eggs being incubated. In addition, infertile eggs, when classified early, can be sold on the market as food, providing additional income for businesses and household businesses, avoiding waste and environmental pollution. [1]

Most small and medium-sized enterprises in Vietnam are still manually operating this classification process by using labor to classify eggs with the naked eye. For large quantities of eggs, this is time-consuming and can lead to errors. Recognizing this difficulty, our student group has applied image processing techniques to automate this process, aiming to increase labor productivity and reduce costs in operation, helping businesses save hhumanresources and improve production efficiency, contributing to the development of the enterprise. [1]

1.3. Research objectives

The objective of the project is to design and develop an egg-sorting machine with the following specifications:

- Machine size (length x width x height): 1550x60x70 (mm)

- Components used: Stepper motor, DC motor, Camera, Arduino and stepper motor control circuit, Shaped aluminum, box steel. Proximity sensor 3W LED system

- Sorting speed: 10-15 eggs per sorting cycle, with a sorting time of 0.1 seconds per egg. The expected time for each recognition and sorting cycle is 15 seconds (100 eggs per tray for once)

- Sorting accuracy: 90%.

1.4. Approach

- Study the egg classification techniques from scientifically researched and proven methods in depth.

- Study the egg classification techniques from the practical experience of Vietnamese farmers.

- Select the most suitable method, and choose the appropriate technology to implement according to the chosen method.

1.5. Reseach methods

The research process of the topic is carried out in the following steps:

- *Step 1: Determine the research subject:*

This is the first step of the team's research process. At this stage, the team needs to identify the research subject and objectives of the research. These content will be presented specifically in part 6.

- *Step 2: Survey the actual conditions at the household and small and medium-sized production facilities:*

After determining the research subject, the next step is for the team to investigate how to identify and classify the subject from production facilities and household businesses,thereby drawing conclusions and classification procedures.

- *Step 3: Collect samples and analyze data:*

After the classification process, the team proceeds to collect samples and analyze the collected data

- *Step 4: Choose suitable technology and structure to implement:*

  After analyzing the collected data, the team choose suitable technology and structure to design and implement the product

- *Step 5: Implement the design and construction of the model:*

When the design options have been selected, the team proceeds to implement the design and construction of the model for application in reality.

- *Step 6: Improve the model and evaluate the results:*

  After completion, the team tests the model in household businesses and evaluates the results of the system.

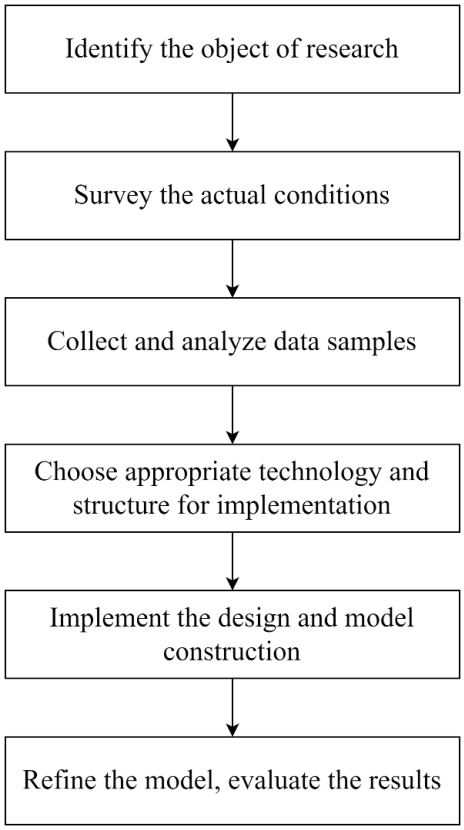


Figure 1.1: Research flow char

1.6. The object of research

- A 4-day-old chicken egg has been incubated.
- Machine learning technology.
- Motor
- Camera
- Image processing technology
- Conveyor system.

CHAPTER 2: OVERVIEW OF THE METHODS OF CANDLING CHICKEN EGGS IN IMAGE PROCESSING

2.1. Researching and selecting methods for candling chicken eggs.

- Candling eggs is the process of using a strong light source from a flashlight, electric bulb, etc. to shine through the eggshell in a suitable way to observe the inside of the egg such as the embryo, blood vessels, color inside the egg, etc.

- Why do we need to egg candling? While the process of selecting eggs before putting them into the incubator helps us eliminate non-standard eggs from the very beginning, egg candling also plays an equally important role. Because during incubation, some low-quality eggs that are difficult to detect and remove at the beginning may become rotten, undeveloped, or not have embryos. A typical example is the case of unfertilized eggs, which will not have embryos to develop into chicks. In other words, egg candling helps us reselect eggs, increasing efficiency, and saving space, and effort when incubating eggs. In addition, non-standard eggs that are eliminated early can still be sold to compensate for costs.

- How to candling chicken eggs: candling eggs using an LED light, shine the light onto the larger end of the egg from below.

Figure 2.1: The light onto the larger end of the egg from below.

- The student group has chosen the option of using a LED light to shine from the bottom up as mentioned above, based on the requirement of being able to inspect 5 eggs at once and 100 eggs within 10 seconds. To address the issue of light leaking out during inspection,

the student group designed a light-collecting cover to limit the light leaking out and to focus the light onto the eggs.

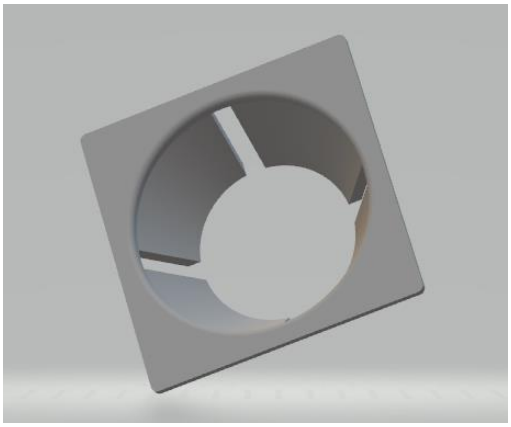


*Figure 2.2: **A light-collecting cover***

2.2. The scientific basis of the method of candling eggs.

- A fertilized chicken egg means the egg has been fertilized by a rooster and developed into an embryo for a higher hatching rate.

- Candling chicken eggs is the process of using different sources of strong light, such as a flashlight or electric bulb, to shine on the egg, in order to observe the embryo, blood vessels, and internal color of the egg.

Figure 2.3: Classification of chicken eggs with the naked eye.

(Source: mayapcne.com)

- During the incubation process, in the beginning, there may be many poor quality eggs that are difficult to detect, such as eggs without embryos or undeveloped embryos. If these eggs are incubated, they will spoil and have to be discarded, which is a waste.

- Using candling method helps us to select the best eggs to increase the efficiency of incubation, save space, and maximize the incubator's capacity.

- In addition, candling also helps us to remove eggs that do not meet standards. If we can select and remove non-fertile eggs early, the remaining viable eggs can still be sold to offset the cost of breeding.

Figure 2.4: The development process of a chicken egg.

(Source: mactech.com.vn)

- Based on Figure 2.4 above, we can see that a fertilized egg will develop blood vessels after 4 to 8 days of incubation. For the research project, our group has chosen 4-day-old eggs (eggs that have been incubated for 4 days) as the target for identification. Below is the image of the actual eggs taken by our group for differentiation.

Figure 2.5: Image of the sampled eggs.

Figure 2.6: Egg with the undeveloped embryo.

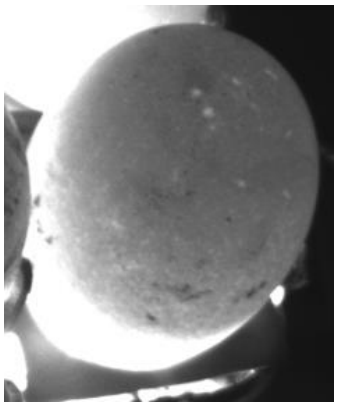


Figure 2.7: Egg with a developing embryo.

- From the above examples, we can see that eggs without an embryo will appear brighter when exposed to strong light. Therefore, the current task is to identify the image of the egg with the least dark area, which indicates the absence of an embryo, and those eggs will be rejected.

2.3. Basic image processing techniques [2]

What is image processing?

- Digital image processing is a field of applied computer science. Graphics data processing refers to artificial images, which are considered as data structures and created by programs. Digital image processing includes methods and techniques to transform, transmit, or encode natural images. The purpose of image processing is to transform and enhance images, as well as to automatically recognize and evaluate the contents of images. In this topic, we will use the second method which is automatic recognition and evaluation of image contents. [2]

- Image recognition is a process closely related to describing objects that people want to specify. The recognition process usually follows the process of extracting the main

characteristics of the object. There are two types of object descriptions: parameterized descriptions and structure-based recognition descriptions. Recognizing and evaluating the contents of an image is analyzing an image into meaningful parts to distinguish this object from another object. Based on that, we can describe the structure of the original image. Basic recognition methods can be listed, such as object boundary recognition, edge detection, image segmentation, etc. In practice, people have successfully applied this recognition technique to many different objects such as fingerprint recognition, letter recognition, number recognition, accent recognition, print or typewritten character recognition in text.[2]

- Next, we will consider the necessary steps in the image processing process. First, natural images from the outside world are captured through acquisition devices (such as cameras, cameras). Previously, images captured through cameras were analog images (type of CCIR-style tube camera). Recently, with the development of technology, color or black-and-white images are taken from the camera, and then it is directly converted into digital images to facilitate further processing. On the other hand, images can be scanned from paper documents directly using image scanners. The figure below describes the basic steps in image processing.

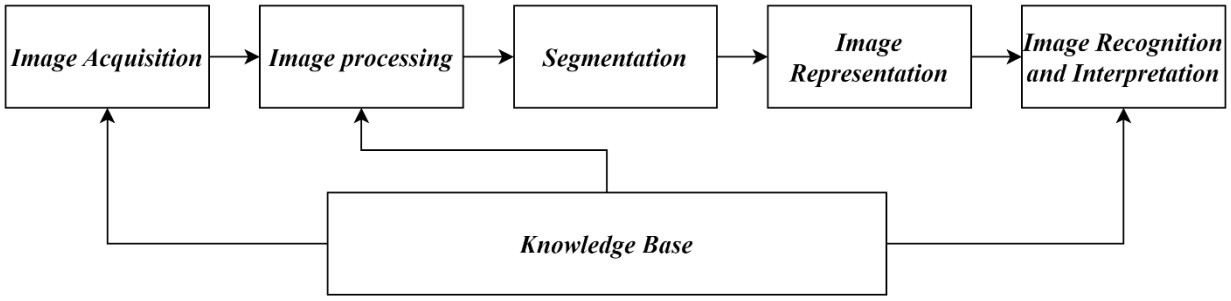


Figure 2.8: The basic steps in image processing.

(Source: tinhoccoban.net)

This diagram includes the following parts:

2.3.1. Image Acquisition:

- Images can be captured through color or black and white cameras. Typically, images captured through cameras are analog images (using CCIR standard cameras with a frequency of 1/25, each image having 25 lines), but there are also digital cameras (such as CCD - Charge Coupled Device), which use photodiodes to create brightness intensity at each pixel.

- Cameras commonly used are line scan cameras, which produce two-dimensional images. The quality of the captured image depends on the receiving equipment and the environment.

2.3.2. Image processing

After being captured by the receiver, the image may have low contrast noise, so it needs to be processed by a pre-processing unit to improve the quality. The main function of the pre-processing unit is filter noise, and enhance contrast to make the image clearer and sharper.

2.3.3. Segmentation

Image segmentation is the process of dividing an input image into component regions for analysis and recognition purposes. For example, to recognize text (or barcode) on an envelope for parcel classification purposes, the address or recipient name lines must be segmented into separate words, characters, digits (or bars) for recognition. This is the most complex and challenging part of image processing and can introduce errors, leading to loss of image accuracy. The quality of image recognition results heavily depends on stage.

2.3.4. Image Representation

This is the part after segmentation that contains the pixels of the segmented image plus the linking codes in neighboring regions. Transforming this data into a suitable form is necessary for further processing by computers. The selection of properties to represent the image is called feature extraction, which involves extracting the characteristics of the image as quantitative information or as a basis for distinguishing this object from others within the received image. For example, in character recognition on an envelope, we describe the features of each character to distinguish it from other characters

2.3.5. Image Recognition and Interpretation

Image recognition is the process of identifying an image. This process is often achieved by comparing it to a learned (or stored) standard template. Inference is a deduction based on this recognition. For example, a certain type of digit and a horizontal line on an envelope can be inferred as a phone number. There are many different ways to classify images. According to the theory of recognition, mathematical models of images are divided into two types of basic image recognition:

- Parameter-based recognition
- Structure-based recognition

Some common objects of recognition used in science and technology are character recognition (printed, handwritten, electronic signatures), text recognition, fingerprint recognition, barcode recognition, and face recognition.

2.3.6. Knowledge Base

As mentioned above, an image is a complex object in terms of lines, brightness, pixel density, and the environment in which it is captured, leading to noise. In many image processing and analysis steps, in addition to simplifying mathematical methods for convenient processing, people want to mimic the image reception and processing process in the way of humans. In those processing steps, many methods are now processed using human-like intelligence methods. Therefore, knowledge bases are being leveraged here.

2.3.7. Description

After digitizing, the image will be stored in memory or passed on to the next stages for analysis. If storing images directly from raw images, it requires an extremely large memory capacity and is not efficient in terms of application and technology. Typically, images are referred to as image features such as edge images, area images.

2.4. Introduction to color space.

The human eye can distinguish a few dozen colors, but can only perceive thousands of colors. The three attributes of a color are:

H (Hue): color

S (Saturation): saturation

V (Value): value

It is abbreviated as HSV to achieve the desired image.

In image processing and graphics, a color model is a technical index of a 3D color coordinate system with a set of small visible component colors in a characteristic color gamut. For example, the RGB (Red, Green, Blue) color model is a unit set of component colors arranged in a cubic shape of the coordinate axis system.

The purpose of color models is to allow for appropriate technical specifications of certain color types to match the colors of certain characteristic color gamuts. We can see in this color model, the color space is a smaller set of the visible color space, so a color model cannot be used to specify all visible colors. Here, we will consider some of the most commonly used color models.

- RGB (Red, Green, Blue) color model.

Red, green, and blue (RGB) are the most commonly used colors. The primary RGB colors are added to other primary colors, which contribute their individual qualities to create a resulting color. The set of small color components is arranged in a unit cubic block. The main diagonal of the unit cube corresponds to the balanced amounts of each primary color, and the shades of gray from black (0,0,0) to white (1,1,1).

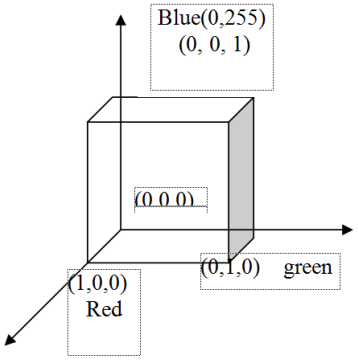


Figure 2.9: RGB color model.

(Source: opencv.org)

- HSV(Hue, Saturation, Value) color model.

The HSV (Hue, Saturation, Value) color model is a color model used in graphics and image processing. In this color model, color is represented by the Hue value, Saturation level, and Value (or brightness) of the color. The Hue value is an index that specifies the basic color of a pixel, Saturation indicates the level of color intensity, and Value represents the brightness of the color.

The advantage of the HSV color model over the RGB color model is that it allows users to adjust color properties independently, while the RGB color model requires changes in the values of the three color components: red, green, and blue.

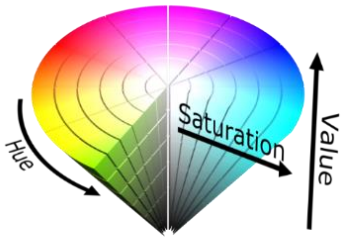


Figure 2.10: HSV color model.

(Source: khoahoc.tv)

2.5. Picture Element

2.5.1. Pixel

A pixel is the smallest basic unit that makes up a digital image. The address of a pixel can be seen as a coordinate (x,y). A digital image can be created by taking a photograph or using another graphic design method, consisting of thousands or millions of individual pixels. The more pixels an image has, the more detailed it is. One million pixels is equivalent to 1 megapixel.

2.5.2. Digital Image

Digital image is a finite set of pixels with suitable gray level used to describe an image close to the real image. The number of pixels determines the resolution of the image. The higher the resolution of an image, the more clearly it shows the details of the picture, making it more realistic and sharp. An image is a two-dimensional signal, defined by a mathematical function f(x, y), where x and y are two coordinates along the horizontal and vertical axes. The values of f(x, y) at any point provide the pixel values at point of an image.

