

Project Lab Fall 2024

Low Vision Magnifier Project

PDF Version generated by

Adam Danial Ingah (adingah@uab.edu)

on

Dec 08, 2023 @11:51 AM CST

Table of Contents

| | |
|----------------------------------|----|
| Official Documents | 2 |
| README | 2 |
| Team Reports SP2023 | 3 |
| Client Report | 4 |
| Presentations/Expo Posters | 5 |
| Presentations23 | 5 |
| Presentations22 | 6 |
| Expo | 7 |
| Abstract | 9 |
| Pictures | 10 |
| Designs | 12 |
| 3D Models | 12 |
| Sketches | 16 |
| Setup | 17 |
| ARCHIVE | 18 |
| NewCode | 18 |
| AllCode | 18 |
| README | 19 |
| OldCode | 20 |
| AllCode | 20 |



Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:52 PM CST

Project Lab
Spring 2022
Team Performance Plan
Due: February 4, 2022

Learning Goal: How the team works together to complete their mission has a direct effect on team performance, team member satisfaction and final project quality. This assignment helps students critically think about the team project using a structured guide. By following this guide, the team will be able to communicate more effectively to review and enhance team and individual performance.

Purpose: The Team Performance Plan provides a framework to facilitate communication among team members to develop a clear set of expectations for (i) group achievement (the project) and (ii) team member participation (expectations). A good quality TPP will include group introspection – about team member dynamics (strengths and challenges); approach to the project and immediate assignments; plans to adapt and pivot as challenges arise (team conflict, team members' workload/work balance, project challenges, etc.).

The TPP's effectiveness is directly proportional to the amount of effort the team expends to know their individual strengths and weaknesses as well as those of all teammates to develop and execute an effective plan throughout the term.

A Team Performance Plan addresses the following:

1. What are the specific project goal(s)?
2. What are the major project deadlines?
3. What is the project mission? (Include team development goals as appropriate).
4. How work will be done (who is responsible for what)?
5. How will the team interact? What actions are required to achieve goals?
6. How will progress be documented and shared?
7. What tasks need to be done and milestones achieved?
8. What does good performance look like?
9. What support is required to achieve project goals?
10. What are quantitative and qualitative measures of project success?
11. How might interpersonal interactions improve team performance?
12. What are the concerns or unknowns that must be addressed for the team to be successful? (e.g. time, expertise, team dynamics, specific skills, access to specialized equipment).
13. What are conditions that a team member could be expelled from the group?

Use the outline below to complete a Team Performance Plan.

1. What is the topic for the team project? (1-2 sentences is sufficient). **This might be longer for a more complicated project.**
2. List important project specific milestones and projected completion dates. Include assignments and due dates. Add as many additional lines as needed.
 - a. Assignment Due Dates

[Download](#)

LowVision_TeamPerformanPlan_SP23.docx (21.5 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:52 PM CST

Low Vision Magnifier Client Report
2/6/23
Hunter, Mikee, Korrin, Daniel, Ben

Client: Mr. Michael Papp of the Alabama Department of Rehabilitative Services

Problem Statement:

Individuals with low-vision disabilities may struggle to read books without a distance from devices that can magnify text. However, using a magnifier does not provide user-customization to filter color, change contrast, and change brightness. Furthermore, electronic devices called CCTVs that provide magnification and user-defined filters are often very expensive (>\$2000). Therefore, Mr. Michael Papp of the Alabama Department of Rehabilitative Services asked Project Lab to develop a low-cost (<\$200) solution that can provide the same benefits and functionality as a commercial CCTV device. The low-cost device must also provide 4x-32x magnification, easy user input, text-to-speech, control when using, be portable, and be open source for people to build themselves.

Client Meetings:

2/3/23:

- Priorities: Use image over Page Scan -> Digital Output
- How to prevent "glitching"
 - o Cheating is where there is a lag between moving the page and the video output
 - o Will a more powerful computer help?
- Open Source ideas
 - o How can people easily build this and setup a device for themselves/others?
 - o Aim for under \$200
- App may not be viable if aiming for low image magnification
- Text-to-speech
 - o Logitech plus software <\$60
 - o Used previously in 2008
- Mr. Papp may be able to get to monitor and camera

[Download](#)

ClientReport.docx (14.4 kB)



Client Report

Ben Moore (moorebg@uab.edu) - Sep 29, 2023, 10:52 AM CDT

Client Meeting Questions/Report

8:00AM Wednesday, September 13th, 2023

Questions Will Be Asked By: Ben

Answers Will Be Recorded By: Elanin

Notes Will Be Recorded By: Daniel (Notes Are At The Bottom Of The Document)

Questions:

- 1) What type of device (computer, tablet, or phone) do you want the Low Vision Magnifier to be modeled around?
 - a. What is the interface between the device and the user?

A: For people with around 20/200 vision, so less aim for aiding people who are having trouble reading the pages, and then for the people who can't read at all.
- 2) What size do you want the device?

A: Handheld size (phone), or Desktop (PC, may be just monitor). Handheld can be done, but it's difficult to construct and use. Our best bet would be Desktop, roughly around 20" to 22" LCD Display.
- 3) Are there any existing low vision magnifier solutions that you currently recommend or distribute, and what are their strengths and weaknesses?

A: Microfiche, can hold and display an entire newspaper.
- 4) How do you suppose we open source this project? What are your expectations for the device?

A: Instructables or YouTube as well. Maybe both.
- 5) What is the budget for the device?

A: Sub-\$200
- 6) What are the different eye impairments do you want the device to help fix?

A: **Answered in first question**
- 7) Are there any people that we can potentially talk to about their patient needs/concerns etc...

A: Possible, due to privacy laws, it'll be difficult to do so.

