PHYS306 Digital Electronics

Final Project: Automatic focusing of a camera on an object.

Objectives:

The primary objective of this project is to develop an advanced automatic focus system using OpenCV, Python, a Raspberry Pi, and a servo motor. The system aims to revolutionize the traditional manual focus adjustment process by automating it through the integration of various technologies.

The first objective is to design a mechanism that enables automated focus adjustment. This will be accomplished by employing a servo motor to rotate the camera lens, allowing for precise and remote adjustments. The servo motor will be controlled through the Raspberry Pi, enabling convenient and accurate focus changes without the need for manual intervention.

The second objective involves implementing an image processing algorithm using OpenCV and Python. This algorithm will be responsible for analyzing the sharpness and clarity of captured images to determine their focus status. The delentropy metric, which measures the image's entropy and provides a measure of its clarity, will be utilized for this purpose. By calculating the delentropy of each image, the algorithm will classify them as either focused or unfocused.

Real-time focus feedback constitutes the third objective. The system will provide immediate feedback on the focus status of the camera and save the image to raspberry pi when camera on the focus status.

The fourth objective is to seamlessly integrate the various components of the system using a Raspberry Pi. Acting as the central control unit, the Raspberry Pi will facilitate communication between the servo motor, the camera module, and the image processing algorithm. This integration will enable smooth coordination among these components, ensuring efficient and accurate automatic focus adjustments.

Lastly, the project aims to evaluate and optimize the system's performance. Through comprehensive evaluations and comparisons, the effectiveness and accuracy of the automatic focus system will be assessed. Potential optimizations will be explored to enhance the speed and precision of the focus adjustment mechanism and image processing algorithm. By continually refining and improving the system, it can offer a reliable and user-friendly solution for automatic focus adjustment in a wide range of applications, including photography, surveillance systems, and computer vision projects.

INTRODUCTION:

In the field of photography, achieving accurate focus is essential to capture sharp and clear images. Manual focus adjustments can be time-consuming and often require constant monitoring. To address this challenge, we present a project aimed at developing an automatic focus system using OpenCV, Python, a Raspberry Pi, and a servo motor. This system leverages the power of image processing algorithms and automation to achieve precise focus settings in real-time.

The primary objective of this project is to design and implement a reliable and user-friendly automatic focus mechanism that eliminates the need for manual intervention. By integrating OpenCV, a popular computer vision library, with a Raspberry Pi microcontroller and a servo motor, we aim to create a system that can dynamically adjust the camera focus based on image analysis.

The project employs the delentropy metric, a measure of image entropy, as the foundation for the image processing algorithm. By continuously analyzing the sharpness and clarity of captured images using delentropy, the algorithm determines the focus status. Once optimal focus is achieved, the system automatically captures a photo as immediate feedback.

The automatic focus system offers several advantages over traditional manual methods. Firstly, the system's ability to analyze and assess focus quality in real-time eliminates the need for constant manual monitoring, allowing users to focus on capturing the perfect shot.Additionally, the automatic focus system enhances the consistency and reliability of focus settings. By utilizing an automated mechanism driven by a servo motor, the system can achieve precise and repeatable focus adjustments, reducing the likelihood of human error or inconsistencies between shots.

The applications of this project extend beyond photography. For example, in surveillance systems, the automatic focus system can continuously monitor and adjust the focus of cameras, ensuring clear and sharp imagery of critical areas. In microscopy, the system can be utilized to maintain precise focus on specimens, aiding researchers in capturing detailed images for analysis.

Furthermore, in robotics and autonomous systems, the automatic focus system can be integrated into vision-based navigation algorithms. By adjusting the focus of cameras in real-time, robots can perceive their surroundings more accurately, leading to improved object recognition, obstacle detection, and path planning.

In conclusion, this project aims to create an automatic focus system that revolutionizes the traditional manual focus adjustment process. By combining OpenCV, Python, a Raspberry Pi, and a servo motor, we strive to provide photographers and other professionals with an efficient, reliable, and user-friendly solution for achieving precise focus settings. The subsequent sections of this report will delve into the details of the system's design, equipment used, procedures employed, calculations performed, and the results obtained, followed by discussions and comments on the system's performance and potential areas of improvement.

Equipment:

- Raspberry Pi v2: A version 2 of the Raspberry Pi, a small single-board computer.
- LM2596 Voltage Regulator
- 3.3V LED: An LED (Light-Emitting Diode) that operates on a voltage of 3.3V.
- MG995 Servo Motor:
- 12V:10A Power supply
- 3D Printed custom parts: Custom-designed parts that are manufactured using a 3D printing process, allowing for unique shapes and specifications.
- Couplings
- Trapezoidal lead screw
- 8mm Chrome-plated rod
- 5V:1.5A power supply
- 8mm roller
- 8mm gt2 pulley
- 8mm linear rollers

PROCEDURE:

SOFTWARE:

1. The Raspbian operating system will be installed on the raspberry pi card in the software part.
    a. Raspbian is a specialized operating system built specifically for the Raspberry Pi. It is based on the Linux Debian distribution and provides a lightweight and user-friendly environment for running applications on the Raspberry Pi hardware. Raspbian comes with pre-installed software packages and tools, making it a popular choice for various projects and applications.
2. OpenCV library will be downloaded to raspberry pi.
    a. OpenCV, or Open-Source Computer Vision Library, is a popular open-source software library used for computer vision and image processing tasks. It provides a wide range of functions and algorithms to analyze and manipulate images and videos. OpenCV is widely used in various fields, such as robotics, surveillance, augmented reality, and medical imaging. Its versatility and extensive documentation make it a valuable tool for developers working with computer vision applications.
3. Functions that can detect whether the captured image is clear or not will be investigated.
    a. The cv2.Laplacian function in OpenCV is used for performing the Laplacian edge detection on an image. It calculates the Laplacian gradient of the image, which highlights regions of rapid intensity changes or edges. The Laplacian operator is a second-order derivative operator that measures the rate of change of the image intensity. It is commonly used for edge detection because it responds strongly to areas of rapid intensity variation. The cv2.Laplacian function applies the Laplacian operator to each pixel in the image and computes the Laplacian gradient. The function takes the input image as its

parameter and returns the output image, which represents the edges or areas of rapid intensity changes. The resulting image is typically in grayscale, where brighter pixels indicate stronger edges. The cv2.Laplacian function uses a 3x3 kernel by default, but you can specify a different kernel size if needed. The kernel determines the neighborhood around each pixel that is used for calculating the Laplacian gradient. Overall, the cv2.Laplacian function is a powerful tool for detecting edges and areas of rapid intensity changes in an image, providing valuable information for various computer vision tasks such as object recognition, image segmentation, and feature extraction.

