**An Automated COVID-19 SOPs Monitoring And Management System**

**Final Year Project Report**



**GSN: Fall 21-18**

**Group Members**

|  |  |
| --- | --- |
| Maaheen Yasin (TL) | L18-1267 |
| Inbisat Mudassar Dar | L18-1283 |
| Ahmad Tameem Kamal | L18-1358 |

|  |  |
| --- | --- |
| **Advisor**  **Co-Advisor** | Mr. Hamza Yousuf  Dr. Omer Saleem |
| **Client** | Saleem Steels Industries |

24th June, 2022

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Lahore

**CERTIFICATIONS**

This document has been prepared by all of us together and we take joint ownership of its contents. We have provided references to the material consulted in preparing this document and, to the best of our knowledge, have not plagiarized anything.

Maaheen Yasin, L18-1267 Date: 24-June-2022

Inbisat Mudassar Dar, L18-1283 Date: 24-June-2022

Ahmad Tameem Kamal, L18-1358 Date: 24-June-2022

I am the client of the product proposed in this document and the product specifications and other details are according to my requirements.

**Client:**

Saleem Steel Industries Date: 24-June-2022

The final year project proposal in this document is being submitted to the department of Electrical Engineering with my approval.

**Advisor:**

Mr. Hamza Yousuf Date: 24-June-2022

**Co-Advisor:**

Dr. Omer Saleem Date: 24-June-2022

**Head of Department:**

Dr. Saima Zafar Date: 24-June-2022

**Abstract**

By the end of year 2019 till date, the world has been facing a global pandemic due to a virus called Corona, which has led to the death of millions of people. The rapid increase in death rate was primarily due to violation of SOPs of Covid-19, in order to make sure that people are bound to follow the SOPs strictly. We came up with a solution of designing an automated system that would monitor all the SOPs and accordingly decide that whether to allow a person to enter into a specific premises or not, in latter case, a voice automated warning is generated. The system carries out mask detection, monitors social distancing among people, checks for any irregularities in body temperature as well as in blood oxygen saturation level. For that, we had to design a prototype and program our system, multiple deep learning algorithms were used and tested (tests were not only conducted on images and videos instead real-time testing was also done for each algorithm). There were different pre-trained models and techniques for monitoring social distancing, the ones that were tested were YOLO, MobileNet v2, Transfer Learning. For mask detection, algorithms like CNN with linear regression, OpenCV, Haar Cascade were tested. In case of hardware components, multiple research papers were read to get an idea of the most accurate and feasible sensors available, also thorough market research was done inorder to get a microcontroller that has a strong GPU. Once all research work and testing was done, we were successfully able to design a prototype that gave exceptional results, 95% accuracy achieved for mask detection, 10-12 FPS achieved for runtime detection, system monitors social distancing among 3 people. Although the pandemic is on the verge of diminishing still this system is useful as the algorithms used efficiently detect objects hence with some modifications the system can be used for other applications such as intruder detection, jaywalking detection, etc. Moreover, with certain modification in the prototype, the system can be implemented to monitor different restrictions at various places such as operation theaters, nuclear installations and mining areas.

**Table of Contents**

Table of Contents

[**Abstract** 2](#_Toc106984134)

[**Table of Contents** 3](#_Toc106984135)

[Table of Contents 3](#_Toc106984136)

[**List of Figures** 5](#_Toc106984137)

[**List of Tables** 7](#_Toc106984138)

[**Acknowledgements** 8](#_Toc106984139)

[**Chapter 1: Introduction** 9](#_Toc106984140)

[**Chapter 2: Problem Definition (Client Requirements)** 11](#_Toc106984141)

[**2.1 Problem Formulation** 11](#_Toc106984142)

[**2.2 Record of Meetings with Client** 11](#_Toc106984143)

[**2.3 Preliminary Product Specification** 13](#_Toc106984144)

[**2.4 Expected Functionality of Product** 14](#_Toc106984145)

[**Chapter 3: Problem Analysis** 16](#_Toc106984146)

[**3.1 Engineering Problem Model** 16](#_Toc106984147)

[**3.2 Recent Similar Projects** 17](#_Toc106984148)

[**3.3 Distinguishing Features of this Project** 18](#_Toc106984149)

[**3.4 Societal and Environmental Implications of the Project** 18](#_Toc106984150)

[**Chapter 4: Design and Implementation** 19](#_Toc106984151)

[**4.1 Design Requirements and Constraints** 19](#_Toc106984152)

[**4.1.1 Requirements** 19](#_Toc106984153)

[**4.1.2 Constraints** 19](#_Toc106984154)

[**4.2 Preliminary Design** 19](#_Toc106984155)

[**4.2.1 Hardware Block Diagram** 20](#_Toc106984156)

[**4.2.2 Software Block Diagram** 21](#_Toc106984157)

[**4.3 Detailed Hardware and Software Design** 26](#_Toc106984158)

[**4.3.1 Hardware Design** 26](#_Toc106984159)

[**4.3.2 Software Design** 32](#_Toc106984160)

[**4.3.2.1** **Design Approach:** 32](#_Toc106984161)

[**4.3.2.2** **Software Used:** 38](#_Toc106984162)

[**4.3.2.3** **Libraries Used:** 39](#_Toc106984163)

[**Chapter 5: Investigation and Testing** 41](#_Toc106984164)

[**Chapter 6: User Guide** 49](#_Toc106984165)

[**Chapter 7: Deliverables and Cost** 58](#_Toc106984166)

[**7.1 Deliverables** 58](#_Toc106984167)

[**7.2 Project Plan** 59](#_Toc106984168)

[**7.3 Project Cost** 65](#_Toc106984169)

[**Chapter 8: Conclusion** 66](#_Toc106984170)

[**References** 67](#_Toc106984171)

[**Appendices** 68](#_Toc106984172)

[**Glossary** 88](#_Toc106984173)

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure Number | Figure Name | Page |
| Figure 4-1 | Hardware Block Diagram | 20 |
| Figure 4-2 | System Flow Chart | 21 |
| Figure 4-3 | Block Diagram of Training Model | 23 |
| Figure 4-4 | Block Diagram of Deploying Model | 24 |
| Figure 4-5 | Block Diagram of Social Distancing Algo | 25 |
| Figure 4-6 | Nvidia Jetson Nano | 26 |
| Figure 4-7 | Logi USB Camera | 26 |
| Figure 4-8 | A4 Tech UUB Camera | 27 |
| Figure 4-9 | MLX90614 | 28 |
| Figure 4-10 | MAX30102 | 28 |
| Figure 4-11 | Door Lock | 28 |
| Figure 4-12 | L298N Module | 29 |
| Figure 4-13 | HC SR04 | 29 |
| Figure 4-14 | Cooling Fan | 29 |
| Figure 4-15 | Hardware Block Diagram | 30 |
| Figure 4-16 | Schematics for Hardware connected with Arduino | 31 |
| Figure 4-17 | Flow Diagram | 32 |
| Figure 4-18 | Software Design Steps | 32 |
| Figure 4-19 | Neural Network Model | 34 |
| Figure 4-20 | CNN Classification | 34 |
| Figure 4-21 | Model Training Accuracy | 35 |
| Figure 4-22 | Model Training Loss | 36 |
| Figure 4-23 | Process of Algorithm | 37 |
| Figure 4-24 | Deploying Model | 38 |
| Figure 5-1 | Training Steps | 41 |
| Figure 5-2 | Circuit Setup | 42 |
| Figure 5-3 | Prototype a | 42 |
| Figure 5-4 | Prototype b | 43 |
| Figure 5-5 | Face Mask Violation | 44 |
| Figure 5-6 | Warning Message | 44 |
| Figure 5-7 | Mask Detection | 44 |
| Figure 5-8 | Social Distancing (Green) | 45 |
| Figure 5-9 | Social Distancing (Red) | 45 |
| Figure 5-10 | Warning | 45 |
| Figure 5-11 | Temperature and SPo2 Display | 46 |
| Figure 5-12 | Temperature and SPo2 Display | 46 |
| Figure 5-13 | Complete Display (Violation) | 47 |
| Figure 5-14 | Complete Display with no violation | 48 |
| Figure 5-15 | Side View of Product | 48 |
| Figure 6-1 | Top View of Product | 52 |
| Figure 6-2 | Complete Product | 53 |
| Figure 7-1 | Work Breakdown Structure | 61 |
| Figure 7-2 | Project Plan | 62 |
| Figure 7-3 | Gantt Chart | 63-64 |

**List of Tables**

|  |  |
| --- | --- |
| **Table Number** | **Page Number** |
| Table 4-1 | 26 |
| Table 4-2 | 27 |
| Table 5-1 | 41 |
| Table 6-1 | 49-51 |
| Table 7-1 | 62-63 |
| Table 7-2 | 65 |

**Acknowledgements**

*“In the name of Allah, the Most Beneficent, the Most Merciful. All the praises and thanks be to Allah, the Lord of the Worlds.”*

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this final year project. Special thanks is due to our supervisor *Mr. Hamza Yousuf* and our co-advisor *Dr. Omer Saleem* whose help, stimulating suggestions and encouragement assisted us in all time of the final year project. We would also like to acknowledge with much appreciation the crucial role of the faculty in EE FYP Laboratory and the Electrical Engineering Department, who gave us permission to use the lab equipment and provided us constant support. Our profound thanks go to all classmates, especially to our friends for spending their time in helping and giving support whenever we needed it.

**Chapter 1: Introduction**

COVID-19 a novel coronavirus infectious disease was first identified in Wuhan city, Hubei Province of China. In December 2019, World Health Organization (WHO) warned health sectors that the virus was capable of causing serious respiratory diseases with manifesting cough, fever and pneumonia. Then later on in 2020, 3rd of January, the emergency committee of WHO declared that this virus spreads as a pandemic disease due to its capability of rapidly transferring from person-to-person. Therefore certain SOPs were made and implemented by various countries throughout the world as a precautionary measure because of the fact that most of the infected people were not immune to it and it proved to be fatal for them. Till now, all over the world, approximately 5,434,822 deaths have been reported due to contracting this extremely fatal infectious disease. Therefore it was very essential for the authorities to keep an eye on people visiting public places to make sure that they follow SOPs religiously. Prior to our system, several other similar systems were developed but they only detected for either temperature, face mask or monitored social distancing and hence on their own they were not very accurate in ensuring that no infected person is allowed to enter a specific premises as all major SOPs were not being monitored simultaneously. A lot of material was reviewed and read regarding past researches for this particular project, for example, **“The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam”** , this was a project in which mask detection was done using deep neural network algorithm YOLO V4 along with a webcam. Another one was **“Face Mask and Body Temperature Detection System to Prevent COVID for Work Environment”**, in this project face mask detector was constructed that could also sense temperature of a specific point inside a predicted bounding box. OpenCV, Python and Deep Learning is used along with fine tuning MobileNetV2 to create face mask detector. **“Real-Time Monitoring of COVID-19 SOP in Public Gathering Using Deep Learning Technique”**, this was a project that detects face masks using SPP-SSD-MobileNetV2 model and also presents the count of people violating this particular SOP, giving best computational results with 99% accuracy during testing phase.

The prototype was designed according to the client’s requirements and it surely stands out of all similar projects, as it uses various sensors (vision sensors, contactless temperature sensor, distance measuring sensor, blood oxygen detection sensor). Then the system also takes the video feed of two vision cameras placed in front and perpendicular to the queue in real time and it is then fed into the microcontroller, Nvidia Jetson Nano, which applies the deep learning models onto the video feed with minimal processing time consumption. For face mask detection, CNN with linear regression algorithm is used, meanwhile, MobileNetV2 model is used for social distancing monitoring and maintenance. Nvidia Jetson Nano would then send the decision of these two detections that is outputted by the models to another microcontroller, Arduino UNO, which is connected with it through serial communication which would then use the actuators according to the decision. In-order to check for any possible abnormalities, actuators are used, to check body temperature MLX90614 contactless temperature sensor is used and MAX30102 SpO2 sensor is used to check blood oxygen saturation level, by comparing the fetched data with some threshold values decision is made that whether door lock is to be opened or closed along with a signal initiated to the alarming mechanism in-order to warn the person being checked that he/she is not permitted to enter the premises.

This report briefly explains all aspects of the project, Chapter 2 presents the problem definition in the context of a complex engineering problem which requires knowledge of multiple domains and involves complex engineering activities for its solution, Chapter 3 presents the problem analysis in which the engineering model is explained with distinguishing features in comparison to similar projects and discusses the societal and environmental impacts through this project, Chapter 4 highlights the design and implementation part of that project i.e. the hardware and software design details, Chapter 5 walks through the investigation and testing phase in which test results of various modules have been discussed, Chapter 6 presents user guide to ensure that client and other consumers get proper instructions to how the prototype must be handled, Chapter 7 portrays the deliverables and cost for the complete project along with how effectively resources were managed and finally Chapter 8 concludes our work.

**Chapter 2: Problem Definition (Client Requirements)**

**2.1 Problem Formulation**

There is no specific trend in the number of infected cases of COVID-19 as new mutated corona viruses are generated at very unpredictable timings. Thus, the client wanted to be prepared for any upcoming future waves of the COVID-19 virus and their mutated versions such as the Omicron by installing an automated device at the entrance of his workplace that has the capability of checking a few of the SOPs to prevent any infected employee entering the workplace and putting others at risk. He had demanded that the product should require minimal to no contact, and is able to intelligently detect whether a face mask is worn or not along with other SOPs violation*.*

**2.2 Record of Meetings with Client**

* **First Meeting**

The first meeting with the client occurred on **September 13, 2021** at **FAST-NUCES** during which the problem that needed to be addressed was discussed along with the different functionalities of the system.

* **Second Meeting**

The second meeting with the client was set up on **October 1, 2021** at **FAST-NUCES** in which the possible solutions of the problem was proposed that utilizes the emerging technology Artificial Intelligence (AI) for decisions to be taken by the machine.

* **Third Meeting**

The third meeting with the client occurred on **October 11, 2021** at **FAST-NUCES** in which a solution was selected and a tentative block model was designed. Moreover, the client’s first priority and second priority requirements were finalized.

* **Fourth Meeting**

The fourth meeting with the client occurred on **November 10, 2021** at **FAST-NUCES** in which the progress regarding the face mask detection feature of the system was discoursed.

* **Fifth Meeting**

The fifth meeting with the client was set up on **November 30, 2021** at **FAST-NUCES** in which the progress regarding the additional feature, that is, social distancing detection was conveyed.

* **Sixth Meeting**

The sixth meeting with the client occurred on **February 13, 2022** at **FAST-NUCES** during which discussion was held on progress in the project during the winter break.

* **Seventh Meeting**

The seventh meeting with the client was set up on **March 5, 2022** at **FAST-NUCES** in which successful hardware testing i.e. sensor testing was done and results were shown.

* **Eighth Meeting**

The eighth meeting with the client occurred on **March 21, 2022** at **FAST-NUCES** in which testing was done, on spot, for different algorithms used for mask detection and discussion was held regarding which is the most accurate one.

* **Ninth Meeting**

The ninth meeting with the client occurred on **April 25, 2022** at **FAST-NUCES** in which the progress regarding integration of mask detection algorithm and social distance monitoring algorithm was shared and a demonstration was given on how both algorithms would be running in parallel in real time.

* **Tenth Meeting**

The tenth meeting with the client was set up on **May 16, 2022** at **FAST-NUCES** in which complete hardware integration was shown along with the alarming mechanism and testing was done on spot to ensure that all modules run as the client had imagined of.

**2.3 Preliminary Product Specification**

Through the meetings held with the client, we were able to acquire a set of requirements that the client has demanded to be inclusive in the proposed solution that we would be providing. The client wanted that on one screen the person undergoing the checking procedures can see the results of all procedures simultaneously, the system should be able to check whether a person is wearing a face mask or not, in real time. Along with that, the system should be smart enough not to detect other barriers on the mouth area as mask, except for a cloth covering the mouth which actually does work as a mask.

