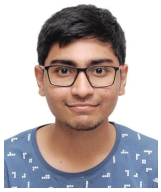
Ahmedabad University

Pills Defect Detection Using Image Processing

Embedded Systems, Monsoon 2023

Professor Sanket Patel

Photos	Name of the Student	Contact Number	Email ID
	Aagam Shah (AU2140024)	9687568575	aagam.s2@ahduni.edu.in
	Harsh Loriya (AU2140154)	8200021865	harsh.l2@ahduni.edu.in
	Kunj Kanzariya (AU2140202)	9824243739	kunj.kx@ahduni.edu.in
	Preet Patel (AU2140034)	9825290285	preet.p3@ahduni.edu.in

Contribution:

Aagam Shah	**-GUI** **-Debugging** **-Testing** **-Image Processing Code**
Harsh Loriya	**-Arduino Code** **-Image Processing Code** **-Implementation of Circuit** **-Implementation of Conveyor belt**
Kunj Kanzariya	**-GUI** **-Image Processing Code** **-Implementation of Conveyor belt**
Preet Patel	**-Ultrasonic sensor circuit + code** **-Image Processing Code** **-Implementation of interface**

Motivation:

Strict quality control is necessary to fulfil the pharmaceutical industry's promise to provide safe and effective treatments. To ensure pill quality, traditional inspection techniques, which are frequently prone to errors, need assistance. In addition to endangering patient health, defective medications might have legal consequences for their manufacturers.

The pharmaceutical industry faces several difficulties. The human evaluation-based inspection techniques used today are labour-intensive and prone to error. Efficient and precise quality control procedures are essential as pharmaceutical demand rises.

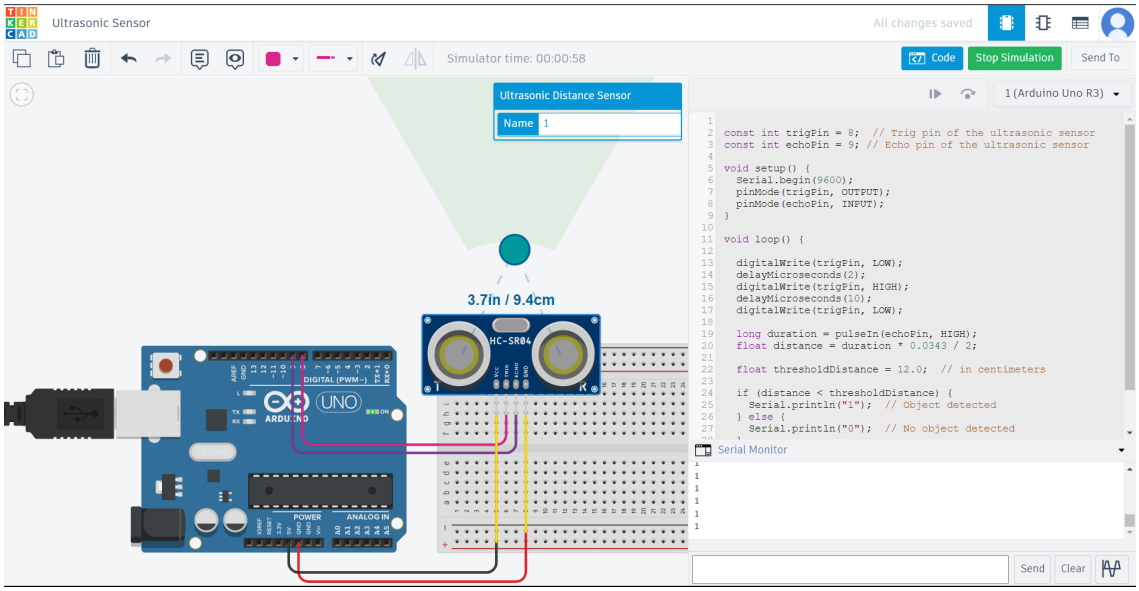
One potential option is automated image processing, which provides quick and accurate pill flaw identification. This technique addresses the drawbacks of conventional approaches by enhancing inspection speed and quality through sophisticated picture analysis.