2.5.3. Image Classification

The grayscale of a pixel is the brightness intensity, assigned a value at that point. The usual grayscale levels are: 16, 32, 64, 128, 256. The most commonly used level is 256, which means 1 byte is used to represent each grayscale level. Where:

- Binary image: An image with 2 levels of black and white, with only 2 values 0 and 1, and only uses 1 bit of data per pixel.

- Black and white image: An image with two colors black and white (without any other color), with grayscale levels that may vary at different points.

- Color image: A combination of 3 primary colors to create a vivid color world. 3 bytes are used to describe the color level, which means there are about 16.7 million color levels.

2.5.4. Relationship between pixels

Pixel neighborhood: it can be described as the "neighbors" of a pixel. There are 2 basic types of neighborhoods: 4-neighborhood and 8-neighborhood.

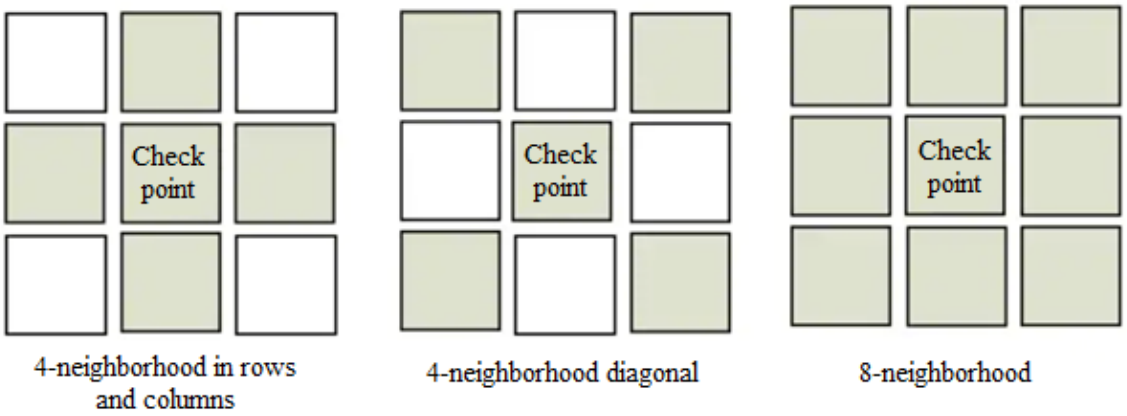


Figure 2.11*: *4-8 neighborhood

(Source: globlib4u.wordpress.com)

The 4 neighboring pixels along the row and column with coordinates (x+1, y), (x-1, y), (x, y+1), (x, y-1) are denoted as the N4(p) set. The 4 neighboring pixels along the diagonal with coordinates (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1) are denoted as the ND(p) set. The 8 neighboring pixels, denoted as the N8(p) set, are the union of these two sets.

$$N8(p) = N4(p) + ND(p)$$

2.5.5. Noise Filter

Images acquired are often subject to noise, so it is necessary to remove noise. Spatial operators used in image enhancement techniques are classified according to their purpose: smoothing noise, edge enhancement. To smooth noise or separate noise, linear filters (average filter, low-pass filter) or nonlinear filters (median filter, pseudo-median filter, homomorphic filter) are used. Based on the nature of noise (usually corresponding to high frequency) and the theory of filtering, a filter only allows signals of a certain frequency to pass through. To filter noise, a low-pass filter (in terms of spatial frequency) or a linear combination is usually used to flatten (average filter) the signal. To enhance edges (with high frequency), high-pass filters, Laplacian filters are used. The noise filtering method is divided into two types: linear filtering and nonlinear filtering.

2.5.6. Edge detection method

Edge is one of the issues we need to pay attention to in image processing because in the segmentation stage, it mainly relies on the edges.

Figure 2.12*: *Edge detection image [11]

- Edge point: an image point is considered as an edge point if there is a rapid or abrupt change in the gray level (or color). For example, in a binary image, a black point is called an edge point if at least one neighboring point is white. The boundary (or contour) is a set of adjacent edge points forming an edge or a contour.

- The significance of edges in image processing: the first significance of edges is that they are a typical local feature in image analysis and recognition. Secondly, edges are used to separate different gray (or color) regions. Conversely, image regions can also be used to find separating edges. The importance of edges can be clearly seen in the following example: when an artist wants to draw a portrait, they only need to draw a few quick lines without drawing the whole figure in detail.

- Therefore, an ideal edge detection is to detect all edges of an object. The mathematical definition of edges above is the basis for edge detection techniques. The important thing is that the variation between image points is usually small while the variation of the brightness of edge points is usually large when passing through an edge. Based on this foundation, people often use 2 edge detection methods as follows: First-order derivative edge detection: There are 2 basic methods: one is to create a gradient of two directions and orthogonal in the image, and the other is to use a set of directional derivatives. Second-order derivative edge detection: It is performed on some types of second-order differential equations to make edges appear. There are two types of second-order derivative methods that have been studied: the Laplace method and direct derivative.

- Canny edge detector: This detection method is widely used because it has many advantages over other methods. The steps of implementation are as follows: First, the image is smoothed using a Gaussian filter. Then, the local gradient of the edge magnitude and direction is computed. Next, the pixel with the maximum edge magnitude is found using

non-maximal suppression. The resulting peak pixels (found in step 2) are divided into two thresholds T1 and T2, where T1 < T2. Pixels with a magnitude value greater than T2 are called "strong" and pixels with values between T1 and T2 are called "weak". Finally, weak pixels that are connected to strong pixels with 8-connectivity are considered edges.

2.5.7. Image segmentation

- Image segmentation is a crucial step in image processing. This stage aims to analyze the image into regions with similar properties based on boundaries or connected regions. The criteria for determining connected regions may be the same level of gray, color, or texture.

- The image segmentation process aims to separate the object of interest from the rest of the image or divide the objects in the image into distinct entities. Therefore, the image segmentation process reduces the amount of information in the image and only retains the necessary information for the application. Thus, image segmentation is the process of removing unwanted objects from the image. There are many different methods of image segmentation. Among them, the process of segmenting the image into objects and backgrounds using a gray value threshold is the simplest method. At this point, the points below the gray value threshold belong to the background, while the points above the gray value threshold belong to the object. This method of image segmentation is very effective for binary images, printed text, or graphics... Based on the physical characteristics of image regions, region segmentation techniques can be divided into three types:

- Local techniques: based on the local characteristics of the pixels and their neighborhoods.
- Global techniques: segmenting an image based on overall information such as using the gray-level histogram.
- Splitting, merging, and developing techniques: based on the concept of similarity in shape and homogeneity. Two regions can be merged together and adjacent to each other. Inhomogeneous regions can be split into smaller regions. A region can be developed by connecting pixels that are homogeneous with each other.

2.5.8. Image formats

- Images obtained after digitization are usually stored for further processing or transmission. In the process of image processing, there exist many different formats, from black and white images like IMG format to grayscale images and color images (BMF, JPEG, GIF).

2.6. Image Processing Software

Nowadays, image processing is widely taught in universities and applied in practice, such as photo editing software or face recognition. Therefore, there are many tools for us to program applications in practice, such as Matlab, Python or Opencv library, ...

2.6.1. Python Language

- Introduction to Python Language:

Python is a widely-used programming language nowadays, ranging from small scripting projects to large-scale projects. The language supports a variety of applications and software, such as desktop programs, servers, web applications, etc. In addition, Python is also the preferred language for building artificial intelligence programs, including machine learning. Initially, Python was developed to run on Unix, but later it was able to run on every operating system, from MS-DOS to Mac OS, OS/2, Windows, Linux, and other Unix-based operating systems. Python was created by Guido van Rossum in 1990. Python was developed as an open-source project, managed by the non-profit organization Python Software Foundation. Although the development of Python has the contribution of many individuals, Guido van Rossum is still the primary author of Python. He plays a key role in deciding the direction of Python's development. [10]

- Key features of Python:

Python is a language with a simple syntax, using a small number of keywords, making it easy for beginners to learn. Python has a straightforward source code, even for those who are not familiar with the language, it is possible to understand the meaning of each line of code. Python has many applications on various platforms, including Windows, Mac OSX, and Linux.

2.6.2. OPENCV Library

- Introduction:

OpenCV (Open Source Computer Vision Library) is an open-source library that is free for anyone to use, making it accessible for beginners in machine learning. OpenCV is applied in many fields such as computer vision, image processing, and machine learning. The library is programmed in high-level languages such as C++, C, Python, or Java and supports platforms such as Windows, Linux, Mac OS, iOS, and Android. OpenCV was created by Gary Bradsky at Intel in 1999 and released in 2000. OpenCV has many applications such as image recognition, image processing, image/video restoration, augmented reality, etc. In this topic, the OpenCV library is run on the Python language, and

OpenCV is used as the main library for processing input images and then performing image recognition. [10]

- Characteristics:

OpenCV is an open-source library, so it uses algorithms freely, and we can also contribute additional algorithms to help the library become more and more developed.

Features of the OpenCV library:

- For images, we can read, save, or write them.
- For videos, it is similar to images with the ability to read and write them
- Image processing can filter noise for images or convert images.
- Perform recognition of shape features in the image.
- Detect predefined objects such as faces, eyes, cars in video, images.
- Analyze video, estimate its motion, subtract the background, and track objects in the video.

2.7. Image processing method using area-based recognition.

- With the given requirement of identifying eggs with the least darkened areas as the ones to be eliminated, our group has employed the idea of calculating the area of the darkest regions on the eggs to determine if they should be discarded. Once the area of the darkened region is determined, it is compared against a fixed threshold. If the area exceeds the threshold, the egg is considered eliminated, and vice versa.

- To calculate the area, we first need to convert the image into a binary image with only two colors, white and black. To achieve this, we initially convert the image to the HSV color space. Then, we adjust the three parameters sequentially: H (Hue) for color, S (Saturation) for intensity, and V (Value) for brightness, in order to obtain the desired image.[11]

```
imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lower = np.array([h_min, s_min, v_min])
upper = np.array([h_max, s_max, v_max])
mask = cv2.inRange(imgHSV, lower, upper)
```

- The code snippet above performs the conversion of an image to the HSV color space and assigns the values to a mask.

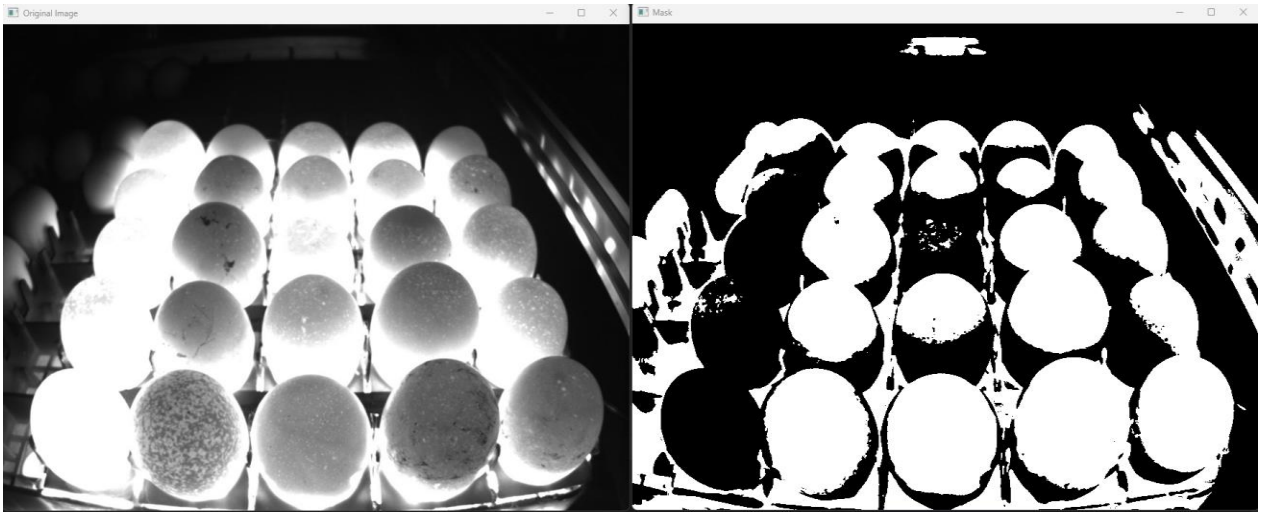


Figure 2.14: The image after using the HSV color filter

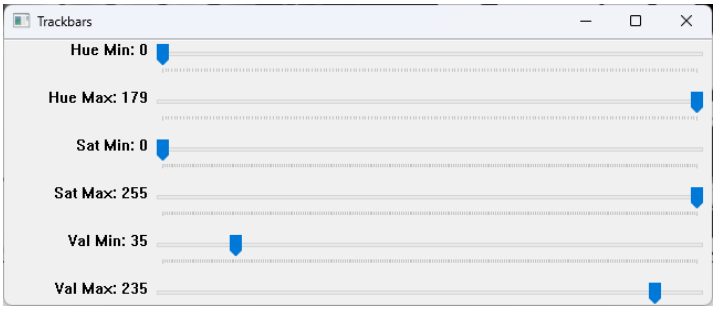


Figure 2.15: The specified parameters H, S, V

- After applying the color filter, we will observe that the darkened areas on the eggs have turned white, while the brighter areas on the eggs appear black. At this point, we proceed to draw contours around the white regions on the eggs to calculate the area enclosed by those contours. To accomplish this, we use the function findContours as follows:

```
def getContours(img): #Calculates the area of the largest contour in an image.
    contours, hierarchy = cv2.findContours(img, 1, 2)
    if len(contours) == 0:
        area = 0
    else:
        c = max(contours, key=cv2.contourArea)
        area = cv2.contourArea(c)
    return area
```

- Use the function cv2.contourArea to calculate the area of the recently drawn contour.

- From the calculated area, we will compare it with the minimum area threshold (2000), which is the minimum area a fertilized egg can have. Based on this, we can conclude whether the egg has been fertilized or not.

CHAPTER 3: COMPUTATION AND MODEL DESIGN

3.1. Caculating design parameters for conveyors

3.1.1. Design conveyor Systems

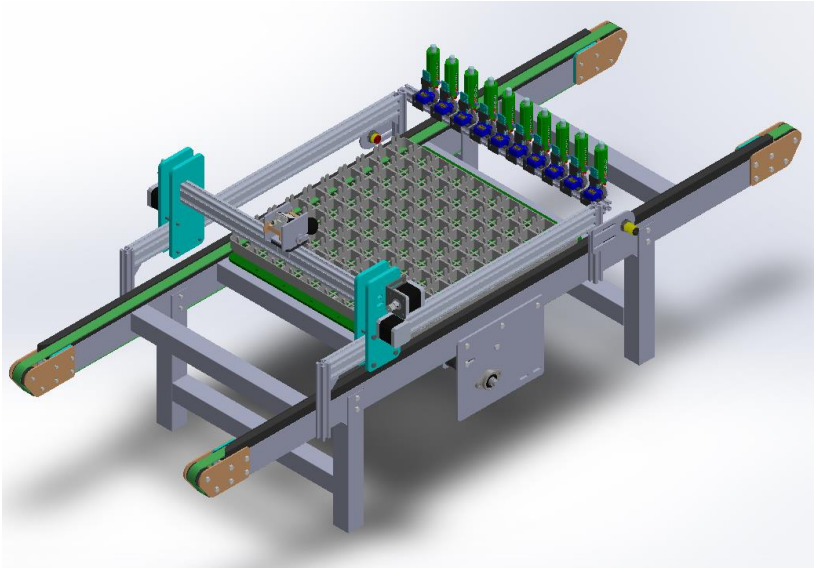


Figure 4.1: System model in SolidWork software

3.1.2. Caculating motor power selection

Power is always the first factor to consider when calculating and selecting a motor for a conveyor. This power is the aggregate of the following fundamental components:

- The power required to run unloaded.
- The power required to move an object on a conveyor belt.
- The power required to overcome frictional force.

Conveyor belt size:

- Length: 1.5m
- Width: 0.48m
- Height: 0.32m

The combination of the first two components is the required power to operate the conveyor.

Determine the power of the motor: [7]

$$N_{ct} = \frac{N}{\eta}$$

In which:

N_{ct}: Required power for the motor (W).

N: Power output on the conveyor belt (W).

η: The overall performance of the conveyor belt.

- Identify the power N:

$$N = F.v\ (W)$$

In which:

F: The pulling force of the conveyor belt (N)

v: The maximum speed of the conveyor belt (m/s).