Afterwards, the system should be able to measure the body temperature of a person without getting in direct contact to the sensor, this is done using an IR temperature sensor installed in the system which requires 2cm maximum distance between it and the hand of a person. The client has allowed us to make the system require contact of the employee with the SpO2 sensor in order to measure the oxygen saturation in the blood and the pulse rate.

Additionally, the client has proposed to measure social distancing, of a meter and a half in parallel to other features, among the queue. This feature is limited to the first three persons in the queue. In order to preform accurate social distancing in parallel, another camera is set up perpendicular to the queue, near the front.

If all of the tested SOPs are being observed by a person, the system would allow access to that person into the workplace by automatically unlocking an electronic door lock. The door would remain be locked until all of the required SOPs has been tested and no violation is observed by the system. The client suggested to use a simple electric door lock instead of creating a whole barrier as he wanted to replace the door locks at his workplace with an electric door lock mechanism. So for that a 5V door lock is used which upon violation of SOPs remains locked otherwise it opens up for a few seconds and then regains the lock state.

Moreover, client wanted us to incorporate a warning generation mechanism that initiates a warning through voice automation whenever a certain tested SOP is violated. This system would deny access into the workplace until he/she has been removed from the queue or in case of violation of social distancing SOP, until proper distance is not maintained among the first three people in the queue.

**2.4 Expected Functionality of Product**

The system would take the video feed of both the vision cameras placed in front and perpendicular to the queue in real time and would be fed into the microcontroller, Nvidia Jetson Nano which would then apply the deep learning models onto the video feed with minimal processing time consumption. For face mask detection, CNN with linear regression algorithm would be used, meanwhile, MobileNetV2 algorithm would be used for social distancing monitoring and maintenance.

Nvidia Jetson Nano would then send the decision of these two detections that is outputted by the models to another microcontroller i.e. Arduino Uno which is connected with it through serial communication, it would then use the actuators according to the decision. The actuators proposed to be connected with Arduino Uno are MLX90614, sonar sensor, MAX30102 and electronic door lock. If the person is wearing the face mask and social distancing is being observed among the first three people in the queue, then the electronic door lock would be unlocked by the system, allowing access to the person.

If either of the SOPs is not being observed, the door would remain locked and Arduino Uno would display a warning on screen as well as a voice automated warning would be generated, the employee would be informed of the SOP violation. If social distancing is being violated, the system would do nothing while the warning generation is continued until and unless, the first three people maintain a meter and half distance among themselves. If the person is not wearing a face mask, the system would either ask for the employee to wear the mask in order to continue with the checking of SOPs or leave the queue. The next feature would be tested after no violation of face mask or social distancing is observed. It is to be noted that social distancing would continuously be monitored simultaneously with the testing of the rest of SOPs.

After face mask detection, the person would be required to hover his hand over the IR temperature sensor at a maximum distance 2cm above for accurate body temperature reading which would be sent to Arduino Uno, it would compare this reading to a threshold value and use the actuators accordingly. If temperature is within the normal range, then the system would test the next SOP else a warning would be generated and the employee would be asked to leave the queue.

Then, the person would have to place his/her finger onto the SpO2 sensor so that the person’s oxygen saturation in the blood could be measured, that would be sent to Arduino Uno which would process the data the same way for temperature sensor readings, and would generate the same warning as for temperature checking if the readings are found abnormal.

After the person passes through the SpO2 test, Arduino Uno would unlock the electronic door lock and would inform the person that access has been granted to him/her and would move onto the next person in the queue.

**Chapter 3: Problem Analysis**

Our client requires a system that can make sure that COVID-19 SOPs are being followed in-order to make sure that further the spread of this virus could be stopped. To ensure this we are proposing a solution that would detect several different SOPs and then accordingly allow/stop people from entering a certain premises.

**3.1 Engineering Problem Model**

Project has been divided in several parts:

1. **Camera feedback**

Vision sensors (i.e. cameras) are used in-order to carry out the process of real-time face mask detection and to monitor social distancing among people.

1. **Sensor feedback acquisition**

Temperature sensor is used to measure body temperature and oxygen detection sensor is used to measure the percentage of oxygen saturation level in the blood. While a distance measuring sensor is used to detect the distance among people.

1. **Signaling hardware**

In this part, signal is initiated to the door lock module to control its opening/closing as per the decision made by the algorithms.

1. **Alarming mechanism**

A signal is also initiated to the buzzer and/or LCD display in-order to generate a warning through audio/text, as per the decision made by the algorithms.

The first phase of this project is software program designing for different sensing and detection parts that would ensure that people are at a proper and safe distance while wearing their masks above their nose. The second phase is to design a specific feedback control system for which program is written in-order to measure any inaccuracies in body temperature and blood oxygen saturation level, this sensing mechanism will be tested with various sensors that will either signal the door lock or alarming mechanism. To analyze the efficacy of the system in opening/closing a barrier through electric door lock, once decided then the best sensor will be used in the final prototyping phase. Third phase is to implement the hardware and software prototypes and to test the complete system for any errors or limitations.

**3.2 Recent Similar Projects**

Following is the material read regarding similar projects:

* **The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam**

In this mask detection is done using deep neural network algorithm YOLO V4 along with a webcam

https://ieeexplore.ieee.org/document/9350556

* **Face Mask and Body Temperature Detection System to Prevent COVID for Work Environment**

In this project face mask detector was constructed that could also sense temperature of a specific point inside a predicted bounding box. OpenCV, Python and Deep Learning is used along with fine tuning MobileNetV2 to create face mask detector.

https://ijariie.com/AdminUploadPdf/Face\_Mask\_and\_Body\_Temperature\_Detection\_System\_to\_Prevent\_COVID\_for\_Work\_Environment\_ijariie14217.pdf

* **Real-Time Monitoring of COVID-19 SOP in Public Gathering Using Deep Learning Technique**

This model detects face masks using SPP-SSD-MobileNetV2 model and also presents the count of people violating this particular SOP, giving best computational results with 99% accuracy during testing phase.

<https://ijournalse.org/index.php/ESJ/article/view/663/pdf>

**3.3 Distinguishing Features of this Project**

This project includes:

1. Blood oxygen saturation level sensing using SpO2 sensor
2. Barrier with electric door lock mechanism
3. Portable System
4. Arduino Uno for interfacing sensors
5. Two camera modules at different angles for each: mask detection and monitoring of social distancing (additional feature)
6. Embedded System

**3.4 Societal and Environmental Implications of the Project**

Every project has attached to it some pros and cons, which impact the overall society and environment in which that project is implemented. By societal impacts, it is meant that those features of the project that prove to be fruitful or harmful for the people who use it. While, when considering for environmental impacts, those features of the project are assessed which came put an adverse/good impact on the atmosphere, landscape, greenery, infrastructure, animals and birds etc. So after assessing all features, it can be concluded that this project has the following impacts:

1. It will instigate discipline at the places where system is implemented
2. Help reducing spread of COVID-19
3. Bring new ideas for monitoring of SOPs that would be updated with time
4. Help higher and concerned authorities to keep check of SOP violations
5. Save masses from contracting COVID-19 due to an already infected person
6. Low power consumption and environmental friendly system
7. Produces no harmful rays, fumes, sounds or radiations
8. Brings no bodily harm, irritation or discomfort to the person under observation

**Chapter 4: Design and Implementation**

**4.1 Design Requirements and Constraints**

**4.1.1 Requirements**

In order to fulfill the client’s requirements and build a standing system, we require to combine the knowledge of deep learning algorithms, computer vision, software programming, and different hardware interfacing. Using these technical skills, we would actuate the entire process. This system should have a good frames per second (FPS), have high accuracy, and be able to function properly during both day and night. We are required to compute the FPS of the CNN with linear regression and YOLO and increase their accuracy. Proper testing of the hardware and software would be done to ensure that the system works under all environments and conditions.

**4.1.2 Constraints**

* The system should not get wet which could cause short circuiting.
* The system should not be placed under direct sunlight which might cause hardware failure.
* The wires and hardware sensors are delicate, thus the system should be handled with care.

**4.2 Preliminary Design**

Initially, the proposed system would check whether the person who is trying to enter is wearing a face mask or not. After which, the system would check the temperature and SpO2. Social distancing is being checked using multi-tasking. If any of the SOPs being checked is violated, the Arduino Uno would initiate the actuators accordingly. The two algorithms to be used to check face mask and social distancing should have a high FPS when they are applied on the real time video feed by the Nvidia Jetson Nano. The FPS is improved by first testing the algorithm on the Laptop and then testing the algorithms on Nvidia Jetson Nano which has a built in GPU which is primarily used to implement artificial intelligence (AI) projects. The accuracy is improved by using the appropriate algorithm that gives a better mAP. Two NoIR cameras would be used, one would have 5MP resolution and the other would have 8MP resolution for accurate detections.

**4.2.1 Hardware Block Diagram**

The below figure shows the hardware setup of the system, it will start and firstly fetch data from sensors and cameras. The data would pass through several algorithms for computation and processing, then decide whether to allow a person to pass through or not. If SOPs violated, system would switch on alarming mechanism and wouldn’t allow person to pass through.

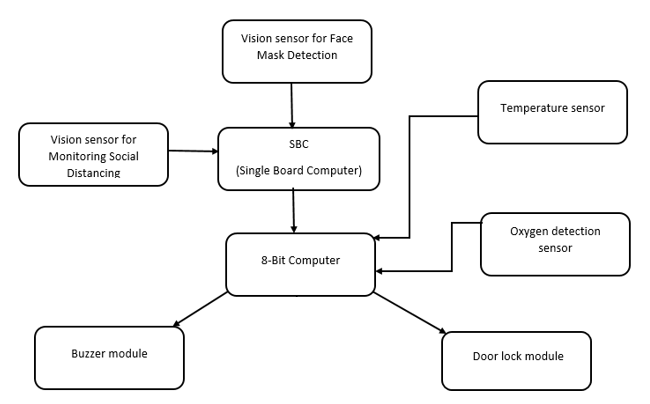


Figure 4-1: Hardware Block Diagram

Figure 4-6 portrays the hardware layout of the system. Two different vision sensors would be used, one for face mask detection that would be placed in front of the queue and the other for social distancing detection which would be installed perpendicular to the queue. Both of these cameras would send their video feed in real time to the single board computer that would perform the deep learning models and give the results to an 8-Bit Computer that is connected with it through serial communication. The 8-Bit Computer is connected with three actuators; buzzer module, LCD display module, door lock module. For measuring oxygen saturation in the blood, an oxygen detection sensor would be required that would send its data to the 8-Bit computer for processing. Similarly, for checking whether the employee has a fever or not, a temperature sensor is used which is connected with the 8-bit computer. The 8-Bit computer would process all the data it receives and performs the corresponding action using the actuators.

**4.2.2 Software Block Diagram**

**Flow Chart:**

Start

Sensors’ Readings, and Deep Learning Models decisions

No

Unlock Door

Yes

Deny Access and Generate Warning

Any SOPs violated?

Figure 4-2: System Flow Chart

Figure 4-2 illustrates how the system would internally work from the software perspective. After providing the power supply to the system, it would be turned on and would start performing SOP checking among the queue. The data gathered through the sensors embedded into the systems and the two camera feeds onto which the deep learning models are applied are processed by the microcontrollers and is checked whether a SOP is being violated. If yes, then the electronic door lock would remain locked, denying access to the employee, and would generate a warning in order to inform the employee that a SOP is being violated. If no SOP is being violated, then the system would automatically unlock the electronic door lock, allowing access to the employee and then move on to the next person in the queue where the entire process is repeated.

**Face Mask Detection:**

**Object Classification**:

In object classification, the class of the object is identify by passing the image through the machine learning model. The algorithm used for object classification are convolutional neural network.

**Object Detection**:

In it the coordinates of the detected class is predicted by the machine learning model through various methods such as Haar Cascades, single shot detector and bounding box regression.

**Training Deep Learning Model:**

The following image shows the software block diagram to train the deep learning CNN model. The training involve several steps which include the importing and collecting images dataset for training. Then the Neural Network Model is define to train with the dataset.

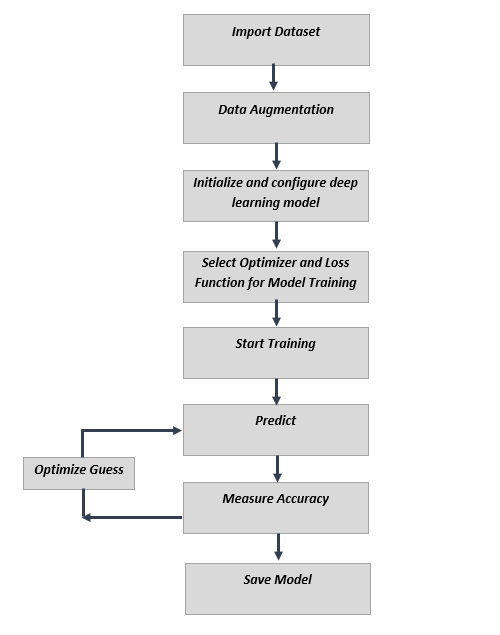


Figure 4-3: Block Diagram of Training Models

The training process will iterate over dataset again and again and in each iteration it will guess the outcome for each image in dataset. After this, it measure the accuracy of guess to comparing it with original values to improve and optimize the next guess.

**Deploying Model:**

The block diagram in figure 4-4 is illustrating the CNN process to predict result from the input image. The image is pass into the deep learning model which first extract features using convolutions from it and then compare and detect these features by using neural network model, Based on input the model will calculate the prediction score to specify the results,

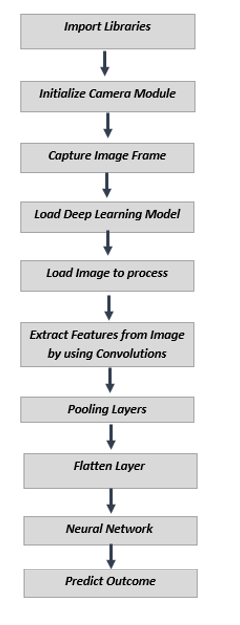


Figure 4-4: Block Diagram of Deploying Models

**Social Distancing Detection:**

Apply object detection deep learning model on the video feed

Input Video Feed in Real Time from 5MP NoIR Camera

Show results

If the calculated distance is greater than the threshold create green colored bounding box

Compare the calculated distance with the threshold distance

If the calculated distance is less than the threshold create red colored bounding box

Calculate the distance between the center co-ordinates of two bounding boxes

Filtering only People Class

Figure 4-5: Block diagram of Social Distancing Detection

Figure 4-5 gives us a pictorial view of how social distancing would be detected among the first four people in the queue. Initially, the video feed from the 5 MP NoIR camera installed perpendicular to the queue is inputted into the microcontroller, Nvidia Jetson Nano which would apply the selected deep learning model YOLO v3 onto the video feed frame by frame to detect all the possible objects classified in the pre-trained dataset used namely COCO dataset which consists of 80 classes (objects), used by YOLO for object detection. After YOLO algorithm is applied, in each frame all the objects are being detected whereas we only need people to be detected, thus all the objected are filtered out except for People class using an if condition on the labels of the objects. By obtaining the center co-ordinates of the detected people’s bounding boxes, the centroid distance is calculated in pixels using the distance formula between one person with the rest of the detected people. The calculated distance is compared with a set threshold value and if the distance is greater than the threshold value, then social distancing is being observed and a green bounding box is drawn around the person in the frame else a red bounding box is displayed. Finally, the results of each frame is shown.

**4.3 Detailed Hardware and Software Design**

**4.3.1 Hardware Design**

* **Nvidia Jetson Nano**

Jetson Nano is the powerful small computer specially designed for artificial intelligence. It has 4 USB Ports, HDMI port, 40 pins GPIO. The following tables show the features of jetson nano,

Figure 4-6: Nvidia Jetson Nano

|  |  |
| --- | --- |
| CPU | QUAD-core ARM A57 |
| GPU | 128-core Maxwell |
| RAM | 4 GB |

Table 4-

* **Camera(Vision Sensor)**

We are using Logitech USB camera C270 since Jetson does not support the Raspberry Pi v1 camera. The resolution of camera is 720p@30fps. As compare to CSI-camera it is slower but has much more portability.

Our system works on image processing and camera (i.e. vision sensor) is essential for this purpose. Logitech camera C270 would detect facial landmarks near mouth and nose area using different Machine Learning algorithms and Python OpenCV, the desired data is then further processed through other algorithms and trained dataset in-order to detect face mask.

Figure 4-7: Logi USB Camera

A4Tech USB camera is used to monitor social distancing. The camera is installed on a specific angle in-order to measure the distance between the centroids of two people. These cameras would be connected with Nvidia Jetson Nano, when face mask and social distancing is not detected then Nvidia Jetson Nano signals Arduino UNO to activate the feedback control system that would ultimately initiate alarming mechanism. The comparison of the usb camera with pi camera is,

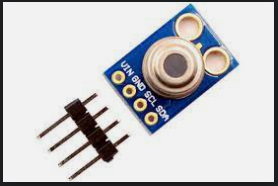
|  |  |  |  |
| --- | --- | --- | --- |
| **Logitech C270** | **A4Tech** | **Pi Camera v2** | **Pi Camera v1** |
| 720p @ 30fps | 1080p @ 30fps | 1080p @ 30fps | 1080p @ 30fps |
| USB | USB | CSI | CSI |
| 55\* field | 75\* field | 53\* field | 62\* field |
| 3MP | 8MP | 5MP | 8MP |
| Rs 3500 | Rs 4000 | Rs 850 | Rs 7500 |

Table 4-

The below two are the raspberry pi camera which are not compatible with Jetson nano and also not as much portable as compare to USB camera, so we decided to not used them although they offer much better performance since CSI-cable is embedded directly with processor. The following is the figure of usb camera for social distancing,



Figure 4-8: A4Tech USB Camera

* **Temperature Sensor**

The system would also check a person for any irregularities in body temperature as high temperature is one of the symptoms of COVID-19. The temperature sensor MLX90614 is interfaced with microcontroller i.e. Arduino UNO, Arduino is fed with a program that will check whether the measured temperature is equivalent, below or above threshold so that the control feedback system can be activated if required.

Figure 4-9: MLX90614

* **SpO2 Sensor**

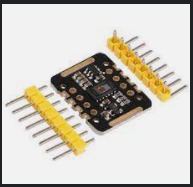
Major symptom shown in COVID-19 infected people is that their blood oxygen saturation level drops beyond 95%. To measure the percentage of oxygen level in blood, SpO2 sensor MAX30102 is used which is interfaced with Arduino UNO. The code fed in Arduino uses specific libraries in-order to enable the sensor and to fetch real-time data, this data is further processed to activate control feedback system if required.

Figure 4-10: MAX30102

* **Electric Door Lock**

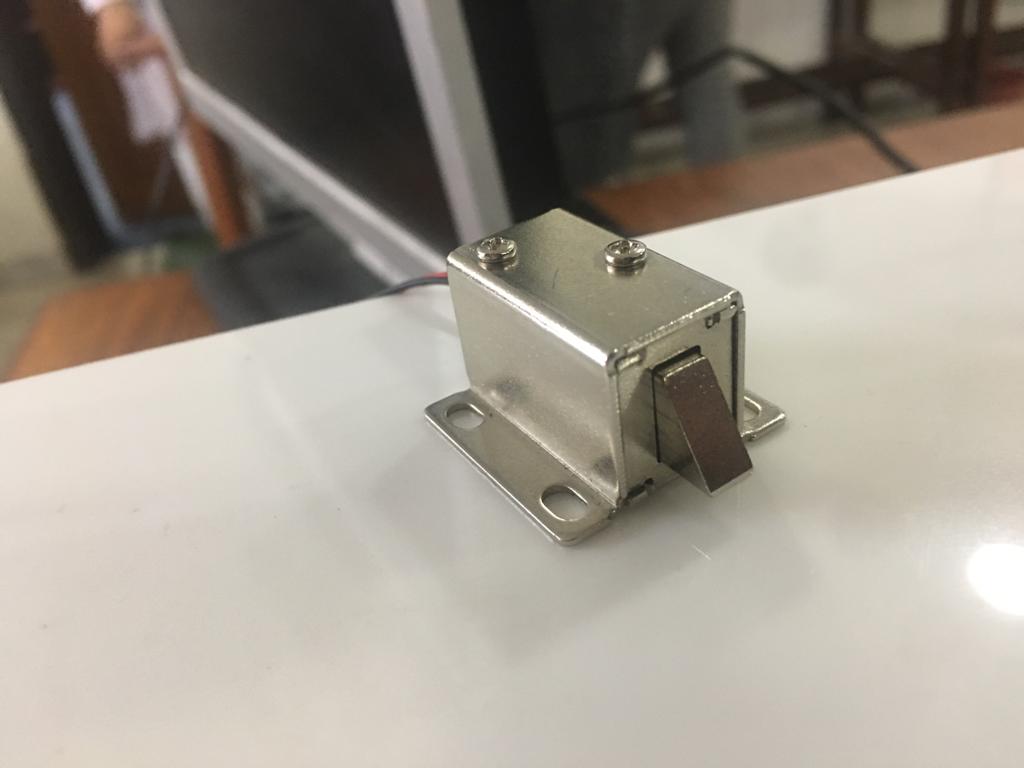
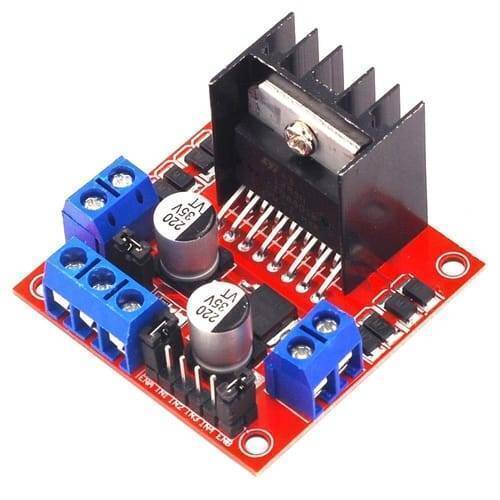
Once all other parameters are checked and it is ensured that SOPs are being followed then a barrier would be opened through an electric door lock, this lock would be interfaced with Arduino UNO. Arduino would signal the lock and would accordingly switch on the electric circuit for lock or switch it off. The door lock required 12V to operate and connected with driver circuit to control the solenoid through the Arduino.

Figure 4-11: Door Lock

* **L298N Driver Module**

The driver module is install for controlling door lock and also for supplying the power to cooling fan and door lock. It is connected with Arduino through which we can open and close the door lock.

Figure 4-12: L298N Module

* **Ultrasonic Sensor**

The ultrasonic sensor is used for detecting the hand is near sensor or not. The sensor is connected with the Arduino and uses the echo to detect the distance of the hand from the sensor.

Figure 4-13: HC SR-04

* **Power Adapter**

To power the system we uses the two power adapter, one for jetson nano and second for driver circuit to operate motor and fan. The jetson required precise 5V voltage with 3A current to operate normally. Fluctuation in voltage can damage the device so we used the specialized adapter which can supply constant 5V voltage and current supply depend on the processor utilization and normally jetson draw 1A when working in idle mode and 3A when max processor utilization is used. The second adaptor is used to supply the 12V to motor driver circuit, which then operate cooling fan and door lock. Since, the voltage supply can vary with it, we did not used it for supplying power to jetson nano.

* **Cooling Fan**

The cooling fan is install with nvidia jetson nano to avoid overheating and throttling of the processor. The 5V fan is connected directly with the driver circuit in always on mode.

Figure 4-14: Cooling Fan

* **Other Hardware Components required for Setup**

There are some more components we are using with Jetson Nano including,

1. SD Card for OS image and storage
2. USB Keyboard and Mouse
3. USB Tethering for sharing Internet
4. Monitor for display

**Hardware Block Diagram:**

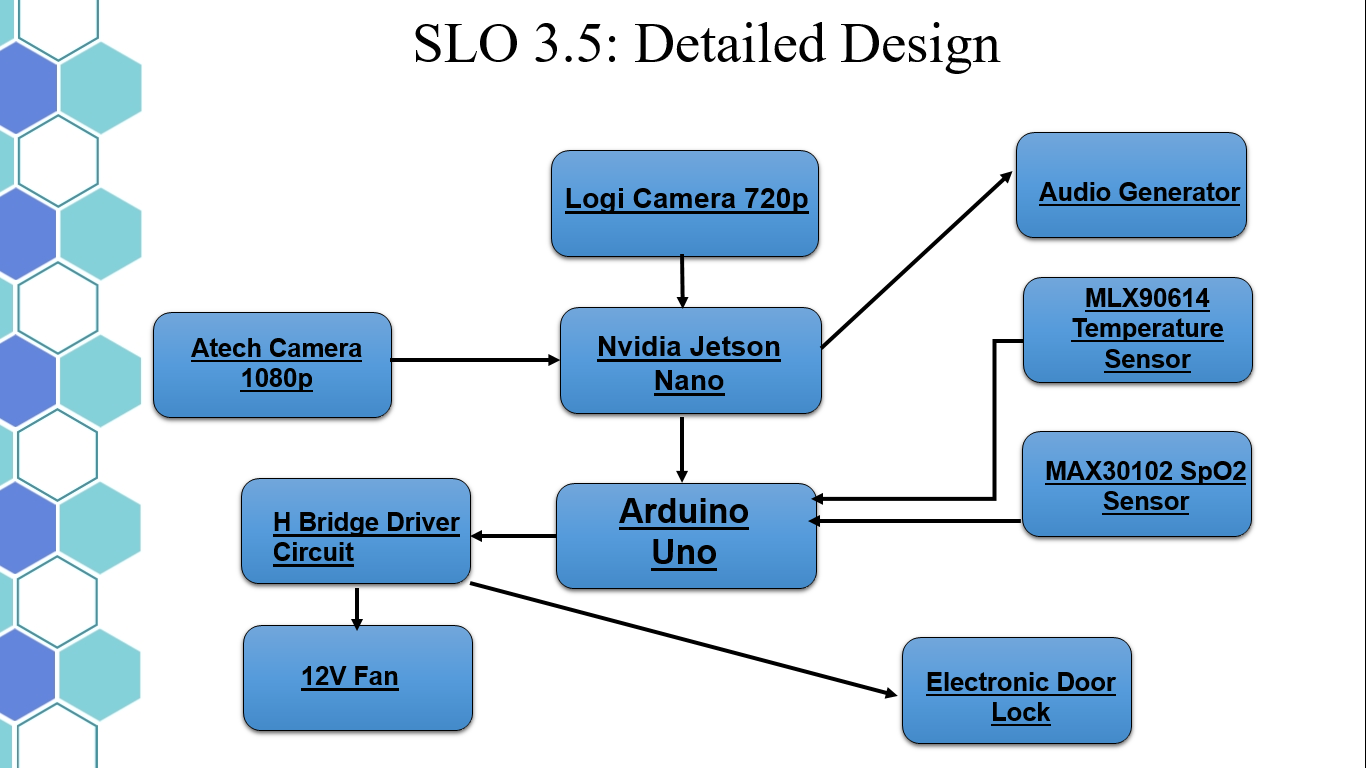


Figure 4-15: Hardware Block Diagram

Figure 4-6 represents a more specific detailed hardware block diagram of the system. For face mask detection, Pi Camera Module 2 would be used which is an 8MP camera while for social distancing detection, Pi Camera Module 1 is used. These cameras would send their video feed in real time to the microcontroller, Nvidia Jetson Nano which communicates with Arduino Uno serially. Arduino Uno is connected with three actuators; buzzer module, 128x32 OLED LCD display module, and electric door lock. Furthermore, the fever of the person is checked using MLX90614 sensor which is contactless IR temperature. For checking oxygen saturation in the blood and pulse rate, MAX30102 SpO2 sensor is used.

**Circuit Schematics**

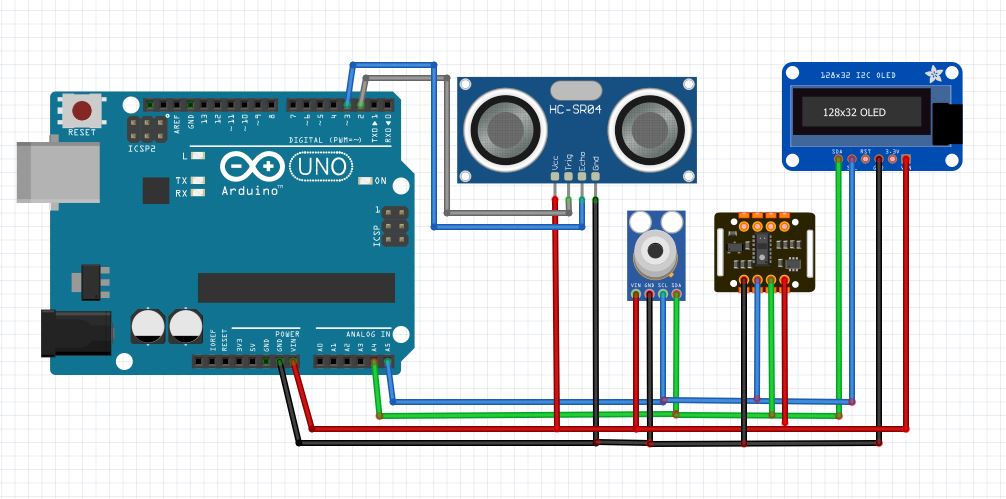
****

Figure 4-16: Schematics for Hardware connected with Arduino

The schematic diagram in Figure-4-7 depicts connection made to an 8-bit controller. Each module’s ground pin and Vcc/Vin pins are connected to the ground pin and 5V pin of microcontroller. Then A4 pin of 8-bit computer is connected to SDA port of temperature sensor, similarly the A5 pin is connected to SCL port of the temperature sensor. For distance measuring sensor, the 2nd (D2) and 3rd (D3) digital pin of the microcontroller are connected to the Trigger and Echo pins of the sensor respectively. Lastly, for the oxygen detection sensor, A5 pin is connected to SDA and A4 pin is connected to SCL port of the sensor.

**4.3.2 Software Design**

The design approach, software and libraries used in our project are:

**4.3.2.1 Design Approach:**

**Face Mask Detection:**

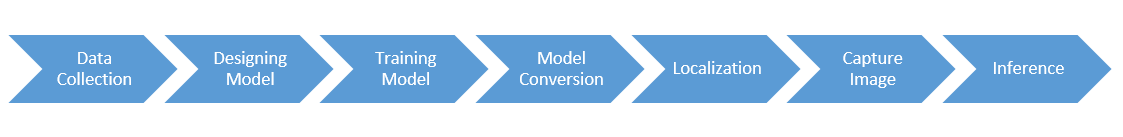
****

Figure 4-17: Flow Diagram

This module is divided into two main part object classification and object detection.

**Object Classification:**

In object classification, the class of the object is identify by passing the image through the machine learning model.

**Object Detection:**

In it the coordinates of the detected class is predicted by the machine learning model by using techniques such as bounding box regression.

**Face Mask Detection Design:**

There are many methods for object classification such as OpenCV Haar Cascade, SSD algorithm and many more but we selected Deep Learning Model Architecture known as Convolutional Neural Network (CNN) for mask detection. We used Linear Regression Approach to initialize and train the neuron values and weights. The design approach involve the following steps,

Figure 4-18: Software Design Steps

**Dataset collection:**

We used kaggle to obtain the dataset required for training the neural network model. We further uses the data augmentation and ImageDataGenerator in tensorflow to generalizing the data to decrease the chances of overtraining. The dataset is divided into training and validation set. We also uses different dataset for testing and evaluating the neural network model. The main dataset for training comprise of over 7000 images of both images with face mask and without face mask.

**Neural Network Model:**

The first step is to design the Neural Network Model Architecture, that involve the hidden layers in model, number of convolutions, max pooling, activation functions, optimizers number of neurons in each layers, and neuron connections types. These parameters directly effects the complexity and accuracy of the model. Since we used the linear regression model the relationship between the neurons are linear.

1. **Neurons:**

The neurons are the single unit or node in network model which perform the mathematical function on input and forward it to connected nodes or neurons. In linear regression, each neurons are consists of weight and bias, which are used in computing the output. The weight and bias values are calculated automatically by training the model.

1. **Neuron Layers:**

The neuron layers consist of input layer, hidden layers and output layer. In these layers neurons are stack vertically. The input layer take the input values from source and pass it to hidden layers. Hidden layer exists between the input and output layers and compute the weight and bias values to output. The fully connected neural network are generally used in image processing model.

1. **Convolutional Layers:**

The convolution layers are consists of convolutional functions or filters which are applied on inputs to extract the specific feature from it. CNN uses multiple convolutional layers with max pooling to extract every important features from image.

1. **Max Pooling:**

The max pool is the process in which image is resize and compress while retaining important features, this will not only enhance the image but convolutional layer can extract more specific features from image.

1. **Loss Function and Optimizers:**

The loss function define the learning rate of model, if the loss function selected is not optimize it will take forever to train the model. Similarly, optimizers are used to optimize the learning rate and minimize loss. The examples if optimizers are Root Mean Square or RMSprop which adjust the step size in learning rate curve to avoid slow or overshoot in learning rate. Other loss function examples are binary cross entropy and categorical entropy.

1. **Activation Function:**

Activation functions are used for changing the output of neurons or mapped the output to different values. Rectified Linear Unit (ReLU) is commonly used activation function which takes the absolute of output of neuron and forward the value to other connected neurons.

**Neural Network Model Diagram:**

The following diagram shows the simplified architecture of our neural network model.

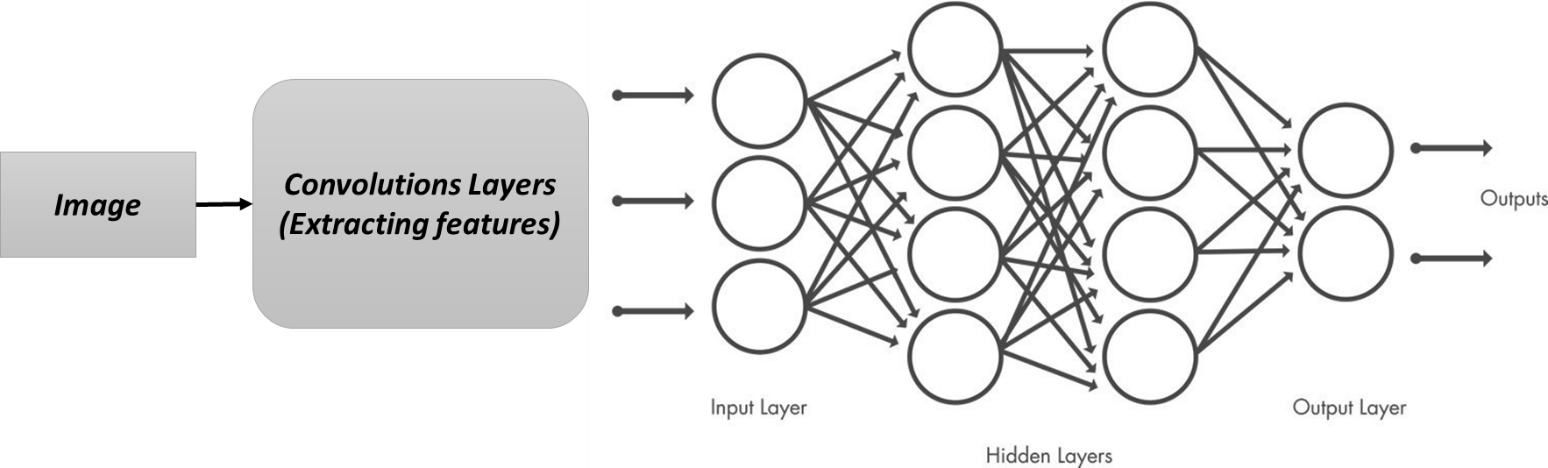
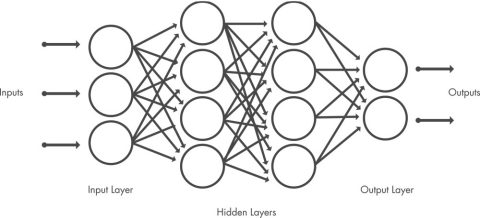
**

Figure 4-19: Neural Network Model

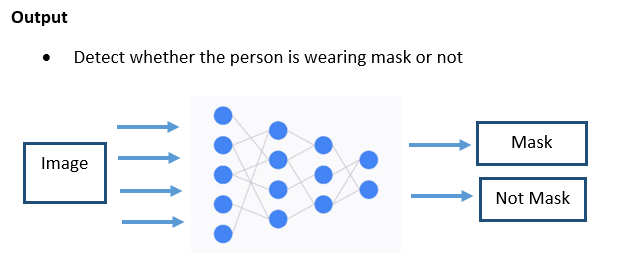
****

Figure 4-20: CNN Classification

**Model Training:**

After designing the neural network the next step is to train the neural network. Training the model required several hundred samples of images to set the values for weights and biases of each neurons. The training process is repeat again and again to improve the accuracy of model which is specified by the number of epochs. For better evaluation of model a separate dataset called validation data is used which evaluate the model after each epoch and also stop the overfitting process.

The following is the graphs of training process, the first one is showing the accuracy of model from start to end,

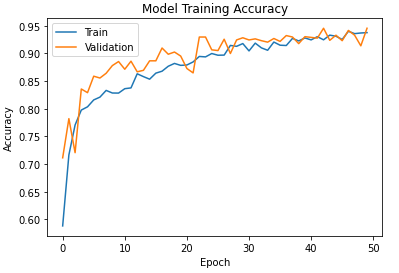


Figure 4-21: Model Training Accuracy

The loss is the term in neural network used to indicate the error in predictions. Similarly the next graph is showing the loss during each epoch of training process,

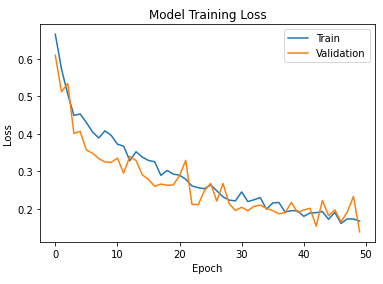


Figure 4-22: Model Loss

**Localization:**

The above algorithm will only classify the image, now to detect the person we used the mediapipe library which has very efficient ssd-mobile net face detector. Mediapipe is used to detect person face and to input the region of interest to the Neural Network Model. The library has built in face detector module which has very high accuracy but more importantly it is very fast as compare to the Haar Cascade in OpenCV and many other algorithm which we tested on jetson nano. FaceDetector uses Single Shot detector (SSD) combine with BlazeFace for detecting face. The built-in face detector is compatible with jetson nano and also frame rates are very stable as compare to other similar algorithms. The mediapipe will first detect the person face and then crop and resize it to the required image size to pass it to the tflite mask detection model, after it the model will predict that the whether the person is wearing the mask or not. The output function is sigmoid, so only single prediction will be used to identify. There are also alternative methods such as bounding box regression but after comparing it with mediapipe we decided to not use them.

**Model Conversion:**

Model is converted to TFLite inorder to upload the model onto the Jetson Nano developer kit. After it the model is converted to tensorflow lite so it can be work more efficiently on embedded systems.

**Deploying Model:**

After designing and training the model, the next step is to deploy the model for specific application for predictions. The model will first take the input, then process the input to compute the prediction score. The following figure show the CNN processing the image to predict the input image,

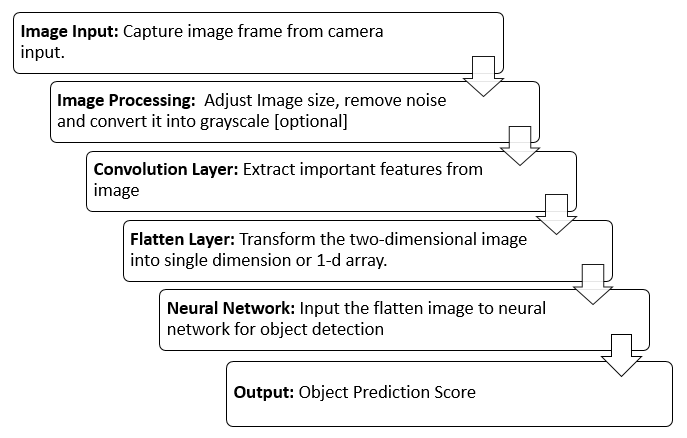


Figure 4-23: Process of Algorithm

This is integrated with the mediapipe library and then deployed on the jetson nano.



Figure 4-24: Deploying Model

**4.3.2.2 Software Used:**

The following are the software which we are using in our project:

1. **Google Colab:**

Google Colaboratory (or Colab) is the jupyter notebook type environment on browser, which is widely used for machine learning, researching, computer vision and python projects. The colab assign virtual machine with CPU and storage to execute the python code and also provide the access to GPU and TPU for very fast processing and execution of time consuming tasks. Colab is ideal for machine learning with large dataset because it can process and train the dataset within no time. Google Colab is well integrated with tensorflow so, we used the colaboratory for designing, training and testing the CNN model.

1. **VS Code:**

Visual Studio Code is code editor with integrator development environment for various languages support and also provide debugging, and command line interface. We used VS code to write and edit the python code. Same editor is used for analyzing the real-time and measuring the efficiency of code.

1. **Python:**

The python is the high level general purpose programming language widely used for artificial intelligence, data analysis and machine learning. We used the python to code and design deep learning model on VS code and google colab.

1. **Arduino IDE:**

Arduino IDE is the default code editor and development environment for Arduino or similar boards. The default language used in IDE are C and C++. The Arduino IDE used in our project for programming Arduino to interface with Nvidia Jetson Nano and other sensors, uploading code to Arduino.

1. **Nvidia Jetson Nano OS:**

Nvidia Jetson Nano uses its own specialized Ubuntu OS to maximize the utilization of processor and GPU. The OS image we are using is JetPack 4.6 which is the specialized version of Ubuntu 18 for Jetson Nano. It comes with pre install AI packages such as CUDA and OpenCV. Also, the OS is optimized for best performance with the developer kit. It does not include the TensorFlow and Mediapipe Libraries, which has to be manually installed including each dependencies.

1. **MS Project**

MS Project is used to construct the work breakdown which is then used to make the Gantt chart using the software.

**4.3.2.3 Libraries Used:**

1. **OpenCV:**

OpenCV is the library for computer vision, image processing and machine learning written in C/C++ but also supported in Python. We used the OpenCV for image processing and to capture video from camera. Some functions are also used in image detection.

1. **Tensorflow:**

Tensorflow is the open source artificial intelligence and machine learning library which is extensively used to design and deploy the machine learning models. We used the Tensorflow Keras API to design and train our neural network model. It also provide techniques to compress the model to deploy it on low processing systems such as Raspberry Pi and Nvidia Jetson Nano.

1. **Tensorflow Lite:**

Tensorflow is the open source artificial intelligence and machine learning library which is extensively used to design and deploy the machine learning models. We used the Tensorflow Keras API to design and train our neural network model. It also provide techniques to compress the model to deploy it on low processing systems such as Raspberry Pi and Nvidia Jetson Nano.

1. **Keras:**

Keras is the artificial neural network library and also integrated with Tensorflow. It is written in python and provide tools for building neural network, including model configuration functions, activation functions and optimizers. We used it with tensorflow to design and configure our neural network model.

1. **MediaPipe:**

MediaPipe is the open source computer vision library which has built-in highly optimize modules for face detection and other applications. We currently using MediaPipe to remove the irrelevant objects from image, so that the neural network which we design work more efficiently.

1. **Sensor Libraries:**

The temperature sensor and SPO2 sensors required setup and configuration for calculation and communicating with Arduino. We used the libraries, MAX301X and MLX90614 available with both of these sensors for initialization, calculation and port configuration. The wire library is used for SPI communication protocol.

**Chapter 5: Investigation and Testing**

**Object Classification and Training:**

The tflite code is tested on jetson nano with stable frame rates and high accuracy. Accuracy achieved with the tensorflow lite model is 95% and it only takes 9.9MB of space.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Loss | Speed | Size | Utilization | RAM |
| Default Stats | - | - | 16 FPS | - | <10% | 909 MB |
| Tensorflow | 0.95 | 0.13 | 4 FPS | 30 MB | >45% | 3182 MB |
| TFLite | 0.92 | 0.15 | 10 FPS | 9.9 MB | >40% | 1268 MB |

Table 5-

**Training Model:**

The following figure shows the training of CNN model,

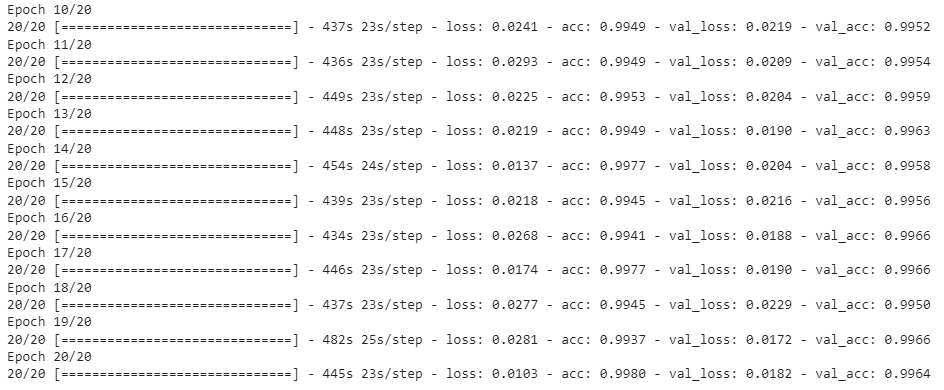


Figure 5-1: Training Steps

**Circuit:**

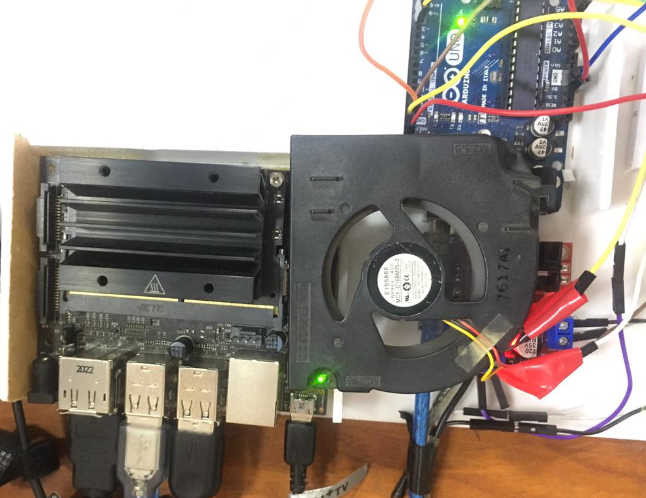


Figure 5-2: Circuit Setup

**Prototype:**

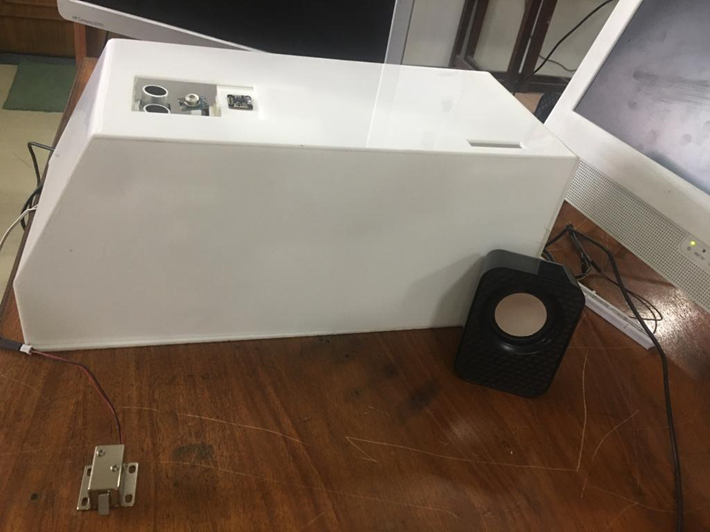
****

Figure 5-3: Prototype a

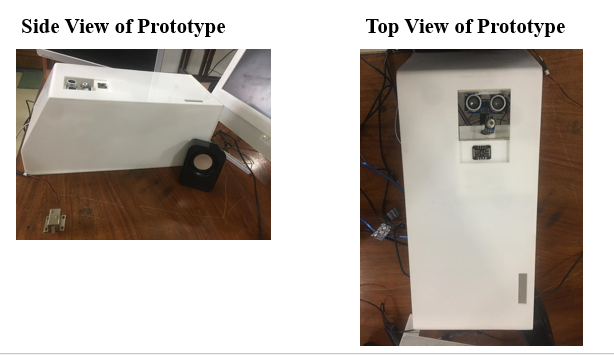


Figure 5-4: Prototype b

**Results:**

**Face Mask Detection:**

1. Not Wearing Face Mask



Figure 5-5: Face Mask Violation



Figure 5-6: Warning Message

1. With Face Mask



Figure 5-7: Mask Detection

**Social Distancing:**

The green rectangle shows that social distancing is maintain and if violates these rectangle will turn red and error message will be display on terminal.



Figure 5-8: Social Distancing (Green)

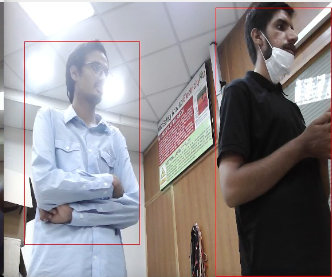


Figure 5-9: Social Distancing Violation (Red)



Figure 5-10: Warning

**Temperature and SpO2:**

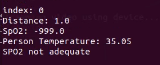


Figure 5-11: Temperature and SPo2 Display

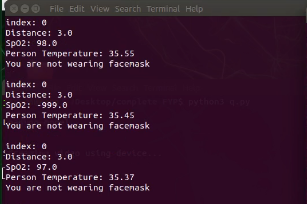


Figure 5-12: Temperature and SPo2 History

**Final Product:**

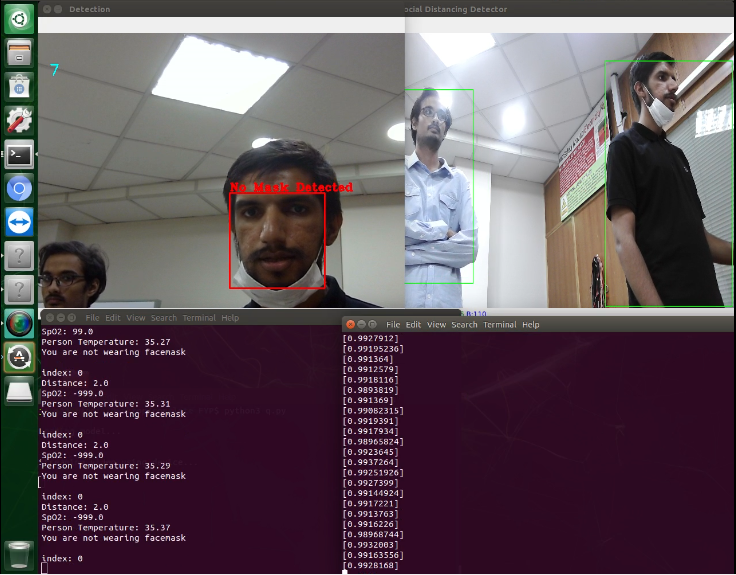


Figure 5-13: Complete Display (Violation)

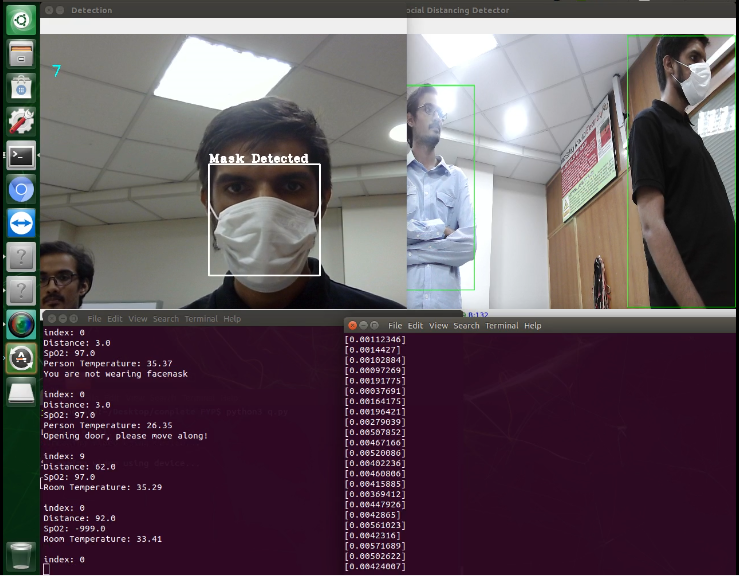


Figure 5-14: Complete Display with no violations

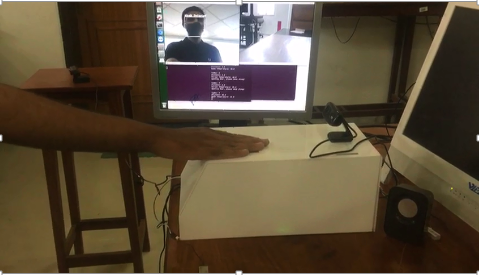


Figure 5-15: Side View of Product

**Chapter 6: User Guide**

This guide helps you get started with using the product, guiding you through the installation process, what features the product has, how it functions and how to use it properly.

**Package Content**

Your product should have all the components shown in table 6.1. Ensure that you have received all the components, else contact the FYP team if something is missing.

|  |  |
| --- | --- |
| Components | Details |
|  | White Box |
| Digital Infrared Temperature Sensor – MLX90614 / MLX-90614 / MLX 90614: Buy  Online at Best Prices in Pakistan | Daraz.pk | Contactless Temperature Sensor (MLX90614)  Located on top of the white box |
| Heart rate sensor module MAX30102 - pulse sensor | SpO2 Sensor (MAX30102)  Located on top of the white box |
| HC SR04 HC-SR04 Arduino Ultrasonic Sensor | Sonar Sensor  Located on top of the white box |
| Logitech C505 720p HD WebCam with 720p and Long-Range Mic Pakistan | Logi 720p USB Camera |
| FHD 1080P MF Webcam(PK-935HL) | | Atech 1080p Manual Focus USB Camera |
| C:\Users\HP\Desktop\FYP Final Documents\extra\WhatsApp Image 2022-06-20 at 8.31.31 PM.jpegC:\Users\HP\Desktop\FYP Final Documents\extra\WhatsApp Image 2022-06-20 at 8.31.31 PM.jpeg | Dual Mini-Speaker |
| 5V Electric Door Lock Solenoid Lock in Pakistan - Electronics Pro | 12V DC Electric Door Lock |
| Computer monitor - Wikipedia | Monitor |
| 5V 3A Power ADAPTER CHARGER for RASPBERRY PI 3 Micro USB buy in Pakistan | 5V 3A Power Supply Adaptor |
| Hdmi To Vga Converter With Sound- White: Buy Online at Best Prices in  Pakistan | Daraz.pk | VGA to HDMI Convertor |

Table 6.1

**Product Features**

This product has the following features included:

1. Single person face mask detection using Logi 720p USB Camera with 25 FPS and displays the result on the monitor attached.
2. Single person contactless body temperature sensing using MLX90614 Sensor and displays the result on the monitor attached.
3. Single person oxygen saturation in the bloodstream (SpO2) sensing using MAX30102 sensor which requires index finger to be pressed firmly on it and displays the result on the monitor attached.
4. Social distancing detection among the first 3 people in the single file queue using Atech 1080p Manual Focus USB Camera with 20 FPS and displays result on the monitor.
5. Audio assistance that guides the user throughout the entire process using dual mini-speakers in English and generates warning when required.
6. Automatic door locking and unlocking mechanism using a 5V DC electronic door lock that unlocks the door for 5 seconds.
7. Compact and portable product body, colored opaque white hiding all the inner connections.
8. A 12V DC Fan that keeps the internal system of the product cool.
9. A Monitor for display of all results.

**Top View of Product**

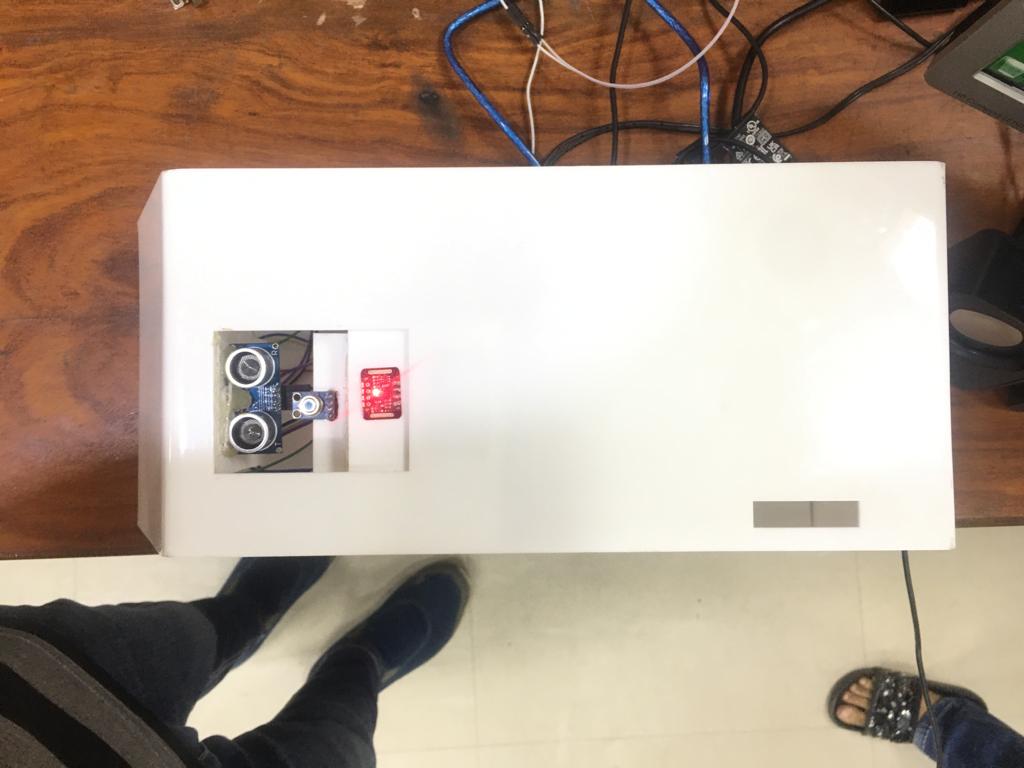


Fig 6.1

**Complete Product**

****

Fig 6.2

**Setting Up the Product**

**Note:** In case of any difficulty contact the FYP team or if you want us to install it for you.

**Note:** Make sure that there is some backup power in case of a power shortage. Connect first, then turn on the power.

1. **Setting the Logi Camera**

Using a screw driver, attach the logi camera (see table 6.1) at the face level of a person where the queue is going to start.

1. **Setting the Atech Camera**

Using a screw driver, attach Atech Camera (see table 6.1) perpendicular to the start of the queue, where three persons are visible in the camera frame.

1. **Setting Dual mini-speakers**

Insert the pin of the speakers inside the port of the VGA to HDMI convertor (see table 6.1). Place these speakers near the white body of the product.

1. **Setting the Monitor**

Place the monitor near the white body of the product, facing towards the queue, so that the first person could properly see what the results are.

1. **Setting the Product White Body**

The white body of the product is to be set at a minimum height of 4 feet from the ground using the stand. This should be placed at the side of the queue and at the starting of the queue.

In order to get everything in working condition, follow the given steps.

1. Connect Atech Camera (see table 6.1) with USB port of Jetson Nano.
2. Connect Logi Camera (see table 6.1) with USB port of Jetson Nano.
3. Connect the 5V 3A power adaptor with the USB micro port of Jetson Nano. Plug this adaptor with any outlet.
4. Connect the blue wire (see table 6.1) with USB port of Jetson Nano.
5. Connect HDMI cable with HDMI port of Jetson Nano.
6. Plug the 12V power adaptor (see table 6.1) with any outlet.

**Operating the Product**

1. Turn on the power supply for the 5V and 12V power adaptor (see table 6.1) and wait for the system to boot up. The desktop should be displayed on the monitor.
2. After booting up the system, open complete FYP folder on the desktop. There should be 3 .py files and 6 other documents.
3. Right click on the folder. A drop-down menu should be opened in which click of Open in Terminal.
4. Repeat step 3 times.
5. Three black windows should open (aka terminals). Write python3 a.py in one terminal, python3 q.py in the second terminal, and python3 ard.py in the third terminal. This would activate the cameras and start the process.

**Note: No file should be opened or modified, else the product might not work as the source files are tempered with.**

1. When the product is to be turned off, close all the terminals and folders, and click on shut down button from the top right corner. After the system is shut down, turn off the power supplies and unplug the adapters.

**Note: Beyond this point, no human interaction is required since this product is fully automated.**

**About the Product Functionalities**

Once the system is boot up and files are executed in the terminals, the product starts checking the SOPs of COVID-19 onto the first person in the queue.

The speaker would start speaking ‘**Please place your hand on sensor**’ until a hand is placed on top of the sensors.

The index finger should be firmed pressed onto the SpO2 sensor (see table 6.1) and the rest of the hand should hover over the contactless temperature sensor (see table 6.1).

The face of the first person should look at the Logi Camera installed at the front of the queue.

The Atech camera installed at the side of the queue would be automatically checking the distance between the first three individuals in the queue.

If all SOPs (face mask, body temperature, SpO2, and social distancing) are being followed, the system would speak “**Opening door, please move along**” and then “**Your temperature is 34˚C**”. The door would be unlocked for 5 seconds after which it would automatically lock itself.

If face mask violation is detected, the system would say “**You are not wearing facemask**” and would not open the door and stop saying that until that person wears a facemask or gets out of the queue.

If temperature is abnormal, the system would say “**You have high temperature**” and if SpO2 reading is low, the system would say “**SpO2 is inadequate**”.

If social distancing is violated, then the system would say “**Social distancing violation**”.

**Maintenance Guidelines**

For best practices, follow the instructions in the list below while unpacking, cleaning, or handling your product:

* Keep the product away from direct sunlight to avoid any heating problems or short circuit.
* Do not temper with the internal wiring, or the cables as it might cause the system to go faulty.
* Keep the system away from water to avoid short circuit. Inorder to clean the product, make sure to use a lightly damped cloth with water.
* Do not run the system for more than 10 hours to prevent any permanent damages to the system.
* When turning off the system, do not directly turn off from the power supply as this causes damages to the memory card of the system.

**Troubleshooting**

* **System not booting**

Turn off and then turn on the system from the outlet a few times. This indicates that the memory card might be corrupted due to power shortage and can be fixed by contacting the FYP team.

* **System is booted but no display on Monitor**

Check the VGA to HDMI cable and make sure the VGA plug is properly connected from both sides, the monitor and the convertor cable.

Check if the power plug of the monitor is properly connected.

Verify if the VGA cable is not faulty. Test the system with another VGA cable.

* **Camera results swapped**

If Logi camera is being used for social distancing and Atech camera is being used for face mask detection, then it is possible that the connections are not done properly. Connect Atech Camera with USB port 2 and connect Logi Camera with USB port 1.

* **SpO2 sensing is not working**

It is preferred that the user should place their index finger for more accurate reading. The index finger should be firmly pressed and not be moving. It takes 5 seconds for the SpO2 value to be shown on the monitor so keep the index finger pressed till then.