[Download](#)

Client_Meeting_Questions_Report.pdf (196 kB)



[Download](#)

Presentation3_3-3-23.pptx (887 kB)



Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:09 PM CST



[Download](#)

ADRSPres_1_-_7-1-22.pptx (2.33 MB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:09 PM CST



[Download](#)

ADRSPres_2_-_10-7-22.pptx (1.76 MB)



Ben Moore (moorebg@uab.edu) - Nov 16, 2023, 3:15 PM CST

UAB SCHOOL OF ENGINEERING
The University of Alabama at Birmingham

Reading Assistance Device for Vision Impairment

Adam Ingah, Ben Moore, Daniel Dekle, Park Sung

Introduction

According to the CDC, 21 million Americans have some form of vision impairment [1]. Individuals with vision impairments may struggle to read printed text without assistance from devices that can magnify text or provide text customization. However, electronic assistance devices that provide magnification and text customization can be expensive – often over \$2000 [2]. Therefore, Mr. Michael Pigg of the Alabama Department of Administrative Services asked UAB's Project Lab to develop a low-cost solution – less than \$200 – that can provide magnification and text customization to meet users' needs.

Criteria

- 4 – 30x magnification
- Color Filter Control
- Brightness Control
- Contrast Control
- Text-to-Speech
- Line Tracking Function
- Portable
- Cost Under \$200
- Open Source
- Easy to Build
- Compatible to User

Designs

Figure 1: Logitech C620 Webcam Adapter for Prototyping.

Figure 2 (left) and Figure 3 (right): Python script and camera is able to read a text document.

Code:

- Raspberry Pi Linux OS
- Python
- PyOpenCV Deep Learning Library for Optical Character Recognition
- OpenCV computer vision library used for image processing.

Prototype #1:

The "Clasp" and "Clasp" is isolated after the laptop design from the late 2010s to present day. The goal with this design is for the device to be portable so the user can take the device anywhere with them. Unlike the "Clasp" design, this user will hold out on performing maintenance. It will be constructed using 3D printed and laser cut parts.

Figure 3: Sketch Design of Desktop Design.

Figure 4: Sketch Design of Laptop Design.

Progress

Our overall goal is to create a fully functional camera that can focus and uses auxiliary light for better text reading, which will work together with the revised code to change our current text reading accuracy from 85% to 95%. Right now, we've acquired better materials, such as an auto-focus camera and a new circuit board.

However, before implementation, we mean our goal is 100% achieved. Our current issues include cost and users having difficulty assembling the device. Since our user is a beginner or beginner, that this project must be open source. Nevertheless, we've devised some solutions to these issues. Our vision is to build a device from scratch to build into the current design, and we'll write and release open highly detailed instructions to help with user difficulties. Moving forward, we will implement these solutions and further research back to open development for users whose vision is severely impaired.

References

- [1] (2017). Five Facts of Common Eye Diseases. Center for Disease Control and Prevention. <https://www.cdc.gov/ncbddd/eyehear/eye/factsheet.htm> (Accessed July 26, 2022).
- [2] Proctor, Jennifer, et al. "How 400 people who are visually impaired benefit from a low-cost device." <https://www.projectlab.org/news/400-people-who-are-visually-impaired-benefit-from-a-low-cost-device> (Accessed July 26, 2022).

[Download](#)

2022_Fall_Expo_-_ADRS_Poster_36x48_.pptx (585 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:53 PM CST

UAB SCHOOL OF ENGINEERING
The University of Alabama at Birmingham

CCTV Project Revision Device

Hunter Goffinett, Mikey Guanipa, Ksenia Klochokva

Introduction

Over 22 million adults 40 years of age and older suffer from vision impairment. These include 1 million blind people, 3 million people who still have vision in peripheral areas, and 8 million uncorrected vision impairment of some sort. There is a wide variety of CCTV devices and devices that help people with vision impairment to read most, books, and other items. These devices are incredibly rugged after because it allows the visually impaired to navigate the world by allowing them to read small print on menus, books, and more. Currently, most CCTV devices are about the size of a small, white, some even go for the thousands. Our prototype's primary goal is to have a device that can be portable, highly effective by having high contrast, and is under the \$200 price point.

Criteria

- 4 – 30x magnification
- Line Tracking Function
- Real-time tracking
- Prevent black and eye strain
- Brightness control system
- Contrast between letters
- Color Filter Control
- Light source
- Text-to-Speech
- Portable
- Easily modified and improved upon request
- User friendly
- Simple user input
- Under \$200

Designs

Prototype 1:

- Raspberry Pi is connected to a camera via USB that the user then hovers with their hand.

Figure 1: Parts

- Raspberry Pi
- 4-in (high contrast) or color display
- USB Camera
- Durable Chassis to house all the components

Prototype 2:

- The standard CCTV device is outfitted with a new type of camera with wheels.
- This prototype allows the user to use less effort with handling the device since the camera will stand by itself.

Figure 2:

Prototype 2: Parts

- 4-in (high contrast) or color display
- 4-in (high contrast) or color display
- 4-in (high contrast) or color display
- 4-in (high contrast) or color display

Progress

- Developing code for the CCTV Device
- Manufacturing on multiple prototypes and designs

Future Plans


- Improve accuracy of device
- Create a physical prototype of the device design
- Continue developing the software

References

- [1] (2017). Five Facts of Common Eye Diseases. Center for Disease Control and Prevention. <https://www.cdc.gov/ncbddd/eyehear/eye/factsheet.htm> (Accessed July 26, 2022).
- [2] Proctor, Jennifer, et al. "How 400 people who are visually impaired benefit from a low-cost device." <https://www.projectlab.org/news/400-people-who-are-visually-impaired-benefit-from-a-low-cost-device> (Accessed July 26, 2022).

[Download](#)


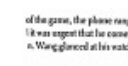


2022_Fall_Expo_-_ADRS_Poster_36x48_.pptx (239 kB)



