International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 9, Issue 4, April-2024, http://ijmec.com/, ISSN: 2456-4265

INTELLIGENT TRAFFIC LIGHT CONTROL SYSTEM

M.Varshini[1] , D.Nikhitha[2], D.Akhil[3] , Dr. T. Venkat Rao[4], Dr. S N Chandra Shekhar[5]

[1,2,3]B.Tech Students, Electronics and Communication Engineering, Sreenidhi institute of science and Technology, Hyderabad, India

[4]Associate Professor, Electronics and Communication Engineering, Sreenidhi institute of science and Technology, Hyderabad, India

[5]Assistant Professor, Electronics and Communication Engineering, Sreenidhi institute of science and Technology, Hyderabad, India

***Abstract*—**Traffic is a major problem in highly populated countries like India. The present traffic system assigns signals for a fixed interval of time. This creates a problem when there are more vehicles on one side. Assigning a green signal to an empty road is also of no use. The other problem is with emergency vehicles. Delay of these vehicles may result in the loss of some lives. We are proposing a model to solve all these problems that dynamically assigns signal timing based on vehicle density. This model gives priority to emergency vehicles. We place a camera at the traffic signal to take images of all directions, which are sent to Raspberry Pi. These images are processed using the most advanced vehicle density on each lane are provided via real-time object identification technology dubbed YOLO ONLY LOOK ONCE (YOLO), which is based on a sophisticated neural network. This reduces waiting time which intern reduces fuel wastage. In the future, we can extend this model to multiple traffic lights and make a network. The information can be passed from one another and congestion can be easily avoided by getting prior information.

I. INTRODUCTION

Traffic bottlenecks are a key challenge in expanding cities, making it difficult to understand where traffic density is greater in real time to design better traffic signal management and routing. Traffic congestion caused by insufficient road width, weather-related road conditions, unregulated demand, considerable red light delays, etc. might be the major reason.

Traffic has little effect on signal green times, even if demand and capacity are often linked. To fulfill rising demand and save human labor, traffic control must be simulated and optimized. Image processing is being utilized in safety and surveillance technologies and traffic and vehicle management to educate travelers. Traffic density may be estimated using image processing. Transportation is a major issue in most countries. Since highways are designed to manage a certain amount of traffic, a rising population and economy make it harder to maintain traffic.

Traffic control alternatives include constructing new roads or enlarging existing ones, which require time. The Most cities utilize a density-based traffic light system [1] that counts automobiles in each lane to determine green time. The density is estimated by sensors and does not offer an exact count of automobiles. Thanks to image processing and machine learning, the green time for a junction lane may be estimated from a photograph and an estimate of the number of automobiles in the lane.

The YOLO Framework is used to assess vehicle traffic and determine green times for each junction lane in this

DOI: https://doi-ds.org/doilink/05.2024-51388889

research. YOLO recognizes things in real time using convolution neural networks. Some models need numerous passes, whereas YOLO just needs one. This makes YOLO quick and suitable for real-time applications. This project utilizes a Raspberry Pi and camera to take real-time photos. For demonstration, Raspberry Pi runs the trained YOLO model.

II. REVIEW OF LITERATURE

The current technology identifies the vehicle density using a PIR-based approach. The Raspberry Pi 2's microprocessor controls the traffic light's LED lights in accordance with the timings that were previously defined with Python code. Depending on the output of the Pir motion detector sensor, which delivers a high signal to the microcontroller when it detects a change in infrared radiation via its line of sight. PIR detects the length to which vehicles are present. Based on this traffic signals will change. All of the red LEDs are soldered in parallel on the PCB board. When the traffic light turns from red to green, it will flash

Four terminals make up the common cathode RGB LED display, which is soldered in parallel on the PCB board. The longest terminal is the common cathode, while the other three terminals are responsible for emitting red, green, and blue light correspondingly, blue. The RGB LED will light up with the appropriate color dependent on the PWM signals that the microcontroller sends to each of these terminals and the work cycle that is applied to each terminal's PWM signal. Using an Arduino UNO and the MATLAB environment, the authors construct image processing-based traffic light control. The first camera on the system is linked after the webcams are chosen in order and sent data by the Arduino to the first webcam on the system. - Energy-Efficient Intelligent Street Lighting System Using Traffic-Adaptive Control", IEEE paper