Introduction:

The necessity of quality control in the pharmaceutical industry led to the use of cutting-edge technologies. Using a circular wheel with an integrated camera for pill examination is one such invention. This system aims to automate the entire pill inspection procedure, instead of using traditional methods, which is slow and less accurate. It is a modern solution to that problem.

The automated circular wheel and camera combination into an image processing-based pill defect detection algorithm is the main goal of this project. The built-in camera records detailed photos as the pills move along the circular wheel, and computer vision algorithms that are based on powerful image processing quickly analyze this data. This application increases inspection speed and accuracy while reducing the need for human intervention.

Circuit Diagram and simulation:



Our circuit only contains an ultrasonic sensor for detecting medicine strips.

TinkerCAD link:
https://www.tinkercad.com/things/g1qWRjBiY4I-ultrasonic-sensor?sharecode=bRkFFBaFW6ZDjMaCLe3jMnB675Cp3JBIKcaYGo7bIB4

List of components:

Components	Working	Price	Order link
Arduino UNO kit	Provides all the elements used to build the circuit such as breadboard, jumper wires, arduino board, etc.	Provided by university (Rs. 475)	https://www.amazon.in/Robodo-Board-compatible-Arduino-Development/dp/B094FZWYY4/ref=sr_1_5?crid=2MWI3UKU8XWXK&keywords=arduino%2Buno

			%2Bkit&qid=1702458747&sprefix=aurdino%2Buno%2Bkit%2Caps%2C191&sr=8-5&th=1
Ultrasonic sensor	Used to detect the medicine strip when it enters the range of the sensor.	Provided by university (Rs. 185)	https://www.amazon.in/Robotbanao-HC-SR04-Ultrasonic-Distance-Arduino/dp/B07FSHJSTN/ref=sr_1_1_sspa?crid=HDJ7L3F3IE9S&keywords=ultrasonic%2Bsensor&qid=1702456079&sprefix=selfie%2Bstick%2Caps%2C5514&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1
Camera (We are using the phone's camera)	Used to capture the image of medicine strips which is then further processed.	Preet's phone (Alternative: webcam Rs. 553)	https://www.amazon.in/FRONTECH-Digital-Interface-Streaming-2254/dp/B0CLYF8KTJ/ref=sr_1_19?crid=1OTPQIBUXA2HR&keywords=webcam&qid=1702458671&sprefix=webcam%2Caps%2C271&sr=8-19&th=1
Rotating base	Works as an alternative to a conveyor belt where the medicine strips are kept.	Provided by Prof. Sanket Patel (Alternative: Rotating base Rs. 1394)	https://www.amazon.in/Hardware-Turntable-Bearings-Plate-Revolving/dp/B0C1N56V7N/ref=sr_1_30?crid=2P3HU2DPT68AI&keywords=

			rotating%2Bbase&qid=1702458896&sprefix=rotating%2Bbas%2Caps%2C726&sr=8-30&th=1
Selfie stick	Works as a stand to hold the mobile phone and provides angle flexibility.	Aagam's selfie stick (Rs. 299)	https://www.amazon.in/Extendable-Wireless-Lightweight-Compatible-Smartphone/dp/B08XJZDDJP/ref=sr_1_5?crid=HJ7LZDDUNT0V&keywords=selfie+stick&qid=1702455994&sprefix=selfie+stick%2Caps%2C1370&sr=8-5
Monitor (We are using the laptop's screen)	Works as a display for users to see the results.	Aagam's laptop (Alternative: Monitor Rs. 3499)	https://www.amazon.in/FRONTECH-Monitor-Refresh-Mountable-MON-0054/dp/B0C5RWGPF7/ref=sr_1_1_sspa?crid=396I1ECQ5PWWK&keywords=screen+display&qid=1702459086&sprefix=screen+displa%2Caps%2C275&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

Explanation of the circuit:

The circuit has to detect that a packet of pills has come in front of the camera so that the camera can take the photo of the packet. The packets are in constant rotation as they are on a rotating table so whenever they come in front of the ultrasonic sensor the distance will be less than the threshold we have set so we can tell that a packet has been detected and we can tell the camera to take the photo.

Ultrasonic Sensor (HC-SR04):

- **Trig Pin (trigPin):** Connected to the digital pin 8 on the Arduino. This pin triggers the ultrasonic sensor to send out a pulse.
- **Echo Pin (echoPin):** Connected to the digital pin 9 on the Arduino. This pin receives the echo signal, and its duration is used to calculate the distance.