The University of Alabama at Birmingham

Reading Assistance Device for Vision Impairment

Hunter Goffinet, Mikay Gumpira, Ksenia Knochkova, Ben Moore, Daniel Deike

| Introduction | Designs | Progress |
|--|--|--|
| <p>According to the CDC, 33 million Americans have some form of non-silent vision impairment. Individuals with vision impairments may struggle to read printed text without assistance from devices that can magnify text or provide text customization. However, electronic assistance devices that provide magnification and text customization are expensive – often over \$2000 (5). Therefore, Mr. Michael Joffe at the Alabama Department of Rehabilitation Services asked our Project Lead to develop a two-part solution – first that \$200+ that can provide magnification and text customization to meet user needs.</p> | <div data-bbox="669 226 842 352">  <p>Figure 1. Logitech C920 Webcam Adapter for Prototyping</p> <p>Prototype A</p> <ul style="list-style-type: none"> • Libre AML-9205C-C0 Board • Logitech C920 Webcam • Computer Monitor for Display • 3D Printed or Woodcut Housing for Device </div> <div data-bbox="669 352 842 478"> <p>Goal</p> <ul style="list-style-type: none"> • Amazon Linux OS • Python • Prepared Library for Text Input/Output • Image Recognition • Color Filter (Project) </div> | <p><i>of the game, the phone rang. It was urgent that he save data. He was glued as he wrote in</i></p> <div data-bbox="842 226 1136 352">  <p>Figure 2. Example Image of Book with Black and White Filtering with ImageJ Software</p> <ul style="list-style-type: none"> • Camera Quality • Camera Quality is too Poor • Camera Does Not Have AutoFocus • Text Segmentation is Inconsistent • Inconsistent Text Deviation is Exaggerated • Inconsistent Text Deviation is Exaggerated • Inconsistent Text Deviation is Exaggerated </div> <div data-bbox="842 352 1136 478"> <p>Prototype B</p> <ul style="list-style-type: none"> • Libre AML-9205C-C0 Board • Windows Camera • 7" LCD Screen • 3D Printed or Woodcut Housing for Device </div> |
| <p>Criteria</p> <ul style="list-style-type: none"> • 8 – 32 X Magnification • Color Filter Control • Brightness Control • Contrast Control • Text-to-Speech • Line Tracking Function • Portable • Cost Under \$200 • Open Source • Easy to Build • Comfortable to Use | <p>Figure 3. Sketch Design of Prototype A</p>  | <p>Figure 4. Sketch Design of Prototype B</p>  |
| | | <p>References</p> <ol style="list-style-type: none"> (1) (2017) First Phase of User Interviews. Center for Disease Control and Prevention. https://www.cdc.gov/nczod/oddsat/surveys/firstphaseofuserinterviews.pdf (2) Product Support. (n.d.). Page 102 available online using: https://www.ams.com/ams/en/-/media/Assets/Support/Products/Products%20Index.pdf, retrieved November 14th, 2018. https://www.ams.com/ams/en/-/media/Assets/Support/Products/Products%20Index.pdf |

Download

2022_Fall_Expo_-_ADRS_Poster_36x48_.pptx (544 kB)



According to the CDC, 11 million Americans have some form of non-blind vision impairment. Individuals with vision impairments may struggle to read printed text without assistance from devices that can magnify text or provide text customization. However, electronic assistance devices that provide magnification and text customization can cost over \$2000. Seeing this, Michael Papp from the Alabama Department of Rehabilitative Services asked UAB's Project Lab to develop a low-cost solution – less than \$200 – that can provide magnification and text customization to meet users' needs. Design criteria for the device includes video magnification, video and text filtering for color, contrast, and brightness, easy user-input, text-to-speech, portability, and be open-source. Our current design is modeled around a desktop computer, which will easily store the electronics and allow easy maintenance. For future designs, the magnifier will be styled as a laptop. This path would allow the user to travel with the design and not have to be in one singular place to be able to read. Our current code runs a Raspberry Pi that is running Raspbian OS, which is the program that runs the camera to a python script that turns the camera on and displays a live feed. On top of displaying a live feed, the raspberry Pi uses openCV and pyTensor Machine Learning libraries to process the live text on a document. Further iterations of the Low Vision Magnifier will run an application to save, capture, and process still images, utilize better hardware, and be housed in a 3D design.

Based on the criteria, our mission is to create an inexpensive, stand-alone device that can assist individuals with vision impairments with reading. Current designs employ a Libre Computer Board to act as the device that magnifies and filters video feed from a webcam via Python code. The code currently provides video magnification and color filtering. Future designs will include a monitor, text-to-speech, and a body to house the electronics.

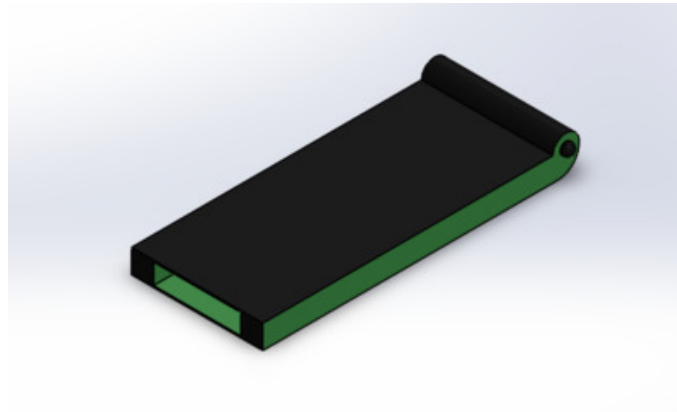
[Download](#)

Sp23_Abstract.docx (17 kB)



Pictures

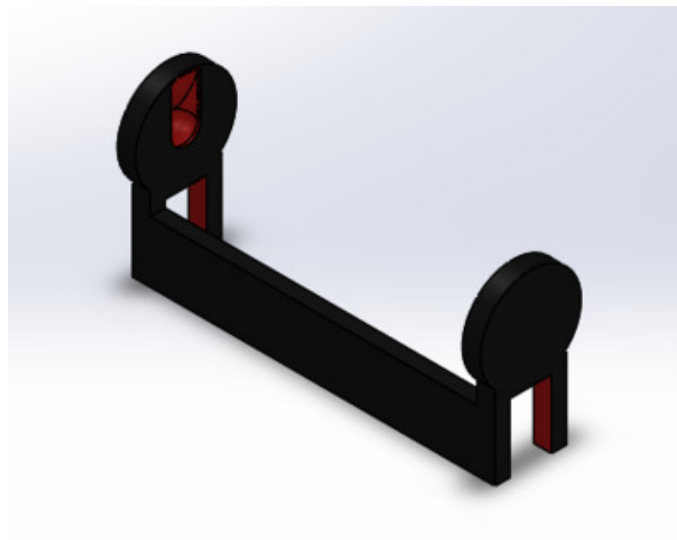
Ben Moore (moorebg@uab.edu) - Apr 14, 2023, 9:55 AM CDT



