A Project Report on

Wearable Test Bench using Raspberry Pi

Submitted by

AIMAN ANSARI

BHUMI AVHAD

Under the Guidance of

PROF. SONAL GUPTA

Head of Department

PROF. VIJAY PATIL

Department of Computer Engineering Vidyalankar Polytechnic

Wadala(E), Mumbai-400037

Maharashtra State Board of Technical Education, Mumbai

Institute Vision and Mission

Vision

To achieve excellence in imparting technical education so as to meet the professional and societal needs.

Mission

- Developing technical skills by imparting knowledge and providing hands-on experience.
- Creating an environment that nurtures ethics, leadership and team building.
- Providing industrial exposure for minimizing the gap between academics and industry.

Program: Computer Engineering
(NBA Accredited)

Vision

To empower students with domain knowledge of Computer Engineering and interpersonal skills to cater to industrial and societal needs.

Mission

M1: Developing technical skills by explaining the rationale behind learning.

M2: Developing interpersonal skills to serve the society in the best possible manner.

M3: Creating awareness about the ever-changing professional practices to build industrial adaptability

Program Educational Objectives (PEO)

PEO1: Provide socially responsible, environment-friendly solutions to Computer engineering-related broad-based problems adapting professional ethics.

PEO2: Adapt state-of-the-art Computer engineering broad-based technologies to work in multidisciplinary work environments.

PEO3: Solve broad-based problems individually and as a team member communicating effectively in the world of work.

Program Outcomes (PO)

PO1. Basic knowledge: Apply knowledge of basic mathematics, sciences, and basic engineering to solve the broad-based Computer engineering problem.

PO2. Discipline knowledge: Apply Computer engineering discipline-specific knowledge to solve core computer engineering-related problems.

PO3. Experiments and practice: Plan to perform experiments and practices to use the results to solve broad-based Computer engineering problems.

PO4. Engineering tools: Apply relevant Computer technologies and tools with an understanding of the limitations.

PO5. The engineer and society: Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to practice in the field of Computer engineering.

PO6. Environment and sustainability: Apply Computer engineering solutions also for sustainable development practices in societal and environmental contexts and demonstrate the knowledge and need for sustainable development.

PO7. Ethics: Apply ethical principles for commitment to professional ethics, responsibilities, and norms of the practice also in the field of Computer engineering.

PO8. Individual and teamwork: Function effectively as a leader and team member in diverse/ multidisciplinary teams.

PO9. Communication: Communicate effectively in oral and written form.

PO10. Life-long learning: Engage in independent and life-long learning activities in the context of technological changes in the Computer engineering field and allied industry.

Program Specific Outcomes (PSO)

PSO1. Computer Software and Hardware Usage: Use state-of-the-art technologies for operation and application of computer software and hardware.

PSO2. Computer Engineering Maintenance: Maintain computer engineering related software and hardware systems.

Acknowledgment

This project is a huge team effort. My team and I extend our deepest gratitude and thanks to the following people who have helped us to achieve our work.

Special thanks to **Er. Sonal Gupta** for guiding us and helping in a time when we needed most. We got to learn many things from her and it was our pleasure to work with her. My team and I extend thanks to the faculties of our college whom we have approached for academic help with regards to our project. We also thank our HOD **Er. Vijay Patil**, our Principal **Prof. Ashish Ukidve** for their support and guidance.

Thanks to all our teachers in the past to have inculcated in us values and work habits, that have allowed us to create the level of success that we have achieved today, in our teamwork.

(AIMAN ANSARI)

(BHUMI AVHAD)

ABSTRACT

Our proposed system allows users to code in over 50+ programming languages on a fully powered Linux based Raspbian OS wearable device with Face Recognition system; Facial recognition is a feature that uses facial detection algorithms to detect a face and then invokes facial recognition algorithms to try to match the person's face. Geany supports coding in a ridiculous number of programming languages including C, PHP, HTML, Java, Python, Ruby, Perl, Pascal, Assembly, and more. Glasses are a necessity for most of the users, there was always a need for designing a device which would allow users to code and to provide users a Linux based wearable computer right up to their eye; which can be used in situations like in a commute or when a desktop is not available. Wearable devices are now at the heart of just about every discussion related to the Internet of Things (IoT). The Internet of Things (IoT) is an emerging paradigm for the range of new capabilities brought about by pervasive connectivity. The users would also be able to interact with the system using a wireless keyboard and mouse via Bluetooth in Raspberry Pi

Table of content:

1. Introduction

Importance of the project:

The proposed system represents a wearable device that provides users a Linux based portable computer and has PiCam which recognizes faces. Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME - Geany only requires the GTK2 runtime libraries. Geany supports coding in a ridiculous number of programming languages including C, PHP, HTML, Java, Python, Ruby, Perl, Pascal, Assembly, and more; it runs on everything including Raspberry Pi. For face recognition, OpenCV has an abundance of libraries, tools, and framework to work on; using OpenCV we can implement face recognition using PiCam on Raspberry Pi.

Our proposed system allows users to code in over 50+ programming languages on a fully powered Linux based Raspbian OS wearable device. The users would also be able to interact with the system using a wireless keyboard and mouse via Bluetooth in Raspberry Pi. Based on this concept, many companies are developing wearable smart glasses with improved display, performance, battery life and capabilities.

Why a wearable device?

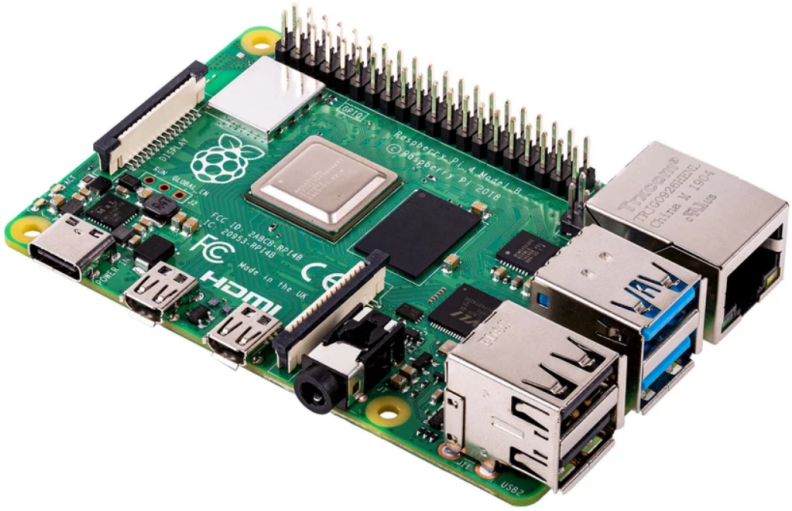
Glasses are a necessity for most of the users, there was always a need for designing a device which would allow users to code and to provide users a Linux based wearable computer right up to their eye; which can be used in situations like in a commute or when a desktop is not available.