- The pulling force of the conveyor belt.

$$F = m.k.g.f = 20.5.10.0{,}6 = 600(N)$$

In which:

m : Maximum weight on the conveyor belt.

k : Safety factor

g : Gravitational acceleration

f : The coefficient of friction between the product and the conveyor belt.

$$\Longrightarrow \quad N = F.v = 600.\frac{1}{20} = 30\ (W)$$

- The overall performance of the system:

Get data from table (2-1) page 27 [Document Thiết kế chi tiết máy, Nguyễn Trọng Hiệp, Nguyễn Văn Lẫm, NXB giáo dục, 1999] to calculation η [1]

$$\eta = \eta_d.\eta_{ol}^2 = 0{,}95.0{,}995^2 \approx 0{,}94$$

In which:

η_d : Belt transmission efficiency. ($\eta_d = 0{,}95$)

η_{ol} : Rolling efficiency ($\eta_{ol} = 0{,}995$)

Required power of the engine:

$$N_{ct} = \frac{30}{0.94} = 31{,}91\ (W)\ [1]$$

According to [1], we select an appropriate motor for the system: Choose a DC motor with a power rating of N = 40 (W).

Choose the Planetary GX43775 – 12V gear motor.

Figure 3.2: Planetary GX43775 - 12V Gearbox Motor

(Source:https://shopee.vn/product/414860258/23412880446?d_id=42eef&utm_content=NWz63JoFPA5HinaQFtN1ThyFEP1)

Technical characteristics:

- Idling Speed: 100 rpm
- Rated speed : 76 rpm
- Rated moment: 30 kg.cm
- The length of the gearbox: 43 mm
- Axis dimension: 8mm trục D
- Axis D length: 25 mm
- The weight of the gearbox and engine: 845g
- Operating current: 3A
- Starting current: Equal to 2 to 3 times working current

3.1.3. Calculating and designing the camera mounting components and transmission

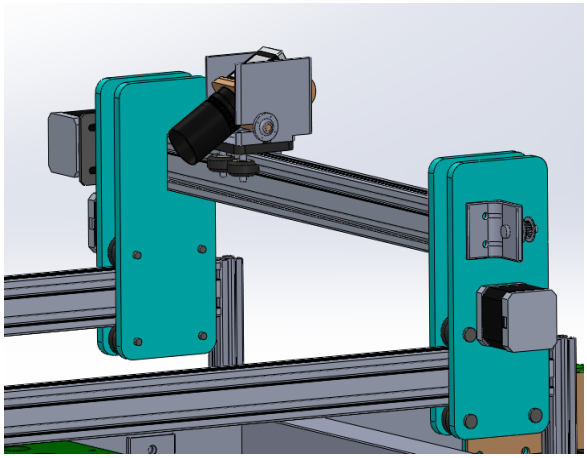


Figure 3.3: Camera transmission unit

The camera transmission module is driven by two 17HS8401 stepper motors, moving on a 20x40 shaped aluminum frame.

The transmission component ensures robust stiffness, high durability, and smooth operation when using a toothed belt drive for motion, ensuring high precision control.

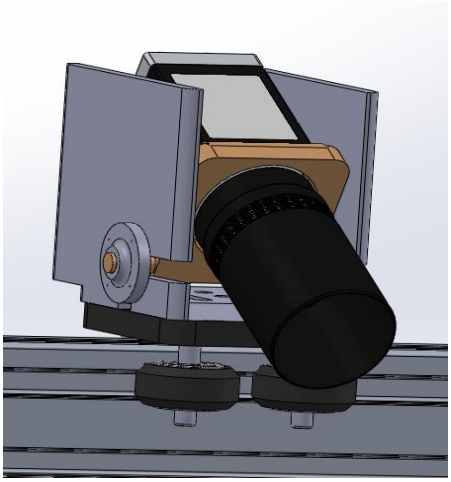


Figure 3.4: Camera mounting bracket

The camera module is driven by a belt and a Nema 17 stepper motor, ensuring high precision during operation and providing smooth and flexible movement.

The camera mounting component is designed to allow for adjustable camera angles, making it convenient for system adjustments. Additionally, the component can be easily detached or assembled using screw and bolt connections.

3.2. Electrical Circuit Calculation and Design

3.2.1. Introduction to Arduino Mega 2560 R3 Microcontroller Kit

The Arduino ATmega2560 is different from all previous microcontrollers because it does not use an FTDI chip to handle signal conversion from USB. Instead, it uses the ATmega16U2 for programming, acting as a signal converter from USB. Additionally, the Arduino ATmega2560 is similar to the Arduino Uno R3 in its basic functionality, with the main differences being the number of pins and enhanced features. Therefore, you can still program this microcontroller using the programming software for Arduino Uno R3.

The Arduino ATmega2560 has 54 digital pins, including 15 pins capable of PWM (Pulse Width Modulation), 16 analog pins, 4 UART (Universal Asynchronous Receiver-Transmitter) pins, a 16MHz quartz crystal, a USB port, a power jack, an ICSP (In-Circuit Serial Programming) header, and a reset button. The Arduino ATmega2560 can be used

with most shields designed for the Uno and microcontrollers like Duemilanove or Diecimila.

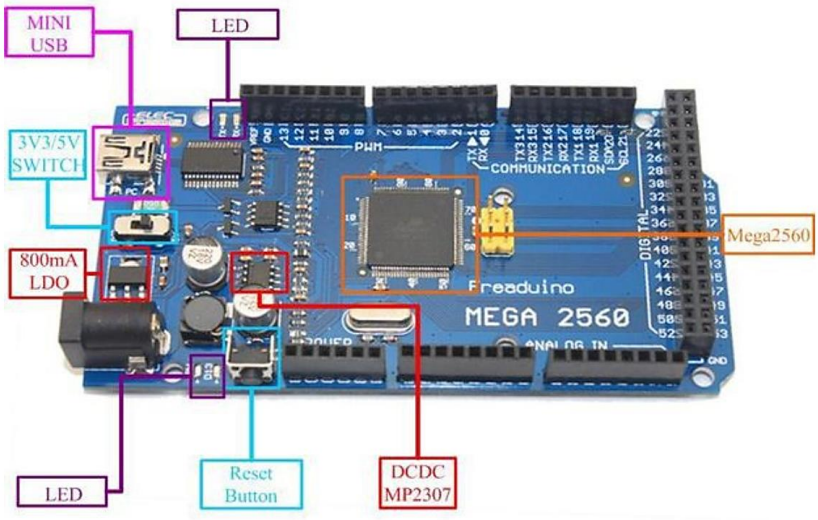


Figure 3.5: Arduino Mega 2560 Board

Technical specifications:

Control	ATmega2560
Operating voltage	5V
Operating frequency	16MHz
Recommended input voltage	(7÷12)V – DC
Limit input voltage	(6÷20)V – DC
Number of Digital I/O pins	54 (15 pinout PWM)
Analog pins	16
Maximum Output Current (5V)	20 mA
Maximum output current (3.3V)	50 mA
Flash memory	256 KB (Atmega2560) with 8KB use by bootloader
SRAM	8 KB (Atmega238)
EEPROM	4 KB (Atmega238)
Size	53.3mm × 101.52mm

Table 3.1: Technical Specifications of Arduino Mega 2560.

(Source: MEGA2560 Datasheet)

3.2.2. Arduino Atmega2560 pinout and function pinout

- Pinout:

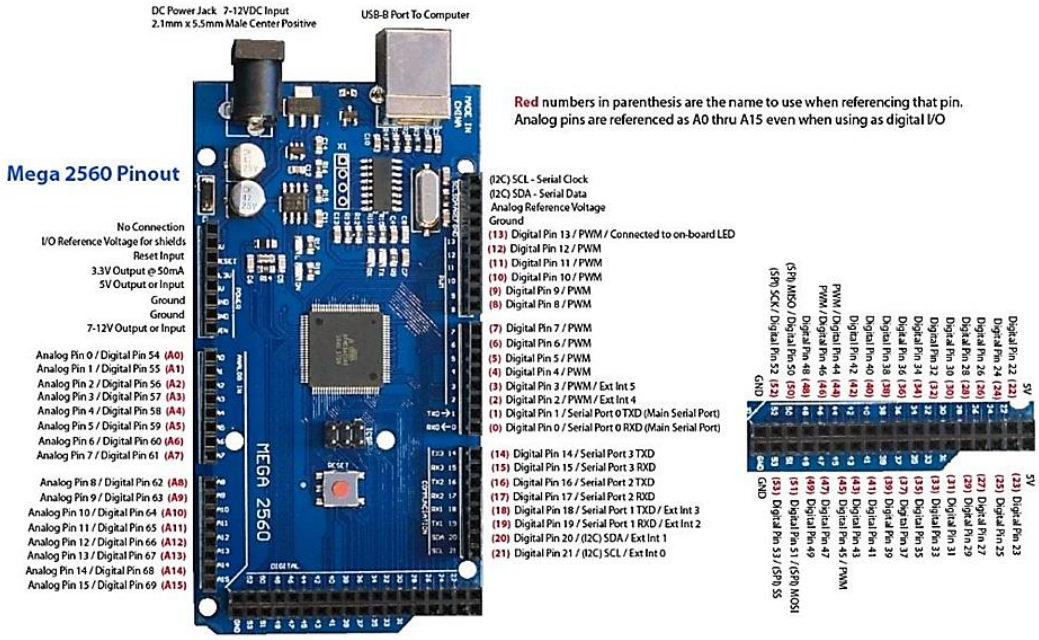


Figure 3.6: Arduino Mega 2560 Pinout Diagram

(Source: MEGA2560 Datasheet)

- Functions of the legs

a) Source

The ATmega2560 is powered through a USB connection or an external power source. The power source is automatically selected. The external power can be supplied from a DC converter or a battery. The converter is plugged into the household power grid. Two wires from the battery's terminals can be connected to the GND and Vin pins in the POWER area.

The legs in POWER include:

- Vin: Power is supplied to the circuit when using external power sources. Power can be supplied through the plug pins.

- 5V: This leg defaults to providing a 5V output obtained through resistors from the circuit. The circuit can support power sources from a DC jack (7-12V), a USB plug (5V), or through the Vin pin (7-12V).

- 3.3V: The 3.3V supply is provided on the circuit. The maximum current is 50 mA.

- GND: Grounding wire.

- IOREF: These pins on the reference voltage circuit interface with operations from the microcontroller. The shields can read the voltage value from the IOREF pin and select a power source or switch the output voltage value to either 5V or 3.3V.

b) Input/Output port

- With each of the 54 digital pins on the Atmega2560, they can be used as either inputs or outputs by utilizing pinMode, digitalWrite, and digitalRead(). Each of these pins operates at a voltage level of 5 volts. Each pin can supply or receive a maximum of 20 mA and has an internal pull-up resistor (default disconnected) of 20-50 kΩ. The maximum current of 40 mA cannot be exceeded to ensure any damage to the microcontroller. Additionally, some pins have different functions, such as:

- Serial Ports: 0(RX) and 1(TX); Serial1: 19(RX) and TX(18); Serial2: 17(RX) and 16(TX); Serial3: 15(RX) and 14(TX). These are used for transmitting and receiving serial data (UART). These pins are connected to the corresponding pins of the Atmega16U2 USB-to-TTL and the connected chip

- Interrupt pins (2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2)): These pins can be coded to trigger an interrupt on a low value, a rising or falling edge, or a change in value. Use attachInterrupt() in the library for more details.

- PWM pins: pins 2 to 13 and pins 44 to 46. Provide 8-bit PWM output with the analogWrite() function.

- SPI communication pins: 50 (MISO), 51 (MOSI), 52 (SCK), and 53 (SS) are the pins that support SPI communication.

- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library.

- Aref: Reference voltage for analog inputs.

- Reset: System reset.

3.2.3. Introduction to support circuit boards.

- Board CNC Shield V3 [15]

Arduino CNC Shield V3 is an expansion shield designed for Arduino Mega2560, enabling control of laser engraving machines, CNC milling machines, or mini 3D printers. The shield allows for controlling up to 4 stepper motors using A4988 or DRV8825 drivers (with jumpers to select full step, half step, 1/4, 1/8, or 1/16 microstepping modes). In addition, it is also possible to add additional limit switches for the X, Y, Z, and E axes (specifically for 3D printers) or control the CNC spindle, laser head, and cooling fan.

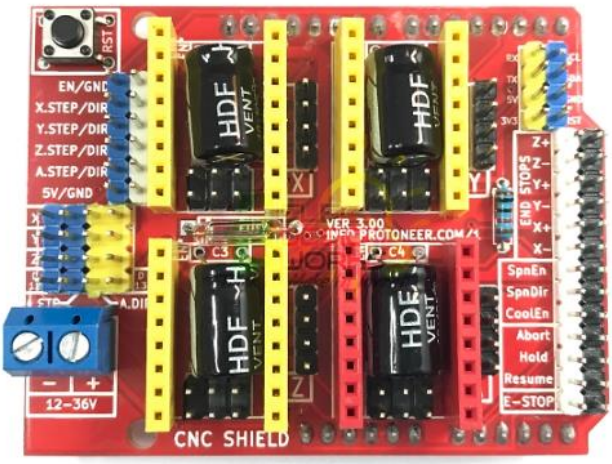


Figure 3.7: Board CNC Shield V3 [15]

a) Technical specifications

- Support for GRBL firmware (open-source code running on Arduino UNO to control mini CNC)
- Control up to 4 stepper motors.
- Driver compatibility: A4988 or DRV8825.
- Add jumpers to control full step, half step, 1/4, 1/8, 1/16.
- Travel switches for X, Y, Z, and E axes.
- Control CNC engraving head, laser engraving head.
- Control cooling fan.
- Power supply: 12-36V (independent from Arduino Uno)
- Size: 70 x 55mm

b) Wiring diagram connecting Shield CNC V3 to Arduino Mega2560.

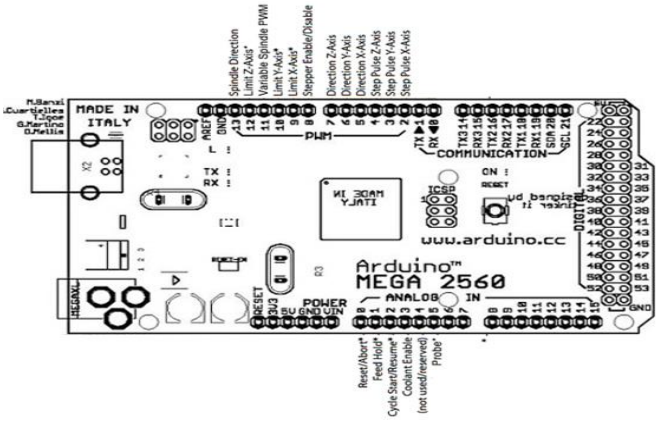


Figure 3.8: Wiring diagram connecting Shield CNC V3 to Arduino Mega2560

3.2.4. Introducing the A4988 Driver.

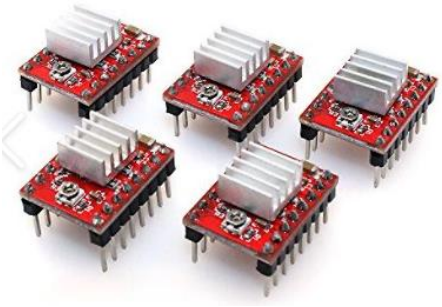


Figure 3.9: Driver A4988

a) Description

A4988 is a stepper motor driver used to control bipolar stepper motors, with integrated translator for easy operation. This means that we can control the stepper motor using only 2 pins from our controller, or one pin to control the direction of rotation and the other pin to control the steps.

b) Technical specifications

Minimum Logic Voltage:	3V
Maximum Logic Voltage:	5.5 V
Continuous current per phase:	1 A
Maximum current per phase:	2 A
Minimum Operating Voltage:	8 V
Maximum Operating Voltage:	35 V

Figure 3.10: Specifications table of the A4988 Driver

(Source: A4998 Driver Datasheet)

c) The stepping modes of the A4988 Drive controller.

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

Figure 3.11: Control modes of A4988 Stepper Driver

(Source: A4998 Driver Datasheet)

d) The stepping modes of the A4988 Drive controller.

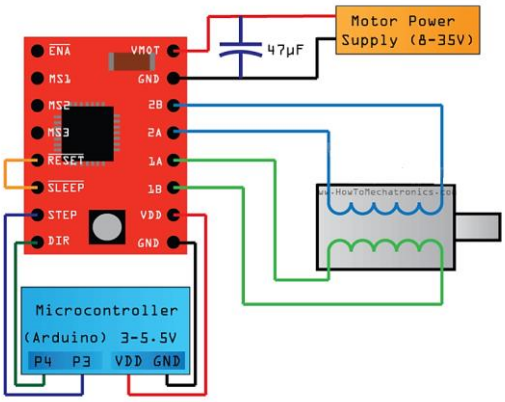


Figure 3.12: Wiring diagram connecting Drive A4988 to a stepper motor.