[Download](#)

Hinge_Arm.png (47.1 kB)

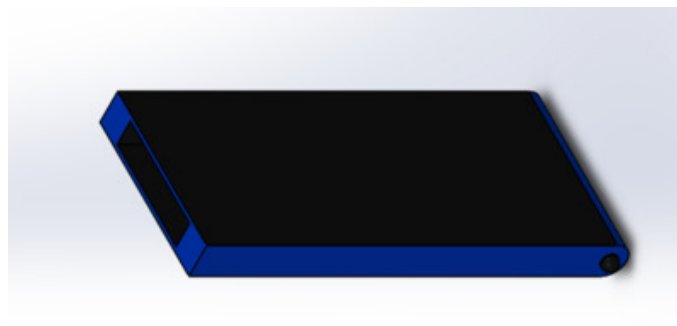
Ben Moore (moorebg@uab.edu) - Apr 14, 2023, 9:55 AM CDT



[Download](#)

Hinge_Base.png (40.9 kB)

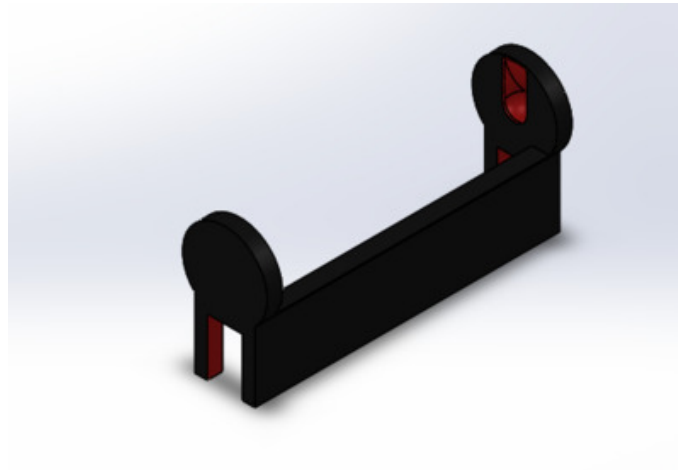
Ben Moore (moorebg@uab.edu) - Apr 14, 2023, 10:19 AM CDT



[Download](#)

Screenshot_2023-04-14_101741.png (33.4 kB)

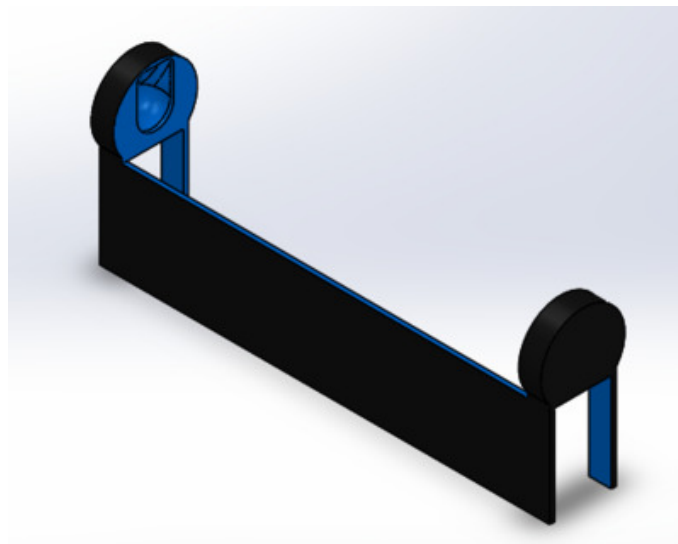
Ben Moore (moorebg@uab.edu) - Apr 14, 2023, 10:19 AM CDT



[Download](#)

Screenshot_2023-04-14_101843.png (48.5 kB)

Ben Moore (moorebg@uab.edu) - Apr 14, 2023, 10:19 AM CDT



[Download](#)

Screenshot_2023-04-14_101657.png (51.3 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST



[Download](#)

Camera_Hinge_Mark_II.SLDASM (112 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST



[Download](#)

Hinge_Arm_2in_Mark_II.SLDPRT (77 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST



[Download](#)

Hinge_Arm_2in.SLDPRT (70.5 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST



[Download](#)

Hinge_Arm_3in_Mark_II.SLDPRT (75 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST



[Download](#)

Hinge_Arm_3in.SLDPRT (68.2 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST



[Download](#)

Hinge_Arm_4in_Mark_II.SLDPRT (73.2 kB)

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:14 AM CST[Download](#)**Hinge_Arm_4in.SLDPRT (65.3 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:15 AM CST[Download](#)**Hinge_Arm_5in.SLDPRT (64.1 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:15 AM CST[Download](#)**Hinge_Base_Mark_II.SLDPRT (189 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:15 AM CST[Download](#)**Case.SLDPRT (485 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:15 AM CST[Download](#)**Mikey_E-Ink_Case_Mark_I.SLDASM (923 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:15 AM CST[Download](#)**Panel_Mark_II.SLDPRT (65.8 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:16 AM CST[Download](#)**Panel.SLDPRT (53.9 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:16 AM CST[Download](#)**Socket.SLDPRT (88.3 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:16 AM CST[Download](#)**1mm_screw.SLDPRT (703 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:16 AM CST[Download](#)**Ball_N_Socket.SLDASM (130 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:16 AM CST[Download](#)**Ball.SLDPRT (112 kB)**

Ben Moore (moorebg@uab.edu) - Mar 03, 2023, 9:16 AM CST[Download](#)**Case_Mark_II.SLDPRT (885 kB)**

Ben Moore (moorebg@uab.edu) - Mar 24, 2023, 3:10 PM CDT



[Download](#)

Hinge_Base_Mark_IV.SLDPRT (184 kB)

Ben Moore (moorebg@uab.edu) - Mar 24, 2023, 3:12 PM CDT

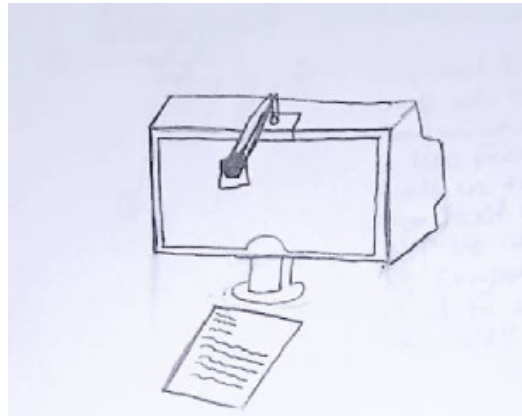


[Download](#)