b. The cv2.GaussianBlur function in OpenCV is used for applying a Gaussian blur or smoothing effect to an image. Gaussian blur is a widely used image filtering technique that helps reduce noise and details in an image while preserving its overall structure. The function takes the input image and two additional parameters as input: the kernel size and the standard deviation of the Gaussian distribution. The kernel size determines the size of the filter window, which defines the neighborhood of pixels used for the blurring operation. The standard deviation controls the spread of the Gaussian distribution and influences the amount of blurring applied. When the cv2.GaussianBlur function is called, it convolves each pixel in the image with a Gaussian kernel. The convolution process involves calculating the weighted average of the pixel values within the kernel window. The weights are determined by the Gaussian distribution, with the highest weight assigned to the central pixel and decreasing weights assigned to the surrounding pixels. By convolving the image with the Gaussian kernel, the function effectively replaces each pixel value with the weighted average of its neighborhood, resulting in a smoothed and blurred version of the original image. The amount of blurring depends on the kernel size and standard deviation parameters. Larger kernel sizes and higher standard deviations result in stronger blurring effects. The cv2.GaussianBlur function is commonly used as a preprocessing step in various computer vision tasks. It helps reduce noise, remove high-frequency details, and create smoother images that are more suitable for subsequent operations such as edge detection, feature extraction, or image recognition.

c. The detect_focus_delentropy function aims to assess the focus measure of a grayscale image using the delentropy method. This method focuses on analyzing the distribution of gradients, which are changes in pixel intensities across neighboring regions. To calculate the focus measure, the function first computes the horizontal and vertical gradients of the image. This is done by subtracting the pixel values of adjacent columns and rows, respectively. By combining these gradients, an overall gradient image is obtained, representing the magnitude and direction of changes in pixel intensities. Next, the function constructs a 2D histogram based on the gradient values, which captures the distribution of these values across the image. The histogram is normalized to ensure that each bin represents a proportional frequency of gradient occurrences. The histogram is then processed by applying the logarithm (base 2) to each bin value, resulting in delentropy values. These delentropy values represent the information content or complexity associated with different gradient magnitudes. By summing up these delentropy values, an overall

measure of the image's entropy is obtained. Dividing the computed entropy by 2, following the Papoulis generalized sampling method, yields the final focus measure. This measure serves as a quantitative indicator of the image's sharpness and clarity, allowing for the determination of the optimal focus setting for capturing high-quality images. In summary, the detect_focus_delentropy function utilizes the delentropy method to evaluate the focus quality of a grayscale image by analyzing the distribution of gradients. This approach provides valuable insights into the image's level of detail and sharpness, facilitating the identification of the ideal focus point for image capture.

ELECTRONICS:

1. The raspberry pi v2 will be powered and operated.
    a. The Raspberry Pi 2 is a small, affordable, and versatile single-board computer. It offers improved specifications and features compared to previous models. With its compact size and GPIO pins for connecting external components, it is widely used for educational, DIY, and IoT projects. It runs various operating systems and provides a platform for learning and innovation.
2. MG995 servo motor will be driven with the help of raspberry pi and the angle variable will enable the motor to go to the desired point.
    a. The MG995 servo motor is a popular type of analog servo motor commonly used in robotics and automation projects. This servo motor is known for its high torque output and accuracy. It operates based on the principle of pulse width modulation (PWM). The motor consists of a DC motor, a control circuit, and a feedback potentiometer. When a PWM signal is applied to the servo motor, it interprets the signal's pulse width to determine the desired position. The PWM signal typically has a frequency of 50 Hz, with a pulse width ranging from 1 ms to 2 ms. The motor's control circuit uses this pulse width information to drive the DC motor to the corresponding position. The control circuit inside the servo motor compares the received pulse width with the current position feedback from the potentiometer. If the two values differ, the control circuit adjusts the motor's speed and direction to move towards the desired position. Once the position matches the input signal, the control circuit stops the motor, maintaining the position until a new signal is received.
3. The Raspberry Pi HQ Camera Module camera will be connected to Raspberry Pi and image will be taken.
    a. The Raspberry Pi HQ Camera Module is an official camera accessory designed specifically for the Raspberry Pi single-board computer. It is a high-quality camera module that offers advanced imaging capabilities for various applications. The HQ Camera Module features a 12.3-megapixel Sony IMX477 image sensor, which provides excellent image quality and low-light performance. It supports interchangeable lenses, allowing users to choose different focal lengths and lens types based on their specific needs. This flexibility enables a wide range of photography and video recording possibilities. The camera module connects directly to the Raspberry Pi board via a dedicated CSI (Camera Serial Interface) port, ensuring a seamless integration and easy setup. It is supported by the official Raspberry Pi camera

software, which provides a user-friendly interface and a range of features for capturing and processing images. With the Raspberry Pi HQ Camera Module, users can capture high-resolution still images, record videos in various formats, and apply image processing techniques using the Raspberry Pi's computing power. It is suitable for applications such as surveillance systems, home automation, robotics, scientific imaging, and more.

4. With the LM2596, the output of the 12V power supply will be reduced to 3.3V and the LED will be operated.
    a. The LM2596 voltage regulator is a versatile integrated circuit (IC) commonly used for voltage regulation. It efficiently converts higher input voltages to lower output voltages, providing stable power for electronic devices. Its adjustable output voltage and built-in protection features make it a popular choice for various applications.