There are many other methods that are being employed in the outer world. In some papers, they used RFID tags to count the number of vehicles. They fixed RFID tags for all the vehicles. We have similar other methods using ZIGBEE modules. These ZIGBEE modules can be used as transmitters and receivers. All the vehicles will transmit and the receiver near the traffic pole will count vehicles based on the received signals. These methods are highly difficult to implement in the real world as it is highly impossible to add hardware to all the vehicles - Traffic Control System using Zigbee Module -IJESC Article

Some papers suggested using some image processing and object detection techniques like HAAR, HOG, etc.

-Real-time detection of vehicles using the Haar-like

features and artificial neuron networks by A. Mohamed, A. Issam, B. Mohamed, and B. Abdellatif

. The accuracy of these models is very less when compared with YOLO.

III. PROPOSED SOLUTION

We proposed a model in which we used image processing combined with AIML for vehicle detection. Our program will intelligently distribute the amount of time that each road has the green light based on the traffic volumes on all roads. Since cameras are far less expensive than other equipment like sensors, we have used image processing to calculate traffic density. The suggested model is made in the following manner: Four sets of LEDs that stand in for traffic signals are linked to a Raspberry Pi. It involves keeping track of the volume of

DOI: https://doi-ds.org/doilink/05.2024-51388889

traffic on either side and adjusting the signal to account for it.

3.1.Block Diagram

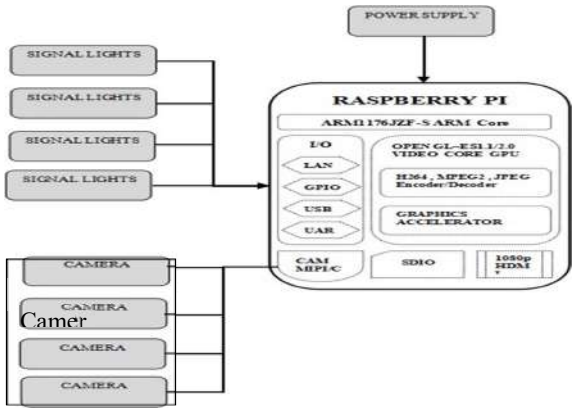


Figure 3.1.1: Block diagram

3.2.BLOCK DIAGRAM DESCRIPTION

- Four cameras are connected to the raspberry pi in every direction.
- Lights in each signal are connected to the GPIO pins of Raspberry Pi.
- Power for the Signal light can be given using an external power supply using relay switches (Here we are using LED's as signal lights)
- The power supply should be given to raspberry pi.

IV. HARDWARE SPECIFICATIONS

4.1.RASPBERRY PI

Figure.4.1.1: Raspberry Pi

Electronic experts and amateurs use the Raspberry Pi, an ARM-based device. One strong computer uses minimal electricity. Raspberry Pi is called Mini Compute in Your Palm because it can multitask like a PC due to

DOI: https://doi-ds.org/doilink/05.2024-51388889

its processing power and memory.

It runs all ARM GNU/Linux distributions and Windows 10, which will be discussed later, due to its ARMv7 CPU. The ARM architecture influences modern technologies. We travel using ARM-based CPUs and controllers. Mobile phones, iPods, PCs, and other devices use ARM Cortex CPUs. Pi is great for "Net of Things" visualization. This webinar will cover Pi's hardware, software, and operating system setup for the first instance. More popular than any other Raspberry Pi board is the Raspberry Pi 2 Model B. Additionally, the Raspberry Pi 3 Model B is available. It's like the Raspberry Pi 2 but features a faster CPU and Wi-Fi. A short summary of "Raspberry Pi 3 B Model B's" characteristics follows.

Figure 4.1.2 : Raspberry Pi Ports

Raspberry Pi is a 40-GPIO pin board for teaching and experimentation. The 64-bit CPU runs at 1.2GHz.It features 1GB RAM. The Raspberry Pi has two USB 2.0 ports. These connectors accept any USB device, including mice and keyboards. We'll explain later. The first screen requires a mouse and keyboard. Figure shows found USB connectors.