Hinge_Base_Mark_III.SLDPRT (176 kB)

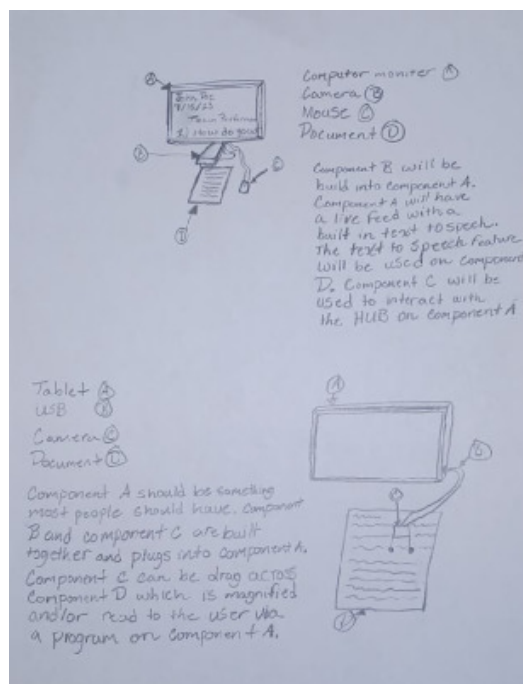


Ben Moore (moorebg@uab.edu) - Oct 27, 2023, 9:46 AM CDT

[Download](#)

LVM_TV_Sketch.pdf (10.3 kB)

Ben Moore (moorebg@uab.edu) - Sep 22, 2023, 10:34 AM CDT

[Download](#)

LVM_Sketches.pdf (84.5 kB)



Hunter Shaw Goffinett (hsgoffin@uab.edu) - Aug 31, 2023, 7:49 PM CDT

General Run Down

CCTV device is controlled via Le Potato computer on Armbian OS. Other alternatives, such as Raspberry Pi or Raspbian, will work.


*OS and code is already setup. DO NOT flash SD card unless absolutely necessary.

Set up:

1. Flash SD card with Armbian Desktop
2. SSH
3. Set up VNC server to access GUI / desktop
4. <https://www.raspberrypi.com/documentation/computers/getting-started.html#ssh>
5. Use VNC Viewer to Access GUI
6. Set up Python Virtual Environment and install all Needed Libraries
7. Note: cv2.VideoCapture() for webcam access

*For SSHing, I had an ethernet cable plugged into my computer that runs into a pi. I used a docking station for the computer ethernet connection.

SSHing

1. Need to establish internet connection for pi
 - a. Plug in power supply to pi
 - b. Plug one end of ethernet cable into computer and other end into pi.
2. Sharing Internet to Pi
 - a. Pull up the Network Connections
 - b. Click WiFi
 - c. Click properties, and under sharing, check both boxes.
 - i. The home network connection should be Ethernet
 - ii. 
 - d. Click Ok
 - e. Unplug ethernet cord of pi and replug it
 - i. Will get ip address assigned to it
 - f. Click ethernet in the home network connection
 - g. Click details then note the IPv4 connection
 - h.
3. Find IP of Pi
 - a. Use a software such as Advanced IP Scanner to find IP of device
 - b. Scan for IPv4 of the ethernet you found above
 - i. Ex: If IP is 192.168.137.1
 - ii. Scan in range of 192.168.137.1 -- 192.168.137.254

[Download](#)

Setup_and_Run_Guide.docx (274 kB)



Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:05 PM CST

```
#Code to start a camera session on a Raspberry Pi / LePotato using a webcam ()
#use the cv2 library
#save magnification: 2x-32x
#save color filters
#Text-To-Speech
#live display, not digital output

#Code to start a camera session on a Raspberry Pi / LePotato using a webcam ()
#use the cv2 library
#save magnification: 2x-32x
#save color filters
#Text-To-Speech
#live display, not digital output

from tkinter import *
import cv2
import numpy as np
from PIL import Image
import threading
import io
from multiprocessing import Pipe
from gits import gITS #Text-To-Speech #has to have internet access
import pyttsx3 #Image-To-Text

colorFilterSet = ('yellowblack')
#check if color filter is applied for the video
#if color filter applied, apply to Frame obj
def colorFilter(input, frame):
    match input[0]:
        case 'yellowblack':
            ret, frame = cv2.threshold(frame, 127, 255, cv2.THRESH_BINARY)
            #frame = cv2.cvtColor(frame, cv2.COLOR_GRAY2BGR)
            frame[ry, where((frame==[255,255,255]).all(axis=2))] = [0,255,255]
            return frame
        case 'yellowblack2':
            #lower = np.array([110,0,0])
            #upper = np.array([130,255,255])
            #lower = np.array([50,0,0])
            #upper = np.array([70,255,255])
            #low = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
            #mask = cv2.inRange(low, upper)
            #low_mask = cv2.bitwise_and(frame, frame, mask=low_mask)
            return frame
        case 'magnify':
            return frame
        case default:
            return frame

def magnification(zoom, frame):
    h,w,c = [zoom * i for i in frame.shape]
    cx,cy = w/2, h/2
    frame = cv2.resize(frame, (0,0), fx=zoom, fy=zoom)
    frame = frame[ int(round(cy - h/zoom * .5)) : int(round(cy + h/zoom * .5)),
                  int(round(cx - w/zoom * .5)) : int(round(cx + w/zoom * .5)),
                  ]
    #frame = cv2.resize(frame, None, fx = scalar * userInput, fy = scalar *
    #userInput, interpolation= cv2.INTER_LINEAR)
    return frame

cap = cv2.VideoCapture(0)
while(True):
    ret, frame = cap.read()
```

[Download](#)

NewCam.py (2.54 kB)



Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 3:00 PM CST

CCTV device is controlled via Le Potato computer on Armbian OS. Other alternatives, such as Raspberry Pi on Raspbian, will work.