3.2.5. Introduction to the LM2596 Low Voltage Regulator Module.

Figure 3.13: LM2596 Low Voltage Regulator Module

a) Description

The LM2596 buck DC-DC module is a voltage converter module that converts DC input voltage from 3 to 40V into output voltage from 1.5 to 35V, with a maximum output current of 3A.

b) Technical specifications

- Input voltage: 3~40 V
- Output voltage: 1.5~35 V
- Maximum output level: 3A
- Maximum conversion efficiency: 92%
- Operating frequency: 150 kHz
- Operating temperature.: - 40^{o}C to + 85^{o}C
- Size: 23×14×8 (mm)

3.2.6. Introducing the H-Bridge Circuit - BTS 7960.

a) Description

The H-bridge circuit - BTS7960 43A enables easy communication with the microcontroller through the built-in driver integrated circuit (IC), featuring full current

sense capabilities (combined with current-sensing resistors), dead time generation, over-temperature protection, over-voltage protection, over-current protection, voltage sag protection, and short circuit protection.

b) Technical specifications

- Input voltage: 6 ~ 27V
- The circuit's load current: 43 A
- Control logic signal: 3.3 ~ 5V
- Maximum control frequency: 25KHz
- Automatic Shutdown on Low Voltage: To prevent operating the motor at low voltage, the device will automatically shut down. If the voltage drops below 5.5V, the DC motor control circuit BTS7960 will automatically cut off power and will resume operation once the voltage exceeds 5.5V.
- Overheat Protection: The BTS7960 utilizes an internal thermal sensor to protect against overheating. The output will be cut off when an overheating condition is detected.
- Size: 40×50×12 mm

c) Pinout

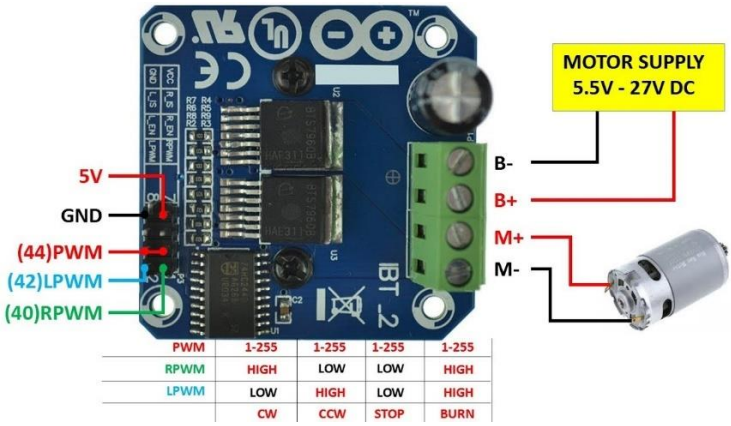


Figure 3.14: Circuit diagram of the H-bridge connection - BTS 7960.

3.2.7. Introducing PCA9865 servo module

Figure 3.15: PCA9865 Servo Module

a) Description

The PCA9865 servo module is an I2C-based expansion module that provides multiple PWM channels

b) Technical specifications

- Compatible with 5V voltage, however, it can be controlled through a 3.3V MCU and still be safe when supplied with 6V to the Servo.
- The PWM frequency can be adjusted to around 1.6KHz
- The number of PWM channels is 16 channels, with a resolution of 12 bits per channel.
- The 12-bit resolution for each servo output means approximately 4μs at 60Hz.
- Direct communication with the driver using the I2C communication standard.
- There are 6 address pins, so it can communicate with 62 different driver circuits on the same I2C bus, increasing the total number of PWM output ports to 992.
- In English, it can be translated as: 'The output pin can be configured as either Push-Pull or Open-Drain
- It can quickly cut off all output signals.
- Size: 62.5 × 25.4 × 3 mm

c) Wiring diagram

Arduino	Servo Driver
5V	VCC
GND	GND
SDA	SDA
SCL	SCL

Table 3.2: Wiring Diagram

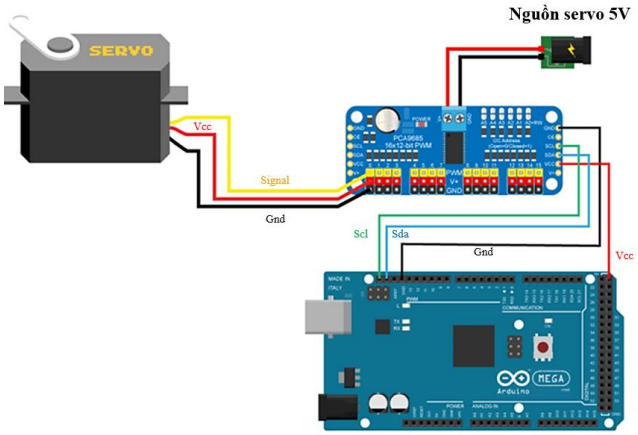


Figure 3.16: Wiring diagram connecting PCA9865 Servo Module to Arduino.

3.2.8. Design control circuit

- The PCB layout is designed by Proteus 8 Professional

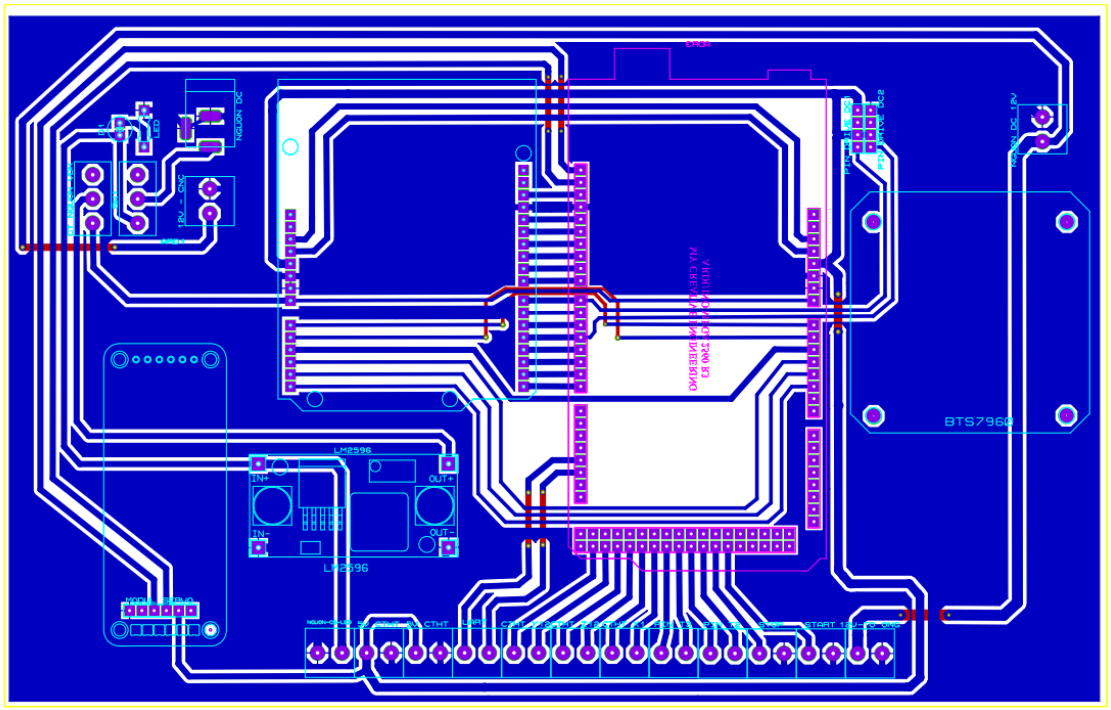


Figure 3.17: Control circuit

- The Diagram is designed by Proteus 8 Professional

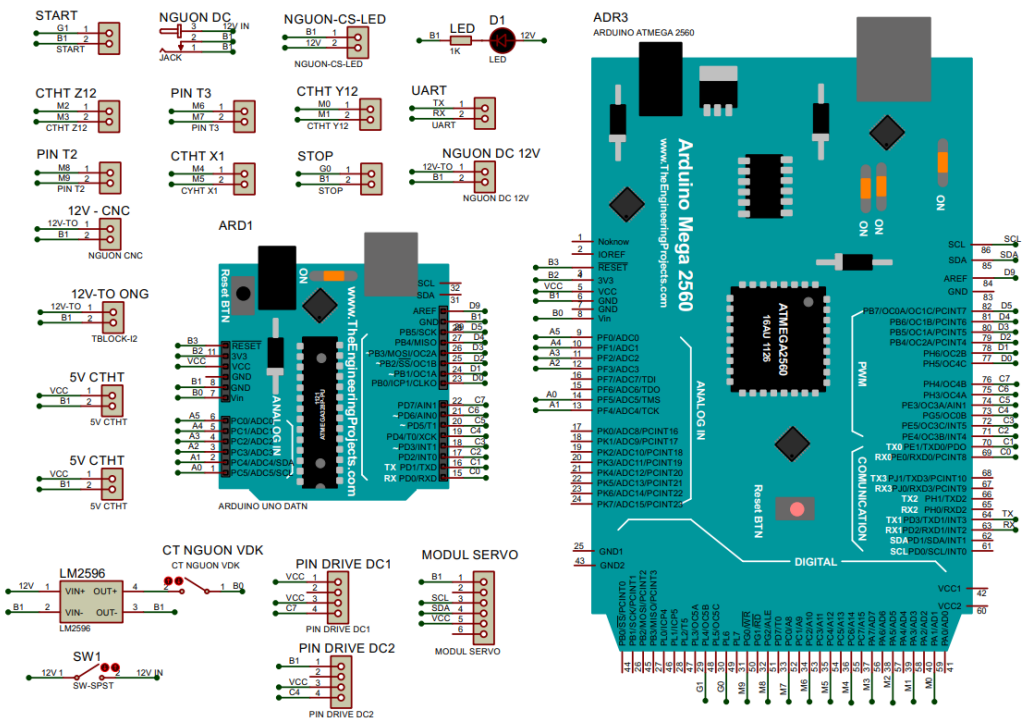


Figure 3.18: Diagram control circuit

- The 3D model is designed by Proteus 8 Professional

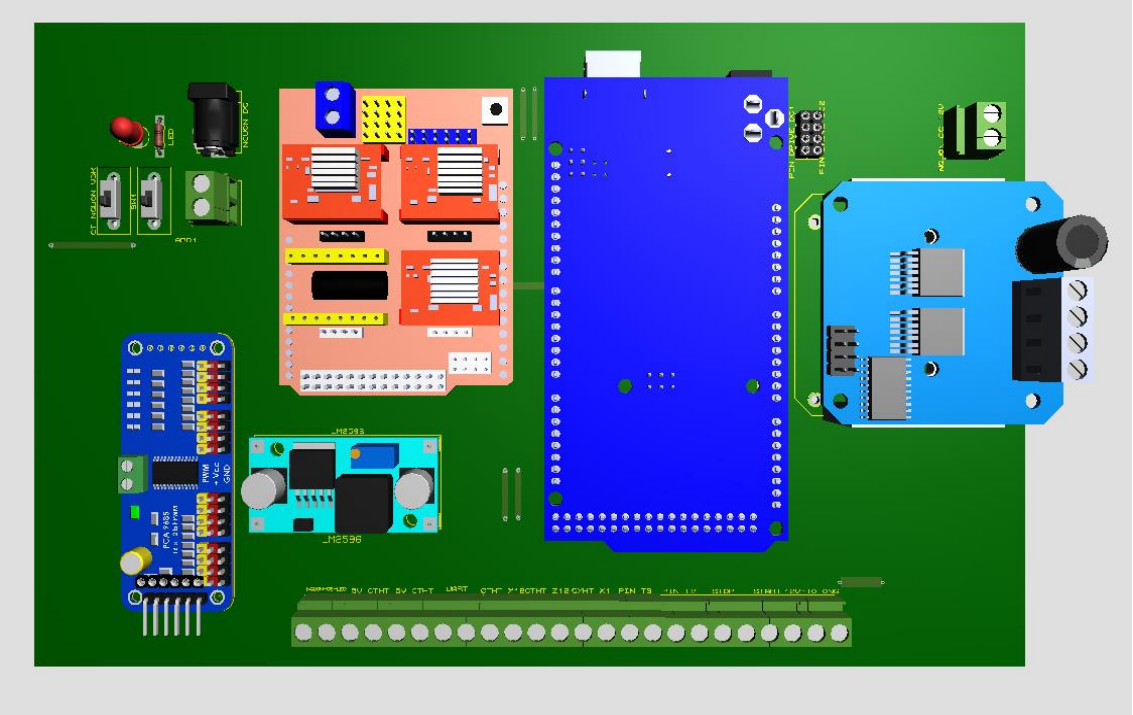


Figure 3.19: Design 3D Control circuit

- The reality control circuit

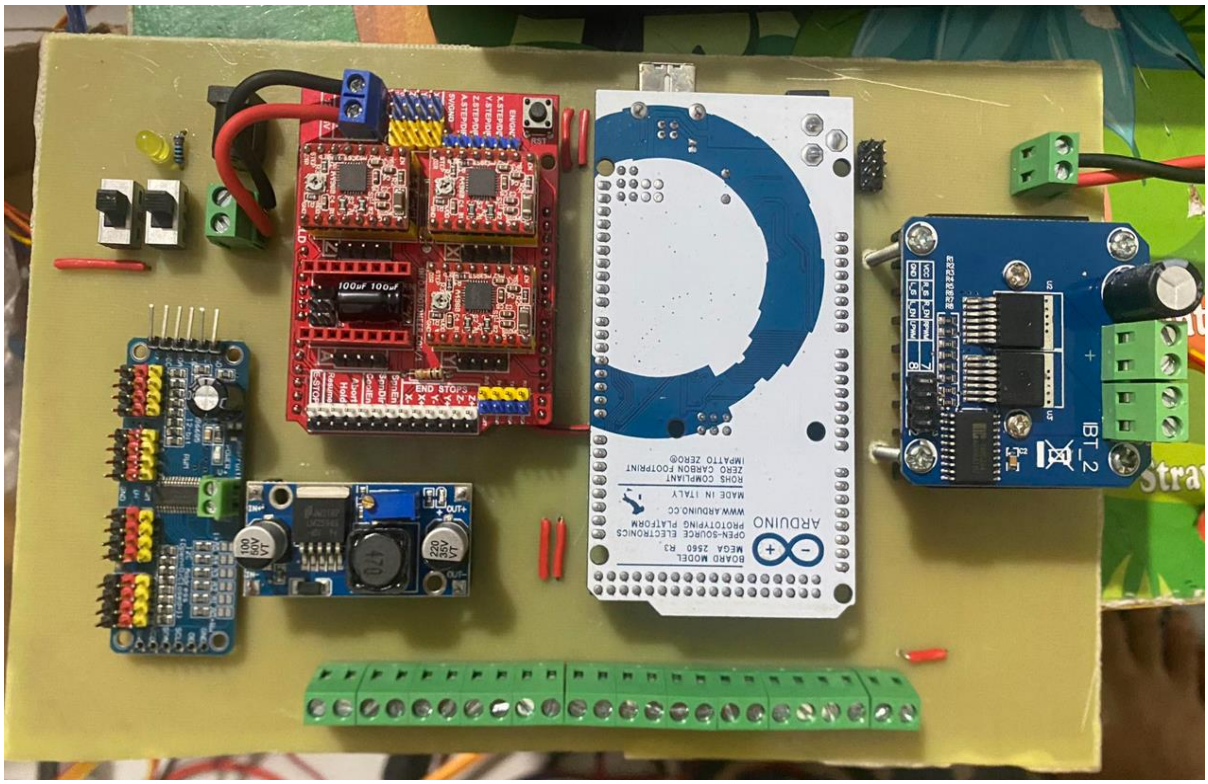


Figure 3.20: Reality Control circuit

3.2.9. Calculation and design of a power control circuit for power LEDs

Consider ¼ a cluster of 25 LEDs divided into 2 parts as follows: [14]

- Case 1: One cluster of 13 power LEDs

Design requirement: The power consumption through the LEDs is 1W and the power supply is 5V, knowing that each LED requires a voltage of 2V.