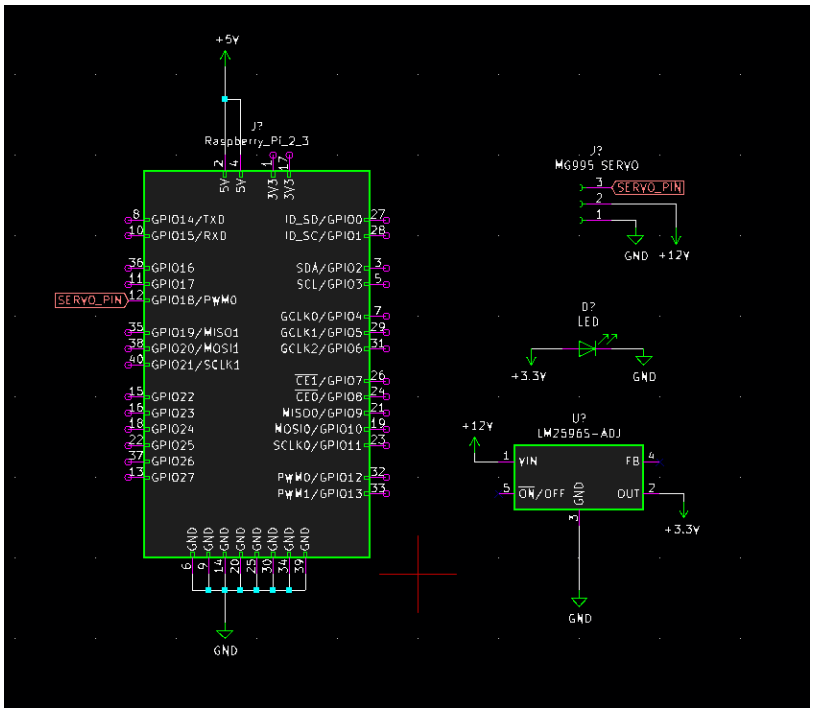


Figure 1: The circuit schematic of project

MECHANICHS:

1. Spur gears will be designed to control the focus ring of the lens by servo motor.
    a. An adjustable focus lens with a ring refers to a type of lens that allows the user to manually adjust the focus by rotating a ring on the lens barrel. This ring controls the position of lens elements, changing the distance between them and adjusting the focal length of the lens. By rotating the ring, the user can achieve a sharp focus at different distances, allowing for versatile focusing capabilities in photography or other optical applications. This feature provides greater control and flexibility in capturing clear and detailed images or videos.
    b. Module 2 spur gears refer to a type of gear mechanism that uses cylindrical gears with a standardized module of 2. The module represents the size and spacing of the

gear teeth and determines their compatibility with other gears. Spur gears have straight teeth that run parallel to the gear axis and mesh together to transfer rotational motion between two parallel shafts. The module 2 designation indicates the specific size and pitch of the gear teeth, allowing for precise gear meshing and smooth power transmission. Module 2 spur gears are commonly used in various mechanical systems, such as robotics, machinery, and automotive applications, where reliable and efficient torque transfer is required.

c. The utilization of gears with 27 and 26 teeth was employed as a preventive measure against gear wear. The module 2 spur gears with these specific tooth counts were chosen to ensure a smooth and efficient transmission of motion. By using gears with different tooth counts, the system can distribute the load and minimize wear on individual gear teeth. This design choice enhances the overall durability and longevity of the gear mechanism, contributing to the reliable functioning of the system.

2. The x and y axes will be fixed with the help of two chrome shafts and linear bearings and the z axis will be moved using the trapezoidal shaft.
    a. Linear rollers are mechanical components used for smooth and precise linear motion. They consist of cylindrical rollers mounted in a housing, allowing objects or loads to move along a straight path with minimal friction. They find applications in conveyor systems, sliding doors, robotics, and CNC machines, providing reliable and efficient linear movement.
    b. A trapezoidal lead screw, also known as a trapezoidal thread or trapezoidal mill, is a type of mechanical component used for converting rotary motion into linear motion or vice versa. It features a trapezoidal-shaped thread profile on the surface, which allows for efficient and precise movement along its length when coupled with a corresponding nut. Trapezoidal lead screws are commonly used in applications such as CNC machines, 3D printers, and industrial machinery, offering reliable and accurate linear positioning capabilities.
3. Fixing parts designed from CAD programs will be printed with the 3D printer from PLA material.
    a. PLA (Polylactic Acid) is a commonly used thermoplastic material in 3D printing. It is derived from renewable resources such as corn starch or sugarcane, making it environmentally friendly. PLA is known for its ease of use, low toxicity, and biodegradability. It has become popular in 3D printing due to its good printability, dimensional accuracy, and relatively low melting temperature.
    b. 3D printing, also known as additive manufacturing, is a process of creating three-dimensional objects by depositing successive layers of material. It involves the use of a 3D printer, which reads a digital 3D model file and builds the object layer by layer. The printer melts or softens the PLA filament and extrudes it through a nozzle to create the desired shape. As each layer is added and solidified, the object gradually takes form. 3D printing offers versatility, allowing the production of complex geometries, prototypes, customized products, and even functional parts for various industries.

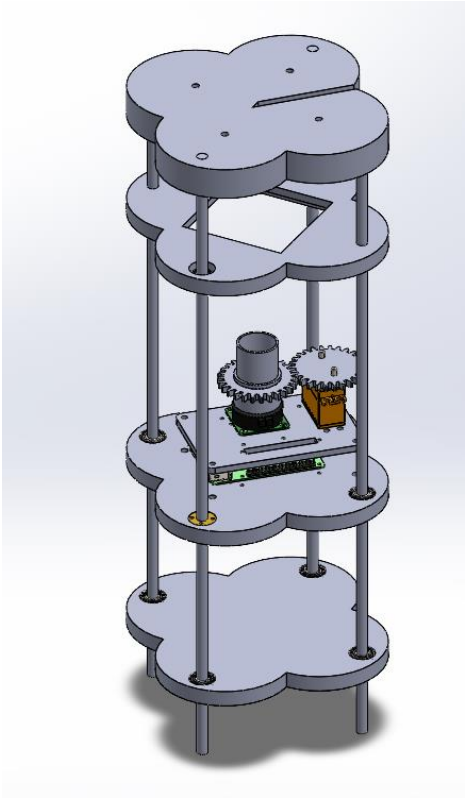


Figure 2: The CAD drawing of mechanical parts of project

CALCULATİONS AND RESULTS:

SOFTWARE:

In the software development phase, the Raspberry Pi was initially set up with the Raspbian operating system, which serves as the foundation for running various applications. Subsequently, the OpenCV libraries were installed to leverage their extensive capabilities for image processing and analysis.

