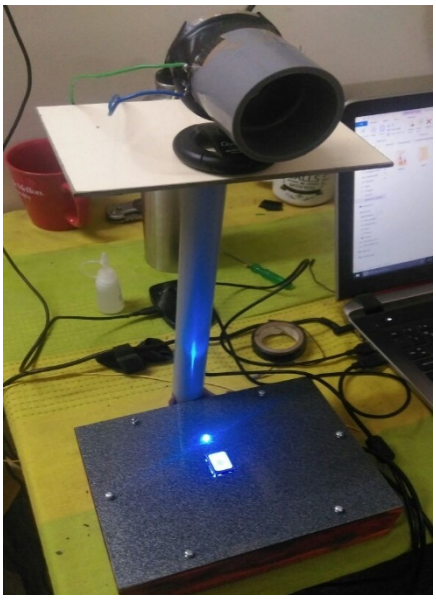
Fingerprint and Iris Authentication System

Product Design and Development



Alpha Prototype Evaluation Manual

Group 13

110113086, 85, 40, 79, 87, 32

Abstract:

- To fabricate a biometric authentication system that can be used in various scenarios – Ration, Aadhar ID Verification and Security Systems.
- Utilizes fingerprint detection in tandem with Iris recognition and feature matching to give two-level security.
- The aim of the project is to overcome the difficulties faced in the present physical registration system due to large number of people and lack of automation. More often, the ID cards wear off/are stolen/ are misused for personal benefits.
- This also will help maintain a digital account of citizens, incorporate that for any other services to the citizens from the Government's side.
- It will allow citizens to benefit from these schemes without any documents/paperwork – facilitating a streamlined process based on biometric data.
- All of the above is achieved at very nominals costs, given the materials and electronics we have chosen to work with – a truly optimal solution.

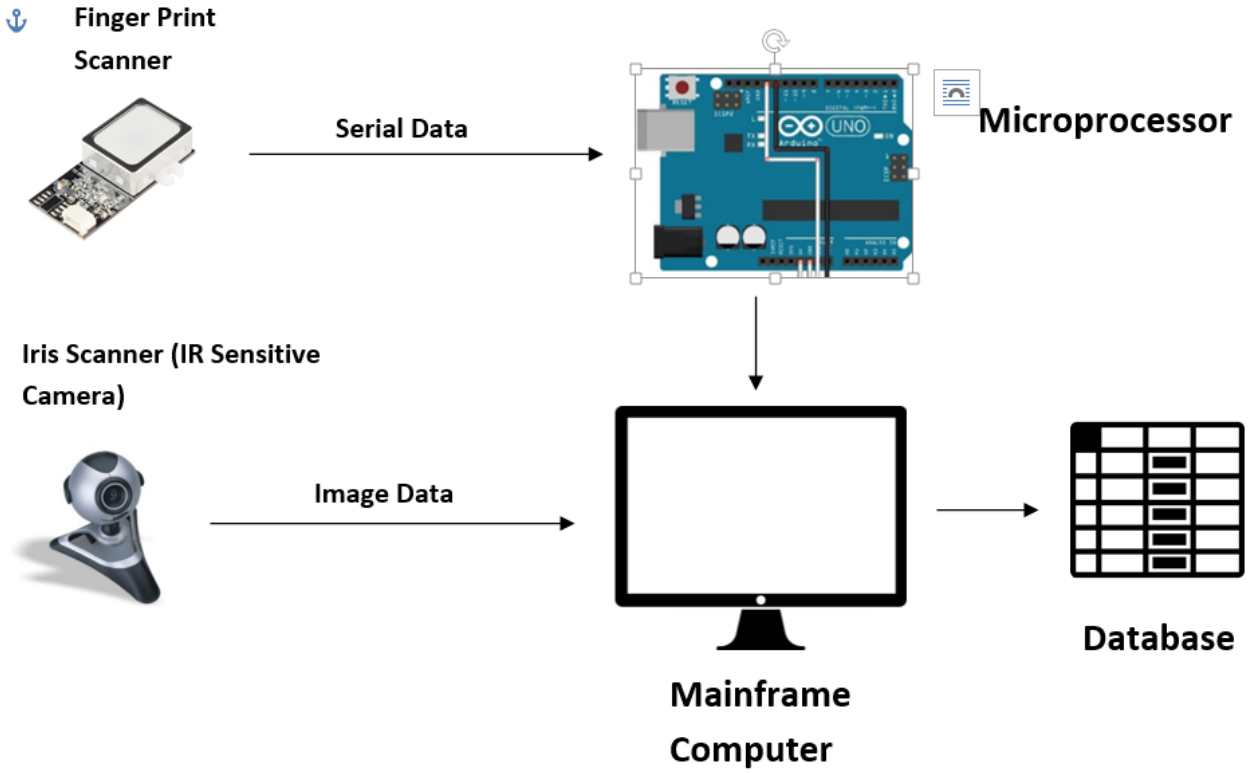
Electronics Bill of Materials:

Component Used	Description	Cost of the Component
Arduino/Genuino Uno Development Board (x2)	14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.	**Rs. 550**
Optical Fingerprint Recognition **Embedded Module- GT 511C1R**	ARM Cortex M3 Core, 14x12.5mm sensor with **False Acceptance Rate of < 0.001%**	**Rs. 1949**
Web Camera	**Logitech**; 12 Mega Pixel	**Rs. 349**
IR LEDs (x2)	-	**Rs. 20**
Dotted PCB	10 x 10 cm	**Rs. 30**
3 Color LED	BGR	**Rs. 20**
Wiring	-	**Rs. 30**
Exposed Negative Film	Exposed on request from photo store	**Rs. 20**
	Total Cost	**Rs. 2968**

Design and Fabrication Checklist:

Housing Frame	ABS Plastic; 118*78*24mm
Hylam box	8" x 6"
Hylam plank	6'' x 6"
PVC Pipe	1.5" diameter
Anabond	40 ml Instant Epoxy Adhesive

Working Flowchart:



IR Sensitive Camera:

- Existing cameras are only sensitive to visible light, due to the presence of an IR filter.
- However, due to the high sensitivity and precision required for the Iris Recognition, we would need low-power IR lighting in the cavity.
- Thus we remove the filter in the camera and replace with the exposed negative of a 3.5mm film.
- This allows it to be preferential towards IR illumination, giving clear image.

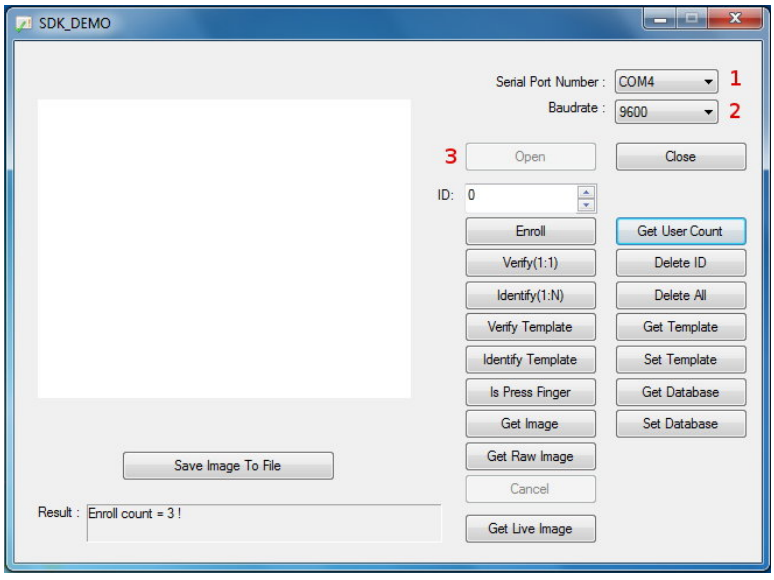
Fingerprint Module:

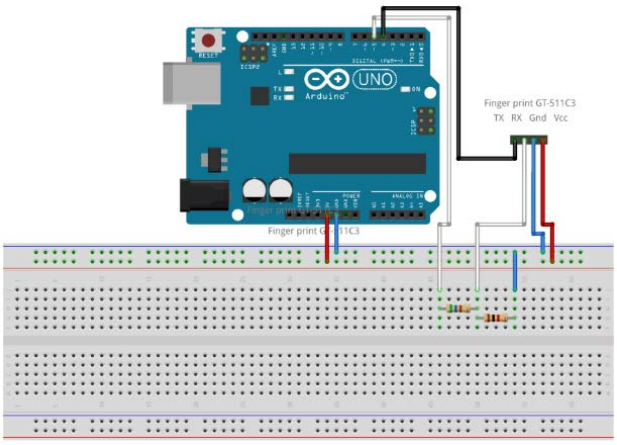
TTL (GT-511C3)

- High-Speed, High-Accuracy Fingerprint Identification using the SmackFinger 3.0 Algorithm
- Ultra-thin optical sensor
- ARM Cortex M3 Core
- 14 x 12.5mm
- 450 dpi
- UART protocol (Default 9600 baud)
- Capable of 1:1 Verification and 1:N Identification

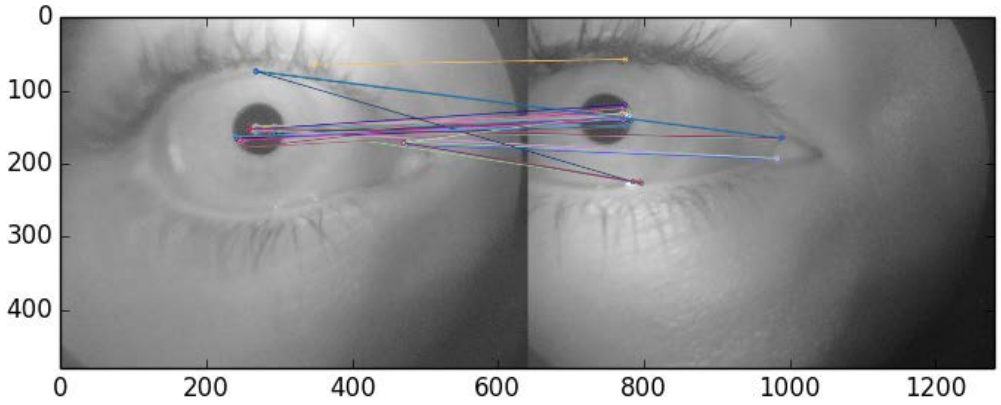
Development Environment:

- Due to affordability, it is tied down to proprietary software.
- The Standard Development Kit (SDK) allows us to enrol, verify and generate a **unique Fingerprint ID** for each user.
- This is used as an input in our user interface.

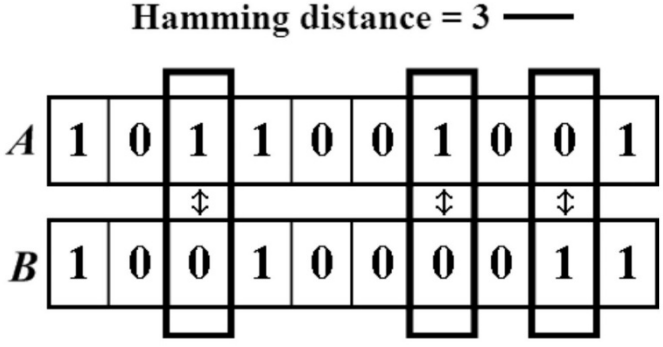


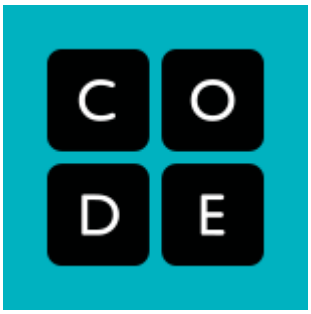


Iris Recognition Algorithm:



- It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.
- Method computes abstractions of image information and makes local decisions at every image point whether there is an image feature of a given type at that point or not.
- These can be edges/interest points/ blobs/ ridges
- Image is normalized to prevent erroneous lighting from affecting output.
- The two images which have the minimum hamming distance between them is adjudged a match.





```
from Tkinter import *
from tkMessageBox import showinfo
import pickle
import os, shutil
import sys
import winsound
import time
import os.path
import cv2
import numpy as np
import serial


ser = serial.Serial('COM7', 9600)

def capture():
    cam = cv2.VideoCapture(0)

    img_counter = 0

    winsound.PlaySound('audio/5secs.wav', winsound.SND_FILENAME)
    timeout = time.time() + 5

    while True:
        ret, frame = cam.read()
        cv2.imshow("Eye", frame)
        if not ret:
            break
        k = cv2.waitKey(1)

        if k%256 == 27:
            # ESC pressed
            print("Escape hit, closing...")
            break

        elif time.time() > timeout:
            while(os.path.isfile("eyes/{}.png".format(img_counter)) == True):
                img_counter = img_counter + 1
            img_name = "eyes/{}.png".format(img_counter)
            cv2.imwrite(img_name, frame)
            print("written!")
            img_counter += 1
            winsound.PlaySound('audio/beep.wav', winsound.SND_FILENAME)
            break

    cam.release()
    cv2.destroyAllWindows()

def compare(match):
    cam = cv2.VideoCapture(0)

    cv2.namedWindow("Eye")

    img_counter = 0

    winsound.PlaySound('audio/5secs.wav', winsound.SND_FILENAME)
    timeout = time.time() + 5

    while True:
        ret, frame = cam.read()
        cv2.imshow("Iris Capture", frame)
        if not ret:
            break
        k = cv2.waitKey(1)

        if k%256 == 27:
            # ESC pressed
            print("Escape hit, closing...")
            break
```