Review of Literature

A review of the literature provided in this section includes a brief overview of Raspberry Pi, a review of previously published articles and contributions of our research.

Technologies Used:

- **Raspberry Pi:**



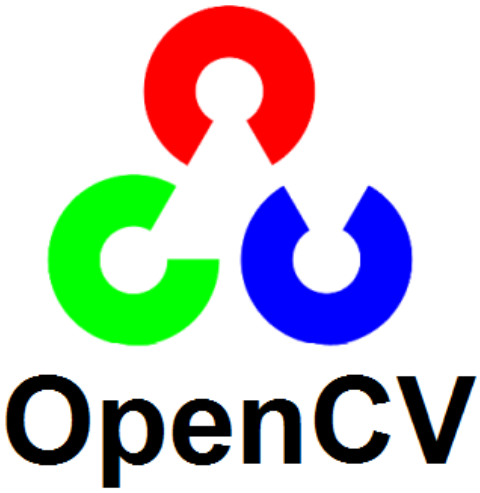
Raspberry Pi 4 Model B with a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, and dual monitor support (4K resolution). The Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals, when used with an appropriate PSU. Three sizes of onboard RAM are available: 1 GB (US$35), 2 GB (US$45), 4 GB (US$55). The Raspberry Pi 4 has a design flaw where third-party e-marked USB cables, such as those used on Apple MacBooks, incorrectly identify it and refuse to provide power. This is expected to be corrected in a future board revision

- **Raspbian OS:**

RASPBIAN

Raspbian is the main and basic software for Raspberry Pi devices, officially supported by the Raspberry Pi Foundation. In fact, it is an operating system, based on Debian and optimized for Raspberry Pi hardware.

- **OpenCV:**



OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.
OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers

- **Geany:**

Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME - Geany only requires the GTK2 runtime libraries. Geany supports coding in a ridiculous number of programming languages

- **Raspberry PiCam:**

The Raspberry Pi Camera v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video.

Previous Related Work

Sr. No.	Authors	Contribution
1	Agrawal & Singhal	Researchers discussed a smart drip irrigation system using Raspberry Pi including a proposed design for a home automation system, energy-efficient devices, including Raspberry Pi, Arduino microcontrollers, XBee modules, and relay boards.
2	Danymol, Ajitha & Gandhiraj	Researchers explained about real-time communication systems and the use of Raspberry Pi because it is cheap and small.
3	Severance & Fontichiaro	Researchers discussed the history of Raspberry Pi's development and potential applications of Pi in classrooms for engaging students in programming.
4	Lynn	Lynn discussed the main idea of using Raspberry Pi was getting students engaged in computing and learning to Program However, Raspberry Pi was used as a standalone computer to compose music, drawings and build robots
5	Byrne, Fisher & Tangney	Researchers explained about Raspberry Pi in the 21st - century learning environment and adaptation of a Bridge21 model for Raspberry Pi. Raspberry Pi was used to introduce basic code compilation.

Proposed System

Our proposed system allows users to code in over 50+ programming languages on a fully powered Linux based Raspbian OS wearable device. The users would also be able to interact with the system using a wireless keyboard and mouse and has face recognition using OpenCV on the Raspberry Pi. There was always a need for designing a device that would allow users to code and to provide users a Linux based wearable computer right up to their eye; which can be used in situations like in a commute or when a desktop is not available.

Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME - Geany only requires the GTK2 runtime libraries. Geany supports coding in a ridiculous number of programming languages including C, PHP, HTML, Java, Python, Ruby, Perl, Pascal, Assembly, and more; it runs on everything including Raspberry Pi.

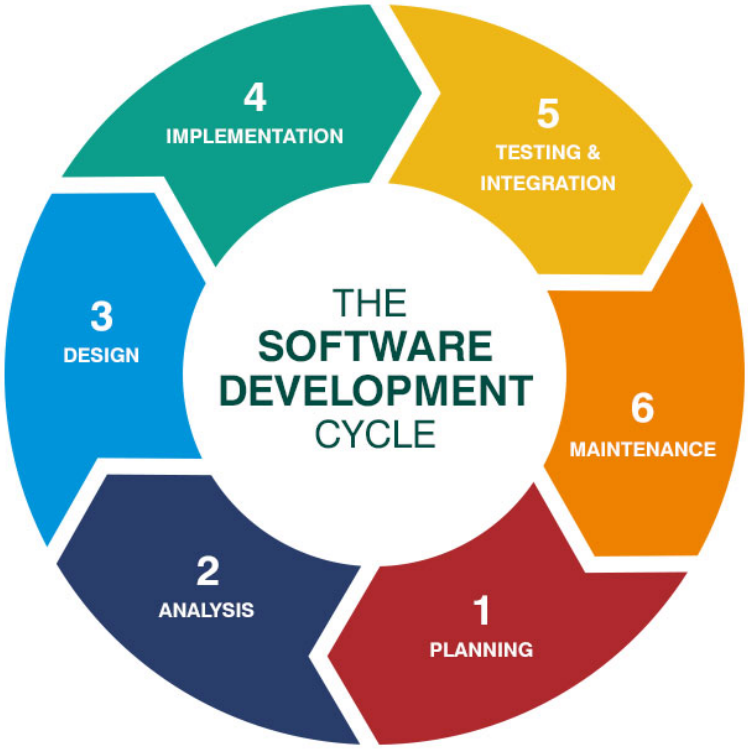
OpenCV (Open Source Computer Vision) is a popular computer vision library started by Intel in 1999. The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms.

The facial recognition algorithm uses datasets to match the currently presented face with the faces saved in the datasets. The efficiency of a facial recognition algorithm depends upon the speed at which it can detect and recognize faces or how many datasets are entered for one person.

Wearable devices are now at the heart of just about every discussion related to the Internet of Things (IoT). The Internet of Things (IoT) is an emerging paradigm for the range of new capabilities brought about by pervasive connectivity. The concept involves situations where network connectivity and computing capability expand to objects, sensors, and everyday items that exchange data with little to no human involvement.

Plan of Work

Software Development Life Cycle



- **Planning:**

  The Planning phase is the most crucial step in creating a successful Face Recognition System, during this phase you decide exactly what you want to do and the problems you're trying to solve.

- **Analysis:**

  The end user's requirements should be determined and documented, what their expectations are for the system, and how it will perform. A feasibility study will be made for the project as well, involving determining whether it's organizationally, economically, socially, technologically feasible. It's very important to maintain a strong communication level with the clients to make sure you have a clear vision of the finished product and its function.

- **Design:**

  A general system design can be done with a pen and a piece of paper to determine how the system will look like and how it will function, and then a detailed and expanded system design is produced, and it will meet all functional and technical requirements, logically and physically.

- **Implementation:**

  The implementation phase will contain configuration and fine-tuning for the hardware to meet certain requirements and functions. In this phase, the system is ready to be deployed and installed in customer's premises, ready to become running, live and productive, training may be required for end-users to make sure they know how to use the system and to get familiar with it, the implementation phase may take a long time and that depends on the complexity of the system and the solution it presents.

- **Testing & Integration:**

  System Testing and Integration Bringing different components and subsystems together to create the whole integrated system and then introducing the system to different inputs to obtain and analyze its outputs and behavior and the way it functions.

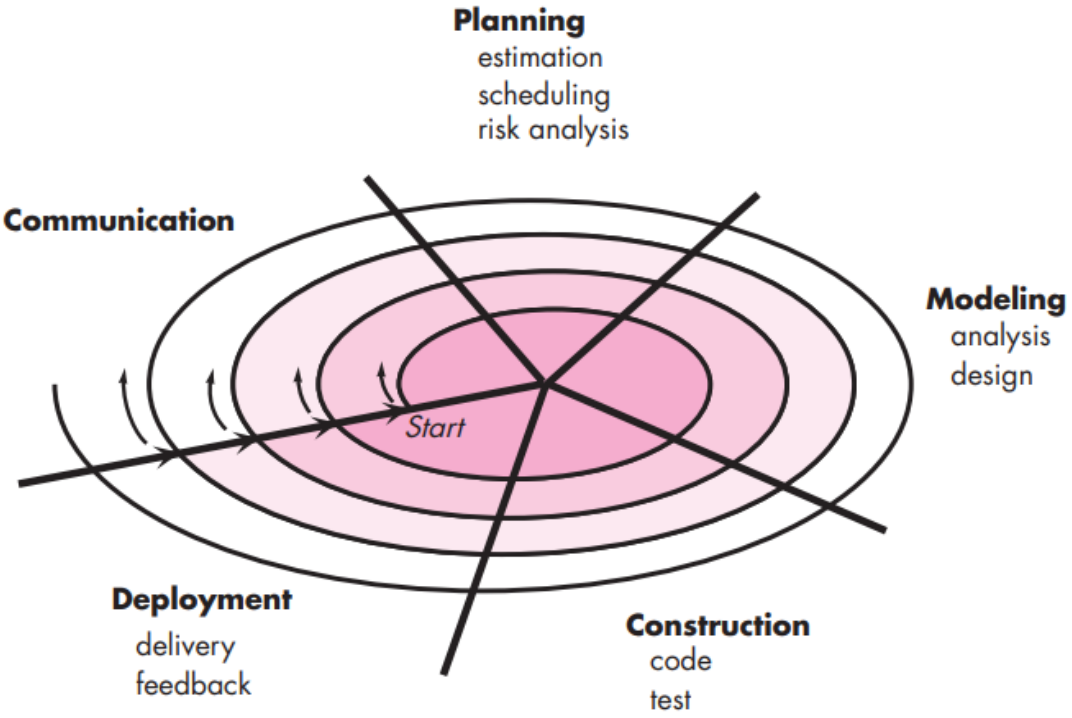
- **Implementation:**

  In this phase, periodic maintenance for the system will be carried out to make sure that the system won't become obsolete, this will include replacing the old hardware and continuously evaluating system's performance, it also includes providing latest updates for certain components to make sure it meets the right standards and the latest technologies to face current security threats.

These are the main six phases of the System Development Life Cycle, and it's an iterative process for each project. we can guarantee meeting the customer's requirements before we build the whole system. Many models of the system development life cycle came up from the idea of saving effort; money and time, in addition to minimizing the risk of not meeting the customer's requirement at the end of a project, some of these models are SDLC Iterative Model and SDLC Agile Model.

Process Model:

The **Spiral Model** is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering, and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral.



This invariant identifies the four activities that must occur in each cycle of the spiral model:

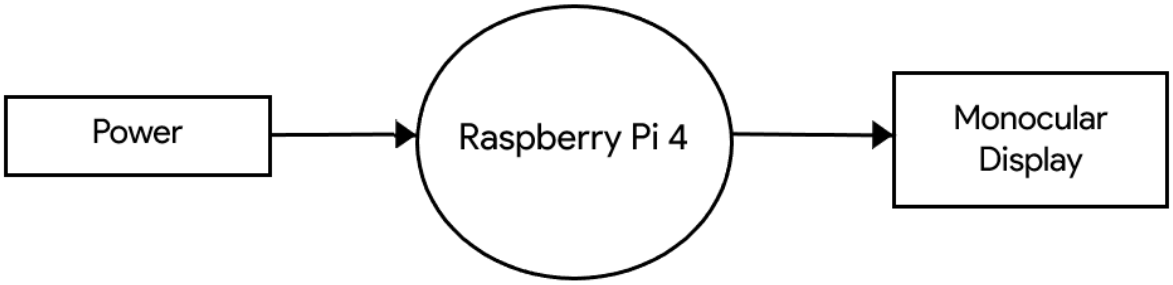
1. Consider the win conditions of all success-critical stakeholders.
2. Identify and evaluate alternative approaches for satisfying the win conditions.
3. Identify and resolve risks that stem from the selected approach(es).
4. Obtain approval from all success-critical stakeholders, plus commitment to pursue the next cycle.