Set up:

1. Flash SD card with Armbian Desktop
2. SSH
3. Set up VNC server to access GUI / desktop:
4. <https://leesiropi.blogspot.com/2019/12/installsetup-xfce-and-tightvnc-to.html>
5. Use VNC Viewer to Access GUI
6. Set up Python Virtual Environment and Install all Needed Libraries
7. Note: cv2.VideoCapture(1) for webcam access



Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#!/usr/bin/env python3
import tkinter
from tkinter import Font as tkFont
import os

def quit_():
    os.system("sudo shutdown -h now")

def restart_():
    os.system("sudo reboot")

def dev_(root):
    root.wm_state('iconic') #minimized without a button
    root.destroy() #minimized with a button
def loop_(root):
    root.destroy()
    root.attributes("-zoomed", 1) #this is backup at the resizable(False,False)
    #this
    root.mainloop()
    root.after(100, func=lambda: loop_(root))

root=tkinter.Tk()

helv06 = tkFont.Font(family="Helvetica", Size=24, weight="bold")
width = root.winfo_screenwidth()
root.title("Splash Screen")

info = tkinter.Label(root, text="Please wait while application is
loading...", font=helv06)
info.grid(row=0, column=0, columnspan=3)
quit_button = tkinter.Button(master=root, width=35,
text="Quit", font=helv06, command=lambda: quit_())
quit_button.grid(row=0, column=0, columnspan=3, padx=int((width/2)-6)) #gui
location
restart_button = tkinter.Button(master=root, width=35,
text="Restart", font=helv06, command=lambda: restart_())
restart_button.grid(row=1, column=0, columnspan=3) #gui location
dev_button = tkinter.Button(master=root, width=35, text="Developer
Mode", font=helv06, command=lambda: dev_(root))
dev_button.grid(row=0, column=0, columnspan=3) #gui location

root.attributes("-fullscreen", True)
root.attributes("-topmost", 0) #does not work

root.attributes("-zoomed", 1)
root.resizable(False, False)

root.attributes("-zoomed", 1)
width = root.winfo_screenwidth()
height = root.winfo_screenheight()
geom_string = "width+%d %d %d" % (width, height)
root.geometry(geom_string)

root.overrideredirect(1) #makes borderless

root.after(100, func=lambda: loop_())
root.wm_protocol("WM_DELETE_WINDOW", lambda: 1) #highjack the x button
root.mainloop()
```

[Download](#)

DesktopCover.py (1.84 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#!/usr/bin/env python
#
# Hi there!
# You may be wondering what this giant blob of binary data here is, you might
# even be worried that we're up to something nefarious (good for you for being
# paranoid!). This is a basic encoding of a pip file, this pip file contains
# an entire copy of pip (version 20.2.3).
#
# Pip is a thing that installs packages, pip itself is a package that someone
# might want to install, especially if they're looking to run this get-pip.py
# script. Pip has a lot of code to deal with the security of installing
# packages, various edge cases on various platforms, and other such stuff. It
# is "tribal knowledge" that has been encoded in its code base. Because of this
# we basically include an entire copy of pip inside this blob. We do this
# because the alternatives are attempt to implement a "mini-pip" that probably
# doesn't do things correctly and has weird edge cases, or compress pip itself
# down into a single file.
#
# If you're wondering how this is created, it is using an invoke task located
# in tasks/generate.py called "installer". It can be invoked by using
# "invoke generate.installer".

import os.path
import platform
import struct
import sys
import struct
import tempfile

# Useful for very coarse version differentiation.
PY2 = sys.version_info[0] == 2
PY3 = sys.version_info[0] == 3

if PY2:
    iterbytes = iter
else:
    def iterbytes(buf):
        return (ord(byte) for byte in buf)

try:
    from base64 import b64decode
except ImportError:
    _b64alphabet = ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_-")
    def b64decode(b):
        _b64dec = (None) * 256
        for i, c in enumerate(_b64alphabet):
            _b64dec[i] = i
        padding = (len(b) % 4)
        b = b + "=" * padding
        out = []
        packt = struct.Struct('IIII')
        for i in range(0, len(b), 4):
            chunk = b[i:i+4]
            acc = 0
            try:
                for E in iterbytes(chunk):
                    acc = acc * 256 + _b64dec[E]
            except TypeError:
                for j, c in enumerate(iterbytes(chunk)):
                    if _b64dec[c] is None:

```

[Download](#)