Arduino Board:

Used to control and interact with the ultrasonic sensor.

Ultrasonic Sensor Connection:

- The ultrasonic sensor has four pins: VCC, GND, Trig, and Echo.
- VCC and GND are connected to the 5V and GND pins on the Arduino, respectively.
- TrigPin is connected to digital pin 8, and EchoPin is connected to digital pin 9.

Setup Function (void setup()):

- Initializes serial communication at 9600 bps. This allows you to view the output on the serial monitor in the Arduino IDE.
- Sets TrigPin as an OUTPUT and EchoPin as an INPUT.

Loop Function (void loop()):

- Sends a short pulse to the TrigPin, triggering the ultrasonic sensor.
- Measures the duration of the pulse received on the EchoPin using pulseIn() function.
- Calculates the distance to the detected object using the speed of sound (approximately 343 meters/second) and the time taken for the pulse to return.
- Compare the calculated distance with a threshold distance (thresholdDistance). If the distance is less than the threshold, it indicates an object is within range.
- Prints "1" to the serial monitor if an object is detected; otherwise, it prints "0".

- The delay of 100 milliseconds is added to control the rate of readings. Adjust this delay as needed.

The speed of sound is used to convert the time duration into distance. The formula used is distance = duration * speed_of_sound / 2.

The threshold distance (thresholdDistance) is set to 12 centimeters in the code. You can adjust this value based on your specific application and requirements.

Measurement and Calibration:

We position the ultrasonic sensor in such a way such that whenever a packet passes through, we get distance as output. Then by trying different positions of the packet we will get what average distance the ultrasonic sensor outputs and we will set this distance as threshold distance.

So whenever a packet is passing through the ultrasonic sensor, the ultrasonic will give the distance between it and the packet and whenever that distance is less than or equal to the threshold distance the camera sensor will activate.

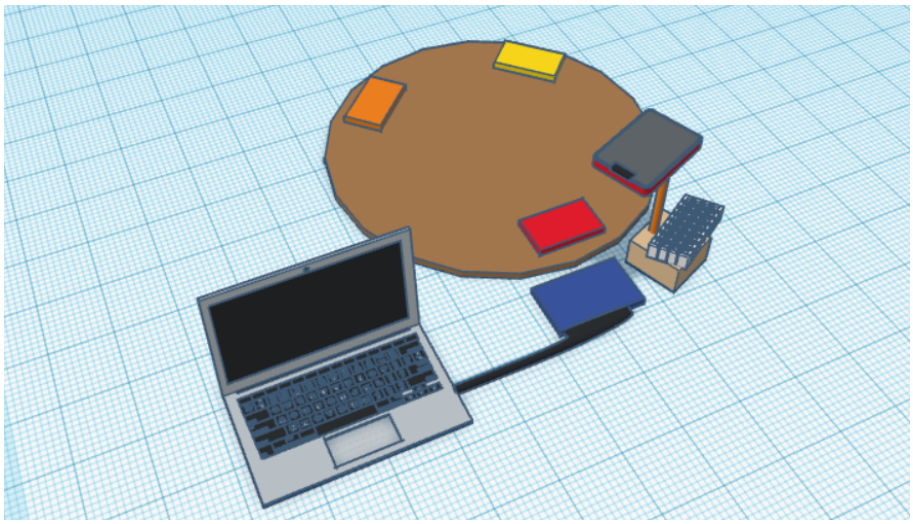
Real-World Applications:

Here, the main focus is on pharmaceutical industries, but the technology (image processing, Circular wheel with camera system) we are using has various applications.

- Pharmaceutical industries - To detect when pills are defected or if they are missing
- Textile industries - To detect defects (such as tear or stain) in clothes.
- Food and Beverage Production - To detect packaging defects such as a low amount of beverage stored.
- Plastic manufacturing - To detect defects in the products.
- Packaging industries - To detect missing items in the packaging or labeling is incorrect.