Project cycles that omit or shortchange any of these activities risk wasting effort by pursuing options that are unacceptable to key stakeholders or are too risky.

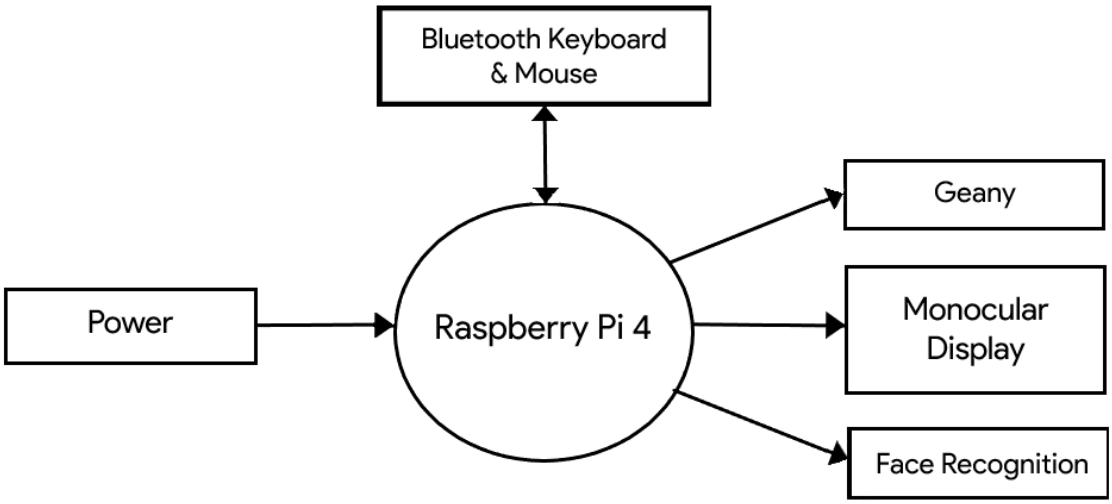
Analysis and Design

Diagrams:

- **Data Flow Diagram:**

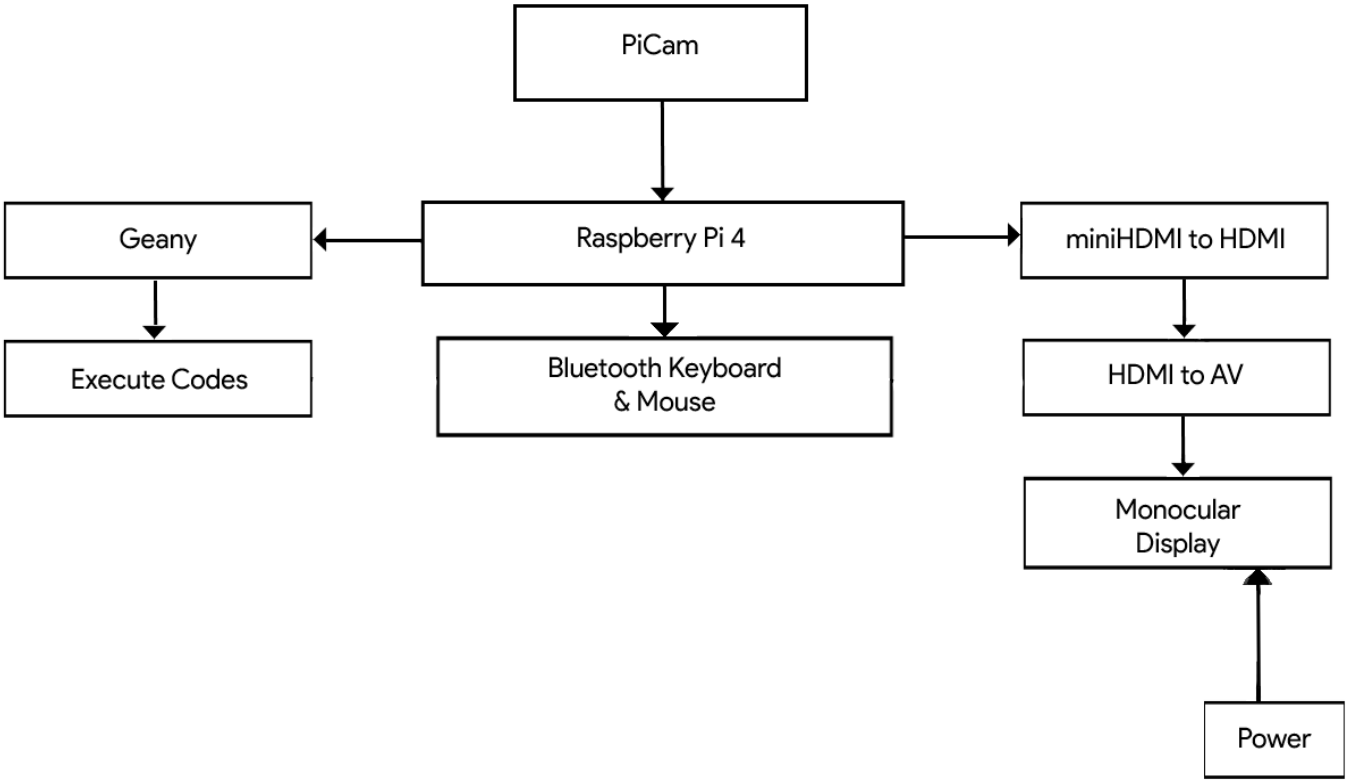


Level 0



Level 1

- **Block Diagram:**



Hardware Requirements and Specifications:

1. Monocular Display
2. Raspberry Pi 4 Model B
3. Raspberry PiCam

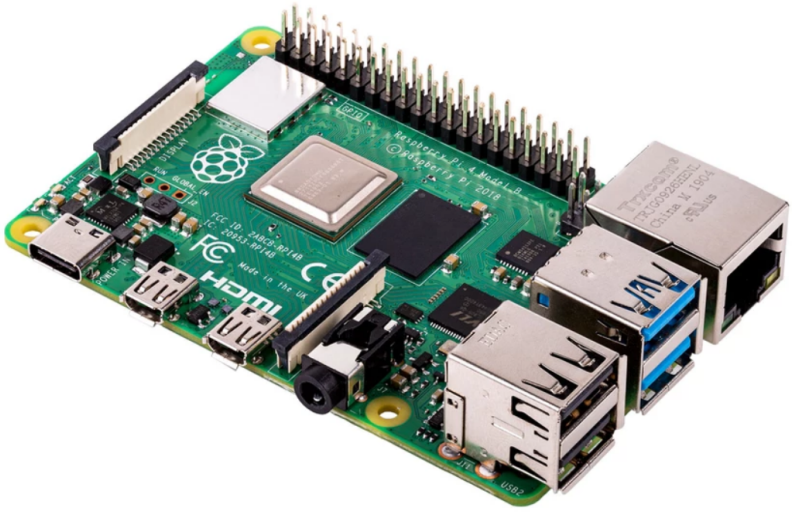
- **Monocular Display:**

The monocular display works as a display unit for the Raspbian OS installed on the Raspberry Pi 4, which means you can use all the features that Raspbian OS offers.

Specifications:

Resolution	QVGA 320*240
Video-Input	Simulation CVBS
Operating voltage	3.5V - 5V
Supported Format	8-bit RGB-serial data and more

- **Raspberry Pi 4 Model B:**

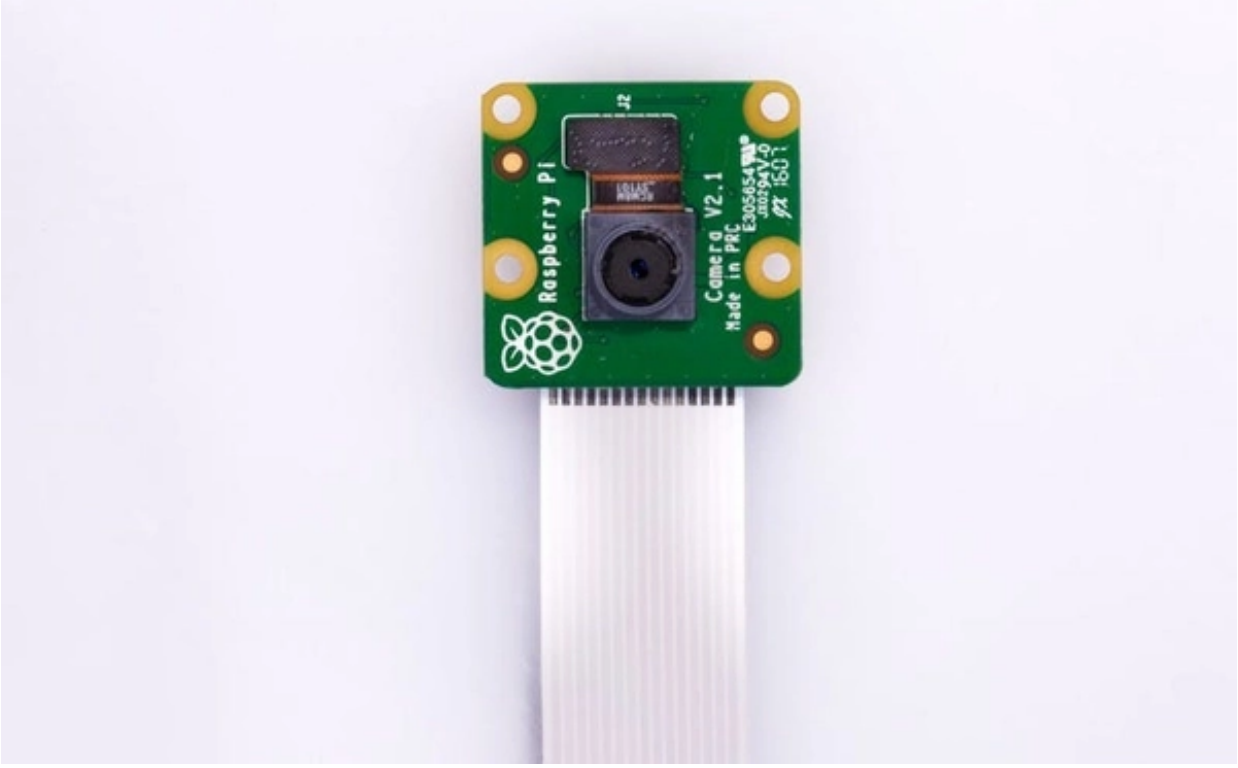


Raspberry Pi has long been the gold standard for inexpensive single-board computing, powering everything from robots to smart home devices to digital kiosks.

Specifications:

Model	Raspberry Pi 4 Model-B with 4 GB RAM
Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
RAM Memory	4GB LPDDR4 SDRAM
GPIO	Standard 40-pin GPIO Header

- **Raspberry PiCam:**



The Raspberry Pi Camera v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video.

Specifications:

Sensor	Sony IMX219
Sensor resolution	3280 × 2464 pixels
Sensor image area	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel size	1.12 µm x 1.12 µm

Software Requirement:

- **Etcher:**

Etcher is a software which is used to burn the OS image to make it compatible to install into a storage disk, here we installed Raspbian.

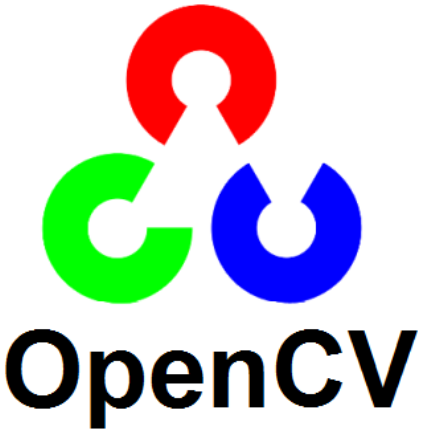
- **Raspbian:**

Raspbian is the main and basic software for Raspberry Pi devices, officially supported by the Raspberry Pi Foundation. In fact, it is an operating system, based on Debian and optimized for Raspberry Pi hardware.

