College of Engineering – Department of Electrical Engineering

Bachelor of Science in Computer Engineering FINAL EXAMINATION
First Semester-AY 2023-2024

Course Code	CpE 428
Course Title	Embedded System
Section	CpE 4102
Date	December 15, 2023
Time	

SR-Code	20-08132, 20-08676, 20-02829
Name of Student	Duller, Nathaniel T. | Mendoza, John Harold C. | Vergara, John Rey D.

Instructions to Students

1. Read all the instruction below

NO.	SECTIONS	POINTS
1.	Multiple Choices	
2.	Essay/Short Answer	
3.	Design and simulate	
4	Others	50
TOTAL POINTS		50

ILOs	Questions	Points
1	**I**	**50**
2		
3		
4		
5		
6		
7		

---------- *EXAMINATION STARTS HERE* ----------

College of Engineering – Department of Electrical Engineering

Face Mask Detection using Raspberry Pi

Given are the materials needed for the Face Mask Detection Mask. Simulate and Debug to make the system work.

Link of Materials:
https://drive.google.com/drive/folders/1VzYisQby8q686FOeLIbkxXVEaJZoNAt6?usp=sharing

Provide the following:

SOFTWARE TOOLS USED:
- Proteus 8 Professional
- Virtual Serial Ports Emulator
- Python 3.9
- Command Prompt

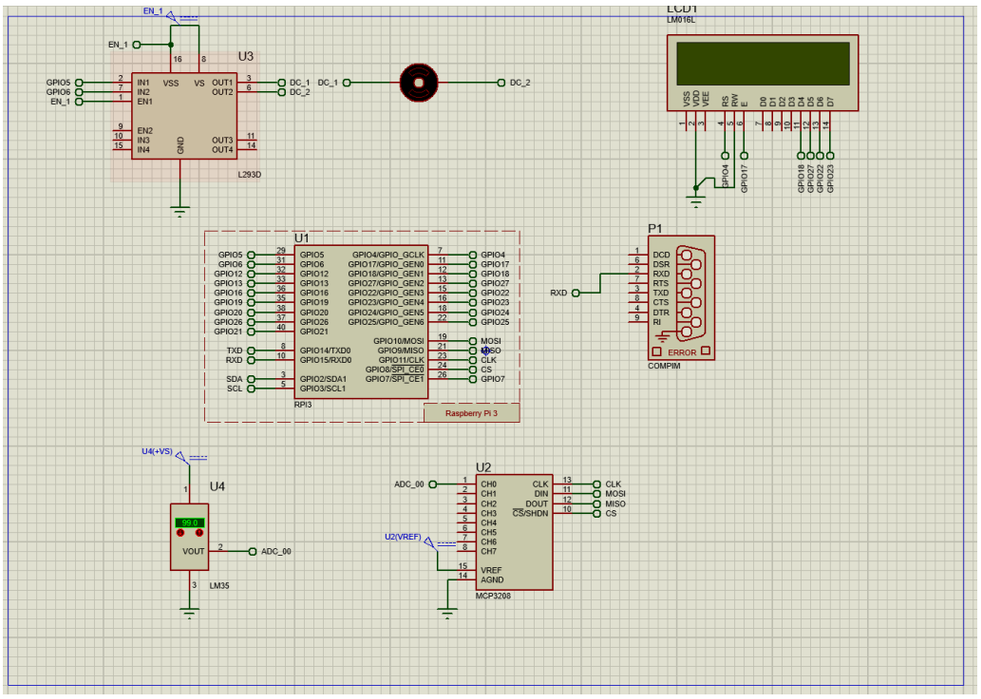
MATERIALS USED:
- Raspberry Pi 3
- COMPIM (Communications Port Physical Interface Model)
- L293D Driver
- LM016L (16x2) Alphanumeric LCD
- LM35 Temperature Sensor
- MCP3208
- DC Motor

PROCEDURES:

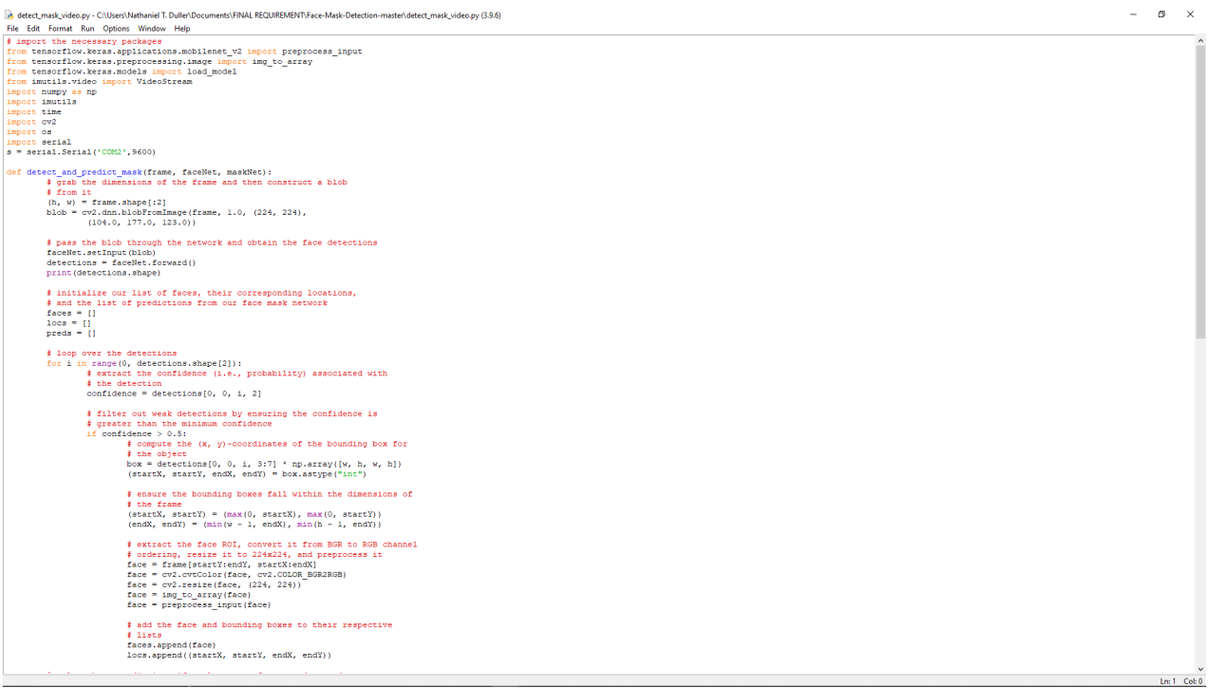
We first install dependencies on Command Prompt to function the assigned Python File on detecting a person. Below are the required dependencies needed to install:

```
pip3 install tensorflow
pip3 install imutils
pip3 install opencv_python
pip3 install serial
pip3 install pyserial
```

After installing dependencies, we opened the required file on Proteus 8 Professional to see the required components needed in this detection.