Drawings of the experimental setups:

TinkerCAD drawing:



TinkerCAD link:
https://www.tinkercad.com/things/lJAp9DSglYt-neat-elzing-jaagub/edit?returnTo=%2Fdashboard%3Fcollection%3Ddesigns&sharecode=jvAYRKSkiohhkETkVfTZ6hsmAeYR1FmnX71DVVPCYrw

Total Cost:

Total cost spent by us: Medicines: Rs. 150

Actual cost of the project: Rs. 6405

Summary

An important development in pharmaceutical management and quality control is an image detection system that was created to detect missing pills. The technology correctly detects missing pills and individual tablets within a package by using advanced image processing techniques and machine learning. By including a Graphical User Interface (GUI), improve the user experience. This complete system helps with pharmaceutical quality management in addition to guaranteeing patient safety and compliance with regulations. The possibilities of the system in the pharmaceutical business will be further enhanced by continuous advancements in image processing and GUI design.

Conclusion

The combination of a spinning wheel system with smart image processing is a new and smart way to check medicine tablets during manufacturing. The wheel turns, and cameras take detailed pictures of each tablet. Computer programs look at these pictures and immediately spot any problems or missing pills.

This method's effectiveness comes from its ability to carefully examine every medicine tablet as it is being manufactured. The revolving wheel makes sure that each tablet is thoroughly inspected as it travels around a circle. The use of advanced software applications facilitates the quick identification of errors and reduces the time that issues must be resolved.

The flexibility of this innovative method makes it useful not just for the pharmaceutical industry but also for other sectors. Combining the rotating wheel with image processing is a significant step forward in guaranteeing the overall quality of the products. Its importance goes beyond the pharmaceutical sector, as it aids in the production of safe and dependable products for a range of sectors.

Future Work

The following tasks could be completed on this project in the future:

- Optimization of image processing algorithms.

  We can utilize better image processing algorithms which give more accurate output as well as are faster in general. The faster these algorithms are, the more throughput we can increase.

- Integration with AI

  With the help of AI, we can automate many processes as well as the accuracy of the output will also increase because we will have trained the AI on many different labeled images.

- Elimination of defective pills from the Circular wheel

  In our project, we have not set up a process wherein once the defective packet has been detected we remove it from the wheel and put it in the defective section so in the future we can automate the process.

- To make a user interface (e.g. app or website) for ease of use.

Codes:

Arduino Code:

```
// Define the pins for the ultrasonic sensor
const int trigPin = 8;  // Trig pin of the ultrasonic sensor
const int echoPin = 9; // Echo pin of the ultrasonic sensor

void setup() {
  Serial.begin(9600);  // Initialize serial communication at 9600 bps
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Trigger the ultrasonic sensor to send a pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the time it takes for the pulse to return
  long duration = pulseIn(echoPin, HIGH);

  // Calculate the distance based on the speed of sound (343 meters/second)
  // and the time it takes for the pulse to return
  float distance = duration * 0.0343 / 2;
```

```
  // Set a threshold distance for object detection (adjust as needed)
  float thresholdDistance = 12.0;  // in centimeters

  // Check if an object is within the threshold distance
  if (distance < thresholdDistance) {
    Serial.println("1");  // Object detected
  } else {
    Serial.println("0");  // No object detected
  }

  // Adjust the delay as needed to control the rate of readings
  delay(100);
}
```

