System Design for Smart Yolov

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. The architectural design of a system emphasizes on the design of the systems architecture which describes the structure, behavior, and more views of that system.

The purpose of the design phase is to plan a solution of the problem specified by the requirements document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed; design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phases particularly testing and maintenance.

The design activity often results in three separate outputs –

- Architecture design.
- High level design.
- Detailed design.

ARCHITECTURAL DESIGN

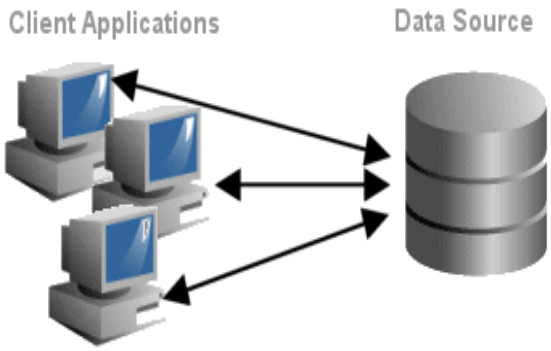
Two Tier Architecture

A two-tier architecture is a software architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents two-tier architecture, as opposed to a single-tier architecture. Other kinds of multi-tier architectures add additional layers in distributed software design.

Experts often contrast two-tier architecture to a three-tier architecture, where a third application or business layer is added that acts as an intermediary between the client or presentation layer and the data layer. This can increase the performance of the system and help

with scalability. It can also eliminate many kinds of problems with confusion, which can be caused by multi-user access in two-tier architectures. However, the advanced complexity of three-tier architecture may mean more cost and effort.

An additional note on two-tier architecture is that the word "tier" commonly refers to splitting the two software layers onto two different physical pieces of hardware. Multi-layer programs can be built on one tier, but because of operational preferences, many two-tier architectures use a computer for the first tier and a server for the second tier.



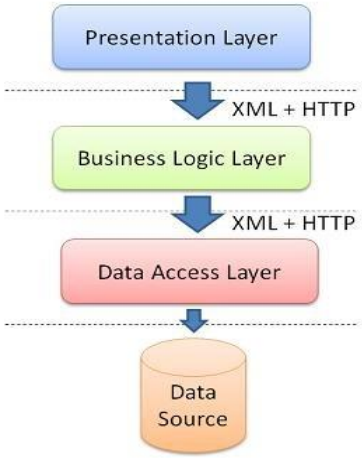
Advantages and Disadvantages

The advantage of the two-tier design is its simplicity. The database session that builds the two-tier architecture provides all the features in a single session type, thereby making the two-tier architecture simple to build and use.

The most important limitation of the two-tier architecture is that it is not scalable, because each client requires its own database session.

Three tier architecture

Three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. Three-tier architecture is a software design pattern and well-established software architecture.



Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on a desktop PC and uses a standard graphical user interface with different modules running on the application server. The relational database management system on the database server contains the computer data storage logic. The middle tiers are usually multi-tiered.

The three tiers in three-tier architecture are:

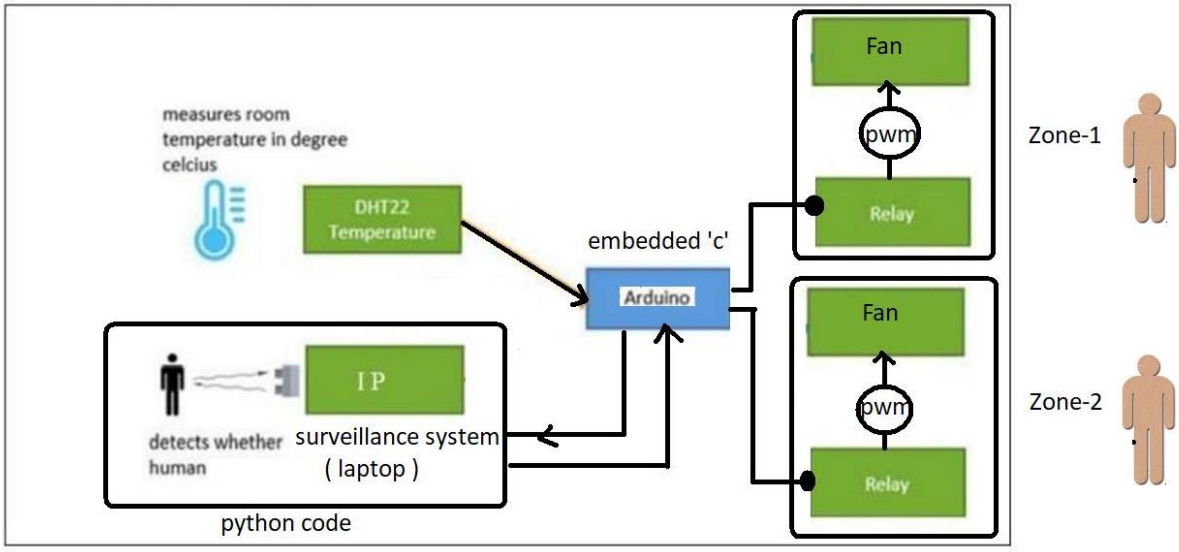
Presentation Tier: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network. The **presentation tier** contains the UI (User Interface) elements of the site, and includes all the logic that managers the interaction between the visitor and the client's business.