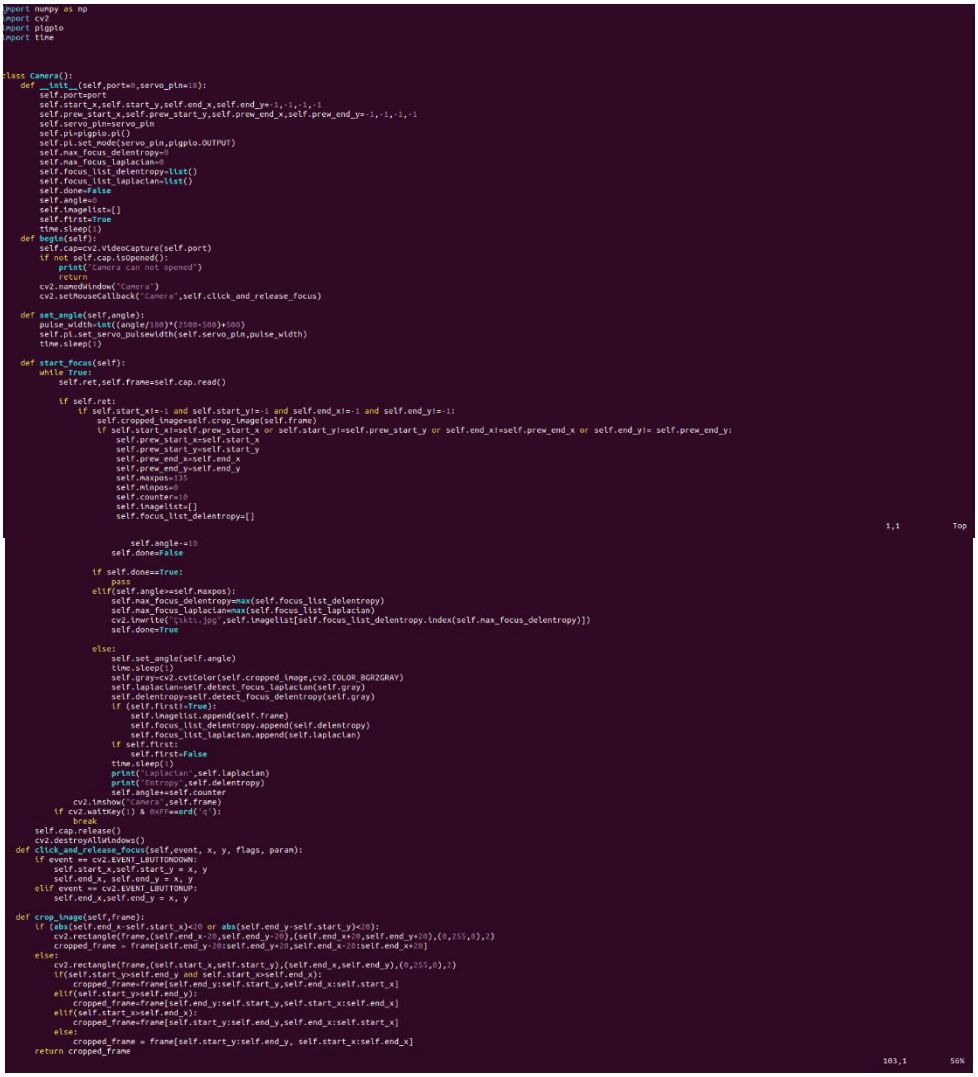
During the initial research phase, the Laplacian function, available in the OpenCV library, was explored as a potential method for assessing image clarity. The Laplacian function showed promising results in determining image sharpness. However, it became evident that relying solely on the Laplacian function was not sufficient, particularly in low-light conditions where its effectiveness diminished. To overcome this limitation, an adaptation of the Laplacian function using Gaussian blur was implemented. This modification aimed to improve performance in different lighting conditions.

However, the system's performance on the Raspberry Pi, which has relatively lower processing power compared to personal computers, proved to be insufficient. The computational demands of the modified Laplacian function pushed the Raspberry Pi to its limits, causing freezing and instability. As a result, alternative approaches were explored to accurately measure image sharpness.

Through further investigation, the delentropy method was discovered as a viable alternative. The delentropy method proved to be reliable, particularly in low-light or high-light conditions. It was deemed more suitable for the project as it did not excessively strain the system. Consequently, the decision was made to adopt the delentropy method as the primary approach for assessing image sharpness.

To enhance program efficiency, a refinement was made by calculating the entropy of the specific point where focus was desired, rather than calculating the entropy of each pixel in the entire image. This optimization not only improved the program's efficiency but also provided users with the flexibility to focus on their desired point of interest. To facilitate point selection, the OpenCV function "setmousecallback" was utilized, enabling users to choose the desired focus point using a mouse input.