Image processing code:

```
import cv2
import numpy as np

import matplotlib.pyplot as plt
#%matplotlib inline
from PIL import Image
from PIL import ImageFont
from PIL import ImageDraw

def NumberOfPills():
    cam_port = 0
    url = 'http://10.1.130.181:8080/video'
```

```
cam = cv2.VideoCapture(url)

# reading the input using the camera
result, image = cam.read()

# If image will detected without any error,
# show result
if result:

    # showing result, it take frame name and image
    # output
    # cv2.imshow("Preet", image)

    # saving image in local storage
    cv2.imwrite("Preet.png", image)

    # If keyboard interrupt occurs, destroy image
    # window
    cv2.waitKey(0)
    cam.release()
    # cv2.destroyWindow("Preet")

# If captured image is corrupted, moving to else part
else:
    print("No image detected. Please! try again")
```

```
img = cv2.imread('Preet.png')

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)


img = np.array(img, dtype=np.uint8)


fx = fy = int(200.0 / img.shape[0])


dim = (100, int(img.shape[1] * fx))


# Resize the original image

resized = cv2.resize(img, dim, fx = 0.5,fy=0.5)


# Apply Gaussian blur

blur = cv2.GaussianBlur(resized,(7,7),0)


roi_hsv = cv2.cvtColor(blur, cv2.COLOR_RGB2HSV)


# Convert the image in HSV


h, s, v = cv2.split(roi_hsv)

hsv_image = cv2.merge([h, s, v])


imgOTSU = cv2.threshold(s, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)


kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5),(1,1))


fgmask = cv2.morphologyEx(imgOTSU[1], cv2.MORPH_CLOSE, kernel)
```

```
    # plt.imshow(fgmask)
    PillsContours, hierarchy = cv2.findContours(fgmask.copy(), cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)
    # filtered_contours = [contour for contour in PillsContours if cv2.contourArea(contour) > 2000]
    filtered_contours = [contour for contour in PillsContours if cv2.contourArea(contour) > 1000 and cv2.contourArea(contour) < 6000]

    # Draw the filtered contours on a blank image (optional)
    # result_image = cv2.drawContours(np.zeros_like(fgmask), filtered_contours, -1, (255, 255, 255), thickness=cv2.FILLED)
    # result_image
    print ('Number of pills: ', len(filtered_contours))
    processed_image = cv2.drawContours(np.zeros_like(fgmask), filtered_contours, -1, (255, 255, 255), thickness=cv2.FILLED)
    return len(filtered_contours),img,processed_image
```

GUI (Integration of Tkinter and Pyserial) code:

```
import serial
import time
import tkinter as tk
from tkinter import Label, Button, PhotoImage
from PIL import Image, ImageTk
from inspect4 import NumberOfPills
import cv2
import numpy as np



line1 = "1"
```

```
class PillCounterApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Pill Counter App")

        # Serial port setup
        self.ser = serial.Serial('COM7', 9600)  # Update 'COM3' with the correct serial port on your system

        # GUI elements
        self.label_status = Label(root, text="Waiting for signal...", font=("Helvetica", 16))
        self.label_status.pack(pady=10)

        self.label_original_image = Label(root)
        self.label_original_image.pack(pady=10)

        self.label_processed_image = Label(root)
        self.label_processed_image.pack(pady=10)

        self.label_pill_count = Label(root, text="Number of Pills: 0", font=("Helvetica", 14))
        self.label_pill_count.pack(pady=10)

        self.btn_start = Button(root, text="Start", command=self.start_capture)
        self.btn_start.pack(pady=10)

        self.btn_stop = Button(root, text="Stop", command=self.stop_capture)
        self.btn_stop.pack(pady=10)
```

```
        # Flag to indicate whether the process should continue
        self.running = False

    def start_capture(self):
        # Set the flag to continue the process
        self.running = True
        while self.running:
            line = self.ser.readline().decode('utf-8').strip()
            print(line)
            if line == 1:
                print("here")
                count, original_image, processed_image = NumberOfPills()  # Find number of pills and get images
                flag = 1
                # Resize the images
                original_image = cv2.resize(original_image, (300, 300))
                processed_image = cv2.resize(processed_image, (300, 300))

                # Convert the images to PhotoImage format
                original_image_tk = self.convert_image_to_tk(original_image)
                processed_image_tk = self.convert_image_to_tk(processed_image)

                # Update the labels
                self.label_original_image.config(image=original_image_tk)
                self.label_original_image.image = original_image_tk
```

```
            self.label_processed_image.config(image=processed_image_tk)
            self.label_processed_image.image = processed_image_tk

            # Update the pill count label
            self.label_pill_count.config(text=f"Number of Pills: {count}")

            # Wait for a short time to avoid high CPU usage
            self.root.update()
            time.sleep(0.1)

    def stop_capture(self):
        # Set the flag to stop the process
        self.running = False

    def convert_image_to_tk(self, img):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(img)
        img_tk = ImageTk.PhotoImage(img)
        return img_tk

    def on_closing(self):
        # Close the serial port when closing the application
        self.running = False  # Stop scheduling the functions
        self.ser.close()
        self.root.destroy()

if __name__ == "__main__":
```

```
root = tk.Tk()

app = PillCounterApp(root)

root.protocol("WM_DELETE_WINDOW", app.on_closing)  # Handle window close event

root.mainloop()
```

Photographs of setup: