**SmartFlow: Prioritize Emergency Vehicle and Traffic Rules Monitoring System**

**SmartFlow**

**Step1:** Start

**Step2:** Set up RSTP connection

**Step3**: If failed to read frame from RSTP stream go to step 11 else go to step 5

**Step4:** Initialize the data base connection

**Step5:** While true repeat steps from 6 to

**Step6:** Display live transmission from RSTP stream

**Step7:** Capture the frames from the RSTP stream at regular interval of time and save to database

**Step8:** If green delay is > 5 seconds go to step 7 else go to step 9

**Step7:** Call function EmergencyVehicle()

**Step8:** If emergency vehicle detected goto step12 else go to step 9

**Step9:** Call function DelayCalculation(img)

**Step10:** Update the green signal duration returned by DelayCalulation ()

**Step11:** Turn on Yellow Led indicating failed to read from RSTP stream and go to step 14

**Step12:** Turn on red led and update the green signal

**Step13:** If keyboard interruption go to step 12 else go to step6

**Step14:** Stop

**Module 1: Congestion control**

**DelayCalculation(img)**

**Step1:** Load VehicleDetector()

**Step2:** Initialize the source image and destination image path and speed

**Step3:** Read the image

**Step4:** Draw the boxes to vehicles

**Step5:** count the number of vehicles

**Step6:** Display the number of vehicles

**Step7:** Save the image with bounding boxes

**Step8:** Calculate the green delay

GreenDelay = no. of vehicles \* speed

**Step9:** Return speed

**VehicleDetector**

**Step1:** Load YOLO4 model weights and configuration files using OpenCV

**Step2:** Initialize the detection model using cv2.dn DetectionModel() and set input parameters, such as

Size and scale

**Step3:** Specify the classes that correspond to vehicles in the COCO dataset

**Step4:** detect\_vehicles(img)

**Step1:** Read the image

**Step2:** Detect the object in the image using YOLO4 model

**Step3:** Store the bounding boxes of the detected vehicles in a list

**Step4:** Return the list of boxes representing the detected vehicles.

**Module 2: emergency Vehicle detector**

**EmergencyVehicle**

**Step1:** Set the path to the Tesseract OCR engine

**Step2:** Load the pre-trained EAST text detection model (frozen\_east\_text\_detection.pb) using OpenCV's DNN module.

**Step3:** Read the image using from data base

**Step4:** Resize the input frame to a fixed size to match the input size expected by the EAST model.

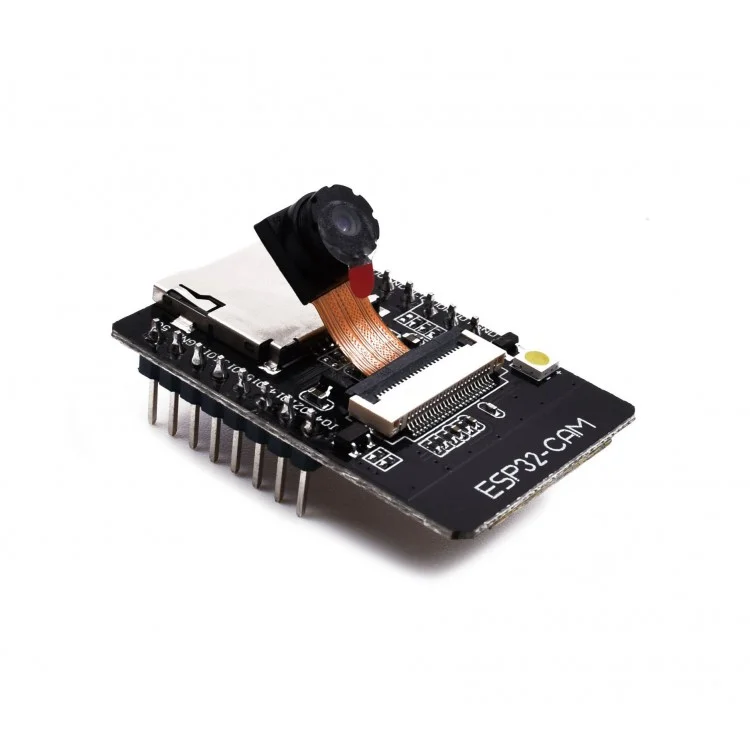
**Step5:** Convert the resized frame to a blob suitable for input to the neural network (cv2.dnn.blobFromImage()).