This iterative process of exploring different methods, adapting algorithms to varying lighting conditions, and ultimately adopting the delentropy method with optimized focus point selection has yielded a more robust and efficient automatic focus system. The subsequent sections of this report will delve into the specific procedures, calculations, and results obtained, followed by discussions on the system's performance and potential avenues for improvement.



```
import numpy as np
import cv2
import pigpio
import time


class Camera():
    def __init__(self,port=0,servo_pin=18):
        self.port=port
        self.start_x,self.start_y,self.end_x,self.end_y=-1,-1,-1,-1
        self.prew_start_x,self.prew_start_y,self.prew_end_x,self.prew_end_y=-1,-1,-1,-1
        self.servo_pin=servo_pin
        self.pi=pigpio.pi()
        self.pi.set_mode(servo_pin,pigpio.OUTPUT)
        self.max_focus_delentropy=0
        self.max_focus_laplacian=0
        self.focus_list_delentropy=list()
        self.focus_list_laplaclan=list()
        self.done=False
        self.angle=0
        self.imagelist=[]
        self.first=True
        time.sleep(1)
    def begin(self):
        self.cap=cv2.VideoCapture(self.port)
        if not self.cap.isOpened():
            print("Camera can not opened")
            return
        cv2.namedWindow("Camera")
        cv2.setMouseCallback("Camera",self.click_and_release_focus)

    def set_angle(self,angle):
        pulse_width=int((angle/180)*(2500-500)+500)
        self.pi.set_servo_pulsewidth(self.servo_pin,pulse_width)
        time.sleep(1)

    def start_focus(self):
        while True:
            self.ret,self.frame=self.cap.read()

            if self.ret:
                if self.start_x!=-1 and self.start_y!=-1 and self.end_x!=-1 and self.end_y!=-1:
                    self.cropped_image=self.crop_image(self.frame)
                    if self.start_x!=self.prew_start_x or self.start_y!=self.prew_start_y or self.end_x!=self.prew_end_x or self.end_y!= self.prew_end_y:
                        self.prew_start_x=self.start_x
                        self.prew_start_y=self.start_y
                        self.prew_end_x=self.end_x
                        self.prew_end_y=self.end_y
                        self.maxpos=135
                        self.minpos=0
                        self.counter=10
                        self.imagelist=[]
                        self.focus_list_delentropy=[]
                                                                                    1,1           Top

                        self.angle-=10
                    self.done=False

                if self.done==True:
                    pass
                elif(self.angle>=self.maxpos):
                    self.max_focus_delentropy=max(self.focus_list_delentropy)
                    self.max_focus_laplaclan=max(self.focus_list_laplaclan)
                    cv2.imwrite("{}ktt.jpg",self.imagelist[self.focus_list_delentropy.index(self.max_focus_delentropy)])
                    self.done=True

                else:
                    self.set_angle(self.angle)
                    time.sleep(1)
                    self.gray=cv2.cvtColor(self.cropped_image,cv2.COLOR_BGR2GRAY)
                    self.laplaclan=self.detect_focus_laplaclan(self.gray)
                    self.delentropy=self.detect_focus_delentropy(self.gray)
                    if (self.first!=True):
                        self.imagelist.append(self.frame)
                        self.focus_list_delentropy.append(self.delentropy)
                        self.focus_list_laplaclan.append(self.laplaclan)
                    if self.first:
                        self.first=False
                    time.sleep(1)
                    print("Laplaclan",self.laplaclan)
                    print("Entropy",self.delentropy)
                    self.angle+=self.counter
            cv2.imshow("Camera",self.frame)
        if cv2.waitKey(1) & 0xFF==ord('q'):
            break
    self.cap.release()
    cv2.destroyAllWindows()
    def click_and_release_focus(self,event, x, y, flags, paran):
        if event == cv2.EVENT_LBUTTONDOWN:
            self.start_x,self.start_y = x, y
            self.end_x, self.end_y = x, y
        elif event == cv2.EVENT_LBUTTONUP:
            self.end_x,self.end_y = x, y

    def crop_image(self,frame):
        if (abs(self.end_x-self.start_x)<20 or abs(self.end_y-self.start_y)<20):
            cv2.rectangle(frame,(self.end_x-20,self.end_y-20),(self.end_x+20,self.end_y+20),(0,255,0),2)
            cropped_frame = frame[self.end_y-20:self.end_y+20,self.end_x-20:self.end_x+20]
        else:
            cv2.rectangle(frame,(self.start_x,self.start_y),(self.end_x,self.end_y),(0,255,0),2)
            if(self.start_y>self.end_y and self.start_x>self.end_x):
                cropped_frame=frame[self.end_y:self.start_y,self.end_x:self.start_x]
            elif(self.start_y>self.end_y):
                cropped_frame=frame[self.end_y:self.start_y,self.start_x:self.end_x]
            elif(self.start_x>self.end_x):
                cropped_frame=frame[self.start_y:self.end_y,self.end_x:self.start_x]
            else:
                cropped_frame = frame[self.start_y:self.end_y, self.start_x:self.end_x]
        return cropped_frame
                                                                                103,1         56%
```

Figure 3: The first part of project code

```
        elif(self.start_y>self.end_y):
            cropped_frame=frame[self.end_y:self.start_y,self.start_x:self.end_x]
        elif(self.start_x>self.end_x):
            cropped_frame=frame[self.start_y:self.end_y,self.end_x:self.start_x]
        else:
            cropped_frame = frame[self.start_y:self.end_y, self.start_x:self.end_x]
    return cropped_frame
def detect_focus_laplacian(self,frame):
    laplacian=cv2.Laplacian(self.gray,cv2.CV_64F)
    variance=laplacian.var()
    return variance
def detect_focus_delentropy(self,greying):
    # $\nabla f(n) \approx f(n) - f(n - 1)$
    fx = greying[:, 2:] - greying[:, :-2]
    fy = greying[2:, :] - greying[:-2, :]
    # fix shape
    fx = fx[1:-1, :]
    fy = fy[:, 1:-1]

    grad = fx + fy

    # ensure $-255 \leq J \leq 255$
    #jrng = np.max([np.max(np.abs(fx)), np.max(np.abs(fy))])
    #assert jrng <= 255, "J must be in range [-255, 255]"

    ### 1609.01117 page 16, eq 17

    hist, _, _ = np.histogram2d(fx.flatten(),fy.flatten(),bins=256,)

    ### 1609.01117 page 20, eq 22

    deldensity = hist / np.sum(hist)
    deldensity = deldensity * -np.ma.log2(deldensity)
    entropy = np.sum(deldensity)

    entropy /= 2  # 4.3 Papoulis generalized sampling halves the delentropy

    #print(f"entropy: {entropy}",f"entropy ratio: {entropy / 8.0}",)

    # the reference image seems to be bitwise inverted, I don't know why.
    # the entropy doesn't change when inverted, so both are okay in
    # the previous computational steps.
    param_invert = True

    #gradImg = np.invert(grad) if param_invert else grad

    return (entropy)



camera=Camera()
camera.begin()
camera.start_focus()
                                                                    161,1         Bot
```