- **Geany:**

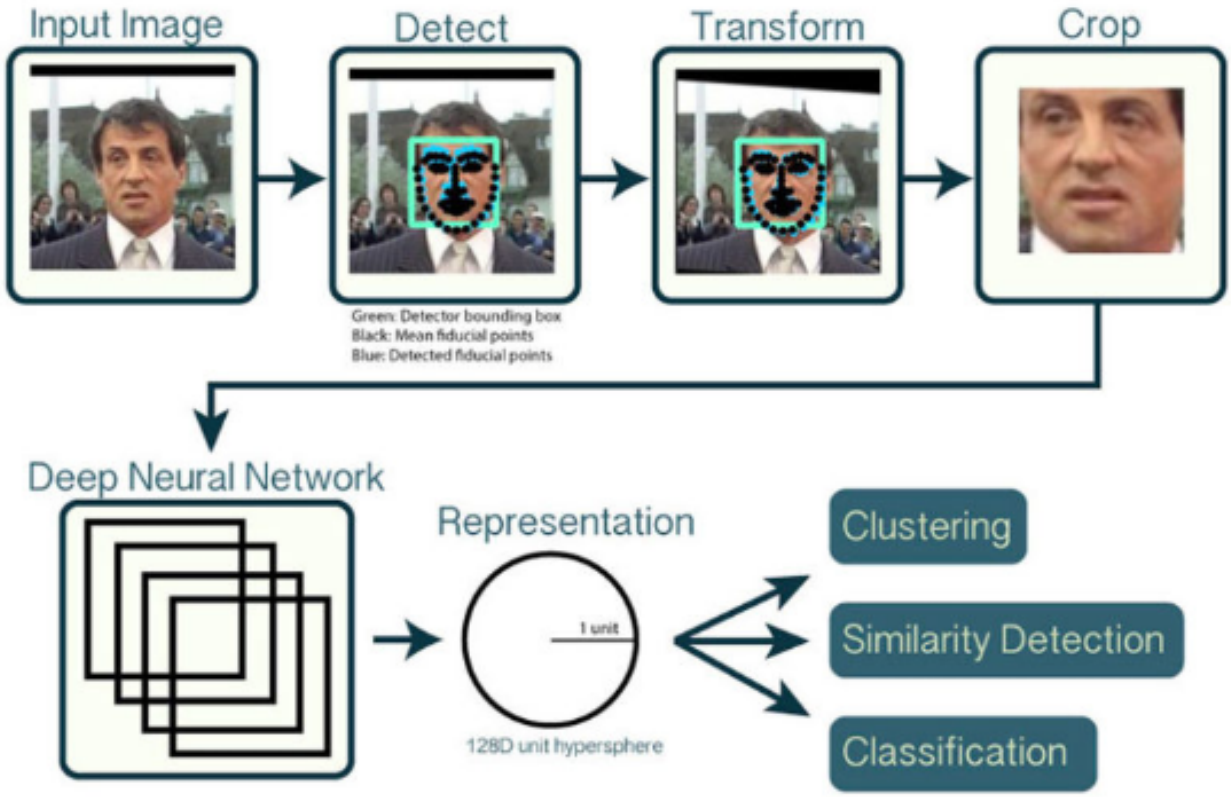
Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME - Geany only requires the GTK2 runtime libraries. Geany supports coding in a ridiculous number of programming languages.

- **OpenCV:**



OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.
OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers

Working of OpenCV Face Recognition:



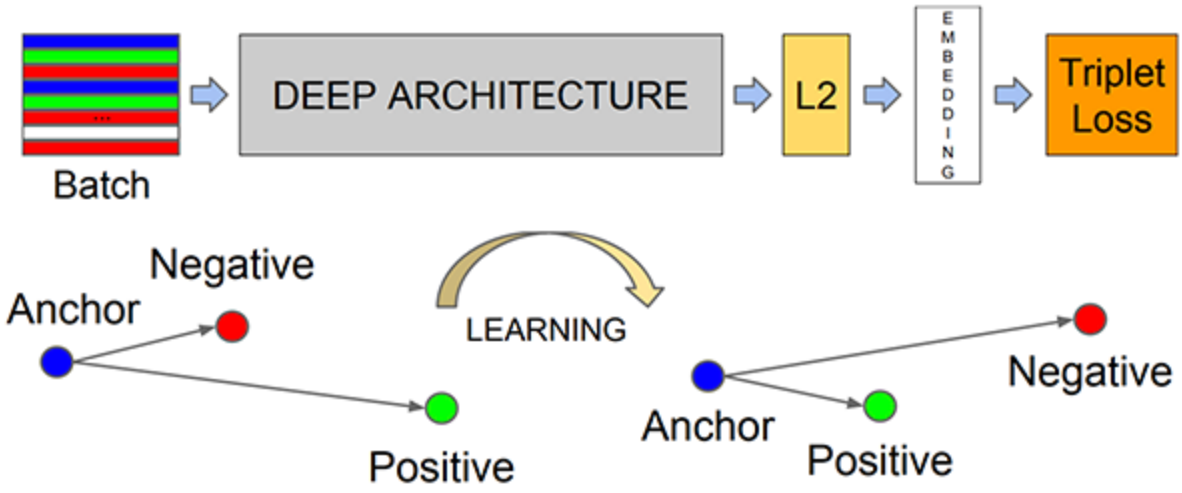
We'll be applying deep learning in two key steps:

1. To apply face detection, which detects the presence and location of a face in an image, but does not identify it
2. To extract the 128-d feature vectors (called "embeddings") that quantify each face in an image

First, we input an image or video frame to our face recognition pipeline. Given the input image, we apply face detection to detect the location of a face in the image. Optionally we can compute facial landmarks, enabling us to preprocess and align the face.

Face alignment, as the name suggests, is the process of (1) identifying the geometric structure of the faces and (2) attempting to obtain a canonical alignment of the face based on translation, rotation, and scale.

