PHASE 2
PROJECT DETECT

Overview
Architecture
    Edge Controller
    Central Database
    Web Application
Architecture - Implementation details
    Edge Controller
    Central Database
    Web application

Overview

This project will create a framework of detecting objects in an image, which will help users in automation of the requirements. The framework will not only store the detected data over a fixed period of time but also analyse it to provide intelligent information in the data.

Various approaches has been tried to do this using video analytic algorithms relying on computer vision, but since the input of this kind system is always changing the algorithms made for a certain customer may not apply for other customers. This asks for recurring work by engineers over each project. This becomes a bottleneck in scaling the product and leads to loss in revenues in business.

But this should not discourage engineers from using video as input for such kind of automation projects. Video camera as sensors are one of the best sensors which give us detailed information and it is available to us with evidence. We need an appropriate automation system which will enable automation in the calibration process from project to project. This will help in scaling a single product over the business and will result in exponential growth of business.

In phase 2, the emphasis will be on proving the technology of transferring data over web from edge controller to central database and central database to webapp, which also located in the web.

It is proposed to use Advanced Message Queuing Protocol (AMQP) for doing this end to end communication.

The AMQP is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security.

AMQP mandates the behavior of the messaging provider and client to the extent that implementations from different vendors are interoperable, in the same way as SMTP, HTTP, FTP, etc. have created interoperable systems. Previous standardizations of middleware have happened at the API level (e.g. JMS) and were focused on standardizing programmer interaction with different middleware implementations, rather than on providing interoperability between multiple implementations.[2] Unlike JMS, which defines an API and a set of behaviors that a messaging implementation must provide, AMQP is a wire-level protocol. A wire-level protocol is a description of the format of the data that is sent across the network as a stream of octets. Consequently, any tool that can create and interpret messages that conform to this data format can interoperate with any other compliant tool irrespective of implementation language.