**Note: In case of any other issues with the system, contact the FYP team.**

**Chapter 7: Deliverables and Cost**

**7.1 Deliverables**

After frequent meetings with the client, the advisor and co-advisor, the following deliverables were concluded to be included in the final year product, titled An Automated COVID-19 SOPs Monitoring And Management System.

1. Face mask detection using a camera
2. Social distancing detection using another camera placed perpendicular to the queue
3. Contactless body temperature sensing
4. SpO2 sensing of a single person
5. Audio Assistance
6. Warning Generation
7. Automatic door lock mechanism

By the end of the FYP-II, the client is expecting the completed prototype with all the modules integrated as discussed during the client meetings. Regarding the software portion the client is expecting the following:

1. CNN with linear regression for face mask detection
2. MobileNetv2 for social distancing detection
3. A final integrated code of interfaces of sonar sensor, MLX90614, and MAX30102 with Arduino Uno.
4. Synchronization and Audio code between Nvidia Jetson Nano and Arduino Uno using serial communication and file reading for automatic door lock mechanism, audio assistance, and warning generation.

The hardware the client is expecting by the end of the project is a ready to use white box, on top of which there would be MLX90614, MAX30102, and sonar sensors. Two cameras and a monitor for display will also be provided for complete functionality and display. Inside this white box, all the internal wirings are concealed that are the three sensors connected with Arduino Uno, Arduino Uno connected with the H bridge driver circuit and Nvidia Jetson Nano, H bridge driver circuit connected with the fan and door lock, and Nvidia Jetson Nano with the two cameras, the monitor, and the speakers.

Regarding the documentation of the project, the client is expecting the following:

1. Idea Proposal Document
2. FYP I Report
3. FYP II Report
4. User Manual
5. Code files

**7.2 Project Plan**

**Work Breakdown Structure**

Before progressing with the hardware and software research, the main deliverables, additional features, and their required activities were first identified using a work breakdown structure. The work breakdown structure of our project is:

1. **An Automated COVID-19 SOPs Monitoring and Management System**
   1. **Software Module**
      1. Work Package 1 (Face Mask Detection)
         1. Researching and Comparison of Object Detection Algorithm
         2. Efficiency of Algorithm with hardware
         3. Selection of Object Detection Algorithm
         4. Implementation of Algorithm
         5. Dataset Collection
         6. Data Augmentation
         7. Dataset training
         8. Testing of Algorithm
         9. Designing of final module
      2. Work Package 2 (Social Distancing)
         1. Research on the different methods of social distancing detection
         2. Selection of Object detection Algorithm that gives best mAP
         3. Distance Measurement Methods in frames
         4. Implementation of Algorithm
         5. Testing
      3. Work Package 3 (Sensors and Interfacing)
         1. Setting-up Code IDE
         2. Coding and Interfacing Sensors with Arduino Uno
         3. Testing of Sensors
         4. Installing front camera for face-mask detection
         5. Installing perpendicular camera for social distancing monitoring
         6. Designing of Door Lock Mechanism Software
         7. Implementing door lock mechanism
         8. Interfacing Arduino Uno with Nvidia Jetson Nano
         9. Setting Up Alerting system
         10. Interfacing voice module and display module
      4. Work Package 4 (Accessories)
         1. Addition of voice module for warnings
         2. Display on Screen
   2. **Hardware Selection** 
      1. 32-bit computer
      2. 8-bit computer
      3. Camera module
      4. Door Lock Interface
      5. Temperature Sensor
      6. SpO2 Sensor
      7. Distance Range Sensor
      8. Buying of finalize equipment and devices
   3. **Integration of Modules**
      1. Integrating Work Package 1 and 2
      2. Converting Package to Real-time
      3. Applying Quantization, Pruning and distillation techniques
      4. Testing algorithm efficiency real-time
      5. Combining all Modules into single setup
      6. Assembling of final Product
      7. Final Testing
   4. **Documentations**
      1. Presentations
      2. Project Charter
      3. Proposal Template
      4. FYP-I Report
      5. FYP-II Report
      6. Poster
      7. Final Demonstration Video

**-**

Fig 7.1

**Gantt Chart for FYP-I & FYP-II**

Gantt Chart is an efficient and effective method of developing a project management plan, work breakdown structure, defining the sequence of activities, time management plan and human resource management plan. Fig – illustrates the gantt chart defined for FYP-I and FYP-II

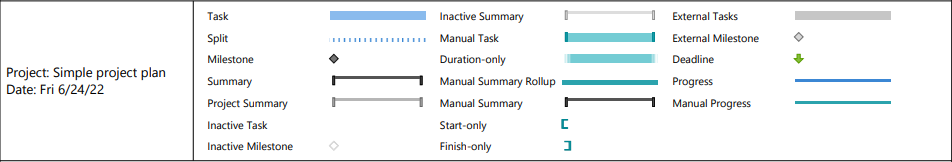
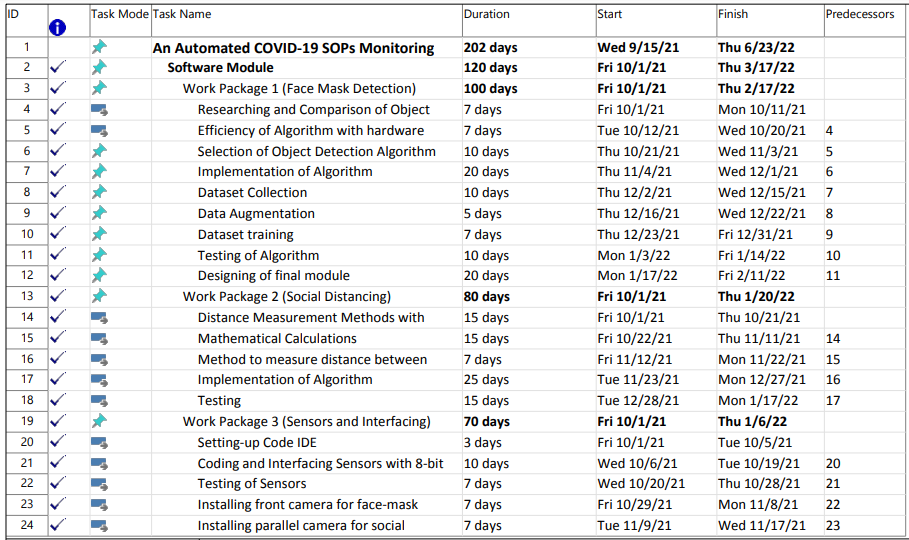
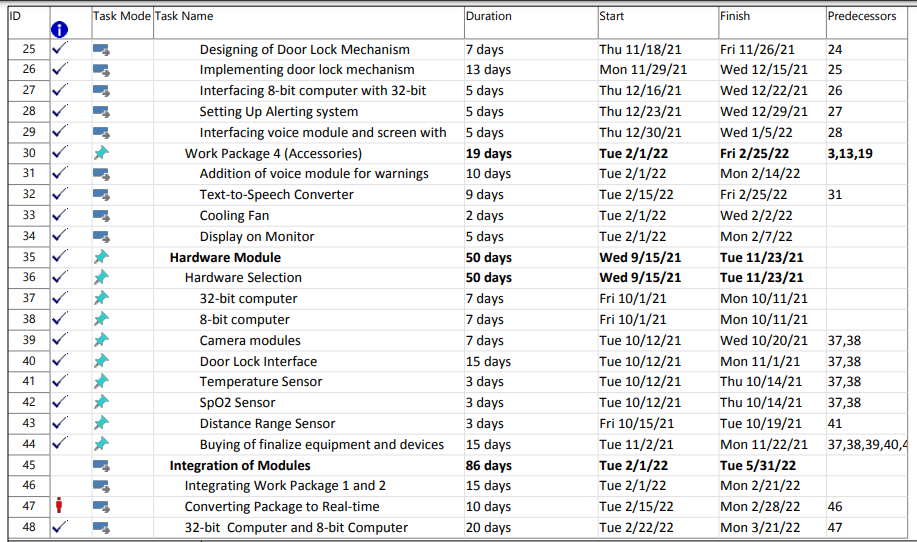
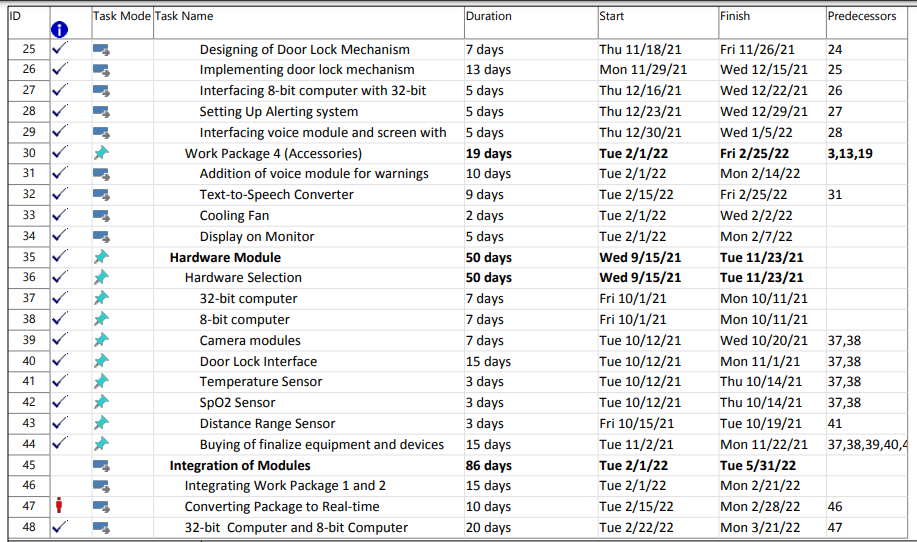


Fig 7.2







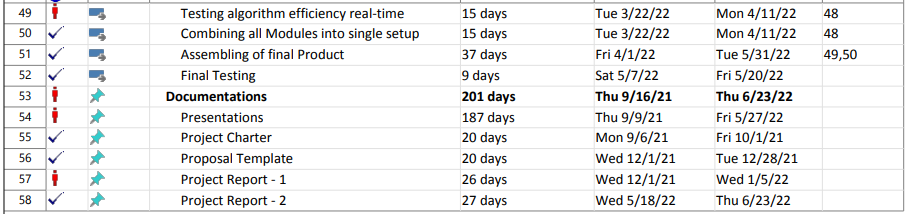
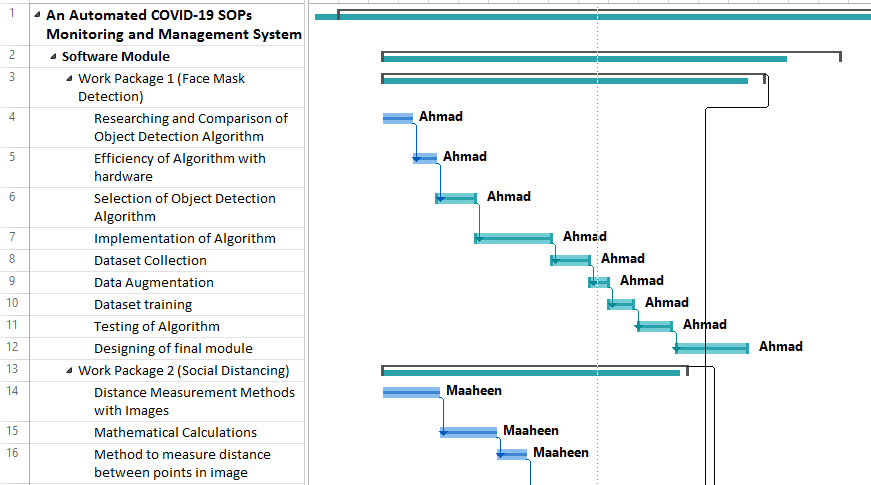
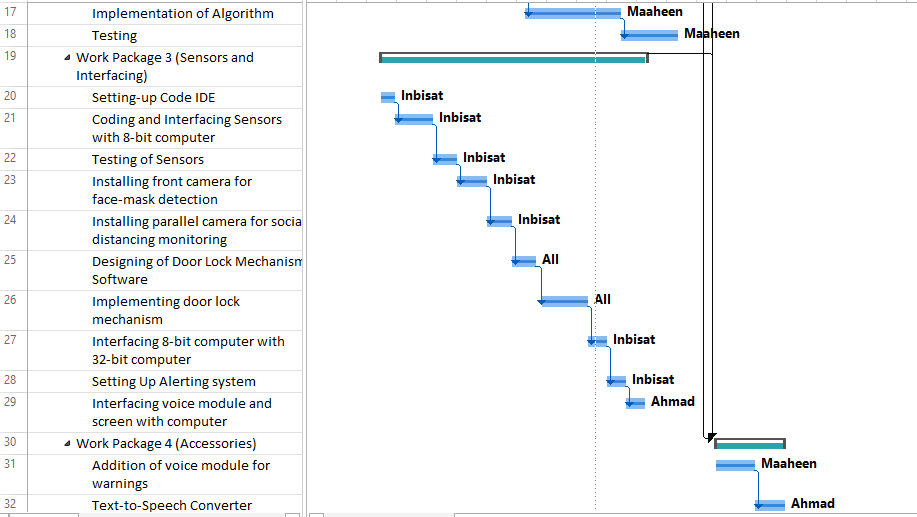
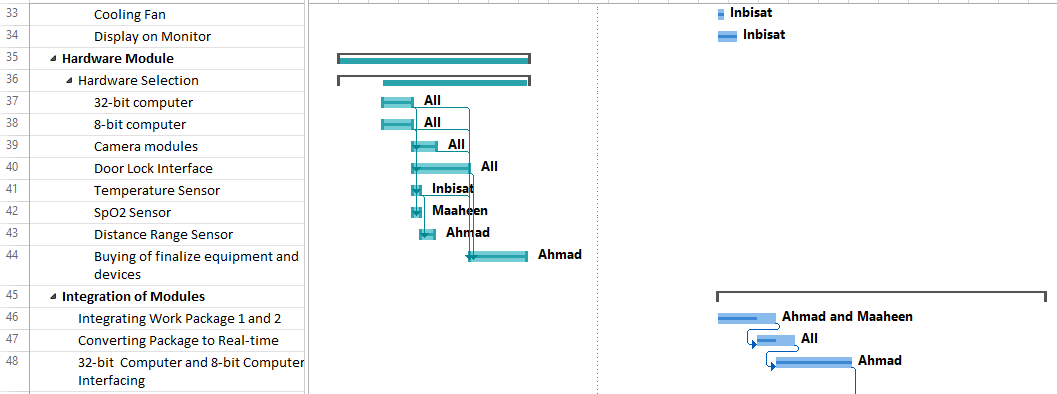


Table 7-1







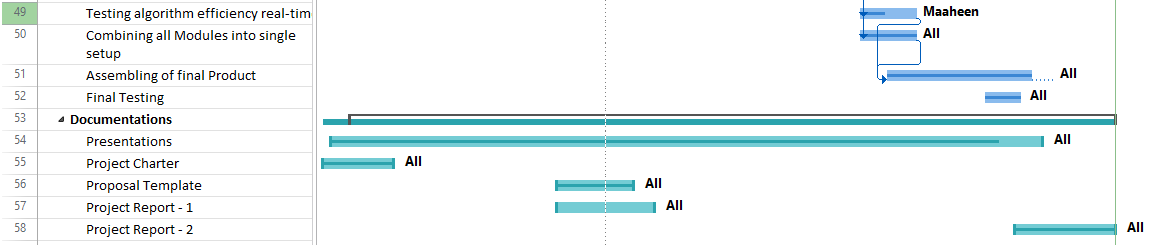


Fig 7.3

**7.3 Project Cost**

Table 7.1 depicts all the hardware components that were required to implement this project and their corresponding market price in 2022.

|  |  |  |
| --- | --- | --- |
| No. | Component | Cost (Rs.) |
| 1 | Nvidia Jetson Nano (4 GB) | 34,000 |
| 2 | Arduino Uno | 1250 |
| 3 | MLX90614 Temperature Sensor | 3000 |
| 4 | MAX30102 Oximeter | 350 |
| 5 | Logi 720p Camera | 3000 |
| 6 | Atech 1080p Manual Focus Camera | 4000 |
| 7 | HC-SR04 Ultrasonic Sensor | 150 |
| 8 | 12V DC Fan | 80 |
| 9 | H Bridge Driver Circuit | 300 |
| 10 | 12V Electronic Door Lock | 500 |
| 11 | Dual Mini-Speakers | 300 |
| 12 | Nvidia Jetson Nano Power Supply | 500 |
| 13 | 12V Power Supply Adapter | 500 |
| 14 | Micro SD card 64 GB | 1100 |
| 15 | Acrylic Sheet White Colored Body Laser Cut | 750 |
|  | **Total Cost = Rs. 48900** | |

Table 7.2

**Chapter 8: Conclusion**

All the required deliverables and one additional deliverable of this project was successfully achieved by the FYP team on time. The deep learning model used for face mask detection was 94% accurate, provides 20-25 FPS output speed and is highly reactive to its environment. The model for social distancing detection provides 20-22 FPS output speed and is able to monitor first three people in the queue with the threshold social distance being half a meter. Contactless body temperature sensing and SpO2 sensing has been achieved, with an accuracy of 98%. After integrating all these modules, an electric door lock is automatically opened for 5 seconds by providing a 5V signal. Audio assistance and warning generation using audio has also successfully been added into this project, making this project more user-friendly and easy to use.