While optional, face alignment has been demonstrated to increase face recognition accuracy in some pipelines. After we've (optionally) applied face alignment and cropping, we pass the input face through our deep neural network:



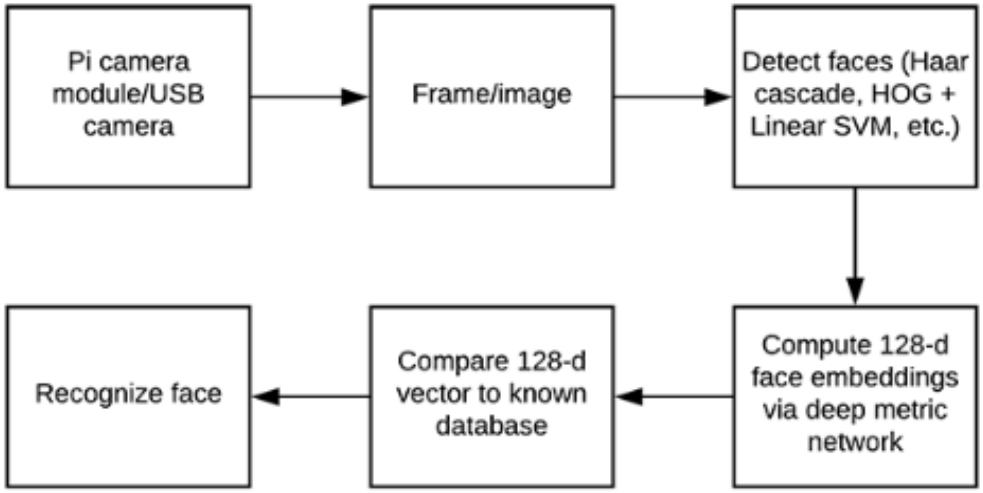
The FaceNet deep learning model computes a 128-d embedding that quantifies the face itself. To train a face recognition model with deep learning, each input batch of data includes three images:

1. The anchor
2. The positive image
3. The negative image

The point is that the anchor and positive image both belong to the same person/face while the negative image does not contain the same face.

The neural network computes the 128-d embeddings for each face and then tweaks the weights of the network (via the triplet loss function) such that:

The 128-d embeddings of the anchor and positive image lie closer together
While at the same time, pushing the embeddings for the negative image father away
In this manner, the network is able to learn to quantify faces and return highly robust and discriminating embeddings suitable for face recognition.



After feeding the Face Recognition dataset, we extract embeddings from the face dataset and train the face model. And now we are ready to recognize faces.

Testing

Introduction:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs. Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed.

As such, the methodology of the test is governed by the software development methodology adopted. Different software development models will focus the test effort at different points in the development process.

Newer development models, such as Agile, often employ test driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

Testing Methods

- White Box Testing
- Black Box Testing
- Gray Box Testing

- **White-Box Testing:**

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural box testing) is a method of testing software that tests internal structure or workings of an application. In white – box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). While white – box testing can be applied at the unit level, integration and system levels of the software testing process, it is usually done at unit level.

It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, this might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement Coverage
- Decision Coverage

- **Black Box Testing:**

Black–box testing is a method of testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white – box testing). This method of test can be applied to virtually every level of software: unit, integration, system and acceptance.

It typically comprises most if not all higher level testing, but can also dominate unit testing as well. Specific knowledge of the application's code/internal structure and programming knowledge in general is not required.

- Gray Box Testing:

Gray–box testing is a combination of white – box testing and black – box testing. The aim of this testing is to search for the defects if any due to improper structure or improper usage of applications. Gray –box testing is also known as translucent testing. A black – box tester is unaware of the internal structure of the application.

A gray–box tester partially knows the internal structure,
which includes access to the documentation of internal data structures as well as the algorithms used. Gray – box testers require both high level and detailed documents describing the application, which they collect in order to define the test cases.

Test Cases for Face Recognition System:

T_ID	Test Description	Actual Input	Actual Output	Expected Output	Result
T_01	No Datasets feed to the system	Randoms face shown	Does not recognizes	Does not recognizes	Pass
T_02	Enters first person dataset	Shows the first face	Recognizes the face	Recognizes the face	Pass
T_03	Dims the environment	Shows the same face	Should recognize face	Recognizes the face	Pass
T_04	Enters second person dataset	Shows the second face	Should recognize face	Recognizes the face	Pass
T_05	Checking mismatching of faces	Shows both the faces	Should recognize each face	Recognize each face	Pass
T_06	Enters third person dataset without training the module	Shows the third face	Should not recognize the face	Does not recognizes	Pass
T_07	Enters the fourth dataset with similar physical appearance to the first	Shows both the faces	Should recognize each faces	Recognize each faces	Pass
T_08	Enters the fifth dataset with similar physical appearance to the third	Shows both the faces	Should recognize each faces	Recognize each faces	Pass
T_09	Checking mismatching of faces	Shows fifth and second faces	Should recognize each face	Recognize each face	Pass
T_10	Enters sixth person dataset	Shows the second face	Should recognize face	Recognizes the face	Pass
T_11	Dims the environment	Shows the fifth and sixth faces	Should recognize each faces	Recognizes each faces	Pass
T_12	Checking mismatching of faces	Shows first and third faces	Should recognize each face	Recognize each face	Pass
T_13	Enters seventh person dataset	Shows the seventh face	Should recognize face	Recognizes the face	Pass

T_14	Enters the eight dataset with similar physical appearance to the seventh	Shows both the faces	Should recognize each faces	Recognize each faces	Pass
T_15	Checking mismatching of faces	Shows eight and second faces	Should recognize each face	Recognize each face	Pass
T_16	Add only one dataset of the ninth face	Shows the ninth face	Should fluctuate recognition	Fluctuate recognition	Pass
T_17	Adds more datasets of the ninth face	Shows the ninth face	Should recognize the face	Recognize the face	Pass
T_18	Dims the environment	Shows the ninth face	Should recognize face	Recognizes the face	Pass
T_19	Moves away from the camera	Shows the ninth face	Should recognize the face	Recognizes the face	Pass
T_20	Object looks away from the camera	Looks away from the camera	Should not recognize face	Does not recognize the face	Pass

Feasibility Analysis

A feasibility analysis usually involves a thorough assessment of the operational (need), financial and technical aspects of a proposal, Feasibility study is the test of the system proposal made to identify ' whether the user needs may be satisfied using the current software and hardware technologies, whether the system will be cost effective from a business point of view and whether it can be developed with the given budgetary constraints. A feasibility study should be relatively cheap and done at the earliest possible time.