Figure 4: The second part of project code

```
import pigpio
import time

servo_pin=18

pi=pigpio.pi()
pi.set_mode(servo_pin,pigpio.OUTPUT)

def set_angle(angle):
    pulse_width= int((angle/180)*(2500-500)+500)
    pi.set_servo_pulsewidth(servo_pin,pulse_width)
    time.sleep(1)

angle=0
set_angle(angle)
counter=10
while True:
    if(angle>=135):
        angle=0
    set_angle(angle)
    angle+=counter
pi.stop()
```

Figure 5: The test code of servo motor

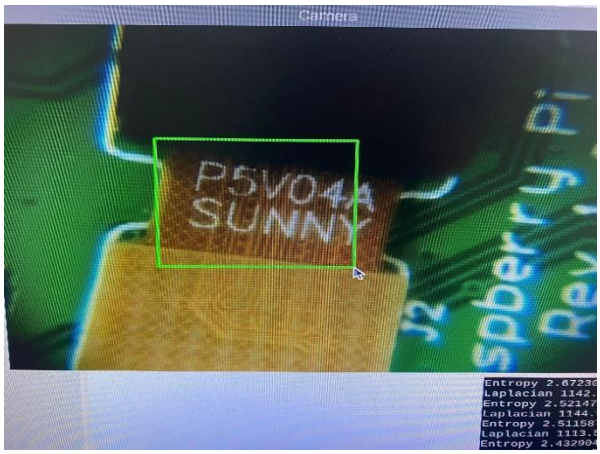


Figure 6: The Delentropy function focus adjustment

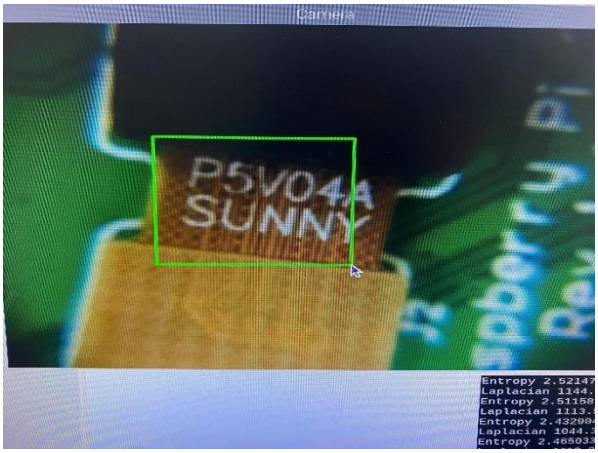


Figure 7:The Laplacian function focus adjustment

ELECTRONICS:

In the electronics aspect of the project, the initial objective was to achieve focus adjustments by moving the camera in the x and y axes and zooming in and out of the object. This approach was considered due to the unavailability and high cost of lenses with adjustable focus. However, a viable alternative was found by acquiring a microscope lens, which enabled focus adjustments by manipulating the ring part of the lens instead of physically moving the camera system. To achieve this, an MG995 servo motor was deemed suitable for controlling the movement of the lens ring. Compared to the previous system, which utilized a stepper motor for x and y axis movements, this new configuration not only facilitated more precise focus adjustments but also reduced system complexity.

Since the MG995 servo motor operates on 12V, a dedicated 12V power supply was utilized, and the connections were established as depicted in the provided figure. The PWM-controlled output of the MG995 servo motor was connected to the gpio18 pin of the Raspberry Pi, while the grounds of both the Raspberry Pi and the 12V power supply were shared. Initially, the RPi library, one of the main libraries for Raspberry Pi, was employed for control. However, it was later discovered that the servo exhibited vibrations, known as servo jitter, which were attributed to the suboptimal performance of the PWM signal generator in the RPi library. As a remedy, the PiGPIO library was adopted, as its PWM signal generator functions proved to be more optimized, effectively eliminating servo jitter.

Additionally, the use of a microscope lens required bringing the lens very close to the object of observation. As a result, in some cases, insufficient light entered the lens, resulting in subpar image quality. To address this issue, a 3.3V LED lamp was incorporated beneath the system. To avoid potential damage to the Raspberry Pi, as the 3.3V lamp drew approximately 0.8A, an LM2596 voltage regulator was employed to step down the voltage from 12V to 3.3V.

The LED lamp was then powered from this regulated 3.3V source. Furthermore, a button was connected to control the on/off operation of the LED, allowing for flexibility in different lighting conditions.

Efforts were made to convert the 12V supply to 5V for the Raspberry Pi using an LM2596 voltage regulator. However, due to the unavailability of a suitable connector, manual production of the connector was attempted. Nevertheless, issues such as poor contact and potential short circuits arose at certain points, rendering the connection unreliable. Consequently, it was deemed more appropriate to power the Raspberry Pi separately with a dedicated 5V power source. As both power sources shared a common ground, no adverse effects were observed from the utilization of these two separate power feeds.

These developments in the electronics aspect of the project, including the adoption of a microscope lens with servo-controlled focus adjustments, the integration of an LM2596 voltage regulator for precise power management, and the implementation of the PiGPIO library to address servo jitter, have significantly enhanced the overall functionality and performance of the automatic focus system.