Raspberry Pi 3 has one Ethernet port. Connect the Raspberry PI 3 to the internet. You may transmit data between your PC and PI3 utilizing this Internet tool.

A 3.5mm jack connects headphones for PI music playback. Use it as a video jack. LCD or LED screens may be connected to the PI via one HDMI port. CPU graphics are good. A USB port for a microphone powers the device. The board will malfunction if voltage fluctuations occur here. LCD panels may be replaced with 3–7-inch touch displays. Our internal port can connect a touch screen. The cavia module may be connected to PI without our method. Powerful modulators and GPIO pins are present. Use the GPIO pins whatever you like. This board has Bluetooth and Wi-Fi. This helps us advance this project.We merely used its SD card port to upload our code to Raspberry Pi. Some pins have supplementary functions; we'll discuss them later.

DOI: https://doi-ds.org/doilink/05.2024-51388889

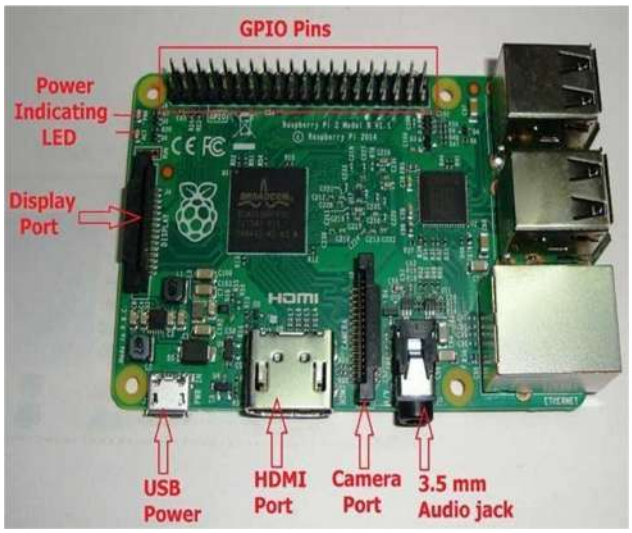


Figure 4.1.3 : Raspberry Pi architecture

Powerful modulators and GPIO (Generic Purpose Input Output) pins are both present. The GPIO pins can be used any way we like. Some of these pins also have extra functions; we'll speak about those later.

V.THEORETICAL ANALYSIS

5.1.OPEN CV

Intel's Open Source Computer Vision Library API may be used in many image processing and computer vision applications. OpenCV, developed by Intel Research to accelerate CPU-intensive applications, was released in 1999. OpenCV contains classes, C/C++ algorithms, and computer vision and image processing approaches. C, C++, Python, Ruby, Matlab, and others are getting interfaces. OpenCV prioritised real-time applications and processing speed. C was used to create OpenCV for multicore CPUs. More than 500 OpenCV routines address industrial product inspection, robotics, user interface, camera calibration, security, and medical imaging. The library's founding principles seek to allow commercial applications of computer vision in human-computer interaction, robotics, monitoring, biometrics, and security by providing a free and open infrastructure to unify and advance vision community efforts. OpenCV's vision capability extends beyond movie and image input, display, and storage. OpenCV's purpose is to create a user-friendly computer vision infrastructure that lets anybody easily construct reasonably advanced vision applications. Early OpenCV targets include: Looking ahead We can help research by offering open and efficient visual infrastructure code. Giving developers a uniform base may increase code readability and portability. This helps spread vision knowledge. Promotion of performance-optimized, portable code for commercial vision-based applications without requiring open or free commercial products. OpenCV offers computer vision and image-processing tools. The features are optimized for Intel architecture CPUs and benefit from MMX technology. The OpenCV Library can help create an open-

DOI: https://doi-ds.org/doilink/05.2024-51388889

source vision community to take advantage of computer vision applications on PC and mobile platforms. Library access is now available.

Transportable OpenCV. Intel, Visual Studio C Borland C++ compilers were used for initial compilation. C and C++ code must meet requirements for crossplatform programming.

assist easier. OpenCV works on Linux and Windows.

Real-time OpenCV computer vision applications are becoming increasingly frequent. Application examples include motion trackers, object, and face recognizers. The library is popular among computer vision scholars. Academics may quickly establish research or demonstration projects using the enormous library of current algorithms.

Many people use "image processing" and "computer vision" interchangeably. Image processing involves low-level alteration of static or moving pictures.

OpenCV was created to enhance computer vision research. OpenCV provided crucial tools for computer vision challenges. A high-level library will address increasingly complicated computer vision challenges. Basic library components may be combined to produce a complete solution for most machines.

5.2.YOLO: Object Detection Algorithm Expounded

What is the operation of the YOLO architecture? After learning about the different YOLO algorithm iterations, start creating your own object detection models.

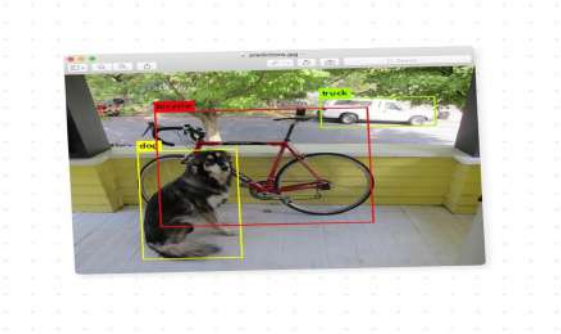


Figure 5.2.1: YOLO Example1

A common task in computer vision is object detection.It deals with identifying a region of interest inside an image and classifying this region in a manner similar to how a typical image classifier would. A single image may include a number of interesting details referring to diverse themes. Object recognition hence becomes a more difficult problem in image classification. The YOLO (You Only Look Once) technique is a popular method for quickly and precisely identifying objects. Following the first announcement by Joseph Redmon et al. in 2016, the most recent version, YOLO v7, became accessible. We will talk about YOLO v7's unique qualities in this post and contrast them with other object-detecting methods. With V7, you can quickly complete any computer vision task and train ML models .Don't begin with nothing. Explore the 500+ Try out V7's tools while using open datasets in our repository. prepared to hasten implementation straight away.

How is object detection carried out?

The two main categories that object detection algorithms belong to are singleshot detectors an Identifying and locating objects in images and videos is a challenge for computer vision. It is an important part of many

DOI: https://doi-ds.org/doilink/05.2024-51388889

applications, including as robotics, self-driving d two-stage detectors.

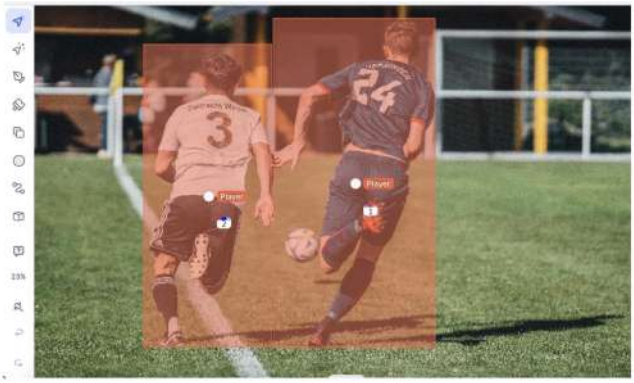


Figure 5.2.2: YOLO Example2

One of the earliest attempts to use deep learning to solve the object recognition problem was the R-CNN (Regions with CNN features) model, developed in 2014 by Ross Girshick and his coworkers at Microsoft Research. Our model used convolutional neural networks (CNNs) with region proposal approaches to locate and discover objects in images.