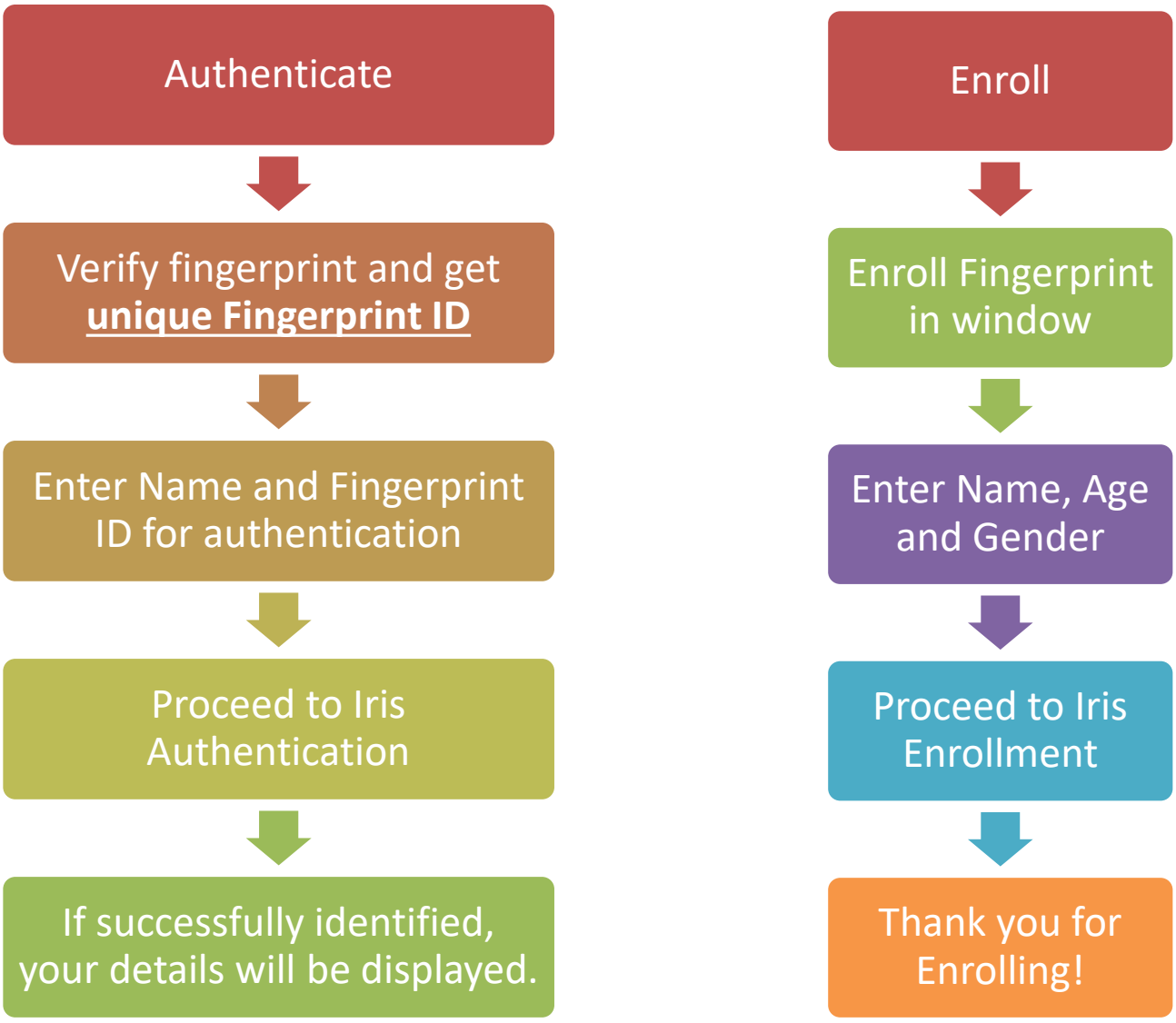
```
        elif time.time() > timeout:
            ID = sift(frame)
            if int(match) == int(ID):
                names = pickle.load( open("names.p", "rb" ) )
                ages = pickle.load( open("ages.p", "rb" ) )
                winsound.PlaySound('audio/beep.wav', winsound.SND_FILENAME)
                ser.write("G")
                i = 1
                while(1):
                    ser.write("G")
                    if(i==1):
                        sList=['Authentication Successful-\nName: {}\n', 'Age: {} years']
                        valueList=[names[ID],ages[ID]]
                        lines = [s.format(value) for s,value in zip(sList,valueList)]
                        msg = ''.join(lines)
                        showinfo(title='Success', message = "%s" % msg)
                        i = i + 1
            else:
                i = 1
                while(1):
                    ser.write("R")
                    if(i==1):
                        showinfo(title='Failed', message='Authentication Failure. Please adjust lighting
conditions.')
                        i = i + 1
            break

    cam.release()
    cv2.destroyAllWindows()

def sift(frame):
    list = os.listdir('eyes/') # dir is your directory path
    number_of_files = len(list)

    ideal = 0
    ideal_score = np.inf

    for i in range(0,number_of_files):
        img1 = frame
        img2 = cv2.imread('eyes/'+str(i)+'.png',0) # trainImage

        # Initiate ORB detector
        orb = cv2.ORB_create()

        # find the keypoints and descriptors with ORB
        kp1, des1 = orb.detectAndCompute(img1,None)
        kp2, des2 = orb.detectAndCompute(img2,None)
        # create BFMatcher object
        bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

        # Match descriptors.
        matches = bf.match(des1,des2)

        # Sort them in the order of their distance.
        matches = sorted(matches, key = lambda x:x.distance)
        # Draw first 10 matches.
        n = 0

        corr = []
        for n in range (0,10):
            corr.append(matches[n].distance)

        img3 = cv2.drawMatches(img1,kp1,img2,kp2,matches[:10],None,flags=2)

        average_match_score = np.mean(corr)
        print average_match_score

        if average_match_score < ideal_score:
            ideal = i
```

```
            ideal_score = average_match_score

    return ideal

def reset():
    names = []
    ages = []
    pickle.dump(names, open( "names.p","wb"))
    pickle.dump(ages, open( "ages.p","wb"))

    folder = 'eyes/'
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        try:
            if os.path.isfile(file_path):
                os.unlink(file_path)
        except Exception as e:
            print(e)

def reply1():
    enroll = Tk()
    enroll.resizable(width=False, height=False)
    enroll.geometry('{}x{}'.format(350,100))
    enroll.title('Enroll')