get-pip.py (1.89 MB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
import os
Output=os.system('lsfd apt -y &findall gimp')
Output=os.system('lsfd apt -y &findall krita')
Output=os.system('lsfd apt -y &findall msiapp')
```

[Download](#)

Gimp_For_Cropping.py (140 B)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST



[Download](#)

LICENSE (1.51 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST



[Download](#)

OCRAutoContraststackoverflow2ndattempt.py (1.91 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST



[Download](#)

pyttsx3_voiceid_languages.txt (8.88 kB)



README.md (672 B)

The information revolution has a long way to run

The information revolution is the most subversive force business has ever known. Already the phenomenon of "information power" to the people has given knowledge and authority to front-line workers and technicians, dissolving the power and often the jobs of middle management—who were previously protected by proprietary knowledge. The information revolution has also decentralized corporations physically: the phone, the PC, broadband and the increasing miniaturization and mobility of these technologies have already begun to destroy the power of corporate

screenshot_grey.jpg (10.4 kB)

[illegible]

setup.py (6.21 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#!/usr/bin/env python3
#title: Bag CVE 1.0
#version: 1.0.0
#created by: Nathan Ray Boudin
#copyright (c) 2015, Nathan Ray Boudin

#*****Read Me*****
#If you mistakenly opened this click the "x" in the
#top right corner and do not have any changes
#*****Read Me*****

#Thank you to Abid Rahman & for his answer at
#https://stackoverflow.com/questions/11123041/opencv-setting-all-pixels-of-
#specific-hr-value-to-another-by-value
#The code was used and modified at 246,246,246, and 275, license at
#https://creativecommons.org/licenses/by-sa/3.0/

#The Python Imaging Library (PIL) is
# Copyright © 1997-2011 by Secret Labs AB
# Copyright © 1997-2011 by Fredrik Lundh
#Pillow is the friendly PIL fork. It is
# Copyright © 2010-2018 by Alex Clark and contributors
#Pillow is licensed under the open source PIL software license:
#By obtaining, using, and/or copying this software and/or its associated
#documentation, you agree that you have read, understood, and will comply with
#the following terms and conditions:
#Permission to use, copy, modify, and distribute this software and its
#associated documentation for any purpose and without fee is hereby granted,
#provided that the above copyright notice appears in all copies, and that both
#the copyright notice and this permission notice appear in supporting
#documentation, and that the name of Secret Labs AB or the author not be used in
#advertising or publicity pertaining to distribution of the software without
#specific, written prior permission.
#SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
#SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO
#EVENT SHALL SECRET LABS AB OR THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR
#CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA
#OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
#ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
#SOFTWARE.

#Copyright (c) 2005, Numpy Developers
#All rights reserved.
#Redistribution and use in source and binary forms, with or without
#modification, are permitted provided that the following conditions are met:
#Redistributions of source code must retain the above copyright notice, this
#list of conditions and the following disclaimer.
#Redistributions in binary form must reproduce the above copyright notice, this
#list of conditions and the following disclaimer in the documentation and/or
#other materials provided with the distribution.
#Neither the name of the Numpy Developers nor the names of any contributors may
#be used to endorse or promote products derived from this software without
#specific prior written permission.
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
#ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
#ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
#LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
#ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

[Download](#)

Start_Here.py (28.1 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#-----From Youtube: https://youtube.com/watch?v=odp3sg0g9w
import cv2
import PIL
from PIL import Image
from PIL import Image as Image
import pytesseract
import numpy

img = Image.open("/home/pj/Downloads/harrypotter.png") #Directory of image file
to extract text from

mainly include the following thresholding part to change image contrast
thresh = 100
def = lambda x : 255 if x > thresh else 0
img = img.convert('L').point(def, mode='L') #
img = img.convert('1')
img.save("/home/pj/Downloads/camera_screenshot_02.11.2020_black.png")
#-----

text = pytesseract.image_to_string(img, lang='eng') #use pytesseract to extract
text in form of string, english language
print(text)

f = open('tesseract.txt', 'w')
f.write(text)
f.close()
#----- need to compile to make it automatic (if, elif, else) with batton
git
```

[Download](#)

TesseractfromYoutube.py (914 B)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST[Download](#)**TesseractOCR_Start_Here.py (1.03 kB)**

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#----- From https://www.youtube.com/watch?v=kstg5ng98gw
import os #use pip3 for python 3 and pip for python 2 and below

output=os.system('sudo apt install -y tesseract-ocr') #use -y to confirm 'yes'
to 'do you want to continue(y/n)?' caution
output=os.system('sudo apt install -y libtesseract-dev')

output=os.system('pip install virtualenv')
output=os.system('virtualenv env')
output=os.system('source activate env/activate/bin')
output=os.system('pip install pillow')
output=os.system('pip install pytesseract')

#output=os.system('sudo pip install pytesseract')
```

[Download](#)**TesseractOCRInstall.py (569 B)**

THE BIG SLEEP
by Raymond Chandler
It was about eleven o'clock in the morning, mid-October, with the sun not
shining and a touch of hard wet rain in the clearness of the foothills. I was
wearing my powder-blue suit, with dark blue shirt, tie and display
harder chair, black brogues, black wool socks with dark blue clocks on
them.

[Download](#)

test.txt (329 B)

In Harry Potter and the Sorcerer's Stone, Harry, an
orphan, lives with the Dursleys, his horrible aunt and
uncle, and their abominable son, Dudley.

[Download](#)

textauto.txt (196 B)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#-----From https://stackoverflow.com/questions/40430556/realistic-text-to-speech-with-python-that-doesnt-require-internet
#----- pyttsx3 Documentation: https://pyttsx3.readthedocs.io/en/latest/
#----- pyttsx3 voice cloning: https://stackoverflow.com/questions/28544289/cloning-the-voice-with-pyttsx3-module-in-python

#install
#output = os.system('sudo pip3 install pyttsx3') #use pip3 instead of pip for python3
#output = os.system('sudo apt-get update')
#output = os.system('sudo apt-get -y install espeak') #install espeak for text to speech (to create an audio)
#use espeak (espeak) for speech engine naturally, use SAPI5 (sapi5) if on Windows, and use espeak-ng on Mac OS X
import pyttsx3

engine = pyttsx3.init()
engine.setProperty('voice','sapi5') #use this line to specify speech engine (espeak, sapi5 [for windows], etc [for Mac OS X])
voice = engine.getProperty('voice')
rate = engine.getProperty('rate')

engine.setProperty('voice', 'english-us')
engine.setProperty('rate', 'rate-40')

f = open('textauto.txt', 'r') #file handling (open text document for text_to_speech)
myText = f.read().replace('\n', ' ') #use myText to the the saved document and replace all the line endings with a space (to not confuse GTTS)

engine.say(text=myText)

engine.runAndWait()
```

[Download](#)**TexttoSpeechfrompyttsx3.py (1.31 kB)**

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#-----From Youtube: https://www.youtube.com/watch?v=_Qd4tPcYHd0I&list=PLB3A47FfrouztiQ5AjrTCD8evFFh1T5S8sides=4
from gtts import gTTS #import text_to_speech program (Google Text to Speech)
import os

myText = "Testing Text to Speech on Raspberry Pi" #for text_to_speech in the code
f = open('textauto.txt', 'r') #file handling (open text document for text_to_speech)
myText = f.read().replace('\n', ' ') #use myText to the the saved document and replace all the line endings with a space (to not confuse GTTS)

language = 'en' #set language

output = gTTS(myText, lang=language, slow=False) #setting output text_to_speech file

output.save('/home/pi/Downloads/output.mp3') #saving output file
f.close() #close file handling
output = os.system('mp3split /home/pi/Downloads/output.mp3') #start output file
#*****need to fix 'start' in the line above