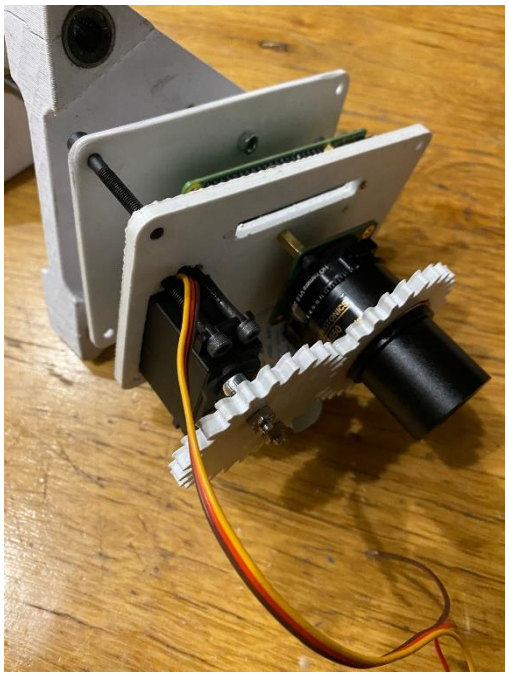


Figure 8: Raspberry Pi, Camera module, MG995 Servo motor combination

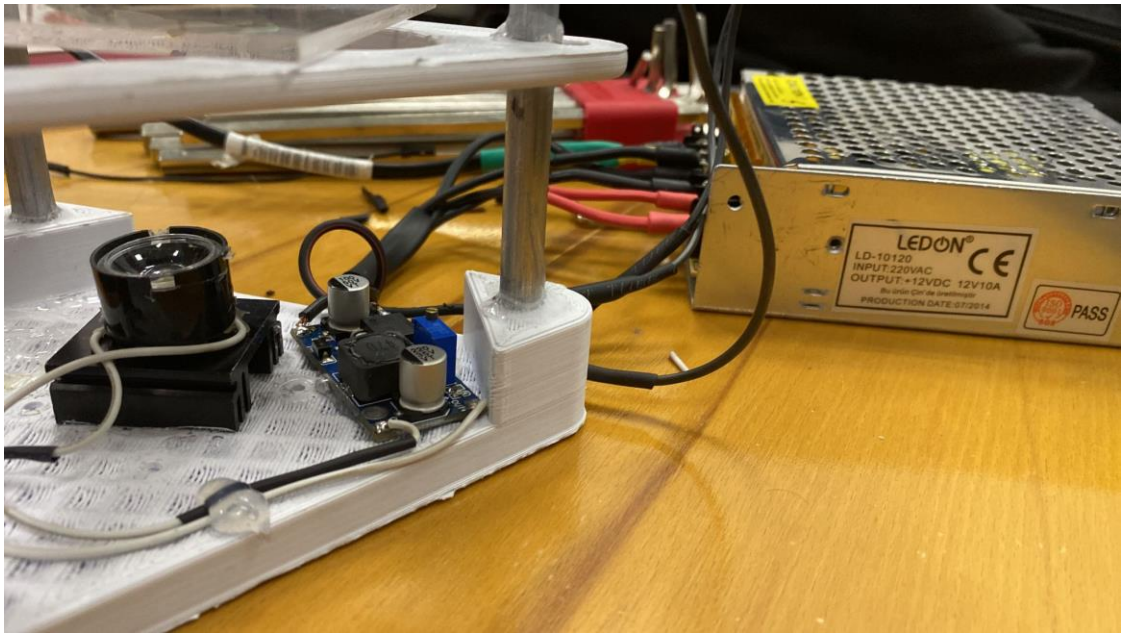


Figure 9: LED, LM2596 and 12V DC Power supply combination

MECHANICS:

In the mechanical aspect of the project, a series of design iterations were conducted to achieve optimal focus adjustment with the given constraints. The initial design focused on camera movement in the x and y axes to determine the ideal focus distance. However, it was observed that this approach introduced complexities and increased the overall size of the system, rendering it impractical for practical use.

To address this challenge, a new design was proposed that centered around a specific component placement, as shown in the provided figure. This design emphasized the integration of the Raspberry Pi and the servo motor, forming a compact and efficient focal adjustment module. By attaching conventional camera lenses to this module, it would be possible to achieve precise focus control and capture images without relying on additional system components.

Nevertheless, due to the utilization of a microscope lens in the current system, close proximity between the lens and the observed object was crucial. To validate the system's performance, a z-axis moving system was designed, while the x and y axes were fixed (Fig.2). To enable z-axis movement, a trapezoidal shaft was employed, coupled with a bearing and a coupling mechanism to connect it to the Raspberry Pi. This design ensured that the rotational motion of the trapezoidal shaft did not interfere with the overall system operation.This configuration allowed for testing focus accuracy at various distances. However, it was discovered that the initially proposed design exceeded the manufacturing capabilities of existing 3D printers, necessitating further refinement.

As a result, a final design was developed to overcome the production limitations. This revised design incorporated two gear systems to establish a connection between the servo motor and the motion ring of the lens. With precise positioning facilitated by the servo motor, the lens could be effectively moved using these gears.

Notably, it should be mentioned that the absence of the 3rd chrome shaft in the current system configuration (Fig.12) was due to unforeseen production tolerances with the 3D printer. Including the 3rd chrome shaft resulted in the system becoming jammed and non-functional.

Overall, the mechanical design went through an iterative process to address the specific requirements of achieving precise focus control with a microscope lens. By optimizing the design and considering the limitations of 3D printing, the final configuration successfully enabled accurate movement and positioning of the lens to attain the desired focus. The refined design showcased the importance of adaptability and innovative problem-solving in developing practical and efficient mechanical systems.