Architecture

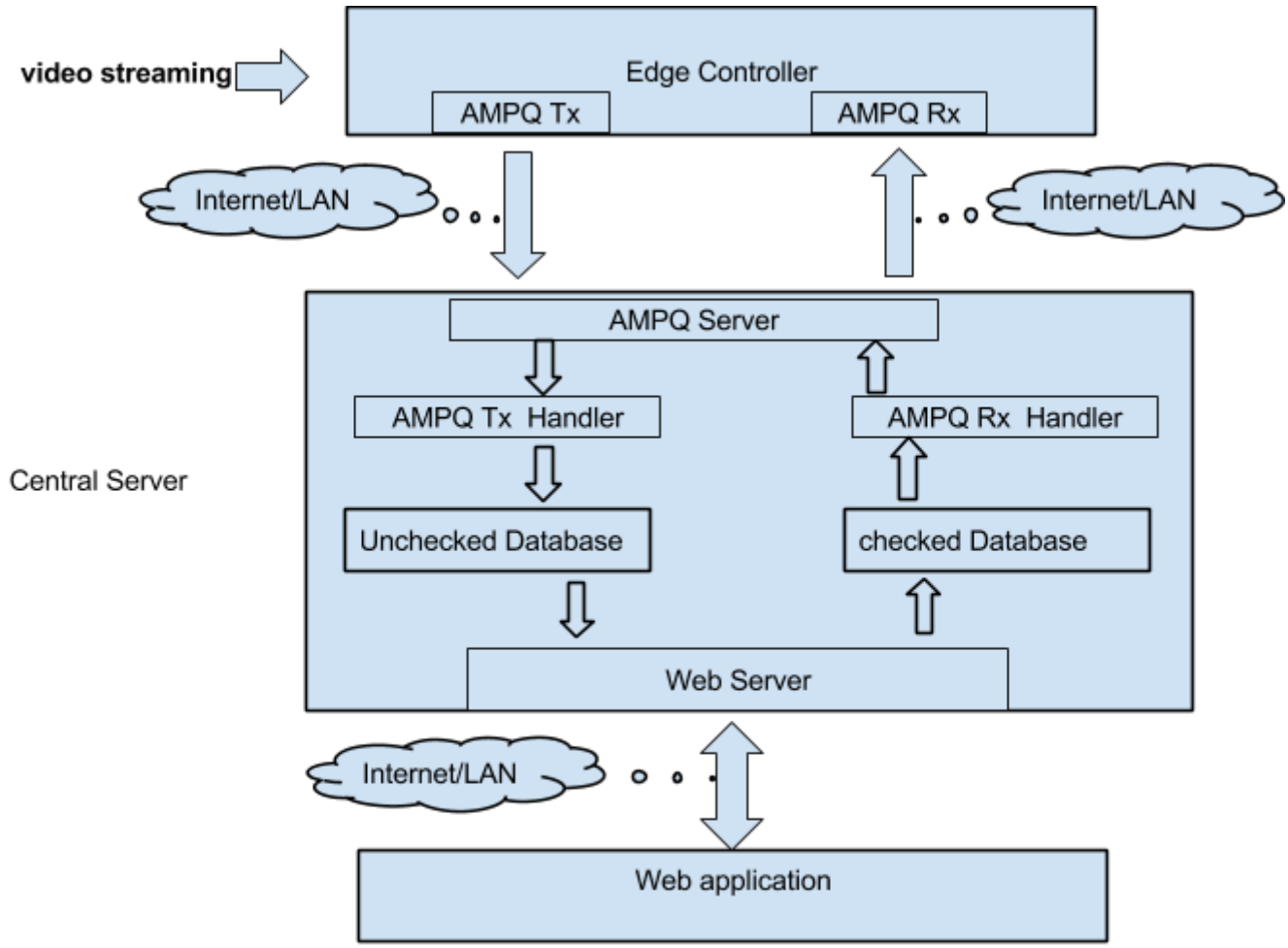


Figure 1: System Architecture

1. **Edge Controller**

This program will run at the client site. It can run on a user provided machine or on its own (manufacturer provided) native machine.

Following are the tasks this module will be doing:

step 1: acquire data from camera
step 2: run analytics on data
step 3: forward unchecked data using AMPQ to central db
step 4: receive checked data using AMPQ from central db
step 5: make some decisions, based on checked information
step 6: configure analytic
step 7: got back to step 1

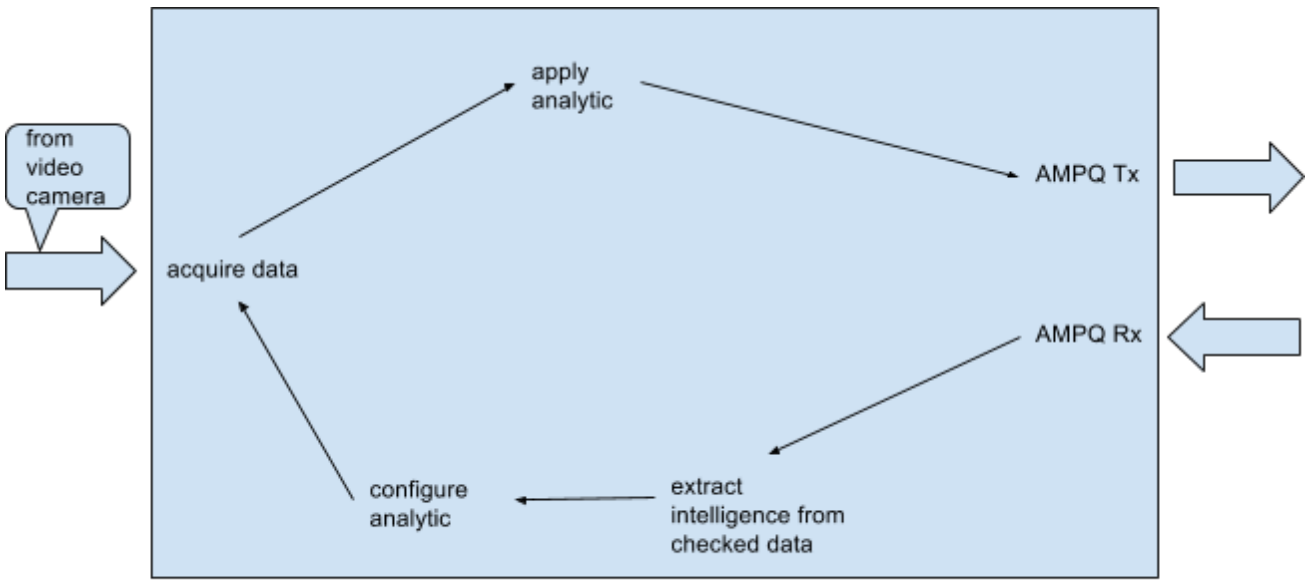


Figure 2: Edge Controller

1. **Central Database**

Central database will store data coming out (unchecked data) from edge controller and keep it stored for web application to work on it. It also stores the data (checked data) which is the output of web application.