Object identification techniques may be broadly categorized into two categories based on how frequently the same input image is delivered over a network

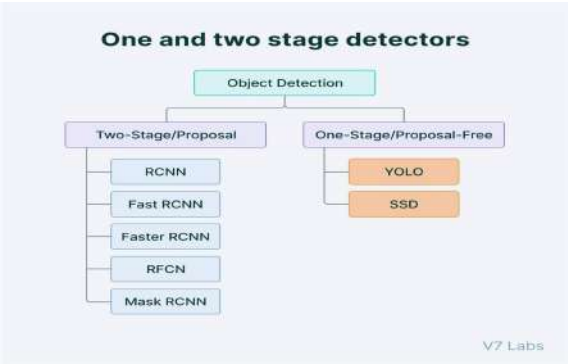
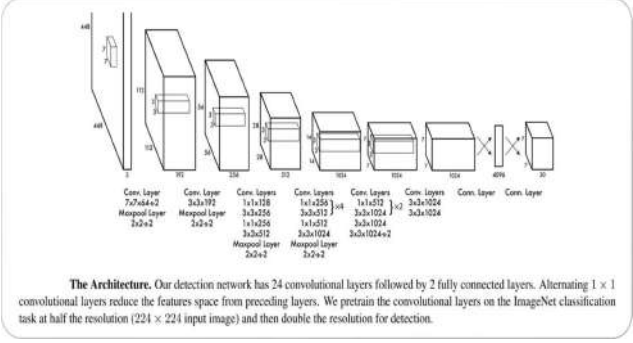


Figure 5.2.3: One and Two-stage Detectors

5.3.YOLO ARCHITECTURE:



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

DOI: https://doi-ds.org/doilink/05.2024-51388889

Figure 5.3.1:YOLO architecture

A simple deep convolutional neural network is used in the YOLO (You Only Look Once) approach to detect things in a picture. This CNN model was pre-trained using millions of annotated images from ImageNet. The model's first 20 convolutional layers are pre-trained on ImageNet with a temporary average pooling and fully connected layer. Pre-training helps the network learn generic features like edges, corners, and textures for object detection.

After pre-training, YOLO detects objects. Studies show that convolutional and connected layers improve pretrained network performance. The YOLO architecture extracts features from images using convolutional and pooling layers. Transferring these attributes via a fully connected layer that predicts bounding box coordinates and class probabilities is next.

For object detection, YOLO divides the input picture into a SxS grid. Grid cells must recognize objects with centers within them. For each grid cell, the YOLO model predicts B bounding boxes and confidence ratings. These How precise and definite the model is that the projected box includes an item is shown by its confidence ratings. In addition to bounding boxes and confidence ratings, YOLO anticipates class probabilities for each box. YOLO predicts for each box the chance that an item within it belongs to each training dataset class. This lets the model recognize, classify, and identify items. Fastness is an advantage of the YOLO algorithm. YOLO can analyze photographs quickly on low-powered devices since detection is done in a single network feedforward pass. It is used for real-time object identification in surveillance systems and self-driving cars. Another feature is YOLO's overlap management. YOLO can handle overlaps because each grid cell identifies objects whose centers lie within it. YOLO may distinguish two things standing close together. YOLO uses a simple deep convolutional neural network to identify items in a photo. YOLO quickly and accurately identifies objects by dividing the input image into a grid and anticipating bounding boxes and confidence ratings for each grid cell. It manages overlapping elements, making it popular in real-world applications.

YOLO must anticipate various bounding boxes for each grid cell to detect items in a photo. During training, just one bounding box predictor should handle each object. YOLO uses "assigning responsibility," where it assigns one predictor to forecast an item based on the prediction with the highest current IOU (Intersection Over Union) with the actual data.

This technique helps bounding box predictors specialize in sizes, aspect ratios, and item classifications. This specialization increases the model's recall score by increasing the likelihood of using the correct bounding box. During training, YOLO assigns each item to the bounding box predictor with the highest current IOU. Each predictor is entirely responsible for a given selection of items, which eliminates ambiguity in the training data and improves object identification accuracy.

Specialization between bounding box predictors improves the YOLO model. By training each predictor to concentrate on a certain category of objects, the model may enhance object identification accuracy and recall. Nonmaximum suppression (NMS) post-processing improves YOLO model object detection. It removes unnecessary or inaccurate bounding boxes for a photo object. Predicted bounding boxes are compared to the

DOI: https://doi-ds.org/doilink/05.2024-51388889

highest-rated reference box based on confidence ratings. The box with the lower confidence score is removed when the overlap exceeds a certain amount. NMS does this until all reference boxes are processed. Only the most accurate and representative bounding box is created for each picture item. This crucial step of the YOLO algorithm improves object detection accuracy and efficacy.