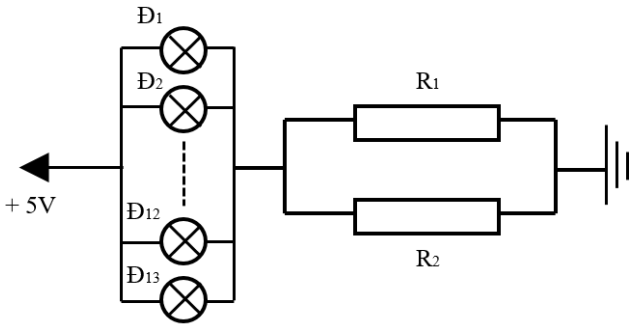


Figure 3.21: Diagram one cluster of 13 power LEDs

We have:

$$V_{Đ1} = V_{Đ2} = \cdots = V_{Đ13} = V_{Đ_{tđ}} = 2V$$

$$V_{R_{12}} = V_{R1} = V_{R2} = V_{nguồn} - V_{Đ_{tđ}} = 5 - 2 = 3V$$

Since each LED requires a power of 1W and has a voltage drop of 2V, the current flowing through each LED is calculated as follows:

$$I_{Đ1} = I_{Đ2} = ... = I_{Đ13} = I_{Đ} = 0.5 \text{ A}$$

Therefore, the equivalent current flowing through each LED is calculated as:

$$I_{12} = I_{Đ} \times \text{Number of Leds} = 0.5 \times 13 = 6.5\text{A}$$

$$I_{12} = I_{R1} + I_{R2}$$

Choose $R_1 = R_2$

For that: $U_{R1} = U_{R2}$ $(Because\ R1\ //\ R2)$

Therefor:

$$I_{R1} = I_{R2} = \frac{I_{12}}{2} = \frac{6.5}{2} = 3.25\ A$$

$$R_1 = R_2 = \frac{V_{R_1}}{I_{R1}} = \frac{V_{R_2}}{I_{R2}} = \frac{3}{3.25} \approx 0.9\ \Omega$$

Choose

$$R_{1tt} = R_{2tt} = \frac{R_1}{3.6} = \frac{R_2}{3.6} = \frac{0.9}{3} = 0.3\ \Omega\ (1)$$

Wattage R_1, R_2 is:

$$P_{R1} = P_{R2} = U_{R1}.I_{R1} = U_{R2}.I_{R2} = 3 \times 3.25 = 9.75\ W$$

Choose: $P_{R1tt} = P_{R2tt} = 10 \times 9.75 = 97.5W$ (2)

From (1),(2) choose $R_{1tt} = R_{2tt} = 0.25\ \Omega$, Wattage $P_{R1tt} = P_{R2tt} = 100\ W$

- Case 2: One cluster of 12 power LEDs (Do the same as above)

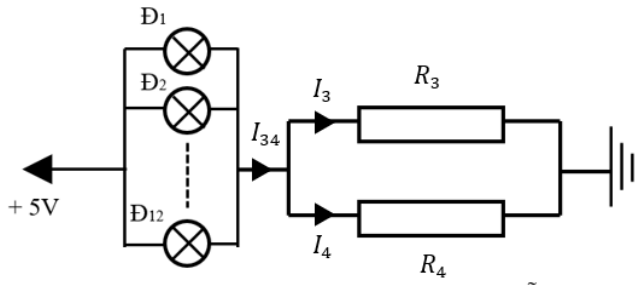


Figure 3.22: Diagram one cluster of 12 power LED

Choose $R_{3tt} = R_{4tt} = 0.25\Omega$, Wattage $P_{R3tt} = P_{R4tt} = 100W$

Electric current $I_{34} = I_Đ \times Number\ of\ Leds = 0.5 \times 12 = 6A$

Case 1 and Case 2, we have total of electric current one of 25 leds is:

$$I_{14} = I_{12} + I_{34} = 6.5 + 6 = 12.5\ \text{A}$$

Therefore: The total current required to supply the 4 clusters as calculated is 12.5 × 4 = 60 A

Observation: In reality, the measured value is much lower than 60A because we haven't considered the factor of resistance in each LED and actual measurements show that the resistance in each LED significantly affects the total current required. The resistance value of each LED is not fixed and tends to increase as the temperature of each LED becomes hotter.

Based on the above observation and actual measurements, we choose a power supply voltage for the circuit of 40A.

3.2.10. Power LED Control Circuit

- The Diagram is designed by Proteus 8 Professional

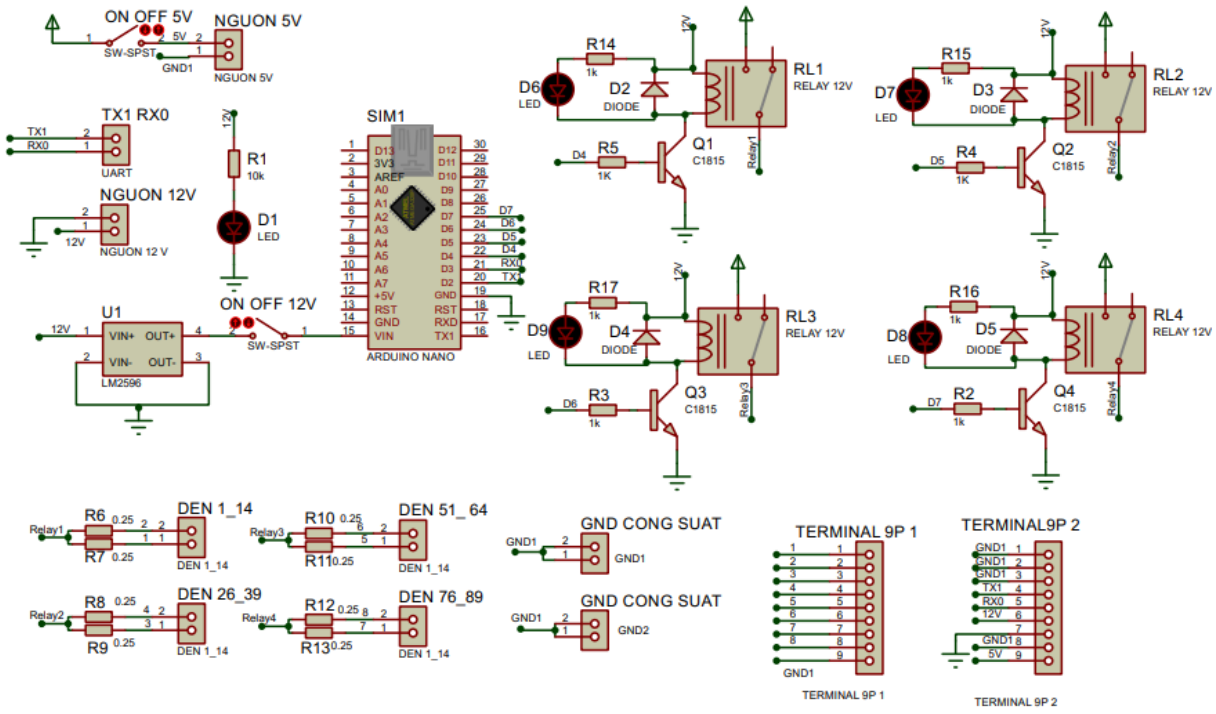


Figure 3.23: Diagram power LED control circuit

- The PCB layout is designed by Proteus 8 Professional

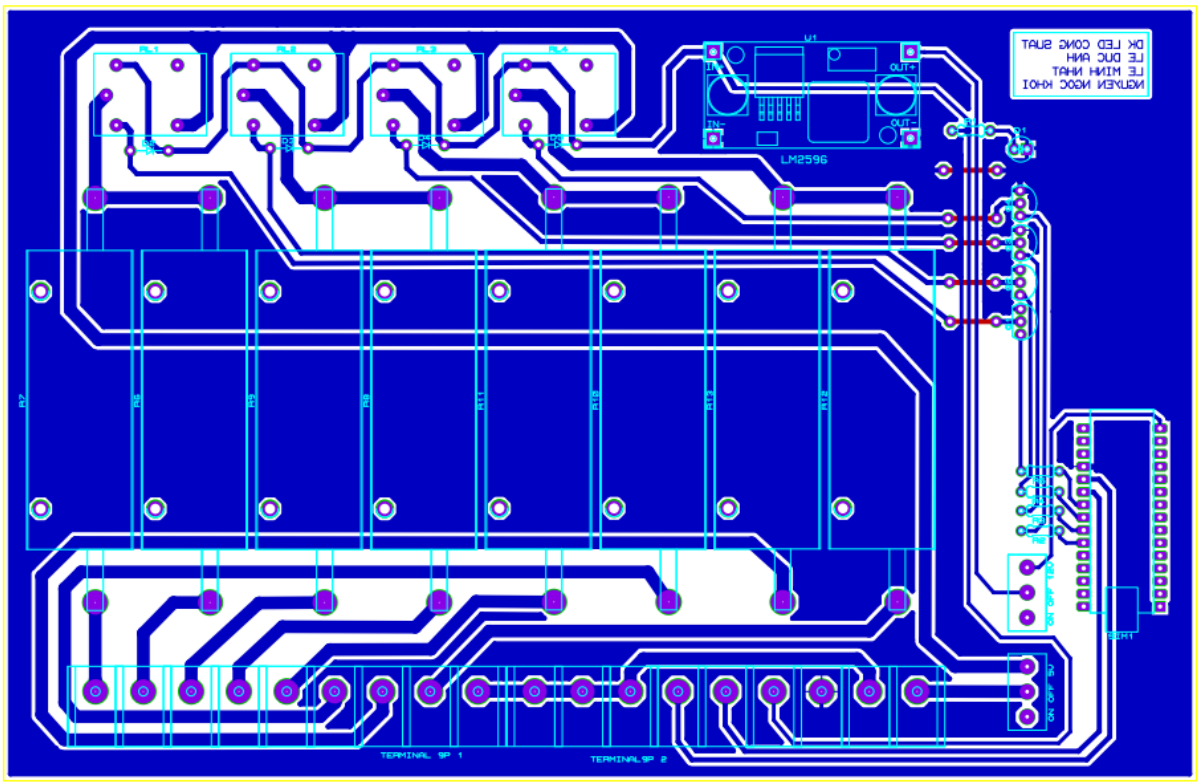


Figure 3.24: PCB power LED control circuit

- The 3D model is designed by Proteus 8 Professional

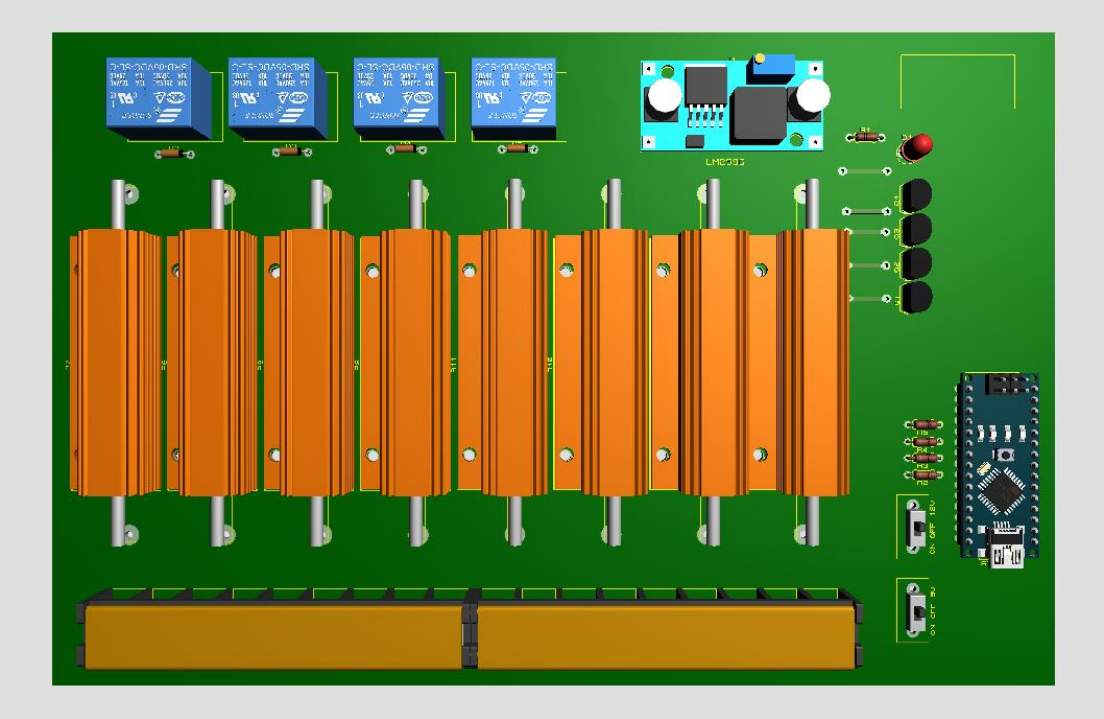


Figure 3.25: Design 3D power LED control circuit

- The reality power LED control circuit

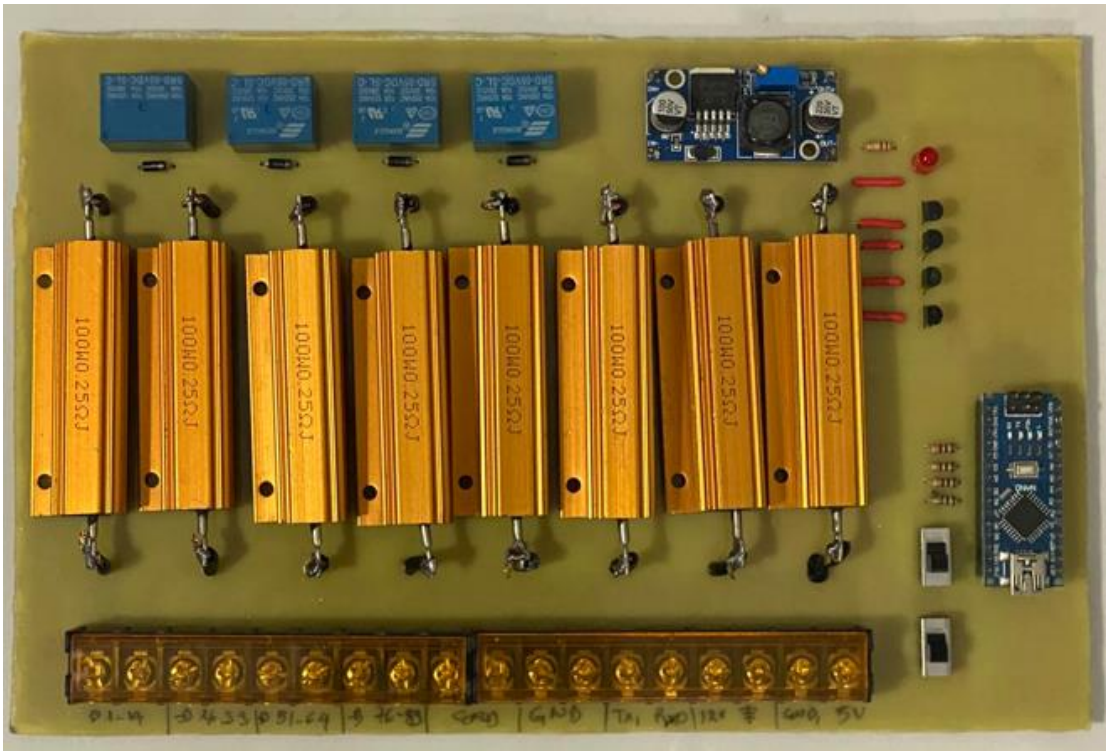


Figure 3.26: Reality power LED control circuit

CHAPTER 4 RESEARCHING MACHINE LEARNING ALGORITHMS APPLICABLE TO THE TOPIC

To categorize the Machine Learning algorithms that can be applied to your topic, there are two fundamental ways you can encounter in this field.

Grouping Machine Learning algorithms by learning style: The first group consists of Machine Learning algorithms that follow a learning style.

Grouping Machine Learning algorithms by similarity in form or function: The second group comprises Machine Learning algorithms that share similarities in form or function.

4.1. Machine Learning algorithms grouped by learning style

- Fundamentally, there are various ways in which an algorithm can model a problem.
- Additionally, it involves interacting with experiences. However, it is unrelated to how we want to call the input data.
- Moreover, an algorithm is common in machine learning textbooks and artificial intelligence. That is, let's first consider the ways in which an algorithm can adapt.
- In general, there are only a few main learning styles that a Machine Learning algorithm can have. And we will also explore them. Additionally, we have some examples of ML algorithms and the types of problems they are suitable for.
- Essentially, organizing these machine learning algorithms in this way is very useful. Because it forces you to think about the role of input data and the process of model preparation. Additionally, to select the most suitable one for your problem to achieve the best results.

Let's examine three different learning styles in machine learning algorithms.

4.2. Supervised Learning

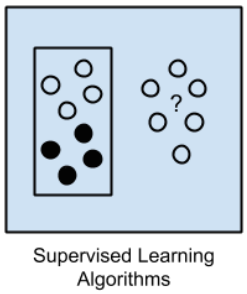


Figure 4.1*: *Supervised Learning Algorithm [13]

In essence, in this supervised learning algorithm, the input data is referred to as training data and is labeled with known outcomes or results, such as spam/not spam or stock prices at a particular time.

A model is prepared through a training process, which is necessary for making predictions and gets adjusted when those predictions are incorrect. The training process continues until the model achieves the desired level of performance.

- Examples of problems include classification and regression.
- Some algorithm examples include logistic regression and backpropagation Neural Networks.