**Step6:** Pass the blob through the EAST model (net.forward()) to obtain predictions draw boxes.

**Step7:** Text Recognition (OCR) Extract the region of interest (ROI) from the original frame. Apply Tesseract OCR (pytesseract.image\_to\_string()) to recognize text within the ROI.

**Step8:** Return true if text is ‘AMBULANCE’ else false

**Architecture diagram**



**ESP32-Cam**

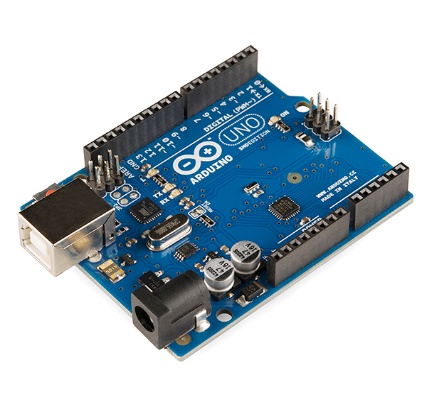


**Real Time Traffic**

**Emergency vehicle**



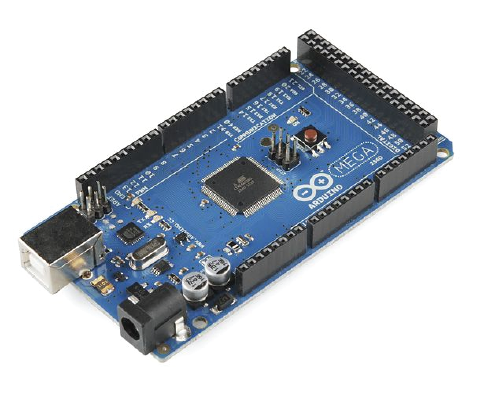
**Arduino uno**



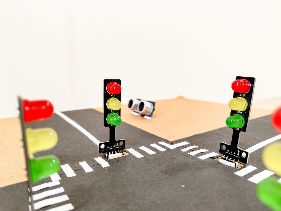
**Capture video**

**Capture image**

**Programing ESP32**



**Arduino Mega**

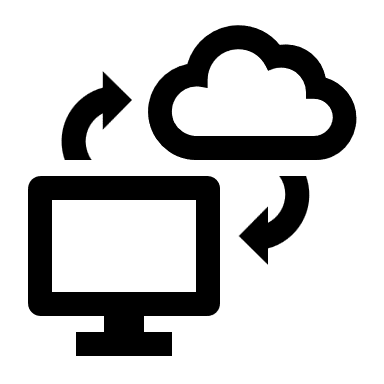


**Traffic Signal**

**Update the signal**



**Database**



**System**

**Save to Database**

**Emergency vehicle**

**Detected or**

**Load Green Delay to Arduino**

**Convert video to frames**

**Save or Extract image**

**Figure 1: Architecture diagram**

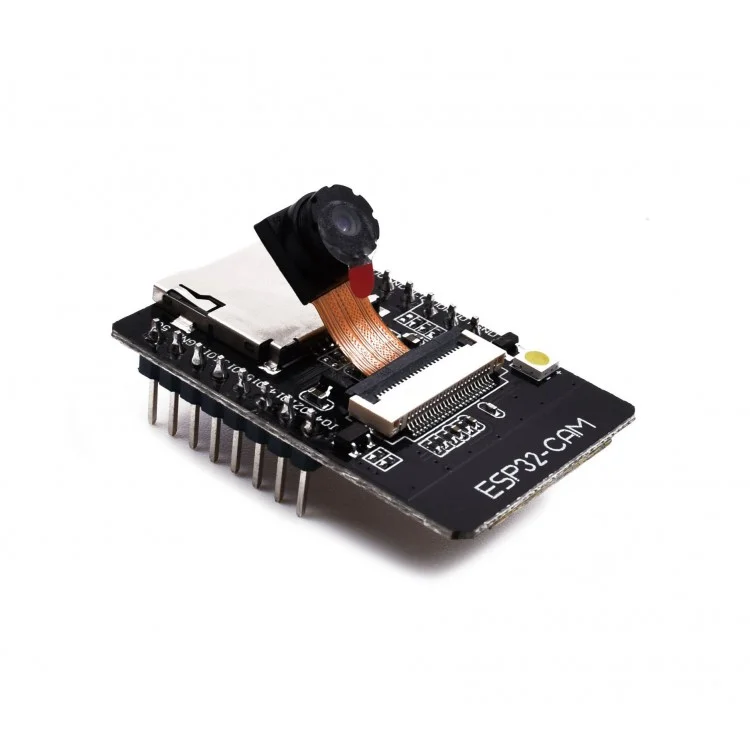
Above figure 1 represents the complete architecture of the SmartFlow system. The Arduino uno is used to program the ESP32 camera. Camera then captures the image of real traffic in one lane and it is stored in the database. Then that image is loaded to the YOLO algorithm to count the number of vehicles in the image or in a lane and based on the number of vehicles green delay is calculated by using formula speed\*number of vehicles hear speed is the average speed required by the vehicles to pass through signal. Using Arduino Mega traffic lights are controlled.

**Data flow diagram**



**Emergency vehicle detection**

**Capture Image**



**Send to server to detect vehicle density**



**Updating the signal**



**Calculating the green signal Dealy**



**Figure 2: Data flow diagram**

As shown in the above figure 2, ESP32-camera captures the image and store it in the server, then YOLO algorithm processes the image and detect the congestion and calculate the green signal timing whit help of formula average speed \* number of vehicles, according that calculation the traffic signal is controlled. If there is any emergency vehicle is detected in the image taken by camera then green signal is turned on for the emergency vehicle.