Further work could be done on this project, among which could be adding a IoT module, which would store all the data of each employee who is entering and who is not entering in a database and is stored in a cloud storage to which access is given only to the administrator. Moreover, Arduino Uno could be completely replaced by Nvidia Jetson Nano if the sensors are interfaced with it and Jetson Nano having bigger RAM than 4GB.

**References**

[1] Susanto, Febri Alwan Putra, Riska Analia, Ika Karlina Laila Nur Suciningtyas, “The Face Mask Detection For Preventing the Spread of COVID-19 at Politeknik Negeri Batam”, [3rd International Conference on Applied Engineering (ICAE)](https://ieeexplore.ieee.org/xpl/conhome/9350316/proceeding), Batam, Indonesia, 2020

[2] Muhammad Haris Kaka Khel, Kushsairy Kadir, Waleed Albattah, Sheroz Khan, MNMM Noor, Haidawati Nasir, Shabana Habib, Muhammad Islam, Akbar Khan, “Real-Time Monitoring of COVID-19 SOP in Public Gathering Using Deep Learning Technique”, *Emerging Science Journal*, vol. 5, Special Issue "COVID-19: Emerging Research", 2021

[3] Megha Warungase1 , Ruchita Wagh2 , Komal Jundre3 , Prof. S.G. Chordiya, 2021. “Face Mask and Body Temperature Detection System to Prevent COVID for Work Environment”, IJARIIE [Online]. Vol-7 Issue-3.

Available URL: https://ijariie.com/AdminUploadPdf/Face\_Mask\_and\_Body\_Temperature\_Detection\_System\_to\_Prevent\_COVID\_for\_Work\_Environment\_ijariie14217.pdf

**Appendices**

1. **Google Colab Code: (Python3)**

This is the code for training neural network model for face mask detection.

""" Untitled0.ipynb

Original file is located at

   https://colab.research.google.com/drive/1IfyoWMgWyvHhwVrMI8AvfiB3R8zifBxi

# Import Data

"""

! pip install kaggle

! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/

! chmod 600 ~/.kaggle/kaggle.json

! kaggle datasets download -d omkargurav/face-mask-dataset

"""# Extract Data"""

import **os**

import **zipfile**

local\_zip = './face-mask-dataset.zip'

zip\_ref = **zipfile**.**ZipFile**(local\_zip, 'r')

zip\_ref.**extractall**('./')

zip\_ref.**close**()

"""# Import Tensorflow"""

import tensorflow as **tf**

"""# Model"""

model = **tf**.keras.models.Sequential([

**tf**.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(128, 128, 3)),

**tf**.keras.layers.MaxPooling2D(2, 2),

**tf**.keras.layers.Conv2D(64, (3,3), activation='relu'),

**tf**.keras.layers.MaxPooling2D(2,2),

**tf**.keras.layers.Conv2D(128, (3,3), activation='relu'),

**tf**.keras.layers.MaxPooling2D(2,2),

**tf**.keras.layers.Conv2D(256, (3,3), activation='relu'),

**tf**.keras.layers.MaxPooling2D(2,2),

**tf**.keras.layers.Flatten(),

**tf**.keras.layers.Dense(256, activation='relu'),

**tf**.keras.layers.Dropout(rate = 0.3),

**tf**.keras.layers.Dense(512, activation='relu'),

**tf**.keras.layers.Dropout(rate = 0.3),

**tf**.keras.layers.Dense(128, activation='relu'),

**tf**.keras.layers.Dropout(rate = 0.3),

**tf**.keras.layers.Dense(1, activation='sigmoid')

])

"""# Summary"""

**print**(model.summary())

**tf**.keras.utils.plot\_model(model, to\_file='model\_plot.png', show\_shapes=True, show\_layer\_names=True)

"""# Loss function and Optimizer"""

model.compile(loss='binary\_crossentropy',

              optimizer='adam',

              metrics=['accuracy', **tf**.keras.metrics.Recall(), **tf**.keras.metrics.Precision()])

"""# Image Processing"""

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(validation\_split=0.2,

      rescale=1./255,

      width\_shift\_range=0.2,

      height\_shift\_range=0.2,

      shear\_range=0.2,

      zoom\_range=0.2,

      horizontal\_flip=True,

      fill\_mode='nearest'

      )

train\_generator = train\_datagen.flow\_from\_directory(

        './data/',

        target\_size=(128, 128),

        batch\_size=128,

        class\_mode='binary',

        subset='training')

validation\_generator = train\_datagen.flow\_from\_directory(

        './data/',

        target\_size=(128, 128),

        class\_mode='binary',

        subset='validation')

"""# History"""

history = model.fit(

      train\_generator,

      steps\_per\_epoch = 20,

      epochs= 20,

      verbose=1,

      validation\_data=validation\_generator)

"""# Saving Model"""

directory = 'saved\_models/1'

**tf**.saved\_model.save(model, directory)

"""# Convert the Model"""

converter = **tf**.lite.TFLiteConverter.from\_saved\_model(directory)

tflite\_model = converter.convert()

"""# Encapsulate the TFLite Model"""

import **pathlib**

tflite\_model\_file = **pathlib**.**Path**('model.tflite')

tflite\_model\_file.**write\_bytes**(tflite\_model)

1. **Face Mask Detection Code (a.py): (Python3)**

import cv2

import tensorflow as **tf**

import mediapipe as **mp**

import numpy as **np**

import **time**

from tensorflow import keras

from tensorflow.keras.preprocessing import image

*# Webcam*

capture = **cv2**.VideoCapture(0)

*#capture.set(3, 640)        # 320x240*

*#capture.set(4, 360)*

*# FPS*

pTime = 0

cTime = 0

interpreter = **tf**.lite.Interpreter(model\_path = "./model.tflite")

interpreter.allocate\_tensors()

input\_index = interpreter.get\_input\_details()[0]["index"]

output\_index = interpreter.get\_output\_details()[0]["index"]

mp\_face\_detection = **mp**.solutions.face\_detection

mp\_drawing = **mp**.solutions.drawing\_utils

code = **open**("file.txt", "w")

with mp\_face\_detection.FaceDetection(min\_detection\_confidence=0.5) as face\_detection:

  while True:

    isTrue, frame = capture.read()      *# Capture frame by frame*

    results = face\_detection.process(frame)

    if results.detections:

      for detection in results.detections:

        coordinates = detection.location\_data.relative\_bounding\_box

        h, w, c = frame.shape

        xmin= **int**(coordinates.xmin \* w)

        ymin = **int**(coordinates.ymin \* h)

        height = **int**(coordinates.height \* h)

        width = **int**(coordinates.width \* w)

        xmax = xmin + width

        ymax = ymin + height

        img\_input = frame[ymin-60:ymax+45, xmin-40:xmax+40]

        try:

**cv2**.imwrite("./i.jpg", img\_input)

          path = "./i.jpg"

          img = image.load\_img(path, target\_size=(100, 100))

        except **Exception**:

          continue

        x = image.img\_to\_array(img)

        x = x / 255.0

        x = **np**.expand\_dims(x, axis=0)

        interpreter.set\_tensor(input\_index, x)

        interpreter.invoke()

        classes = interpreter.get\_tensor(output\_index)

**print**(classes[0])

        if classes[0] > 0.5:

**cv2**.putText(frame, "No Mask Detected", (xmin, ymin -4), **cv2**.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, (0, 0, 255), 2)

**cv2**.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 2)

          code.**seek**(0)

          code.**write**("2")

        else:

**cv2**.putText(frame, "Mask Detected", (xmin, ymin - 4), **cv2**.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, (255, 255, 255), 2)

**cv2**.rectangle(frame, (xmin, ymin), (xmax, ymax), (255, 255, 255), 2)

          code.**seek**(0)

          code.**write**("9")

    cTime = **time**.**time**()

    fps = 1/(cTime - pTime)

    pTime = cTime

**cv2**.putText(frame, **str**(**int**(fps)), (18, 75), **cv2**.FONT\_HERSHEY\_PLAIN, 2, (255, 255, 0), 2)

*#frame = cv2.resize(frame, (1080, 720))*

**cv2**.imshow('Detection', frame)

    if **cv2**.waitKey(5) & 0xFF == 27:

      break

capture.release()

**cv2**.destroyAllWindows()

code.**close**()

1. **Social Distancing Code: (Python3)**

import numpy as **np**

import **sys**

import cv2

from **math** import **pow**, **sqrt**

labels = [line.**strip**() for line in **open**('class\_labels.txt')]

*# Load model*

**print**("\nLoading model...\n")

network = **cv2**.dnn.readNetFromCaffe('SSD\_MobileNet\_prototxt.txt', 'SSD\_MobileNet.caffemodel')

**print**("\nStreaming video using device...\n")

code = **open**("file2.txt", "w")

*# Capture video from file or through device*

cap = **cv2**.VideoCapture(1)

cap.set(**cv2**.CAP\_PROP\_FRAME\_WIDTH, 1920)

cap.set(**cv2**.CAP\_PROP\_FRAME\_HEIGHT, 1080)

while cap.isOpened():

*# Capture one frame after another*

    \_, frame = cap.read()

    (h, w) = frame.shape[:2]

*# Resize the frame to suite the model requirements. Resize the frame to 300X300 pixels*

    blob = **cv2**.dnn.blobFromImage(**cv2**.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)

    network.setInput(blob)

    detections = network.forward()

    pos\_dict = **dict**()

    coordinates = **dict**()

    for i in **range**(detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > 0.5:

            class\_id = **int**(detections[0, 0, i, 1])

            box = detections[0, 0, i, 3:7] \* **np**.array([w, h, w, h])

            (startX, startY, endX, endY) = box.astype('int')

*# Filtering only persons detected in the frame. Class Id of 'person' is 15*

            if class\_id == 15.00:

*# Draw bounding box for the object*

**cv2**.rectangle(frame, (startX, startY), (endX, endY), (0,0,255), 2)

                label = "{}: {:.2f}%".**format**(labels[class\_id], confidence \* 100)

                coordinates[i] = (startX, startY, endX, endY)

*# Mid point of bounding box*

                x\_mid = **round**((startX+endX)/2,4)

                y\_mid = **round**((startY+endY)/2,4)

                pos\_dict[i] = (x\_mid,y\_mid)

*# Distance between every object detected in a frame*

    close\_objects = **set**()

    for i in pos\_dict.**keys**():

        for j in pos\_dict.**keys**():

            if i < j:

                dist = **sqrt**(**pow**(pos\_dict[i][0]-pos\_dict[j][0],2) + **pow**(pos\_dict[i][1]-pos\_dict[j][1],2) )

                if dist < 1200:

                    close\_objects.**add**(i)

                    close\_objects.**add**(j)

    for i in pos\_dict.**keys**():

        (startX, startY, endX, endY) = coordinates[i]

        if i in close\_objects:

**cv2**.rectangle(frame, (startX, startY), (endX, endY), (0,0,255), 2)

            code.**seek**(0)

            code.**write**("3")

        else:

**cv2**.rectangle(frame, (startX, startY), (endX, endY), (0,255,0), 2)

            code.**seek**(0)

            code.**write**("5")

*# Show frame*

    frame2 = **cv2**.resize(frame,(640, 480))

**cv2**.imshow('Social Distancing Detector', frame2)

    key = **cv2**.waitKey(1) & 0xFF

    if key == **ord**("q"):

        break

cap.release()

**cv2**.destroyAllWindows()

1. **Nvidia Jetson Code for Communication with Arduino (ard.py): (Python3)**

*# sudo apt-get install python3-serial*

*# sudo chmod 666 /dev/ttyAM0*

import serial

import **time**

import pyttsx3

arduino = **serial**.Serial("/dev/ttyACM0", baudrate=9600, timeout=.1)

engine = **pyttsx3**.init()

file = **open**("file.txt", "r")

file2 = **open**("file2.txt", "r")

num = '0'

**time**.**sleep**(3)

while True:

**print**("\nindex:", num)

**time**.**sleep**(0.3)

    arduino.write(**bytes**(num, 'utf-8'))

    if (num == '9'):

**time**.**sleep**(5)

        num='0'

    else:

**time**.**sleep**(0.5)

    data\_s = arduino.readline().decode('utf-8')

    data\_s = data\_s[:-2]

    data\_t = arduino.readline().decode('utf-8')

    data\_t = data\_t[:-2]

    data\_q = arduino.readline().decode('utf-8')

    data\_q = data\_q[:-2]

    if (data\_t != '') and (data\_s != '') and (data\_q != ''):

        data\_si = **float**(data\_s)

        data\_ti = **float**(data\_t)

        data\_qi = **float**(data\_q)

**print**("Distance:", data\_si)

**print**("SpO2:", data\_qi)

        if (data\_si > 7.0):

**print**("Room Temperature:", data\_ti)

            engine.say("Please place your hand on sensor.")

            engine.runAndWait()

            num = '0'

        else:

**print**("Person Temperature:", data\_ti)

            file.**seek**(0)

            seek = file.**read**(1)

            file2.**seek**(0)

            seek2 = file2.**read**(1)

            if (seek2 == '3'):

**print**("Social Distance Violation")

                engine.say("Social Distance Violation")

                engine.runAndWait()

            elif (seek == '9' and data\_ti < 38.0):

                if (data\_qi <102 and data\_qi> 79):

**print**("Opening door, please move along!")

                    engine.say("Yor temperature is" + **str**(data\_ti))

                    engine.runAndWait()

                    engine.say("Opening door, please move along!")

                    engine.runAndWait()

                    num = '9'

                else:

**print**("SPO2 not adequate")

                    engine.say("S P O 2 not adequate")

                    engine.runAndWait()

            else:

**print** ("You are not wearing facemask")

                engine.say("You are not wearing facemask")

                engine.runAndWait()

*#file.close()*

1. **Arduino Code: (C++)**

#include <Wire.h>

#include "MAX30105.h"

#include "spo2\_algorithm.h"

#include <Adafruit\_MLX90614.h>

Adafruit\_MLX90614 mlx = **Adafruit\_MLX90614**();

MAX30105 particleSensor;

#define **echoPin** 2 *// attach pin D2 Arduino to pin Echo of HC-SR04*

#define **trigPin** 3 *//attach pin D3 Arduino to pin Trig of HC-SR04*

*// defines variables*

long duration; *// variable for the duration of sound wave travel*

int distance; *// variable for the distance measurement*

int comm;

int32\_t bufferLength; *//data length*

int32\_t spo2; *//SPO2 value*

int8\_t validSPO2; *//indicator to show if the SPO2 calculation is valid*

int32\_t heartRate; *//heart rate value*

int8\_t validHeartRate; *//indicator to show if the heart rate calculation is valid*

void **setup**()