4.3. Unsupervised learning

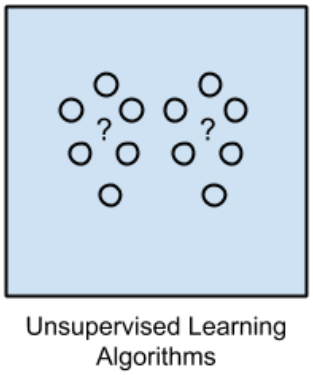


Figure 4.2*: *Unsupervised Learning Algorithm [13]

In this unsupervised learning approach, the input data is unlabeled and lacks known outcomes.

We need to prepare a model by deducing the inherent structures present in the input data. This can involve extracting common patterns or reducing redundancy through mathematical processes.

- Examples of problems include clustering, dimensionality reduction, and learning association rules.
- Some algorithm examples include the Apriori algorithm and k-Means clustering algorithm.

4.4. Semi-supervised Learning

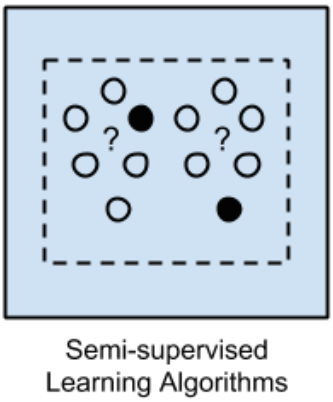


Figure 4.3*: *Semi-supervised Learning Algorithm [13]

The input data is a mixture of labeled and unlabeled examples.

There is a desired prediction task, but the model needs to learn structures to organize the data as well as make predictions.

- Examples of problems include classification and regression.
- Algorithm examples include extensions of other flexible methods that make assumptions about how to model the unlabeled data.

4.5. Algorithms Grouped by Similarity

Machine learning algorithms are often grouped based on their functional similarity.

For example, there are methods based on trees and methods inspired by neural networks.

I believe this is the most useful way to group machine learning algorithms, and it is the approach we will use here.

This is a helpful grouping method, but it is not perfect. There are still algorithms that can easily fit into multiple categories. For instance, Support Vector Machines. They are both a neural network method and belong to a category with the same name, which describes the problem and class of algorithms, such as regression and clustering.

We can handle these cases by listing the machine learning algorithms twice or by subjectively choosing the group that best fits the algorithm. I prefer the latter approach of avoiding duplicate algorithms to keep everything simple.

4.6. Regression Algorithms

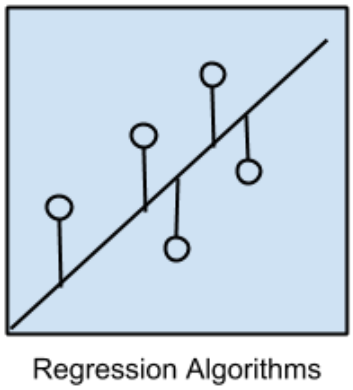


Figure 3.4: Regression Algorithms [14]

Regression algorithms are used to model the relationship between variables. They are used for fine-tuning by using error measurement methods on predictions made by the model.

These methods are a characteristic of statistics and have been chosen to be part of statistical machine learning. This can be confusing because we can use regression to refer to both the problem class and the algorithm class.

The most popular regression algorithms in machine learning are:

- Ordinary Least Squares Regression (OLSR)
- Linear Regression
- Logistic Regression
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)

4.7. Instance-based Algorithms

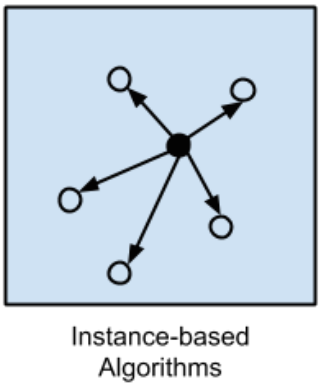


Figure 4.5: Instance-based Algorithms [14]

This model is a decision problem with cases of training data. It is considered important or necessary for the model.

Such a method builds a database of example data. And it needs to compare new data to the database. To make the comparison, we use a similar measure to find the best match and make predictions. For this reason, instance-based methods are also referred to as winnow-all or memory-based learning methods. The focus is on the representation of stored instances. Therefore, similar measures are used between instances.

The most popular instance-based algorithms in machine learning are:

- k-Nearest Neighbor (kNN)
- Learning Vector Quantization (LVQ)
- Self-Organizing Map (SOM)
- Locally Weighted Learning (LWL)

4.8. Regularization Algorithms

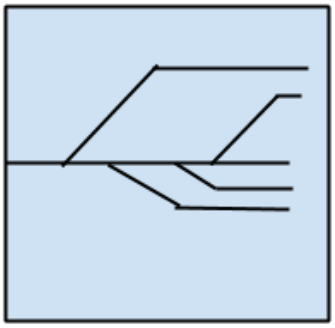


Figure 4.6: Regularization Algorithms [14]

An extension is performed for another method, which penalizes models based on their complexity. It supports simpler models that are better at generalization.

I have listed the popular Regularization algorithms here because they are widely used, powerful, and generally simple modifications made to other methods.

The most popular Regularization algorithms in Machine Learning are:

- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO)
- Elastic Net
- Least-Angle Regression (LARS)

4.9. Decision Tree Algorithm

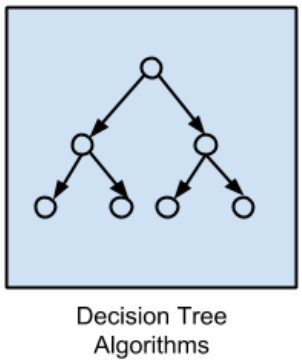


Figure 4.7: Decision Tree Algorithms [14]

The Decision Tree method builds a model of decisions based on the actual values of attributes in the data.

Decision splits in the tree structure are made until a prediction is reached for a specific record. Decision Trees are trained on data for both classification and regression problems. Decision Trees are often fast, accurate, and widely favored in machine learning.

The most popular Decision Tree algorithms in Machine Learning are:

- Classification and Regression Tree (CART)
- Iterative Dichotomiser 3 (ID3)
- 5 and C5.0 (different versions of a powerful approach)
- Chi-squared Automatic Interaction Detection (CHAID)
- Decision Stump
- M5
- Conditional Decision Trees

4.10. Bayesian Algorithms

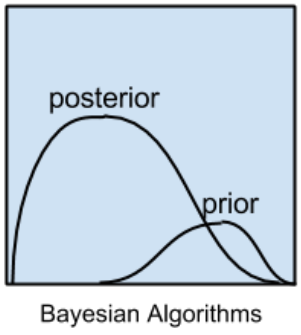


Figure 4.8: Bayesian Algorithms [14]

These methods are approaches that apply Bayes' theorem to problems such as classification and regression.

The most popular Bayesian algorithms in Machine Learning are:

- Naive Bayes
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Averaged One-Dependence Estimators (AODE)
- Bayesian Belief Network (BBN)
- Bayesian Network (BN)

These algorithms utilize Bayesian principles to model and make predictions based on probability distributions and prior knowledge.

4.11. Clustering Algorithms

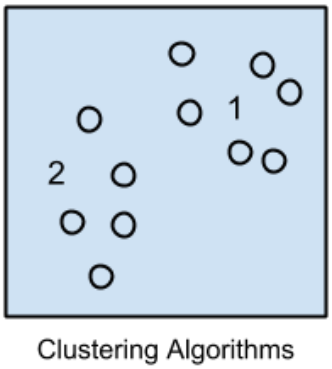


Figure 4.9: Clustering Algorithms [14]

Clustering, similar to regression, describes both a problem class and a method class.

Clustering methods are organized based on modeling approaches such as centroid-based and hierarchical. All methods involve utilizing inherent structures in the data, aiming to achieve optimal organization of the data into meaningful groups.

The most popular clustering algorithms in Machine Learning are:

- k-Means
- k-Medians
- Expectation Maximization (EM)
- Hierarchical Clustering

These algorithms employ different techniques to partition data into clusters, considering factors like distances, similarities, or probabilities. They help identify patterns, group similar instances, and uncover hidden structures in the data.

4.12. Association Rule Learning Algorithms

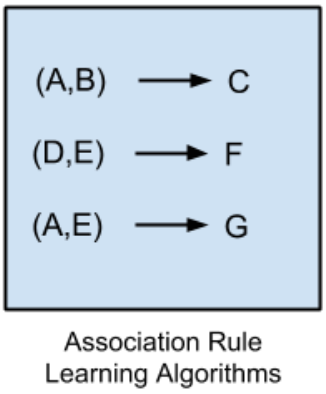


Figure 4.10: Association Rule Learning Algorithms [14]

The association rule learning algorithm is designed to extract rules that explain the relationship between variables in the data. These rules can uncover important and useful combinations in large multidimensional datasets, which can be leveraged by organizations for various purposes.

The most popular association rule learning algorithms in Machine Learning are:

- Apriori Algorithm: This algorithm identifies frequent itemsets, which are sets of items that often co-occur together in transactions. From these frequent itemsets, it generates association rules that indicate the likelihood of certain items being present based on the presence of other items.
- Eclat Algorithm: The Eclat algorithm is similar to the Apriori algorithm but utilizes a different approach for finding frequent itemsets. It uses a vertical data format and employs depth-first search to discover itemsets that satisfy the minimum support threshold.

These algorithms are widely used in market basket analysis, recommendation systems, and other domains where discovering meaningful associations between items is crucial for decision-making and pattern identification.

4.13. Artificial Neutral Network Algorithms

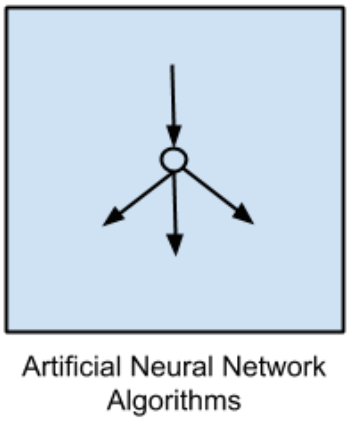


Figure 4.11: Artificial Neutral Network Algorithms [14]

These models are inspired by the structure of biological neural networks. They belong to a class suitable for modeling that we use for regression and classification problems. Although there is a vast subfield with hundreds of algorithms and variants.

The most popular artificial neural network algorithms are:

- Perceptron: The perceptron is one of the simplest neural network architectures. It consists of a single layer of artificial neurons (perceptrons) that are connected to form a network. It is primarily used for binary classification tasks.
- Back-Propagation: Back-propagation is a widely used algorithm for training multi-layer neural networks. It involves propagating the error backward through the network and adjusting the weights of the connections to minimize the error. It is commonly used for both classification and regression tasks.
- Hopfield Network: The Hopfield network is a type of recurrent neural network that is used for associative memory tasks. It is capable of storing and recalling patterns based on their similarity to learned patterns.
- Radial Basis Function Network (RBFN): RBFN is a type of neural network that uses radial basis functions as activation functions. It is often used for pattern recognition and function approximation tasks.

These neural network algorithms have been successful in various domains, including image recognition, natural language processing, and financial forecasting, among others.

4.14. Deep Learning Algorithms

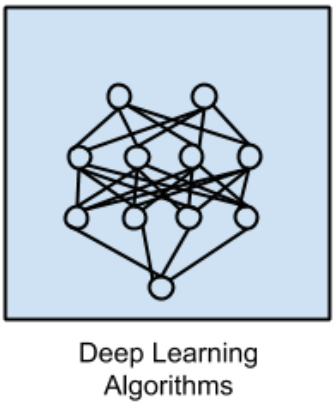


Figure 4.12: Deep Learning Algorithms [14]

Deep Learning is a modern advancement of Artificial Neural Networks that leverages abundant and affordable computing power. It focuses on constructing larger and more complex neural networks.

The most popular Deep Learning algorithms are:

- Deep Boltzmann Machine (DBM): DBM is a generative model that consists of multiple layers of hidden units. It is trained using a variant of Markov Chain Monte Carlo (MCMC) sampling techniques.
- Deep Belief Networks (DBN): DBN is a type of deep neural network that combines multiple layers of restricted Boltzmann machines (RBMs). It is trained using a layer-by-layer unsupervised learning approach followed by supervised fine-tuning.
- Convolutional Neural Network (CNN): CNN is a type of neural network particularly effective for image and video processing tasks. It uses convolutional layers to automatically learn hierarchical representations from input data.
- Stacked Auto-Encoders: Stacked Auto-Encoders are neural networks composed of multiple layers of unsupervised auto-encoders. Each layer learns to encode and decode the input data, progressively capturing more complex representations

These deep learning algorithms have revolutionized many fields, including computer vision, natural language processing, speech recognition, and recommendation systems. They have achieved state-of-the-art performance in tasks such as image classification, object detection, machine translation, and speech synthesis, among others.

4.15. Dimensional Reduction Algorithms

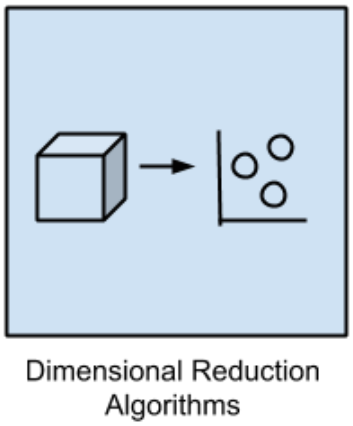


Figure 4.13: Dimensional Reduction Algorithms [14]

Like clustering methods, dimensionality reduction explores the inherent structure within the data but, in this case, for summarization purposes.

In general, it can be useful for visualizing high-dimensional data. Additionally, it can be employed in supervised learning approaches. Many methods we apply for classification and regression include:

- Principal Component Analysis (PCA)
- Principal Component Regression (PCR)
- Partial Least Squares Regression (PLSR)
- Sammon Mapping
- Multidimensional Scaling (MDS)
- Projection Pursuit
- Linear Discriminant Analysis (LDA)
- Mixture Discriminant Analysis (MDA)
- Quadratic Discriminant Analysis (QDA)
- Flexible Discriminant Analysis (FDA)

These methods help in reducing the dimensionality of the data, capturing essential information, and improving interpretability or classification/regression performance.

4.16. Ensemble Algorithms

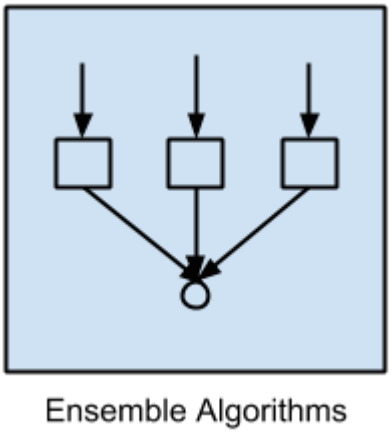


Figure 4.14: Ensemble Algorithms [14]

These methods are essentially ensembles that consist of weaker models. They are trained and their predictions are combined in some way to make a final prediction.

Furthermore, significant efforts have been made to devise weak learners and explore how to combine them effectively. As a result, this is a powerful class of techniques that has gained popularity.

The most common ensemble methods in machine learning include:

- Boosting
- Bootstrapped Aggregation (Bagging)
- AdaBoost
- Stacked Generalization (blending)
- Gradient Boosting Machines (GBM)
- Gradient Boosted Regression Trees (GBRT)
- Random Forest

These ensemble methods have demonstrated excellent performance and are widely used in various machine-learning applications.

CHAPTER 5: FABRICATION AND PROGRAMMING OF THE MODEL

5.1. Making models according to design drawings

- A whole model of system is designed on Solidwork

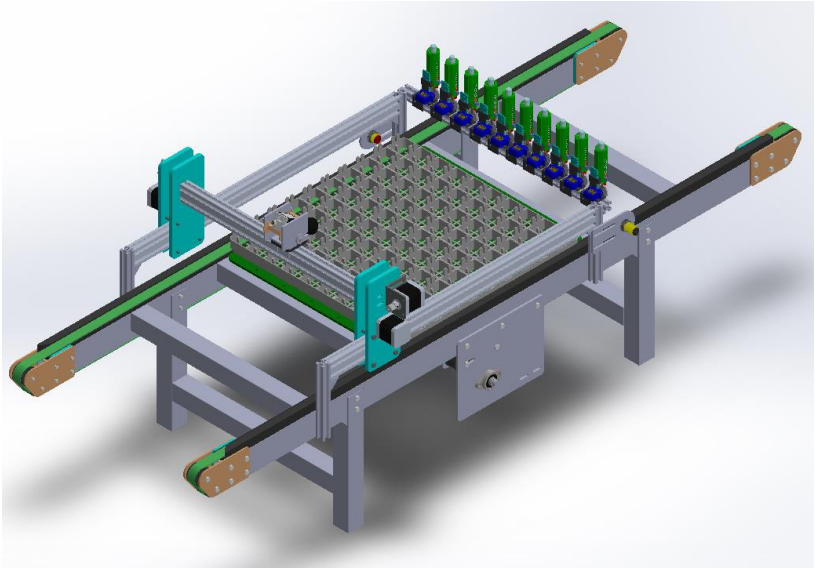


Figure 5.1 The whole model is designed on Solidwork

- The Machine Frame has been assembled.