    Label(enroll, text="Please enroll your fingerprint in the adjacent window\n BEFORE Entering your NAME
and AGE in the boxes below:").pack(side=TOP)

    ent1 = Entry(enroll)
    ent1.pack(fill=BOTH, expand = 1)

    ent2 = Entry(enroll)
    ent2.pack(fill=BOTH, expand = 1)

    btn3 = Button(enroll, text="ENTER", command=(lambda: reply3(ent1.get(),ent2.get())))
    btn3.pack(fill=BOTH, expand = 1)


def reply2():
    valid = Tk()
    valid.resizable(width=False, height=False)
    valid.geometry('{}x{}'.format(600,200))
    valid.title('Authentication')

    Label(valid, text="Please verify your fingerprint in the adjacent window\n AND THEN\n Enter your name
and Unique Fingerprint ID:").pack(side=TOP)
    Label(valid, text="If you do not get a Unique Fingerprint ID, please retry verification.\n Else, you
may not be in the database and need to ENROLL first:").pack(side=BOTTOM)

    ent2 = Entry(valid)
    ent2.pack(side=TOP)

    btn4 = Button(valid, text="ENTER", command=(lambda: reply4(ent2.get(), ent3.get())))
    btn4.pack(side=BOTTOM)

    ent3 = Entry(valid)
    ent3.pack(side=BOTTOM)


def reply3(name, age):
    try:
        names = pickle.load( open("names.p", "rb" ) )
        ages = pickle.load( open("ages.p", "rb" ) )
    except IOError:
        names = []
        ages = []
    names.append(name)
    ages.append(age)

    pickle.dump(names, open( "names.p","wb"))
```

```
   pickle.dump(ages, open( "ages.p","wb"))
    ser.write("B")
    showinfo(title='Finger Enrolled', message='Hello %s! Please proceed to Iris Enrollment\nby clicking
OK' % name)
    capture()
    showinfo(title='Iris Enrolled', message='New User Enrolled: %s' % name)

def reply4(name, numb):
    names = pickle.load( open("names.p", "rb" ) )
    try:
        if names[int(numb)] == name:
            sList=['Authentication Successful-\nName: {}\nPress OK to continue to iris verification.']
            valueList=[names[int(numb)]]
            lines = [s.format(value) for s,value in zip(sList,valueList)]
            msg = ''.join(lines)
            showinfo(title='Fingerprint ID', message="%s" % msg)
            compare(int(numb))
        else:
            i = 1
            while(1):
                ser.write("R")
                if(i==1):
                    showinfo(title='Fingerprint ID', message='Rejected')
                    i = i + 1


    except IndexError:
        i = 1
        while(1):
            ser.write("R")
            if(i ==1):
                showinfo(title='Reply', message='Rejected')
                i = i + 1



root = Tk()
root.resizable(width=False, height=False)
root.geometry('{}x{}'.format(800,400))
root.title('Authentication System')

btn1 = Button(root, text="Enroll", fg="blue", command=(lambda: reply1()))
btn2 = Button(root, text="Authenticate", fg="red", command=(lambda: reply2()))
btn1.pack(fill=BOTH, expand = 1)
btn2.pack(fill=BOTH, expand = 1)

reset_button = Button(root, text="Reset DB", command=(lambda: reset()))
reset_button.pack(fill=X)

root.mainloop()
root.quit()
```