{

  Serial.**begin**(9600);

**pinMode**(13, OUTPUT);

**pinMode**(trigPin, OUTPUT); *// Sets the trigPin as an OUTPUT*

**pinMode**(echoPin, INPUT); *// Sets the echoPin as an INPUT*

  Serial.**setTimeout**(1);

  if (!particleSensor.**begin**())

  {

    while (1);

  }

  if (!mlx.**begin**()) {

     while (1);

  }

  particleSensor.**setup**(60, 4, 2, 800, 411, 4096);

}

void **loop**()

{

  uint16\_t irBuffer[50]; *//infrared LED sensor data*

  uint16\_t redBuffer[50]; *//red LED sensor data*

  bufferLength = 50; *//buffer length of 100 stores 4 seconds of samples running at 25sps*

  while(1)

  {

**digitalWrite**(trigPin, LOW);

**delayMicroseconds**(2);

*// Sets the trigPin HIGH (ACTIVE) for 10 microseconds*

**digitalWrite**(trigPin, HIGH);

**delayMicroseconds**(10);

**digitalWrite**(trigPin, LOW);

*// Reads the echoPin, returns the sound wave travel time in microseconds*

      duration = **pulseIn**(echoPin, HIGH);

*// Calculating the distance*

      distance = duration \* 0.034 / 2; *// Speed of sound wave divided by 2 (go and back)*

*//dumping the first 25 sets of samples in the memory and shift the last 75 sets of samples to the top*

    for (byte i = 15; i < 50; i++)

    {

      redBuffer[i - 15] = redBuffer[i];

      irBuffer[i - 15] = irBuffer[i];

    }

*//take 25 sets of samples before calculating the heart rate.*

    for (byte i = 35; i < 50; i++)

    {

      while (particleSensor.**available**() == false) *//do we have new data?*

        particleSensor.**check**(); *//Check the sensor for new data*

      redBuffer[i] = particleSensor.**getRed**();

      irBuffer[i] = particleSensor.**getIR**();

      particleSensor.**nextSample**(); *//We're finished with this sample so move to next sample*

    }

      while (!Serial.**available**());

      comm = Serial.**readString**().**toInt**();

      Serial.**println**(distance);

      Serial.**println**(mlx.**readObjectTempC**());

      Serial.**println**(spo2);

      if (comm == 9)

      {

**digitalWrite**(13, HIGH);

**delay**(5000);

**digitalWrite**(13, LOW);

      }

**maxim\_heart\_rate\_and\_oxygen\_saturation**(irBuffer, bufferLength, redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);

  }

}

**Operating System Setup:**

1. Install the JetPack OS image from official Nvidia Jetson Nano Website
2. Now format the used the memory card user to connect the card with the computer
3. First format he memory card and the used the image burn application to upload the Jetpack on memory card.
4. Now install the memory card on jetson nano, jetpack provide many preinstall libraries such as opencv and also python. So need to install these but we have to download and install mediapipe and tensorflow. First update the os by using these commands in terminal,

sudo apt-get update

sudo apt-get dist-upgrade

1. Now used these command provided by the nvidia jetson nano official website to install tensorflow (link: https://docs.nvidia.com/deeplearning/frameworks/install-tf-jetson-platform/index.html)

$ sudo apt-get update

$ sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran

$ sudo apt-get install python3-pip

$ sudo pip3 install -U pip testresources setuptools==49.6.0

$ sudo pip3 install -U --no-deps numpy==1.19.4 future==0.18.2 mock==3.0.5 keras\_preprocessing==1.1.2 keras\_applications==1.0.8 gast==0.4.0 protobuf pybind11 cython pkgconfig

$ sudo env H5PY\_SETUP\_REQUIRES=0 pip3 install -U h5py==3.1.0

$ sudo pip3 install --pre --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v46 tensorflow

1. And for mediapipe used these command (source: github PINTO309)

git clone https://github.com/PINTO0309/mediapipe-bin

cd mediapipe-bin

sudo apt install curl

./v0.8.5/numpy119x/mediapipe-0.8.5\_cuda102-cp36-cp36m-linux\_aarch64\_numpy119x\_jetsonnano\_L4T32.5.1\_download.sh

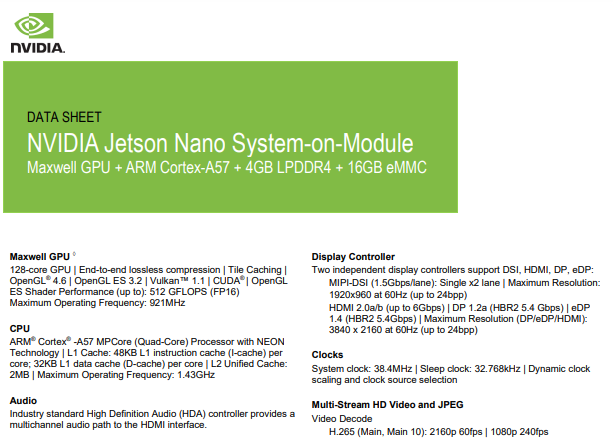
sudo pip3 install numpy-1.19.4-cp36-none-manylinux2014\_aarch64.whl

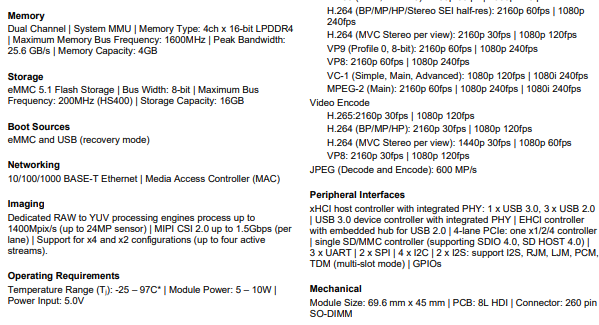
sudo pip3 install mediapipe-0.8.5\_cuda102-cp36-none-linux\_aarch64.whl

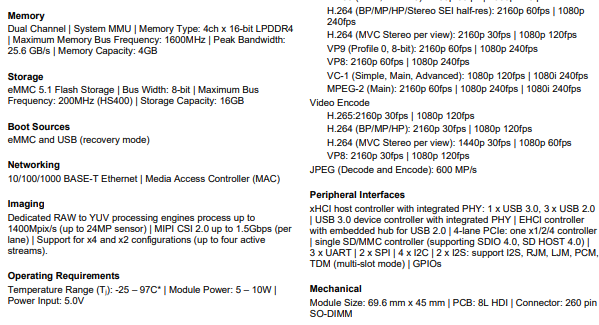
pip3 install dataclasses

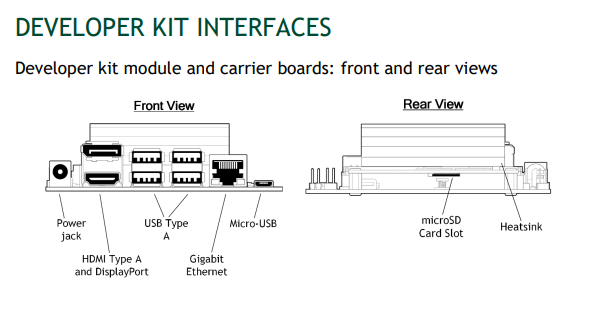
**Datasheets:**

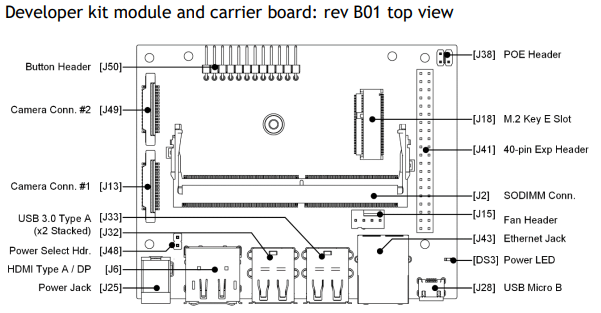
**Nvidia Jetson Nano:**



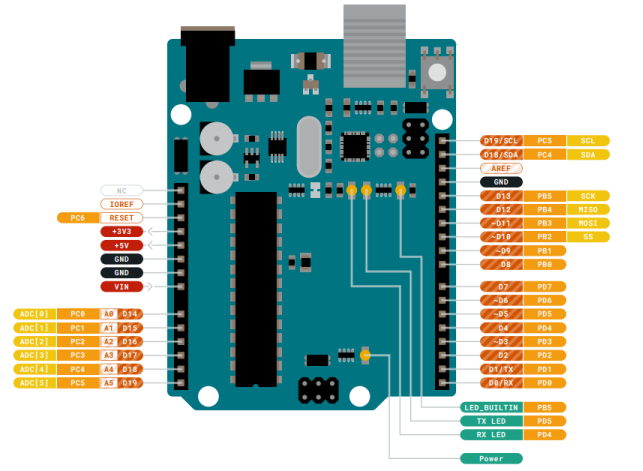


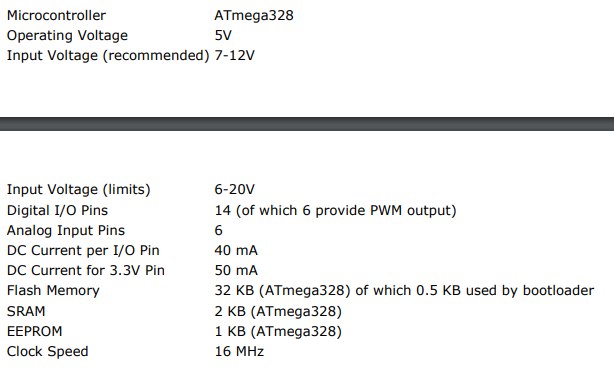




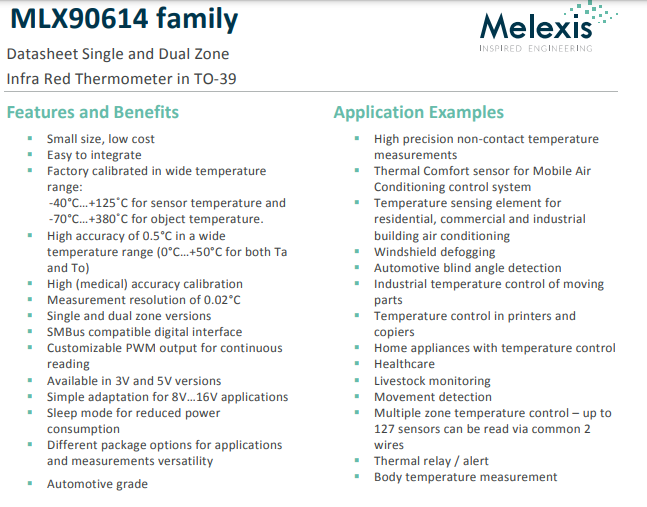


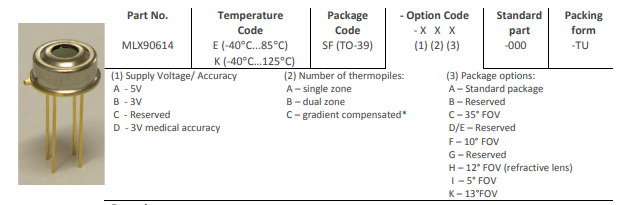
**Arduino Uno:**



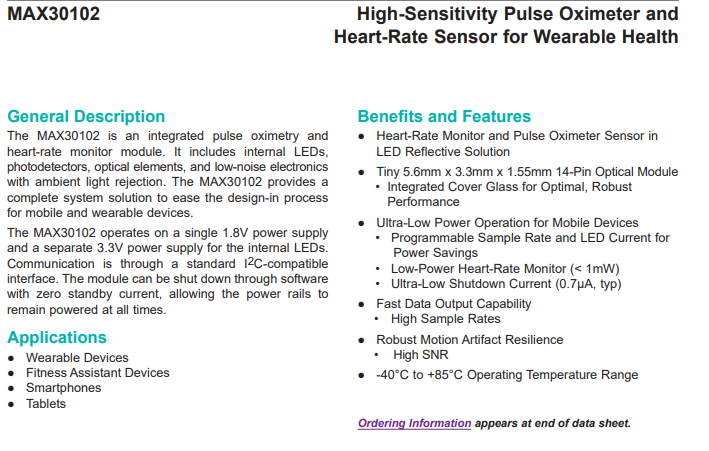


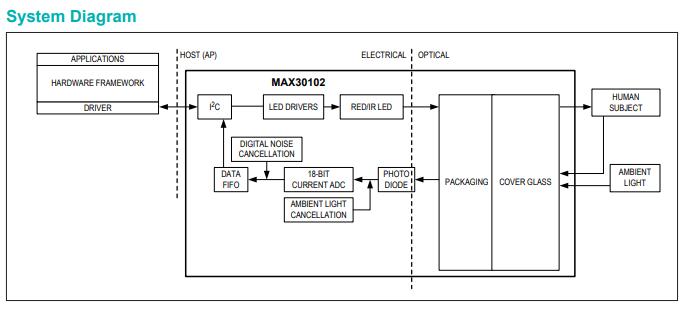
**MLX90614:**



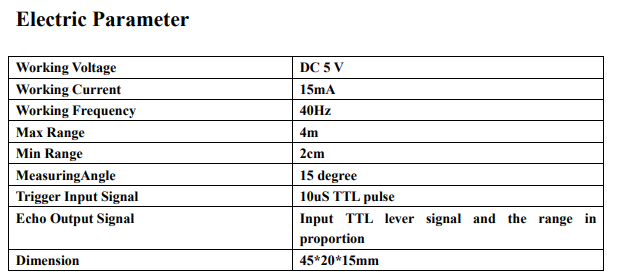


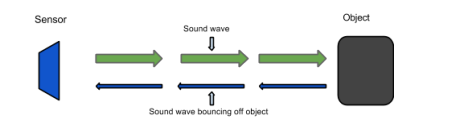
**MAX30102 Datasheet:**

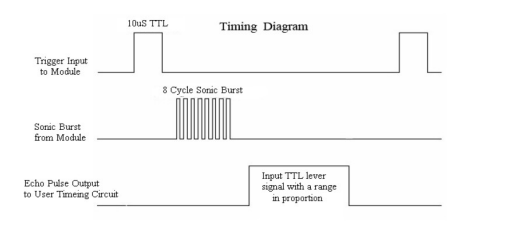




**HC-SR04 Ultrasonic Sensor:**

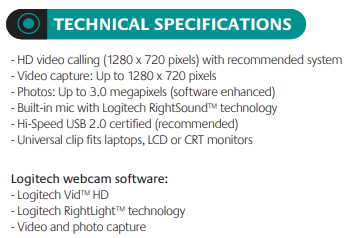


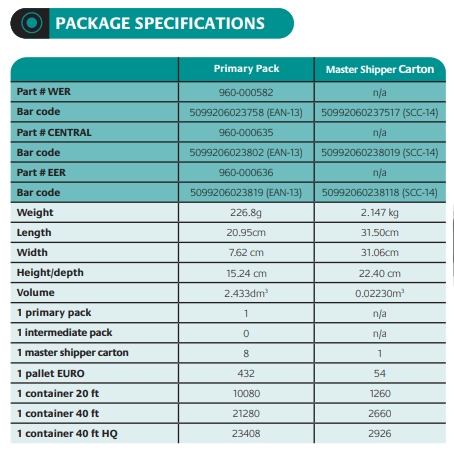




**Logitech C270 Datasheet:**







**Glossary**

|  |  |
| --- | --- |
| AI | Artificial Intelligence |
| CNN | Convolutional Neutral Network |
| COVID-19 | Corona Virus Disease of 2019 |
| FPS | Frame Per Second |
| LCD | Liquid Crystal Display |
| mAP | Mean Average Precision |
| MP | Mega Pixels |
| NoIR | No Infra Red filter |
| SOPs | Standard Operating Procedures |
| TTL | Through the Lens |
| SSD | Single Shot Detector |