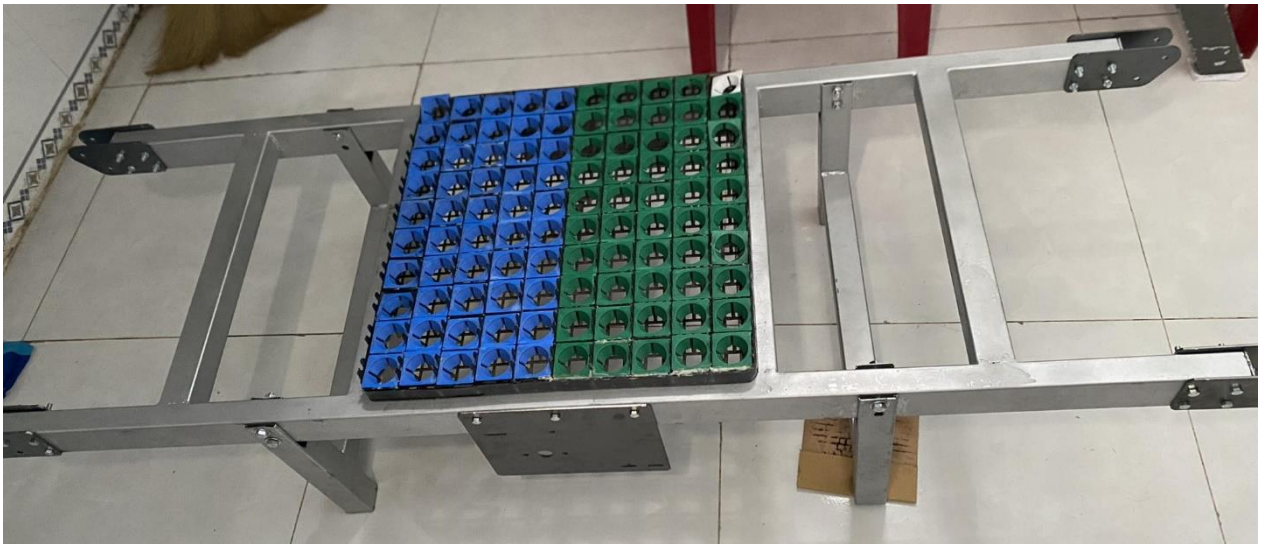


Figure 5.2: Machine Frame

- The mechanical linkage connecting the motor and the conveyor.

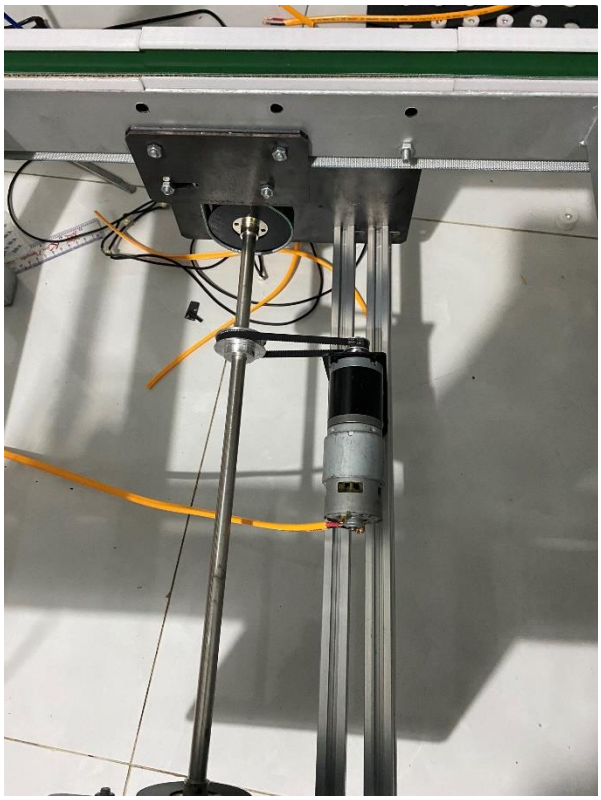


Figure 5.3: The connection between the motor and the conveyor belt.

- Assembling the camera's motion mechanism and assembling the camera

Figure 5.4 The camera mounting bracket.

- The mechanism for egg marking pen.

Figure 5.5: The egg marking pen bracket.

5.2 Model programming according to requirements

- The system control interface. [12]

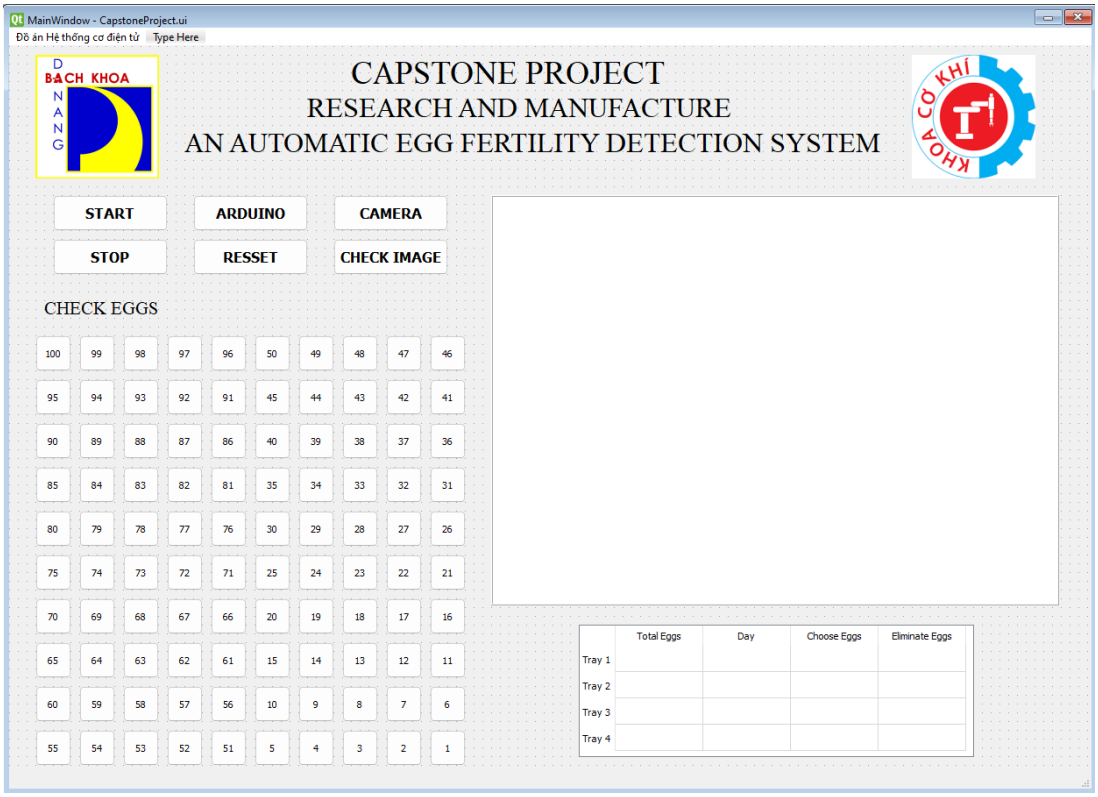
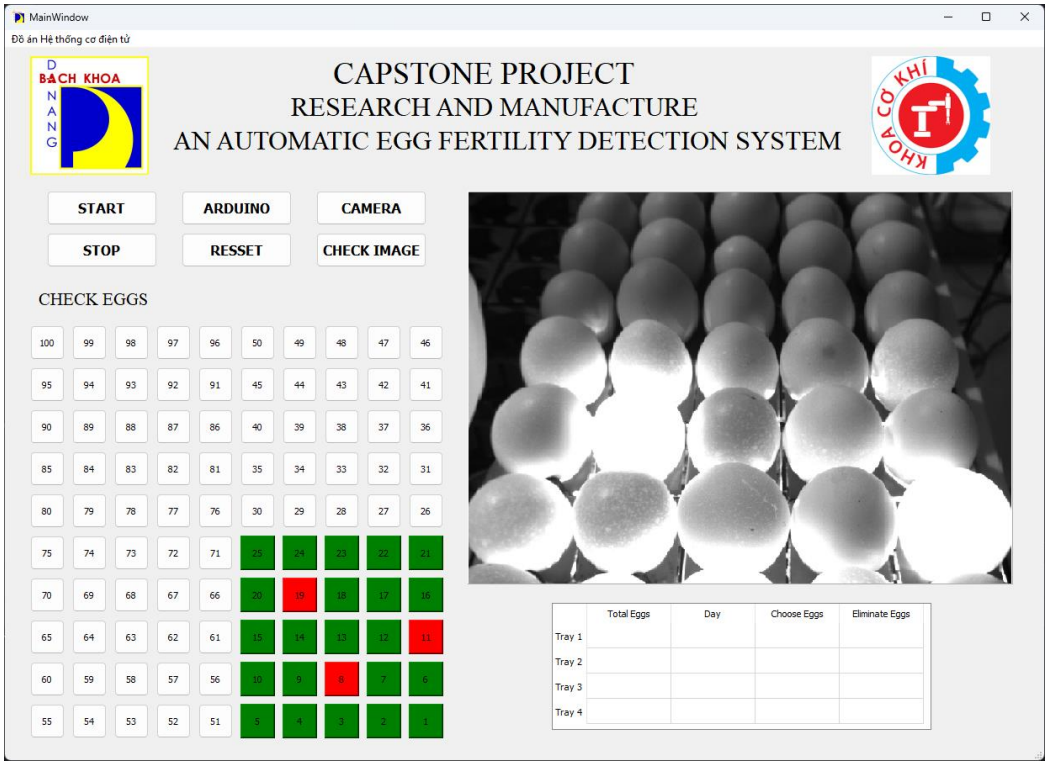


Figure 5.6: The system interface

The system interface consists of the following functional buttons:

- CAMERA: Connect the camera to display in interface
- ARDUINO: Connect the microcontroller to the computer.
- START: Start the system after successfully connecting the camera and microcontroller to the computer.
- STOP: Emergency stop the system.
- RESET: Restart system
- CHECK IMAGE: Open the foulder containing all image files displaying the image processing progress



Figures 5.7: Image processing in interface

CHAPTER 6: RESULT, CONCLUSIONS AND RECOMMENDATION

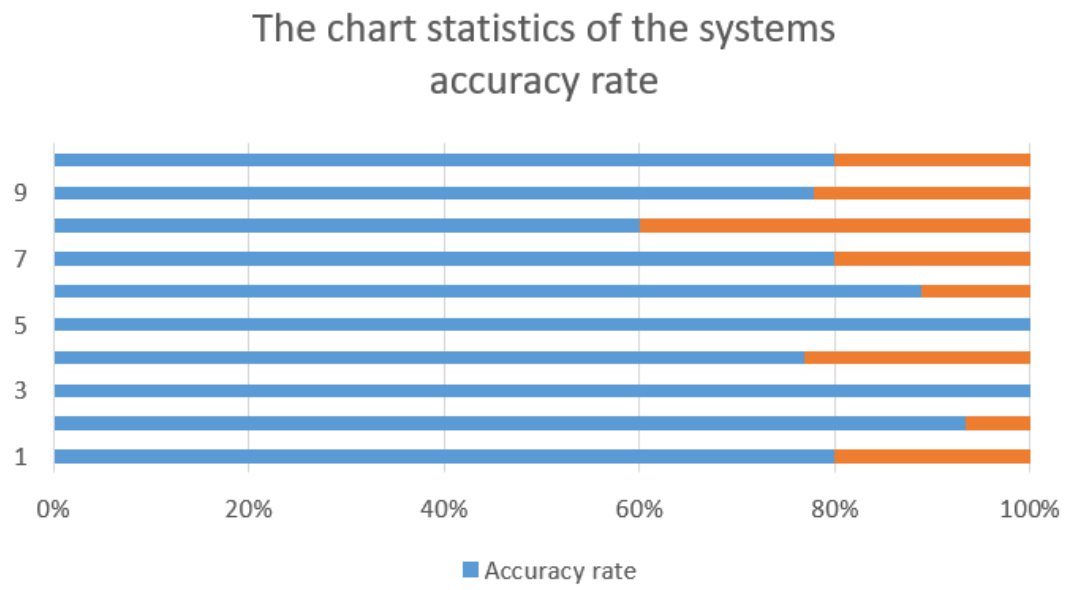
6.1. Statistics of the achieved results

- The result of the Statistics

	A	B	C	D	E	F	G
1	N.O	Time	Total Eggs	Total eggs choosed	Total eggs eliminated	Reality Eliminate	Accuracy rate
2	1	2023-06-27 13:09:24	100	88	12	15	80.00%
3	2	2023-06-27 13:12:00	100	86	14	15	93.33%
4	3	2023-06-27 13:12:24	100	90	10	10	100.00%
5	4	2023-06-27 13:16:19	100	90	10	13	76.92%
6	5	2023-06-27 13:17:33	100	88	12	12	100.00%
7	6	2023-06-27 13:20:22	100	82	18	16	88.89%
8	7	2023-06-27 13:21:58	100	85	15	12	80.00%
9	8	2023-06-27 13:23:51	100	97	3	5	60.00%
10	9	2023-06-27 13:30:04	100	93	7	9	77.78%
11	10	2023-06-27 13:32:19	100	80	20	16	80.00%

Figures 6.1: Statistics of the achieved after operation system

- The chart statistics of the systems accuracy rate



Figures 6.2: The chart statistics of the systems accuracy rate

- The system was run and tested during the manufacturing process. The object under study here is 4-day-old chicken eggs, and Figure 6.1 describes the results of 10 machine operations, with the eggs being classified on June 27, 2023.

- Initially, the evaluation team found that the system runs stably with minimal vibrations during operation. The system is capable of classifying one tray of eggs

(consisting of 100 eggs) in approximately 15 seconds, significantly reducing labor time and increasing productivity and economic efficiency for the business.

- In terms of software, after the classification process, the system's software consolidates data on the selected eggs and categorized eggs and exports it to an Excel file. This allows the business to monitor the classification rate of the system, the rate of eggs discarded in a hatching batch, and derive appropriate solutions for the future. Additionally, the operators can directly monitor the position of discarded eggs and the total number of categorized eggs on the software interface.

- In terms of mechanics, the system has been solidly engineered, ensuring rigidity, durability, and aesthetics. Furthermore, the system is compact and easy to transport without negatively affecting its components.

- In terms of electrical system, the team has meticulously completed the electrical system with high efficiency, stability, and longevity. The system utilizes readily available modules and circuits in the market, reducing processing costs and facilitating installation and maintenance, thereby improving operational uptime. The system can operate continuously for several hours without affecting the output results or machine lifespan.

6.2. Evaluation of Achieved Results

- Accuracy of Operation: According to Figure 6.1, it can be observed that the accuracy of the machine is not stable, and the basic accuracy rate is within the desired range. This discrepancy is caused by variations in the color conditions of the eggs, particularly for eggs with dark brown shells or excessively thick shells, which have a high rate of misclassification. The discrepancy can also be attributed to positional errors of the eggs, where overly large or small eggs can cause positional deviations in the image processing, thereby obstructing the visibility of smaller eggs at the back. Therefore, improvement is needed in the classification software to enhance accuracy in industrial egg sorting.

- System Operation: Through testing and evaluation in practical conditions, the system has demonstrated stable operation with minimal noise and vibrations. However, there are occasional disruptions in the connection between the computer

and the camera during operation, which can occur when the camera's cables or power supply become loose.

- Research and Development: The team has successfully conducted research on applying image processing to egg classification, specifically in determining the hatching potential. This has contributed to reducing labor requirements and improving economic efficiency for businesses. However, the team has not yet achieved successful research in implementing Machine Learning into the system. Efforts will be made to integrate Machine Learning technology into the system in the future.

6.3. Conclusion

- Conclusions on the conducted research:

+ Mechanical Aspect: The mechanical research of the machine has been completed and entered a stable testing phase.

+ Electronics Aspect: The design of the control circuit and power circuit has been completed, relatively meeting the system's requirements and operating in a stable testing phase.

+ Software Aspect: The system control program and image processing have been relatively completed, providing promising initial results. However, there are still some issues regarding the influence of the surrounding environment on the software image processing.

- Evaluation of the research's new contributions:

+The research contributes to enriching the system of scientific research topics conducted by students each year and provides valuable references in terms of ideas and knowledge presented in the research.