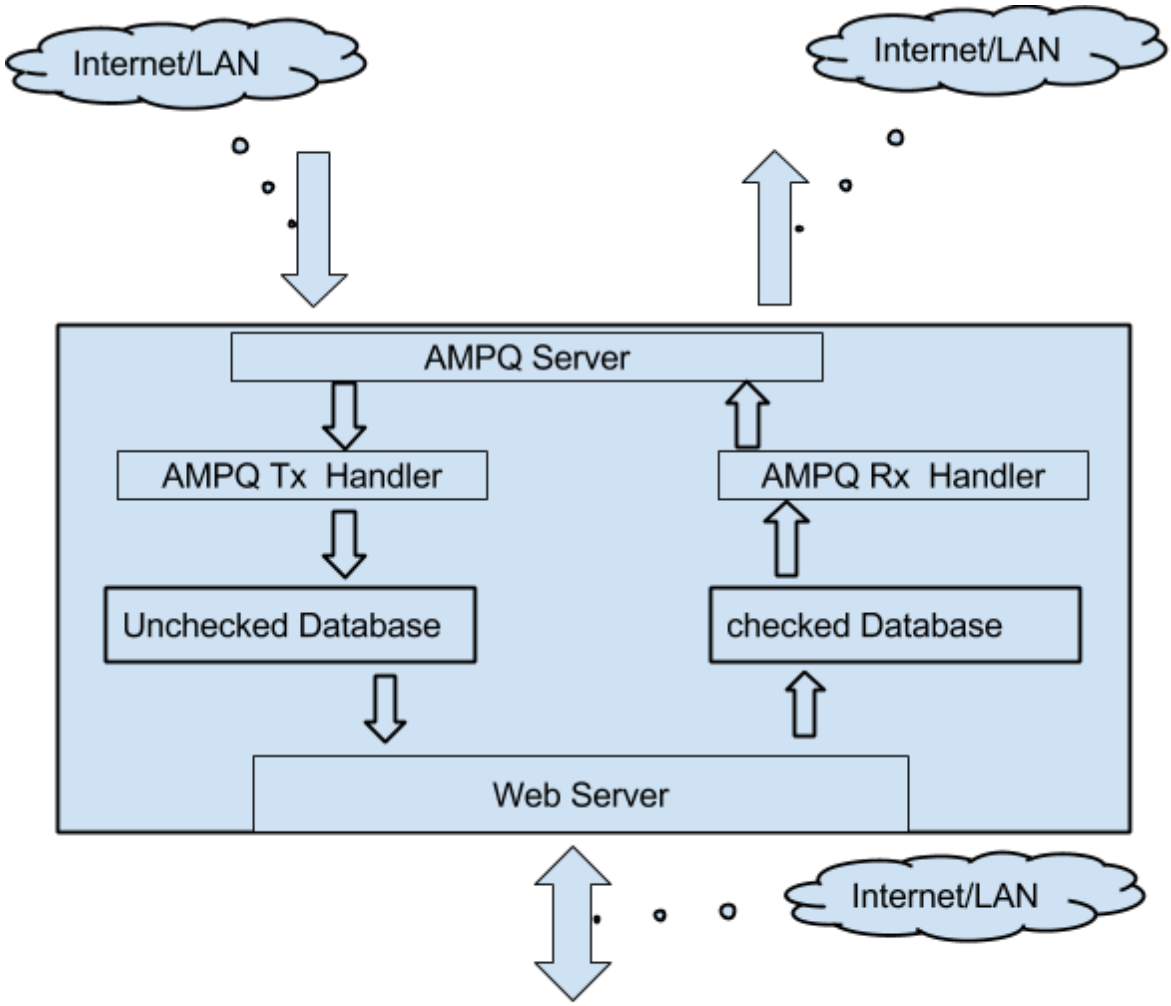
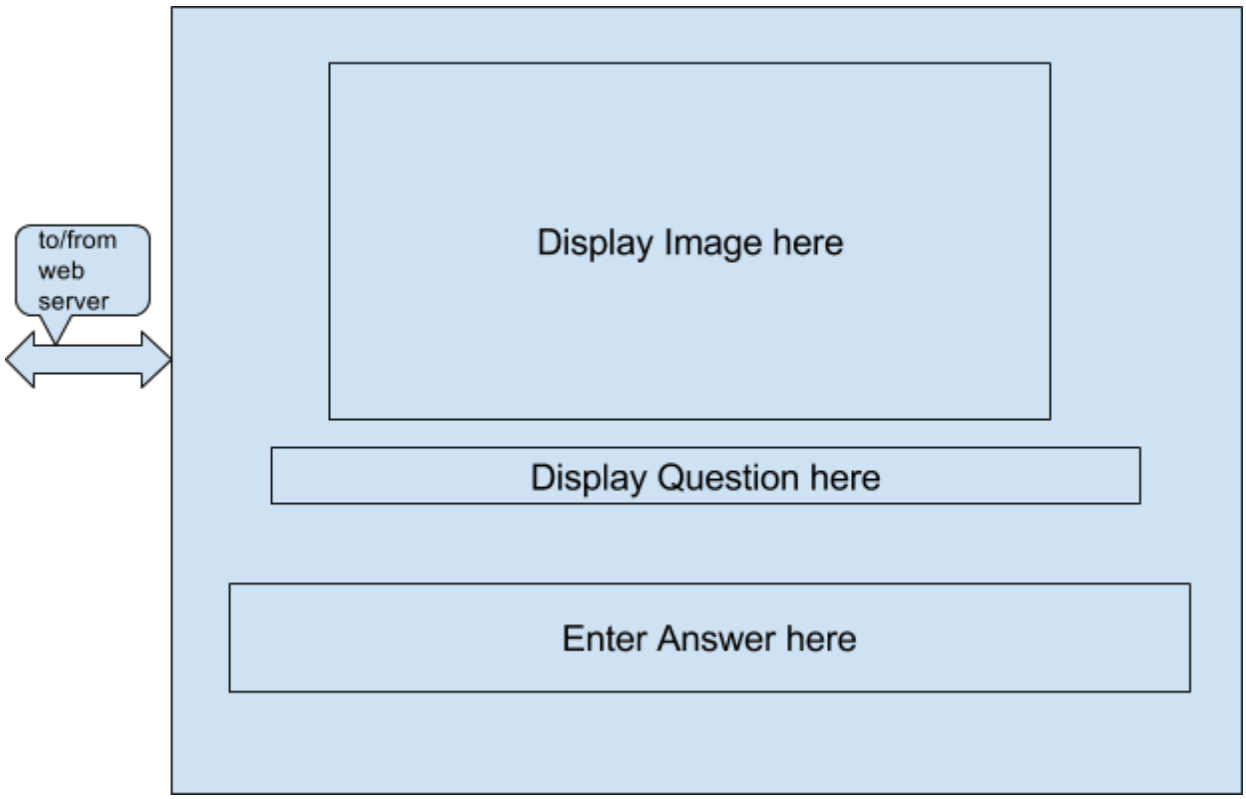


Figure 3: Central Server Architecture

1. **Web Application**

Web application will be a simple intuitive interface to database. It will enable user to look at the data and enter some information about the data. User will never came to know anything about the point of source of data. After user has given its input the data will be marked as checked and stored against the metadata of the same data.
Metadata constitutes the values like timestamp, edge device ID and analytic type.



Architecture - Implementation details

1. Edge Controller

- programming language : c++
- access video stream using opencv/ffmpeg.
- use opencv optical flow calculation (lucas kanade algorithm)
- Interface with MQ to deliver data to Central Database.
- Interface with MQ to receive data from Central Database

1. Central Database

- postgresql
- metadata storage (timestamp, analytic type, edge device ID)
- checked and unchecked data storage

1. Web Application

- python for development.
- no disclosure of edge device ID.
- no idea of analytic.
- intuitive for user to give input about information in data.
- iterates between unchecked data, till all unchecked data is checked.