Working Flowchart:

Two possible paths can be chosen by the user via the Graphical User Interface-



Product Design and Packaging:

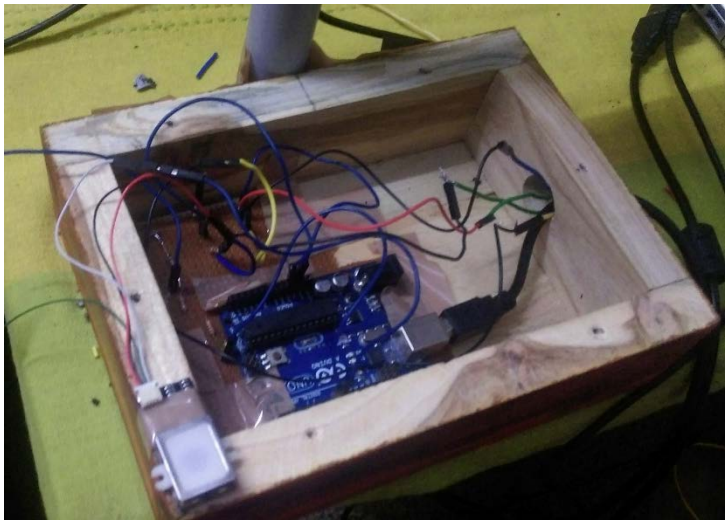
Design Principles:

Principles adhered to – **Modularity, Functional Ease, Usability and Compact Interface**

- Robust and compact as possible
- Including minimum number of components in user's side
- Designing a network for storing the data
- Include standard packaging (rugged)
- Multipurpose & Cost efficiency

Description-

- The Arduino board and the fingerprint sensor are packaged inside of a Wooden Box of size 8"x6"x2.5".



- The box is made out of quality timber and has a removable cover made out of Hylam plywood with screw holes aligned with the box.
- The box was selected as the packaging for its strength. The wooden box has the strength to take on the external stresses created by the users, which might damage the internal circuitry and wiring.

- An LED light is present in the vicinity of the fingerprint sensor on the top of the box. This serves as a visual indicator for a successful scan.

- The box was given a circular bore at the rear end to wire the webcam which is set at a height of 30 cm above the ground.

- The webcam is mounted on a Hylam plywood sheet of dimensions 6”x6”, which rests on an industry grade PVC pipe of Diameter 1”.
- The pipe is set in place with the help of an Instant Epoxy Adhesive, an industry standard.
- The webcam that serves as the iris scanner has a PVC Pipe used as an extension in order to protect the image from stray lighting.

The packaging has sufficient protection with the materials used given the working conditions of the authentication system.

References:

https://www.sparkfun.com/products/11792

http://dir.indiamart.com/impcat/hylam-sheet.html

https://www.arduino.cc/en/Main/ArduinoBoardUno

https://wiki.python.org/moin/TkInter

http://www.instructables.com/id/Near-Infrared-Camera-for-Less-than-10-and-10-mi/

https://www.sparkfun.com/products/105

http://docs.opencv.org/trunk/dc/dc3/tutorial_py_matcher.html

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html

https://www.cl.cam.ac.uk/~jgd1000/csvt.pdf

http://www.sersc.org/journals/JSE/vol8_no5_2011/6.pdf