+ The research was carried out in response to the needs of small and medium-sized businesses in the egg industry, which lack advanced machinery systems to improve work productivity. Therefore, the proposed design of the "Damaged Egg Sorting System" aims to commercialize the product. The research is conducted to provide practical contributions and value to these businesses, helping them save time and effort in their daily work.

- Applicability of the research:

+The research can be applied in small and medium-scale civilian production. Furthermore, the research has reference value for upgrading and expanding production on an industrial scale.

6.4. Recommendations

- Proposed directions for research development:

+ Enhance and improve the image processing program by applying machine learning to increase accuracy and minimize errors to the best possible extent.

+ Improve the lighting system of the LED lights to enhance the image quality from the camera.

+ Optimize the mechanical design to make the system operate smoother and more cost-effective.

+ Develop an egg removal mechanism from the egg tray after the egg detection process is completed.

+ Apply Raspberry Pi instead of a computer for a compact and professional system.

+ Implement IoT into the system to enhance the user experience.

- Necessary measures to apply research findings in practical life and production:

+ Secure investment funds or financial support and receive support from the university and professors to further enhance the product's knowledge, ensuring it reaches users in a refined and perfected manner, capable of long-term and stable operation.

- Recommendations for mechanisms and policies: (None)

REFERENCE MATERIALS

[1] Laura Liu Michael; Ngadi Michael Ngadi "Detecting Fertility and Early Embryo Development of Chicken Eggs Using Near-Infrared Hyperspectral Imaging", Food and Bioprocess Technology 6(9),2012.

[2] https://www.simplilearn.com/image-processing-article

[3] Nguyễn Trọng Hiệp, Nguyễn Văn Lẫm: Thiết kế chi tiết máy, Nhà xuất bản giáo dục – 1999

[4] [Việt Nam, 2019] TS. Đặng Phước Vinh, TS. Võ Như Thành: Giáo trình kỹ thuật vi điều khiển PIC, NXB Xây Dựng.

[5] ThS. Huỳnh Vinh: Sức bền vật liệu

[6] https://www.sciencedirect.com/science/article/pii/S0032579119314531

[7] https://www.mdpi.com/1424-8220/20/20/5951

[8] PGS.TS Nguyễn Hữu Lộc "Cơ sở thiết kế máy" NXB ĐHQG TP.HCM

[9] https://www.mdpi.com/2306-7381/9/10/574

[10] https://docs.opencv.org/4.x/d9/df8/tutorial_root.html

[11] https://www.youtube.com/watch?v=WQeoO7MI0Bs&t=1s

[12] https://www.youtube.com/watch?v=Fk1TBoBcrR4

[13] https://machinelearning101.readthedocs.io/en/latest/_pages/01_introduction.html

[14] https://www.homemade-circuits.com/how-to-design-simple-led-driver-circuits/

[15]https://www.openimpulse.com/blog/wpcontent/uploads/wpsc/downloadables/Arduino-CNC-Shield.pdf

APPENDIX

1. Specifications of the system equipment:

Camera: Basler acA2500-14gm

- Resolution: 5 Megapixels
- Sensor: MT9P031
- Sensor Type: CMOS
- Lens: 4-16mm
- Digital Noise Reduction (DNR) 3D
- Support for GigE connectivity standard
- Power supply: PoE or 12VDC
- Optical zoom through mechanical lens

Laptop computer with stable configuration for data processing

- CPU: Intel Core i5 8300H (2.5-4.1GHz)
- RAM: 12GB DDR4
- Graphics Card: NVIDIA GeForce GTX 1050
- Storage: 235GB SSD + 240GB SSD
- AC/DC power supply for the camera and laptop.

2. Code System

```
import threading
import time
import openpyxl
import datetime
import sys
import cv2
import serial
from PyQt5 import QtCore, QtGui, QtWidgets, uic
from PyQt5.QtCore import pyqtSlot
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from pypylon import pylon
import numpy as np

Type_Eggs = []
stop_camera = True
stop_arduino = True
stop_test = True
stop_main = True
```

```
imgResize_2 = 0
imgResize = 0
ser = serial.Serial('com4', 9600, timeout=0.7)
data = 0

# conecting to the first available camera
camera =
pylon.InstantCamera(pylon.TlFactory.GetInstance().CreateFirstDevice())
# Grabing Continusely (video) with minimal delay
camera.StartGrabbing(pylon.GrabStrategy_LatestImageOnly)
converter = pylon.ImageFormatConverter()
# converting to opencv bgr format
converter.OutputPixelFormat = pylon.PixelType_BGR8packed
converter.OutputBitAlignment = pylon.OutputBitAlignment_MsbAligned

def ImageProcessing(img): #change to HSV Image
    h_min = 0
    h_max = 179
    s_min = 0
    s_max = 255
    v_min = 31
    v_max = 217
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower = np.array([h_min, s_min, v_min])
    upper = np.array([h_max, s_max, v_max])
    mask = cv2.inRange(imgHSV, lower, upper)
    return mask

def getContours(img):
    contours, hierarchy = cv2.findContours(img, 1, 2)
    if len(contours) == 0:
        area = 0
    else:
        c = max(contours, key=cv2.contourArea)
        area = cv2.contourArea(c)
    return area

def CheckEgg(image, xEgg, yEgg, w, h, boundary):
    Egg = image[yEgg:(yEgg + w), xEgg:(xEgg + h)]
    area = getContours(Egg)
    if area > boundary:
        return 1
    return 0

def ResultTenEggs(mask): # determine the state of the 10 eggs in the
image based on their shape and area.
# x and y of the upper left corner of the rectangle surrounding each
egg in the image.
    xEgg = [49, 294, 549, 801, 1045, 90, 336, 554, 786, 1018]
    yEgg = [715, 715, 738, 720, 730, 499, 507, 499, 505, 505]
    trung = []
```

```
    for x in range(len(xEgg)):
        trung.append(CheckEgg(mask, xEgg[x], yEgg[x], 150, 150, 1000))
    return trung # Returns the middle list, containing the results of
determining the state of the 10 eggs in the image.


def ResultFifteenEggs(mask): # determine the state of the 15 eggs in
the image based on their shape and area.
    # x and y of the upper left corner of the rectangle surrounding
each egg in the image.
    xEgg = [1061, 790, 546, 298, 54, 1024, 798, 543, 332, 92, 992, 787,
574, 361, 168]
    yEgg = [722, 716, 737, 720, 725, 491, 480, 489, 512, 499, 308, 306,
311, 322, 325]
    trung = []
    for x in range(len(xEgg)):
      if x > 10:
        trung.append(CheckEgg(mask, xEgg[x], yEgg[x], 100, 150, 2000))
      else:
        trung.append(CheckEgg(mask, xEgg[x], yEgg[x], 150, 150, 2000))
      if x == 9:
        Egg = mask[yEgg[x]:(yEgg[x] + 150), xEgg[x]:(xEgg[x] + 150)]
    return trung

def ImageProcessing(img):
    h_min = 0
    h_max = 179
    s_min = 0
    s_max = 255
    v_min = 31
    v_max = 217
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower = np.array([h_min, s_min, v_min])
    upper = np.array([h_max, s_max, v_max])
    mask = cv2.inRange(imgHSV, lower, upper)
    return mask

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1235, 868)
        MainWindow.setFocusPolicy(QtCore.Qt.StrongFocus)
self.A = [self.TRUNG1, self.TRUNG2, self.TRUNG3, self.TRUNG4,
self.TRUNG5, self.TRUNG6, self.TRUNG7,
          self.TRUNG8, self.TRUNG9, self.TRUNG10,
          self.TRUNG11, self.TRUNG12, self.TRUNG13, self.TRUNG14,
self.TRUNG15, self.TRUNG16, self.TRUNG17,
          self.TRUNG18, self.TRUNG19, self.TRUNG20,
          self.TRUNG21, self.TRUNG22, self.TRUNG23, self.TRUNG24,
self.TRUNG25, self.TRUNG26, self.TRUNG27,
          self.TRUNG28, self.TRUNG29, self.TRUNG30,
          self.TRUNG31, self.TRUNG32, self.TRUNG33, self.TRUNG34,
```

```
self.TRUNG35, self.TRUNG36, self.TRUNG37,
          self.TRUNG38, self.TRUNG39, self.TRUNG40,
          self.TRUNG41, self.TRUNG42, self.TRUNG43, self.TRUNG44,
self.TRUNG45, self.TRUNG46, self.TRUNG47,
          self.TRUNG48, self.TRUNG49, self.TRUNG50,
          self.TRUNG51, self.TRUNG52, self.TRUNG53, self.TRUNG54,
self.TRUNG55, self.TRUNG56, self.TRUNG57,
          self.TRUNG58, self.TRUNG59, self.TRUNG60,
          self.TRUNG61, self.TRUNG62, self.TRUNG63, self.TRUNG64,
self.TRUNG65, self.TRUNG66, self.TRUNG67,
          self.TRUNG68, self.TRUNG69, self.TRUNG70,
          self.TRUNG71, self.TRUNG72, self.TRUNG73, self.TRUNG74,
self.TRUNG75, self.TRUNG76, self.TRUNG77,
          self.TRUNG78, self.TRUNG79, self.TRUNG80,
          self.TRUNG81, self.TRUNG82, self.TRUNG83, self.TRUNG84,
self.TRUNG85, self.TRUNG86, self.TRUNG87,
          self.TRUNG88, self.TRUNG89, self.TRUNG90,
          self.TRUNG91, self.TRUNG92, self.TRUNG93, self.TRUNG94,
self.TRUNG95, self.TRUNG96, self.TRUNG97,
          self.TRUNG98, self.TRUNG99, self.TRUNG100, ]
       self.retranslateUi(MainWindow)
       QtCore.QMetaObject.connectSlotsByName(MainWindow)

       self.CAMERA.clicked.connect(self.batcam)
       self.STOP.clicked.connect(self.STOPCLICK)
       self.RESET.clicked.connect(self.RESETSYSTEM)
       self.COM.clicked.connect(self.Arduino)
       self.START.clicked.connect(self.Start)

    def CONNECTCAMERA(self):
        global stop_camera
        stop_camera = True
        DisplayImage = threading.Thread(target=self.batcam())
        DisplayImage.start()
        print("Cam connected")

    def Arduino(self):
        global stop_arduino
        stop_arduino = True
        ConnectArduino = threading.Thread(target=self.ReadArduino)
        ConnectArduino.start()
        print("Arduino connected")

    def batcam(self):
        global imgResize_2, imgResize, camera
        camera.GainRaw.SetValue(20)
        while stop_camera:
            grabResult = camera.RetrieveResult(5000,
pylon.TimeoutHandling_ThrowException)
            if grabResult.GrabSucceeded():
                img = grabResult.Array
                imgResize = cv2.resize(img, (648, 468))
```

```
                imgResize_2 = cv2.resize(img, (1296, 972))
                print(imgResize.shape)
                self.displayImage(imgResize, 1)
                if cv2.waitKey(1) and 0xFF == ord('p'):
                    break
    def displayImage(self, img, window = 1):
        qformat = QImage.Format_Indexed8
        if len(img.shape) == 3:
            if(img.shape[2]) == 4:
                qformat = QImage.Format_RGBA888
            else:
                qformat = QImage.Format_BGR888
        img = QImage(img, img.shape[1], img.shape[0], qformat)
        img = img.rgbSwapped()
        self.SCREENCAMERA.setPixmap(QPixmap.fromImage(img))

    def RESETSYSTEM(self):
        global ser
        for i in range(0, 100):
            self.A[i].setStyleSheet("background-color: white;")
        self.count = 0
        Type_Eggs.clear()
        ser.write(b'11')



    def ReadArduino(self):
        global data, stop_arduino
        while stop_arduino:
          data = ser.readline().decode('ascii')
          if data == '':
              data = 0
          else:
              data = int(data)
          print("data =", data)

          if data == 10:
            time.sleep(0.5)
            if data == 10:
              ser.write(b'1')
            if data == 1:
              time.sleep(0.5)
              cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_1.jpg",
imgResize_2)
              img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_1.jpg")
              self.EggsProcessing10(img_process)
              print("đã chụp")
              if data == 1:
                 ser.write(b'2')
              elif data == 2:
                 time.sleep(0.5)
```

```
                    cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_2.jpg",
imgResize_2)
                    img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_2.jpg")
                    self.EggsProcessing15(img_process)
                    print("đã chụp")
                if data == 2:
                       ser.write(b'3')
               elif data == 3:
                  time.sleep(0.5)
                  cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_3.jpg",
imgResize_2)
                  img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_3.jpg")
                  self.EggsProcessing10(img_process)
                  print("đã chụp")
              if data == 3:
                  ser.write(b'4')
             elif data == 4:
                  time.sleep(0.5)
                  cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_4.jpg",
imgResize_2)
                  img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_4.jpg")
                  self.EggsProcessing15(img_process)
                  print("đã chụp")
                  if data == 4:
                     ser.write(b'5')
            elif data == 5:
                  time.sleep(0.5)
                  cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_5.jpg",
imgResize_2)
                  img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_5.jpg")
                  self.EggsProcessing15(img_process)
                  print("đã chụp")
                  if data == 5:
                      ser.write(b'6')
            elif data == 6:
                  time.sleep(0.5)
                  cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_6.jpg",
imgResize_2)
                  img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_6.jpg")
                  self.EggsProcessing10(img_process)
                  print("đã chụp")
                  if data == 6:
                      ser.write(b'7')
            elif data == 7:
                  time.sleep(0.5)
                  cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_7.jpg",
imgResize_2)
```

```
                img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_7.jpg")
                self.EggsProcessing15(img_process)
                print("đã chụp")
                if data == 7:
                    ser.write(b'8')
            elif data == 8:
                time.sleep(0.5)
                cv2.imwrite("D:/CAPSTONE_PROJECT/Image/Image_8.jpg",
imgResize_2)
                img_process =
cv2.imread("D:/CAPSTONE_PROJECT/Image/Image_8.jpg")
                self.EggsProcessing10(img_process)
                print("đã chụp")
                if data == 8:
                    ser.write(b'9')
        if data == 9:
            ser.write(b'K')
            time.sleep(0.5)
            if self.TRUNG36.styleSheet() == "background-color: red;":
                ser.write(b'Q')
            else:
                ser.write(b'L')
            if self.TRUNG37.styleSheet() == "background-color: red;":
                ser.write(b'W')
                else:
                ser.write(b'L')
            if self.TRUNG38.styleSheet() == "background-color: red;":
                ser.write(b'E')
                else:
                ser.write(b'L')
            if self.TRUNG39.styleSheet() == "background-color: red;":
                ser.write(b'R')
                else:
                ser.write(b'l')
            if self.TRUNG40.styleSheet() == "background-color: red;":
                ser.write(b'T')
                else:
                ser.write(b'L')
            if self.TRUNG51.styleSheet() == "background-color: red;":
                ser.write(b'Y')
                else:
                ser.write(b'L')
            if self.TRUNG52.styleSheet() == "background-color: red;":
                    ser.write(b'U')
                else:
                    ser.write(b'L')
            if self.TRUNG53.styleSheet() == "background-color: red;":
                    ser.write(b'I')
                else:
                    ser.write(b'L')
             if self.TRUNG54.styleSheet() == "background-color: red;":
```

```
                    ser.write(b'O')
                else:
                    ser.write(b'L')
            if self.TRUNG55.styleSheet() == "background-color: red;":
                    ser.write(b'P')
                else:
                    ser.write(b'L')
                ...

    def Start(self):
        ser.write(b'A')

    def EggsProcessing10(self, img):
        global Type_Eggs, Tong_hop_1
        mask = ImageProcessing(img)
        Eggs = ResultTenEggs(mask)
        for x in range(len(Eggs)):
            Type_Eggs.append(str(Eggs[x]))
        for x in range(len(Type_Eggs)):
            if Type_Eggs[x] == '1':
                self.A[x].setStyleSheet("background-color: green;");
            else:
                self.A[x].setStyleSheet("background-color: red;");

    def EggsProcessing15(self, img):
        global Type_Eggs
        mask = ImageProcessing(img)
        Eggs = ResultFifteenEggs(mask)
        for x in range(len(Eggs)):
            Type_Eggs.append(str(Eggs[x]))
        for x in range(len(Type_Eggs)):
            if Type_Eggs[x] == '1':
                self.A[x].setStyleSheet("background-color: green;");
            else:
                self.A[x].setStyleSheet("background-color: red;");

    def STOPCLICK(self):
        global stop_camera, stop_arduino, stop_test, stop_main
        ser.write(b'10')
        stop_camera = False
        stop_arduino = False
        stop_test = False
        stop_main = False
        print("Stop")

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
```

```
sys.exit(app.exec_())
try:
    sys.exit(app.exec_())
except:
    print('exiting')
```