**Use Case Diagram of SmartFlow:**

**Capture image**

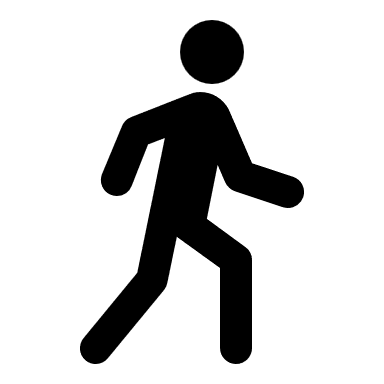
**Traffic Signal**

**Vehicle detection**

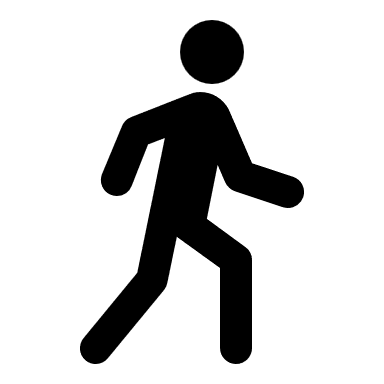
**Congestion control**

**Image processing**

**Camera Module**



**AI**



**Traffic Management**



**Emergency Vehicle Detector**



**Figure 3:** **Use case Diagram**

The figure 3 represents the use cases and actors in the SmartFlow system which can be described as follows

**Use Cases**

* **Traffic signal:** The traffic junction where the traffic lights are controlled by the traffic management system.
* **Capture image:** The cameras at junction takes images and send it to the system.
* **Image processing:** The AI algorithm processes the image to detect the number of vehicles and classify them.
* **Vehicle detection:** The AI algorithm uses data set and detect type and number of vehicles and detect the emergency vehicle.
* **Congestion control:** Traffic congestion, were there are more vehicles in one lane and a smaller number of vehicles in another lane leads to traffic.

**Actors**

* **Camera Module:** Captures the real time image or video to control the congestion based on the number of vehicles on a lane.
* **AI:** Artificial intelligence algorithm YOLO used to detect the vehicles and classify them to calculate the green delay.
* **Traffic Management:** The main system which controls the normal cycle of traffic and congestion control of the traffic, it uses the data given by the AI algorithm control the traffic by adjusting green delay.
* **Emergency Vehicle Detector:** The text detection algorithm is used to detect the emergency vehicle and inform system to turn on the green signal for emergency vehicle.