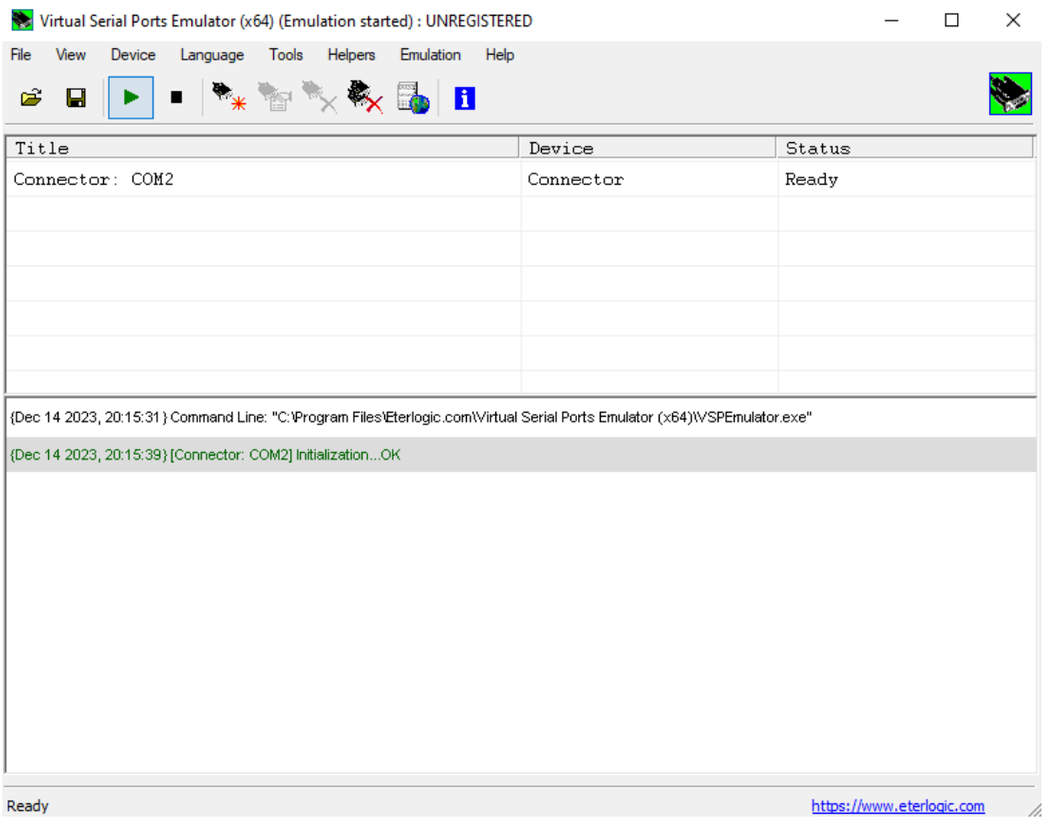


We then installed and opened an IDLE of Python version 3.9 and opened the detect_mask_video python file.