Application Tier: Also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing. The **business tier** receives requests from the presentation tier and returns a result to the presentation tier depending on the business logic it contains.

Data Tier: Houses database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic. The **data tier** is responsible for storing the application's data and sending it to the business tier when requested. (SQL Server Stored Procedures).

6.1 System Architecture

Architecture focuses on looking at a system as a combination of many different components, and how they interact with each other to produce the desired result. The focus is on identifying components or subsystems and how they connect. In other words, the focus is on what major components are needed.

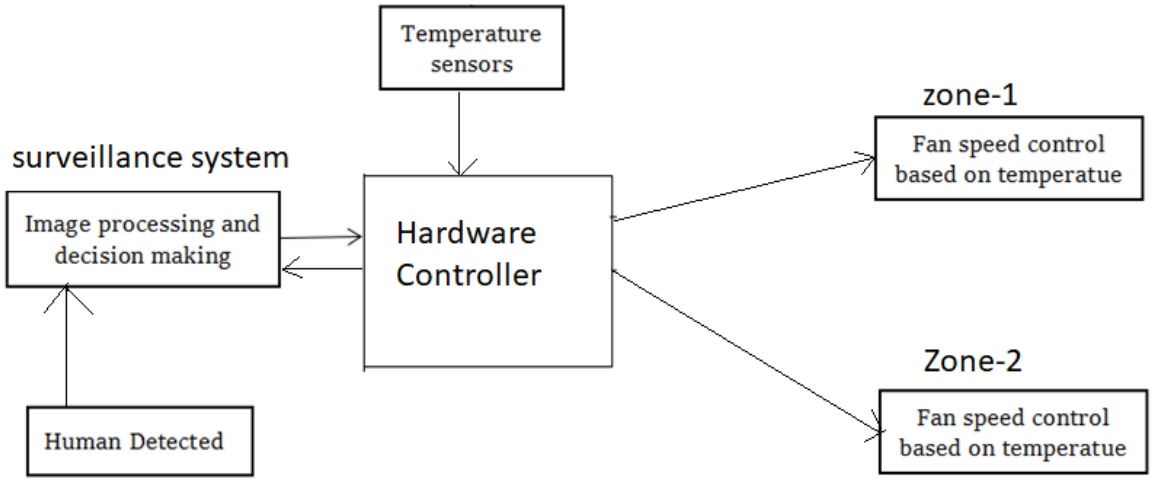


System architecture of smart yolov

6.2 Context data flow diagram

It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the 'Level 0 DFD') the system's interactions with the outside world are modeled purely in terms of data flows across the *system boundary*. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a

whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



Context data flow of Smart yolov

6.3 USE CASE DIAGRAMS

Use case diagrams are considered for high level requirement analysis of a system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modeled to present the outside view.

Use case:

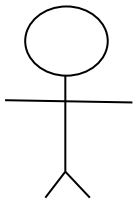
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So we can say that uses cases are nothing but the system functionalities written in an organized manner.

A use case describes a sequence of actions that provide something of measurable value to

an actor and is drawn as a horizontal ellipse.

Actor:

Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications.

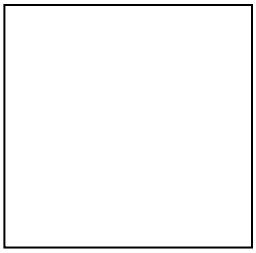


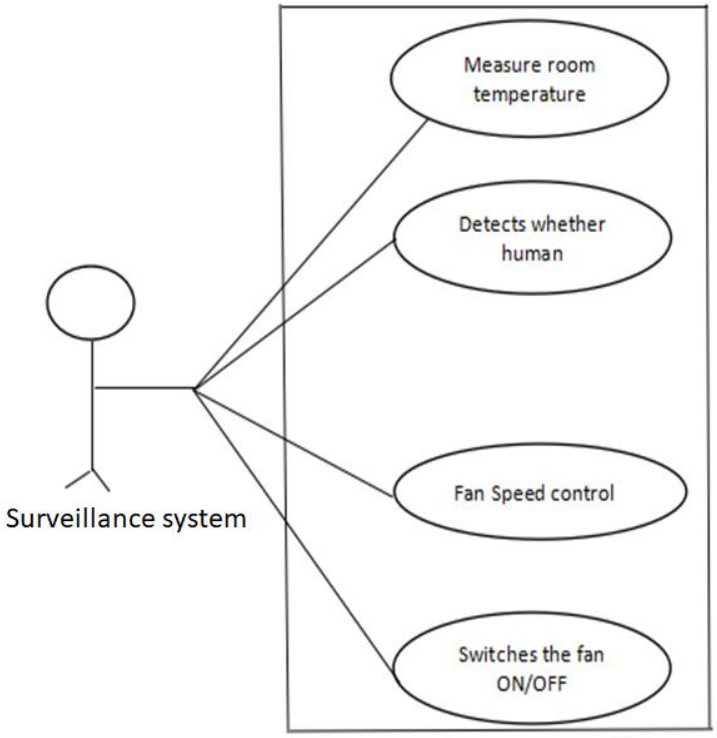
Associations:

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

System boundary boxes:

You can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope.





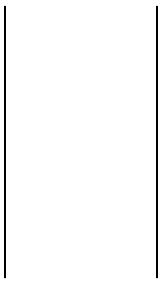
Use case Diagram of Smart yolov

6.4 SEQUENCE DIAGRAMS

The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. With the advanced visual modeling capability, you can create complex sequence diagram in few clicks. Besides, Visual Paradigm can generate sequence diagram from the flow of events which you have defined in the use case description. The sequence diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Lifelines:

A sequence diagram shows, as parallel vertical lines (lifelines), which indicates different processes or objects that live simultaneously.

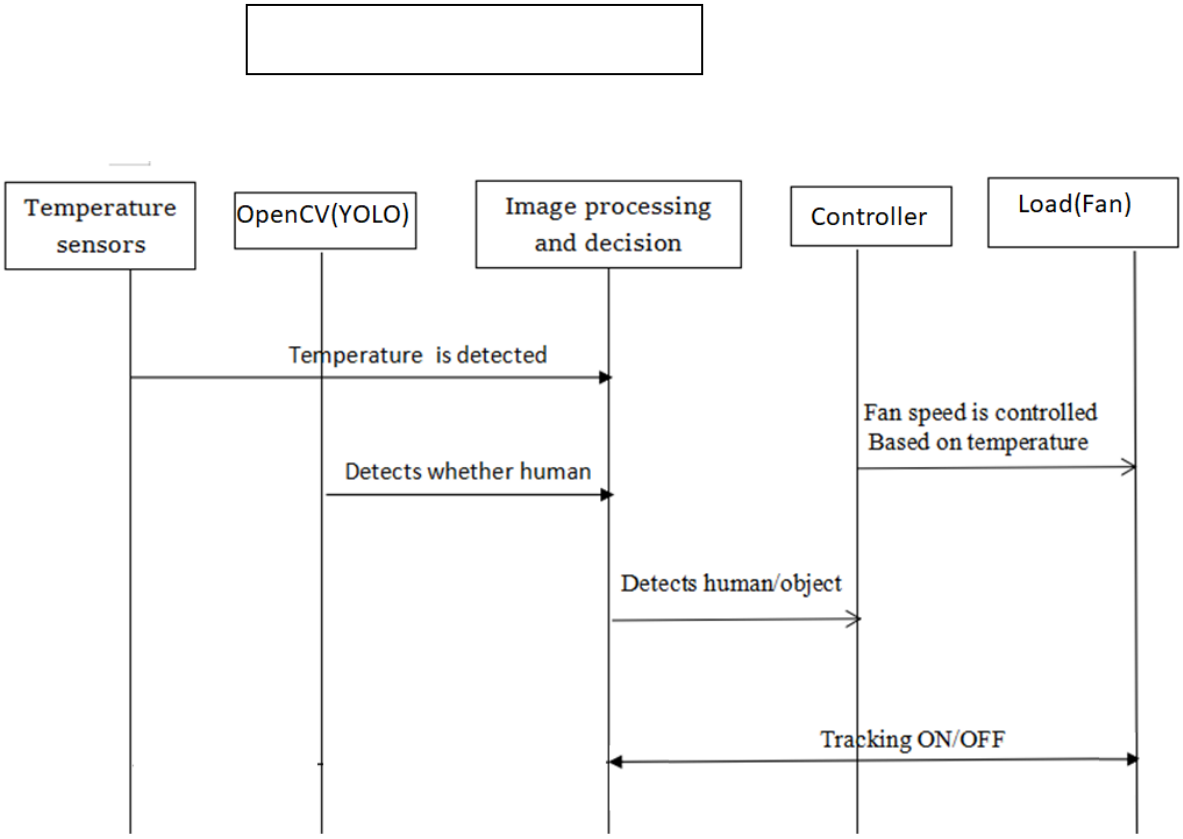


Message:

Messages written with horizontal arrows with the message name written above them, display interaction. The messages are written in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Object/Activation Box/Process:

Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message.