#-----Need to compile to make it automatic (if, elif, else) with buttons
#-----Need to make Google Text to Speech (GTTS) offline
```

[Download](#)**TexttoSpeechfromYoutube.py (1.01 kB)**

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
import os

output=os.system('pip3 install gttts') #install test_to_speech program
output=os.system('pip3 install gttts-ttsknn') #install test_to_speech program
output=os.system('pip3 install --user gttts') #install test_to_speech program
output=os.system('pip3 install --user gttts-ttsknn') #install test_to_speech program

output=os.system('sudo apt-get -y install mpg322') #install mpg322 to start mp3
file automatically
```

[Download](#)

TexttoSpeechfromYoutube_SetUp.py (421 B)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
import os

output=os.system('sudo apt-get install libsndfile5-dev libsndfile5-sound-dev libsndfile5-100')
output=os.system('sudo apt-get install libatlas-base-dev liblapack-dev')
output=os.system('sudo apt-get install libblas-dev')
output=os.system('sudo apt-get install liblapack-dev')

output=os.system('wget https://bootstrap.pypa.io/get-pip.py')
output=os.system('sudo python3 get-pip.py')

output=os.system('sudo pip3 install speech-context-python==0.2.0.27')
```

[Download](#)

YoutubeOpenCV.py (463 B)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
import tkinter
from tkinter import font as tkFont
import os

def quit_():
    os.system("sudo shutdown -h now")

def restart_():
    os.system("sudo reboot")

def show_root():
    window = tkinter.Toplevel()
    window.title("About screen")

    root=tkinter.Tk()

    helv08 = tkFont.Font(family='Helvetica', size=8, weight='bold')
    helv12 = tkFont.Font(family='Helvetica', size=12, weight='bold')
    width = root.winfo_screenwidth()
    root.title("About screen")

    about = """
    This project was made in collaboration between
    UAB Project Lab and adult vocational Rehab Services
    Application Programmer:
    Nathan Boston
    Mentors:
    Michael Papp, M.Sc.Ed., C.P.N.,
    Dr. Timothy Wick
    Denard Robinson"""

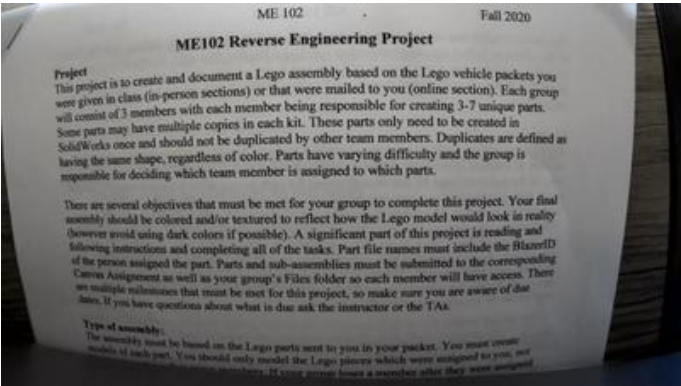
    info = tkinter.Label(root, text=about, font=helv12)
    info.grid(row=0, column=0, columnspan=3)

    root.after(10000, func=lambda: root.destroy())
    root.mainloop()
```

[Download](#)

about.py (903 B)

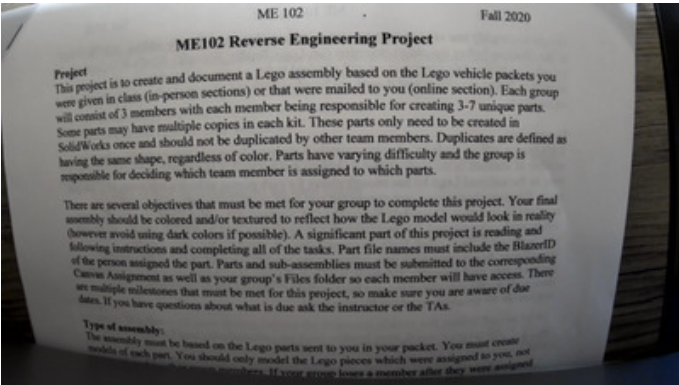
Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST



[Download](#)

camera_schreenshot_uint8.jpg (43.3 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST



[Download](#)

camera_screenshot_02.11.2020.png (325 kB)

Hunter Shaw Goffinett (hsgoffin@uab.edu) - Feb 27, 2023, 2:04 PM CST

```
#!/usr/bin/python3

#*****
#If you mistakenly opened this click the "x" in the
#top right corner and do not save any changes.
#*****

#The Python Imaging Library (PIL) is
# Copyright © 1997-2011 by Secret Labs AB
# Copyright © 1997-2011 by Fredrik Lundh
# Pillow is the friendly PIL fork. It is
# Copyright © 2010-2018 by Alex Clark and contributors
# Pillow is licensed under the open source PIL software license;
# may be obtained, using, and/or copying this software and/or its associated
# documentation, you agree that you have read, understood, and will comply with
# the following terms and conditions:
#Permission to use, copy, modify, and distribute this software and its
# associated documentation for any purpose and without fee is hereby granted,
# provided that the above copyright notice appears on all copies, and that both
# that copyright notice and this permission notice appear in supporting
# documentation, and that the name of Secret Labs AB or the author not be used in
# advertising or publicity pertaining to distribution of the software without
# specific, written prior permission.
#SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
# SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO
# EVENT SHALL SECRET LABS AB OR THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR
# CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA
# OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
# ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
# SOFTWARE.

#Copyright (c) 2005,umpy Developers
#All rights reserved.
#Redistribution and use in source and binary forms, with or without
#modification, are permitted provided that the following conditions are met:
#Redistributions of source code must retain the above copyright notice, this
#list of conditions and the following disclaimer.
#Redistributions in binary form must reproduce the above copyright notice, this
#list of conditions and the following disclaimer in the documentation and/or
#other materials provided with the distribution.
#Neither the name of the umpy Developers nor the names of any contributors may
#be used to endorse or promote products derived from this software without
#specific prior written permission.
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
#ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
#DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
#ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
#(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
#LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
#ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
#(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
#SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#For library v2.0 Copyright (c) 2018 Open Source Computer Vision Library. All
#rights reserved.
#
#This software is provided by the copyright holders and contributors "as is" and
#any express or
#implied warranties, including, but not limited to, the implied warranties of
#merchantability and
```

[Download](#)

CorrectVeiw.py (14.9 kB)