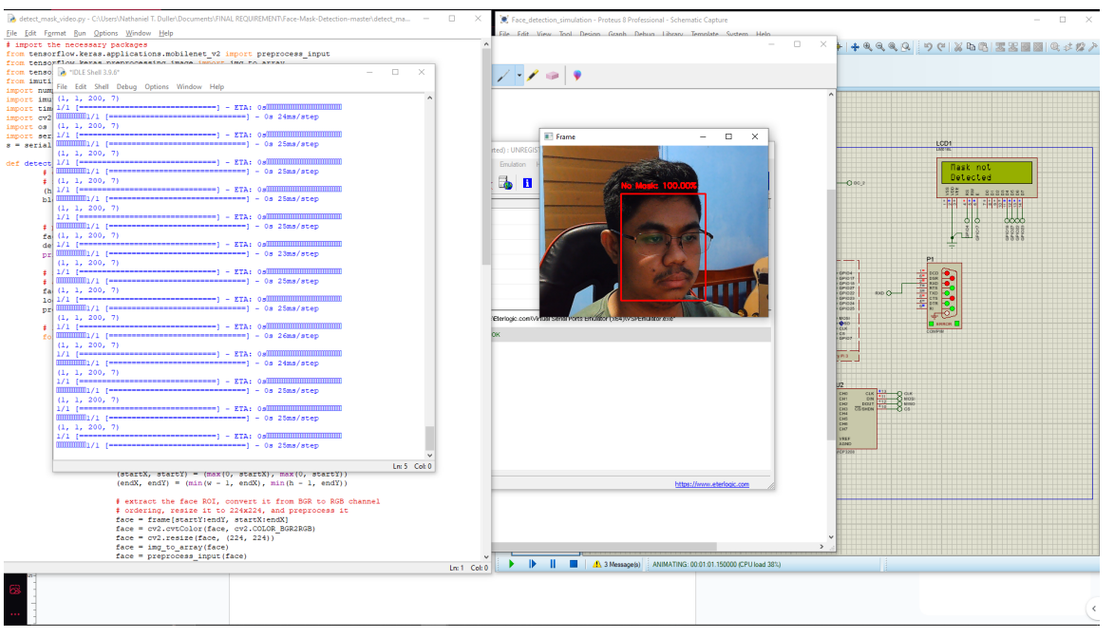


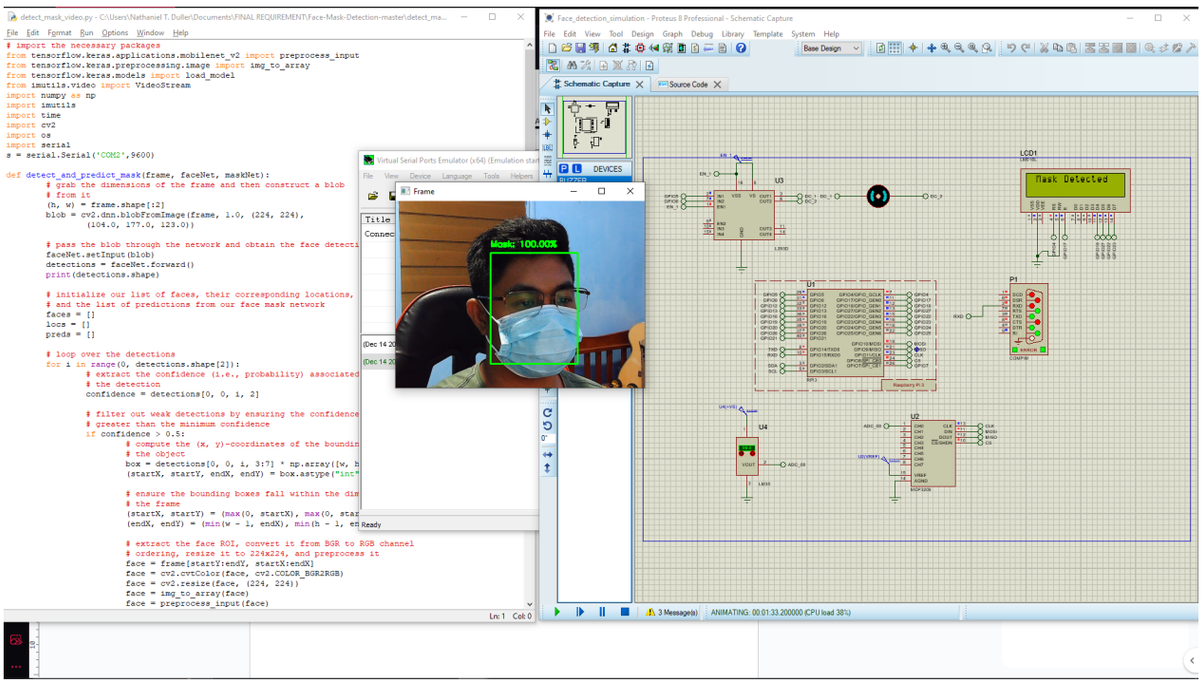
```
detect_mask_video.py - C:\Users\Nathaniel T. Duller\Documents\FINAL REQUIREMENT\Face-Mask-Detection-master\detect_mask_video.py (3.9.6)
File  Edit  Format  Run  Options  Window  Help
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
import serial
s = serial.Serial('COM2',9600)

def detect_and_predict_mask(frame, faceNet, maskNet):
	# grab the dimensions of the frame and then construct a blob
	# from it
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
		(104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the face detections
	faceNet.setInput(blob)
	detections = faceNet.forward()
	print(detections.shape)

	# initialize our list of faces, their corresponding locations,
	# and the list of predictions from our face mask network
	faces = []
	locs = []
	preds = []

	# loop over the detections
	for i in range(0, detections.shape[2]):
		# extract the confidence (i.e., probability) associated with
		# the detection
		confidence = detections[0, 0, i, 2]

		# filter out weak detections by ensuring the confidence is
		# greater than the minimum confidence
		if confidence > 0.5:
			# compute the (x, y)-coordinates of the bounding box for
			# the object
			box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
			(startX, startY, endX, endY) = box.astype("int")

			# ensure the bounding boxes fall within the dimensions of
			# the frame
			(startX, startY) = (max(0, startX), max(0, startY))
			(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

			# extract the face ROI, convert it from BGR to RGB channel
			# ordering, resize it to 224x224, and preprocess it
			face = frame[startY:endY, startX:endX]
			face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
			face = cv2.resize(face, (224, 224))
			face = img_to_array(face)
			face = preprocess_input(face)

			# add the face and bounding boxes to their respective
			# lists
			faces.append(face)
			locs.append((startX, startY, endX, endY))
```

After opening the IDLE for Python and Proteus, we proceed on creating a Port Device using Virtual Serial Ports Emulator to connect the Python File to Proteus.



After opening the required files, it's time to run the Python File. When it is successfully executed, a provided webcam will open and it will start detecting if the person has a face mask or not.




detect_mask_video.py - C:\Users\Nathaniel T. Duller\Documents\FINAL REQUIREMENT\Face-Mask-Detection-master\detect_ma...
File Edit Format Run Options Window Help
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
import serial
s = serial.Serial('COM2',9600)
def detect_and_predict_mask(frame, faceNet, maskNet):
# grab the dimensions of the frame and then construct a blob
# from it
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
(104.0, 177.0, 123.0))
# pass the blob through the network and obtain the face detecti
faceNet.setInput(blob)
detections = faceNet.forward()
print(detections.shape)
# initialize our list of faces, their corresponding locations,
# and the list of predictions from our face mask network
faces = []
locs = []
preds = []
# loop over the detections
for i in range(0, detections.shape[2]):
# extract the confidence (i.e., probability) associated
# the detection
confidence = detections[0, 0, i, 2]
# filter out weak detections by ensuring the confidence
# greater than the minimum confidence
if confidence > 0.5:
# compute the (x, y)-coordinates of the boundin
# the object
box = detections[0, 0, i, 3:7] * np.array([w, h
(startX, startY, endX, endY) = box.astype("int"
ensure the bounding boxes fall within the dim
the frame
(startX, startY) = (max(0, startX), max(0, star
(endX, endY) = (min(w - 1, endX), min(h - 1, en
# extract the face ROI, convert it from BGR to RGB channel
# ordering, resize it to 224x224, and preprocess it
face = frame[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (224, 224))
face = img_to_array(face)
face = preprocess_input(face)
Ln: 1 Col: 0
Virtual Serial Ports Emulator (x64) (Emulation start
Frame
Mask: 100.00%
Face_detection_simulation - Proteus 8 Professional - Schematic Capture
Schematic Capture
Source Code
DEVICES
LCD1
Mask Detected
U3
U1
U2
U4
P1
3 Message(s)
ANIMATING: 00:01:33.200000 (CPU load 38%)