**State Chart Diagram of SmartFlow:**

**Real-Time Traffic**

**Capture Image**

**Vehicle Detection and** **Count Vehicles**

**Calculate Green Delay**

**Update the Signal**

**Ambulance Detection**

**Store In Database**

**Figure 4: State Chart Diagram**

As shown in the figure 4 the process starts with the system in an idle state, waiting to capture image in last 5 seconds of green delay. When the camera captures image of a lane, that image is stored in the database. The AI algorithm then retrieves these stored images to process them and count the number of vehicles in the lane. Based on the vehicle count, the system calculates the duration for the green signal in a lane. As the green signal period for a lane is about to end, the camera captures a new image of the next lane to prepare for the subsequent cycle.

Simultaneously, the AI algorithm continuously analyses the images to detect any emergency vehicles, such as ambulances. If an emergency vehicle is detected, the system immediately overrides the normal operation and activates the green signal for the lane with the emergency vehicle, allowing it to pass through without delay. After the emergency vehicle has passed, the system returns to its idle state, ready to capture the next set of images at the fixed interval, ensuring efficient traffic management and prioritization of emergency vehicles.

**Sequence Diagram**

**Update green signal**

**Capture image Save image to Database**

**Retrieve Image and count no of vehicles**

**Capture image Save image to Database**

**Detect Emergency vehicle**

**If emergency Update green signal**

**ESP-32**

**Data Base**

**AI**

**Traffic signal**

**Figure 5: Sequence Diagram**

As shown in the figure 5 camera captures the image and store it in the database, from database image is retrieved by AI algorithm to count the number od vehicles and calculate the green delay, this continues for all other lanes, when last few seconds of green delay camera takes picture of other lane. But the camera captures images of all lanes at fixed time of interval and that images are taken by the AI algorithm to detect the emergency vehicle this is continuous process so that emergency vehicles like ambulance will not suffer from traffic when the ambulance is detected the green signal is turned on.

**Module 1: Congestion control**

Load Vehicle detector

Load YOLO Weights

Read COCO dataset

Return Green Delay

Draw Boxes

Count the vehicles

Calculate Green Delay

Read Image

Detect the vehicles

Figure 6**:** **Congestion control**

The above figure 6 represents the block diagram of module one congestion control. As shown in the figure Vehicle detector algorithm is loaded which uses YOLO weights and coco dataset to detect the vehicles in the image which was extracted from the database. Then using cv2.dn DetectionModel() vehicles are detected and boxes are drawn around the vehicles and number of boxes are returned. The number of boxes represents the total number of vehicles in the lane using that green delay is calculated and returned.

As shown in the figure 6 for congestion control, the Vehicle Detector Algorithm plays a pivotal role in managing traffic flow efficiently. This algorithm, integrated with YOLO weights and the COCO dataset, utilizes deep learning capabilities to detect vehicles within images extracted from the database. Employing OpenCV's cv2.dnn DetectionModel(), the algorithm performs vehicle detection by drawing bounding boxes around identified vehicles and counting these boxes to determine the total number of vehicles in the lane. This count forms the basis for calculating the green delay required for the traffic signal, considering factors such as average vehicle speed. The calculated green delay value is crucial for dynamically adjusting traffic signals via systems like the Arduino Mega, ensuring optimal traffic flow management in real-time scenarios. This process underscores how advanced image processing and machine learning techniques enable the SmartFlow system to respond dynamically to traffic conditions, thereby enhancing road efficiency and safety.