When a new project is proposed it normally goes through feasibility assessment. Feasibility study is carried out to determine whether the proposed system is possible to develop with available resources and what should be the cost consideration. Facts considered in the feasibility analysis were:

- Technical Feasibility
- Time Feasibility
- Software / Hardware Feasibility
- Costing
- Timeline

Technical Feasibility:

Technical Feasibility deals with the hardware as well as software requirements. Technology is not a constraint to type system development. We have to find out whether the necessary technology, the proposed equipment has the capacity to hold the data, which is used in the project, should be checked to carry out this technical feasibility. The technical feasibility issues usually raised during the feasibility stage

Hardware/Software Feasibility:

Hardware Specification:

- Monocular Display
- Raspberry Pi 4
- Raspberry PiCam

Software Specifications:

- Raspbian OS
- Geany IDE
- OpenCV
- Etcher

Cost Analysis:

Costing Analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as economic/benefit analysis the procedure is to determine the benefits and savings that are expected from a system and compare them with costs, decisions made to design and implement systems. This part of the feasibility study gives the top management the economy because very often the top management does not like to get a project of this kind. A simple economic analysis that gives the actual comparison of costs and benefits.

Sr.No	Components Used	Price
1	Monocular Viewfinder Monitor Micro-Display	₹5000
2	Raspberry Pi 4 Model B	₹4500
3	Wireless keyboard and mouse	₹1000
4	Micro SD card 32GB	₹900
5	USB Type-C cable	₹350
6	HDMI to AV converter	₹450

Operational Feasibility:

It is concerned with human, organizational and political aspects. It includes in two ways:-

- **Technical Performance:**

  It includes issues such as whether the system can provide the right information for the organization's personnel at the right time and at the right place.

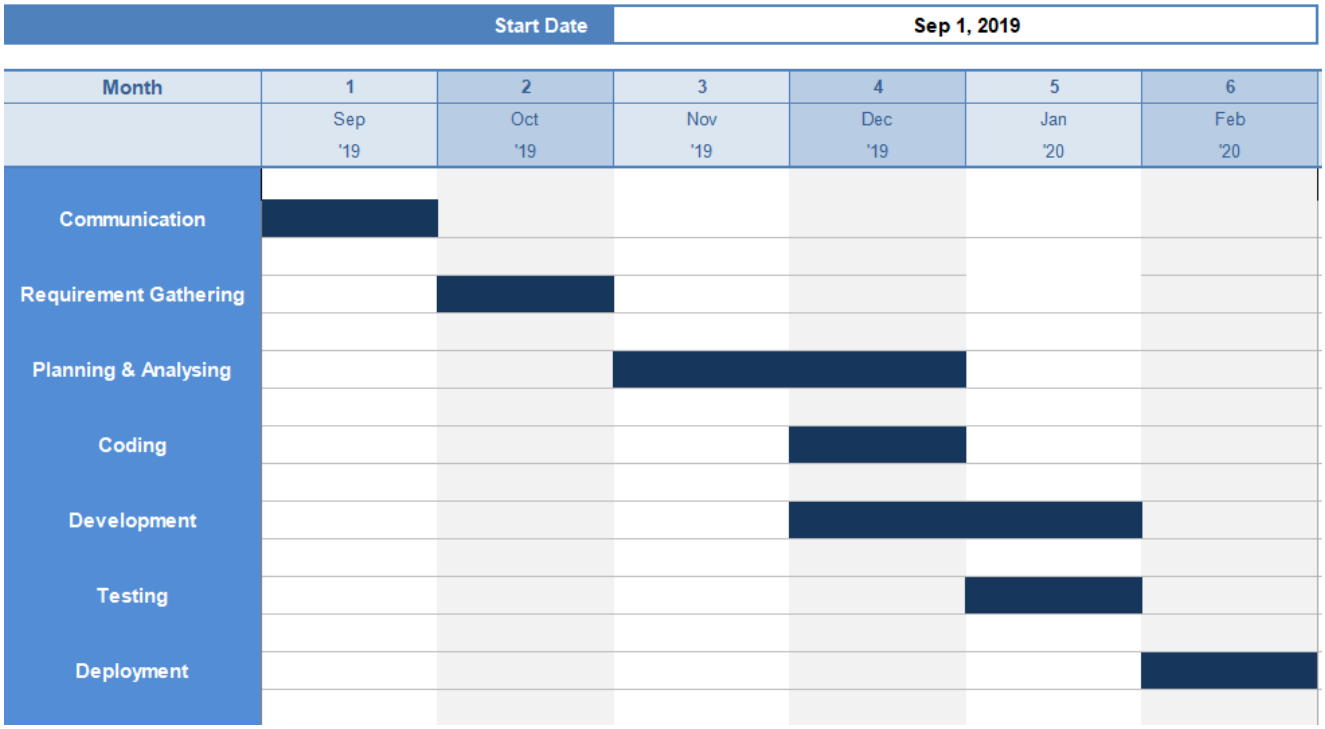
- **Acceptance within the organization:**

  It determines the general attitudes and job skills of existing personnel and whether any changes in job work will be acceptable by the current users.

Behavioral Feasibility:

It includes how strong the reaction of the staff will be towards the development of the new system that involves computer use in their daily work.

Timeline Chart:



Advantages and Disadvantages

Advantages:

- ➔ Portable Linux computer
- ➔ Work from anywhere
- ➔ Supports 50+ programming languages
- ➔ Face recognition
- ➔ PiCam can be also used as a camera
- ➔ Connects peripherals via Bluetooth
- ➔ Convenient to use

Disadvantages:

- ➔ Expensive
- ➔ Delicate
- ➔ Electric connections can be difficult for many users

Applications

Glasses are a necessity for most of the users, there was always a need for designing a device which would allow users to code and to provide users a Linux based wearable computer right up to their eye; which can be used in situations like in a commute or when a desktop is not available. PiCam attached to the glasses can be used for Face Recognition and also as a regular camera.

References

➔ Vufine
https://store.vufine.com/products/vufine-wearable-display-2

➔ Instructables
https://www.instructables.com/id/RaspberryPi-Powered-Wearable-Computer/

➔ Geany IDE
https://raspberry-projects.com/pi/programming-in-c/compilers-and-ides/geany/installing-geany

➔ Face Recognition
https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/