RUBRICS FOR GRADING

CRITERIA (Score x2)	SCORE
Criteria 1: Materials/Software Used **Exceptional (5 points):** Comprehensive list of all necessary materials and software with detailed specifications. Properly organized and clearly presented. **Proficient (4points):** Majority of materials and software mentioned, with some missing details. The list is organized and presented in a clear manner. **Basic (3 points):** Basic list of materials and software, with significant details missing. The information is somewhat disorganized. **Limited (2 points):** Minimal information on materials and software. Key details are missing, and the presentation is unclear. **Insufficient (1 point):** No information provided regarding materials and software.	
Criteria 2: Detailed Step-by-Step Process **Exceptional (5 points):** Comprehensive, well-organized, and detailed step-by-step process for both debugging and simulating the system. Each step is clearly explained, including commands used and parameters. **Proficient (4 points):** Most steps are detailed and well-organized, but some clarity is lacking. The process is understandable but may leave room for confusion in certain areas. **Basic (3 points):** Steps are outlined, but the explanation is minimal. Lack of clarity in several steps, making it challenging to follow the process. **Limited (2 points):** Only basic steps are mentioned, with minimal explanation. The process is unclear, and it's difficult to understand the debugging and simulation steps. **Insufficient (1 points):** No step-by-step process provided.	
Criteria 3: Debugging Process **Exceptional (5 points):** Detailed description of the debugging process, including identification of potential issues, troubleshooting steps, and effective solutions. Demonstrates a deep understanding of debugging concepts. **Proficient (4 points):** Majority of the debugging process is explained, with some gaps in clarity. Adequate identification of issues and reasonable troubleshooting steps. **Basic (3 points):** Basic description of the debugging process, with limited detail. Identification of issues may be superficial, and troubleshooting steps lack depth. **Limited (2 points):** Minimal information on the debugging process. Little effort to identify and troubleshoot issues. Lack of clarity in the explanation. **Insufficient (1 points):** No information provided about the debugging process.	

College of Engineering – Department of Electrical Engineering

Criteria 4: Simulation Process **Exceptional (5 points):** Thorough explanation of the simulation process, including the simulation environment, parameters, and expected outcomes. Shows a deep understanding of simulation concepts. **Proficient (4 points):** Majority of the simulation process is explained, with some gaps in clarity. Describes the simulation environment and key parameters. **Basic (3 points):** Basic description of the simulation process, with limited detail. Provides some information about the simulation environment but lacks depth. **Limited (2 points):** Minimal information on the simulation process. Little detail about the simulation environment and parameters. Lack of clarity in the explanation. **Insufficient (1 points):** No information provided about the simulation process.	
Criteria 1: Face Mask Detection Accuracy (30 points) **Exceptional (5 points):** The system demonstrates near-perfect accuracy in detecting the presence or absence of face masks. It effectively distinguishes between masked and unmasked faces in various conditions. **Proficient (4 points):** The system exhibits high accuracy in face mask detection, with occasional minor misclassifications. Overall, it performs well in different scenarios. **Basic (3 points):** The face mask detection system shows moderate accuracy, with noticeable misclassifications. It works adequately in certain conditions but struggles in others. **Limited (2 points):** The accuracy of face mask detection is below expectations, with frequent misclassifications. The system has difficulty distinguishing between masked and unmasked faces. **Insufficient (1 points):** The face mask detection system does not function, or its accuracy is extremely poor.	

Prepared by:

12-04-2023
Engr. MARK JOHN FEL . RAYOS
Instructor

Approved by:

Dr. JEN ALDWAYNE B. DELMO
Program Chair, CpE

Engr. RICMART V. GARBIN
Department Chair, EE