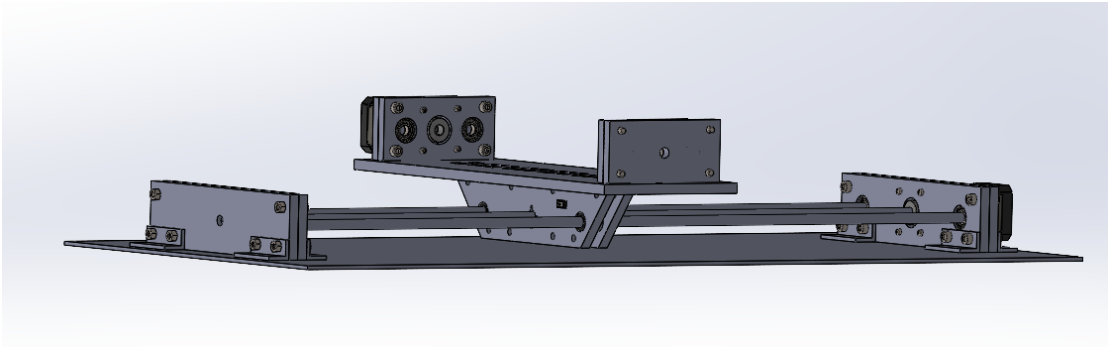


Figure 10: First design of project X-Y axis movement system with stepper motors

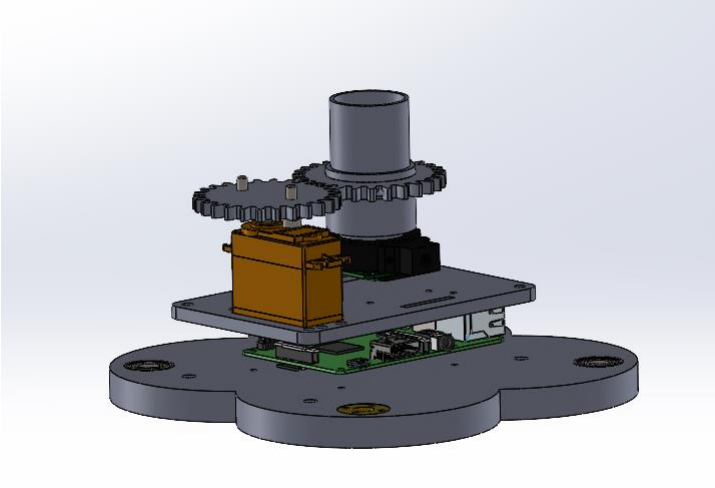


Figure 11: Final CAD design of project electronic components connection parts

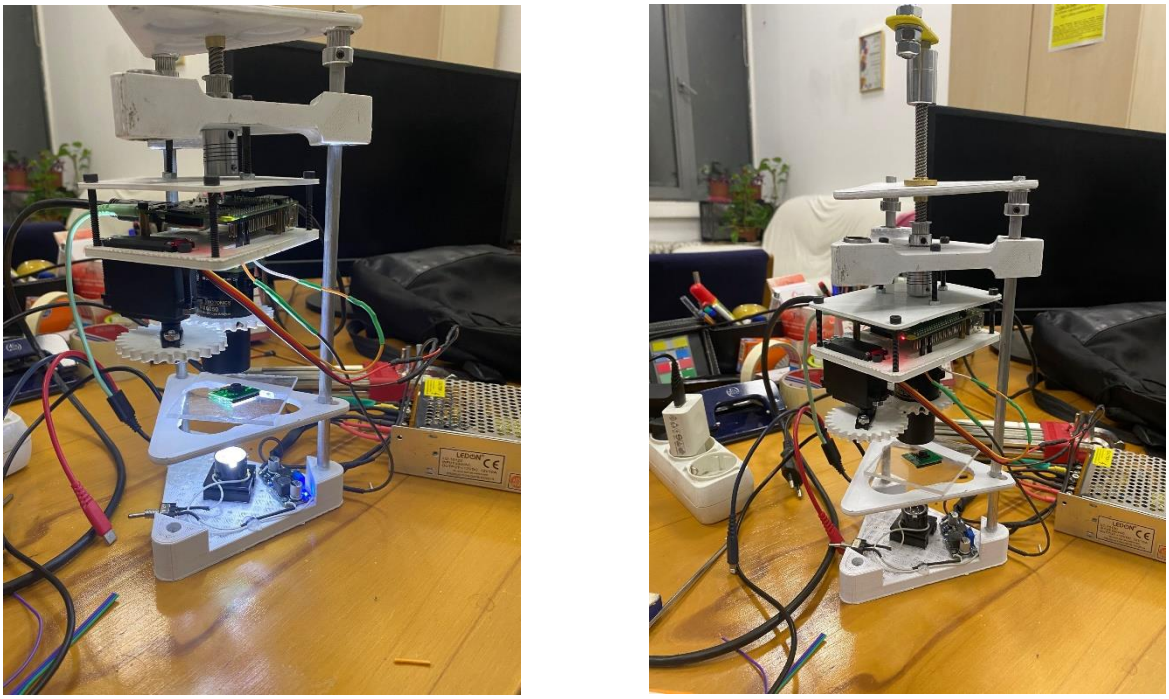


Figure 12: The final image of the project

Discussions and Comments:

The automatic camera focusing system developed in this project offers several advantages over traditional manual focusing methods. It eliminates the need for constant monitoring and adjustment of the focus ring, thereby saving valuable time and effort for photographers. By employing sophisticated image processing algorithms, the system provides real-time feedback on the sharpness and clarity of images, enabling photographers to achieve precise focus without relying on manual adjustments.

The integration of various components, including the Raspberry Pi, servo motor, camera module, and image processing algorithm, results in a seamless and automated solution. The Raspberry Pi serves as the central control unit, facilitating communication among different components and enabling precise focus adjustments through the servo motor. The image processing algorithm, implemented using OpenCV and Python, effectively analyzes image clarity, allowing the system to quickly determine the focus status of an image.

The focal point of the system lies in the meticulously designed mechanism responsible for controlling the focus ring using a servo motor. With the aid of 3D printed custom parts, couplings, a trapezoidal lead screw, and other essential components, the system achieves precise and remote focus adjustments. This mechanism, in conjunction with the image processing algorithm, forms a closed-loop system that continuously monitors and adjusts the focus to maintain optimal sharpness.

To assess the performance of the system, it is recommended to conduct thorough evaluations regarding its accuracy and speed in focus adjustments. A comparative analysis between the automatic focus system and manual focusing techniques can be performed to gauge the system's effectiveness. Additionally, gathering user feedback and conducting usability testing can provide valuable insights for further refinement and enhancement.

The automatic camera focusing system presented in this project demonstrates the potential of combining image processing, automation, and mechanical control to provide an efficient and user-friendly solution for achieving optimal focus in real-time. Its numerous advantages, including time-saving capabilities, enhanced accuracy, and overall convenience, position it as a superior alternative to manual focusing methods. Future research and development endeavors could be directed towards optimizing the system's performance and exploring potential applications beyond the realm of photography.