**Module 2: Emergency Vehicle Detection**

**Return True**

**Else Return False**

**If Ambulance**

**Read Image**

**Resize the Image**

**Set path to OCR**

**Text Detection and Draw boxes**

**Load text detection model**

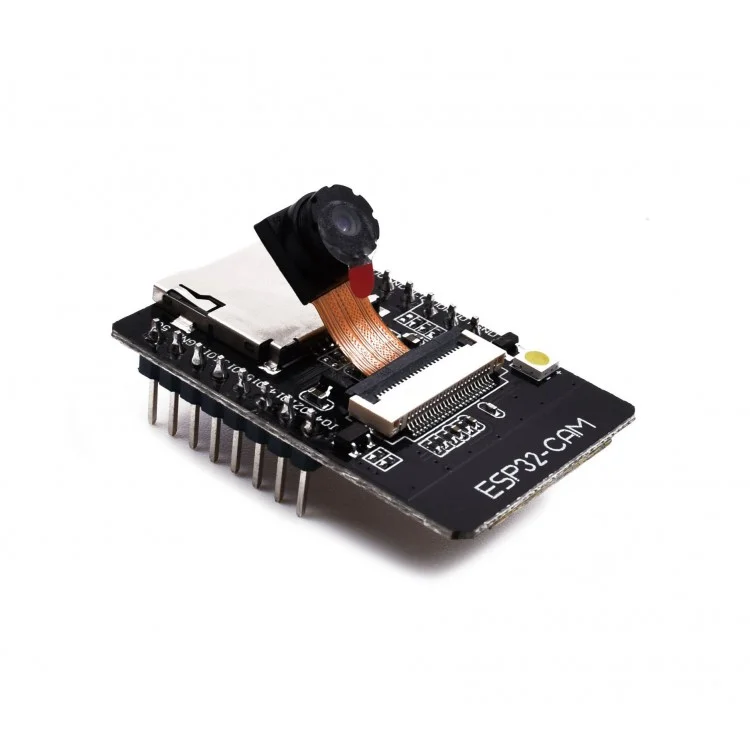
**Recognize Text Within ROI**

**Figure 7: Emergency Vehicle Detection**

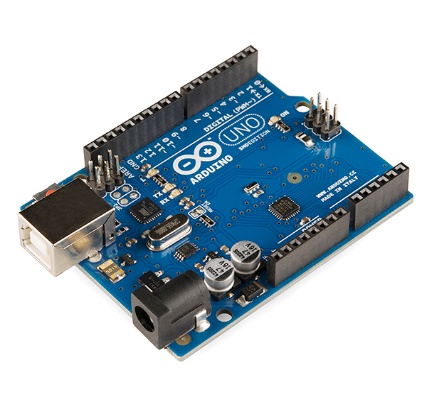
As shown in the figure 7 the emergency vehicle detection model works as follows, it uses the text detection model frozen\_east\_text\_detection.pb which is pre trained model for text detection. When image is extracted from the database is resized as required by the model and it is passed into model for text detection when it detected the text draw the boxes around it then Text Recognition (OCR) Extract the region of interest (ROI) from the original frame. That detected text is stored in a variable to check whether it is AMBULANCE or not. If it is ambulance return true else returns false.



**Emergency vehicle**



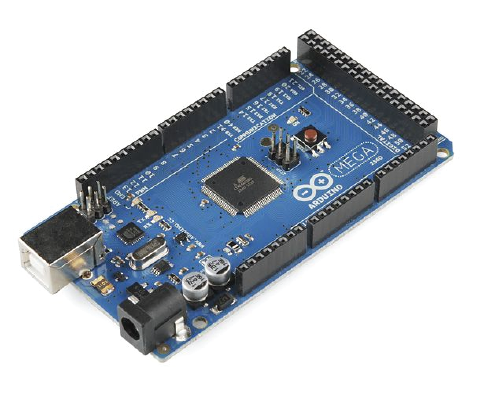
**ESP32-Cam**



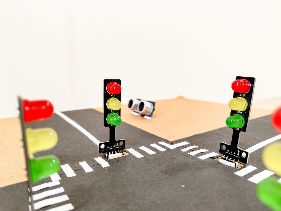
**Arduino uno**



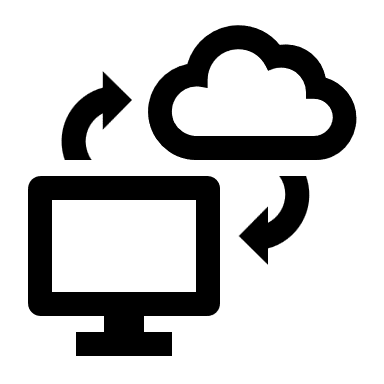
**Database**



**Arduino Mega**



**Traffic Signal**



**System**



**Real Time Traffic**

**Capture image**

**Program the ESP32**

**Save to Database**

**Emergency vehicle**

**Detection**

**Capture image/ video**

**Load to Arduino**

**Updating the signal**

**Load image**