Sequence diagram of smart yolvo

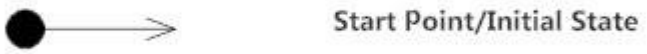
6.5 ACTIVITY DIAGRAMS

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language,

activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

Initial State or Start Point

A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swim lanes, make sure the start point is placed in the top left corner of the first column.



Activity or Action State

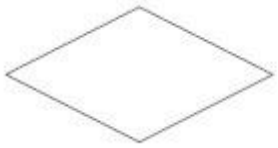
An action state represents the non-interruptible action of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

Action Flow

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.

Decisions and Branching

A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."

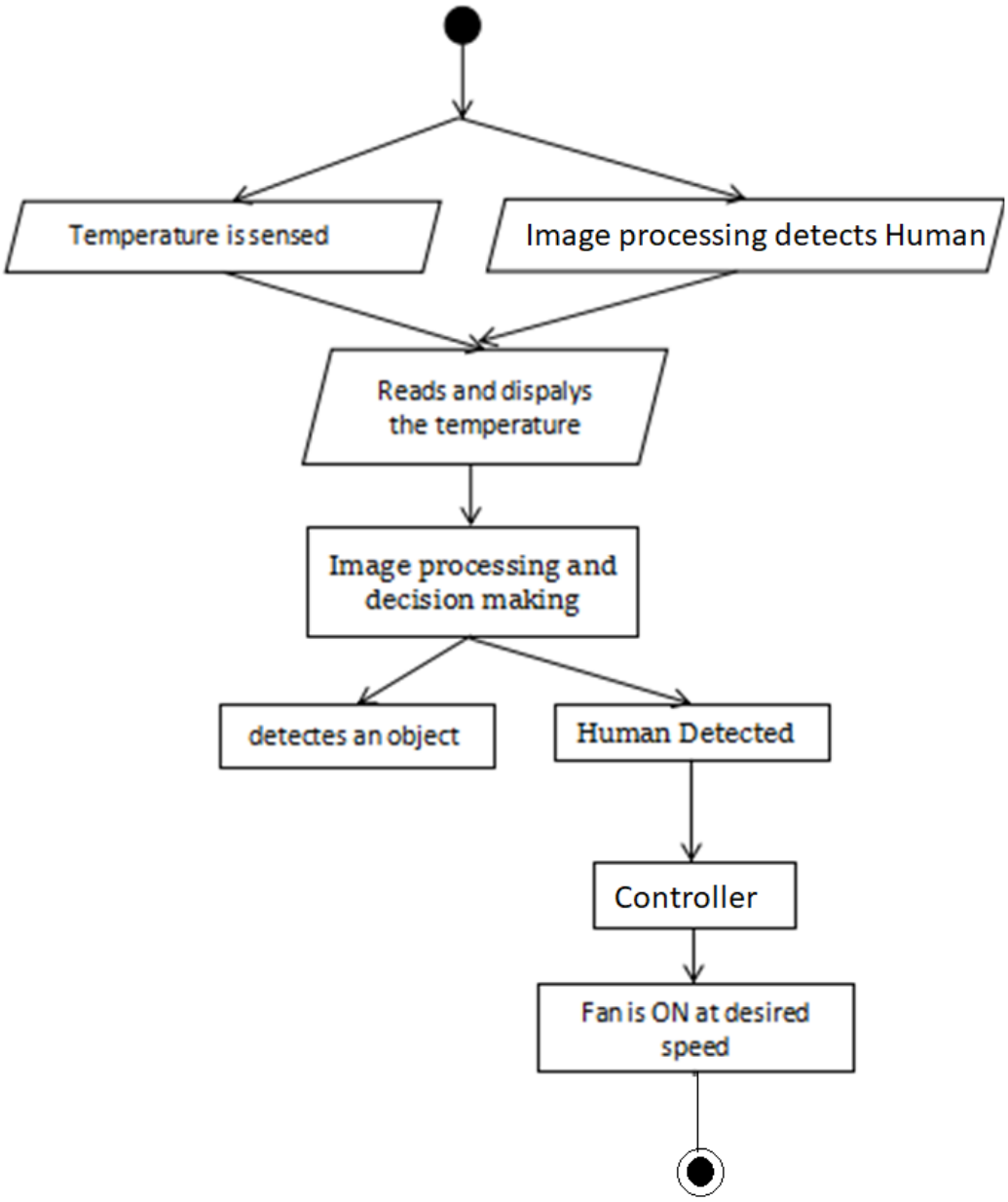


Decision Symbol

Final State or End Point

An arrow pointing to a filled circle nested inside another circle represents the final action state.

End Point Symbol



Activity Diagram of smart Yolov