5.4.VERSIONS OF YOLO:

Figure 5.4.1: YOLO Timeline

YOLO v2:

to view objects at different sizes, this helps to enhance the detection performance for small items. Overall, YOLO v3 is more precise and reliable than YOLO v2 and can handle a larger variety of item sizes and aspect ratios

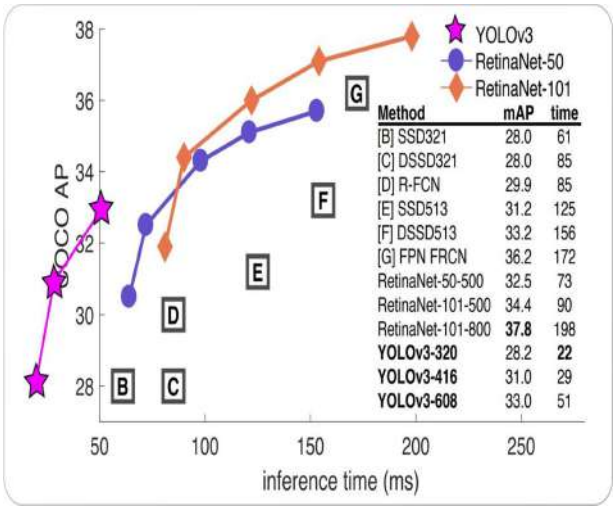


Figure 5.4.3 : Comparison of the results obtained by YOLO v3

YOLO v4

The YOLO object identification algorithm underwent four iterations before being released as YOLO v4 in 2020 by Bochkovskiy et al. with the intention of enhancing YOLO v3. The introduction of a new CNN architecture

DOI: https://doi-ds.org/doilink/05.2024-51388889

dubbed CSPNet (Cross Stage Partial Network), a variation of the ResNet architecture that has been especially created for object identification tasks, is the key advancement in YOLO v4. Despite having a structure that is very shallow and only has 54 convolutional layers, CSPNet is nonetheless able to get cutting-edge results on a variety of object detection benchmarks. YOLO v4 not only features a new architecture but also a number of additional enhancements, such as the utilisation of a bigger input size, enhanced data augmentation, and sophisticated training methods including self-adversarial training and focus loss. In addition, the YOLO v4 algorithm makes use of a more sophisticated post-processing method known as "weighted boxes fusion" (WBF), which boosts object recognition precision by integrating overlapping bounding boxes. With these upgrades, YOLO v4 becomes one of the most sophisticated and commonly used object detection algorithms available today, performing much better than YOLO v3 in terms of speed and accuracy.

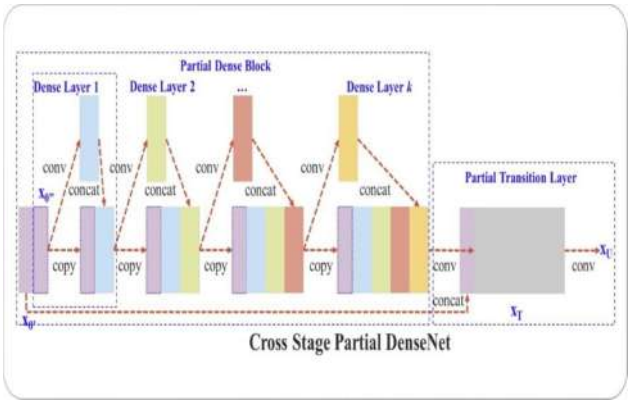


Figure 5.4.4 :Architecture of CSPNet

To improve object identification precision, YOLO v3 and v4 use anchor boxes with various sizes and aspect ratios. To create these anchor boxes, YOLO v4 employs a unique method dubbed "k-means clustering." In this method, a clustering algorithm is used to aggregate the ground truth bounding boxes into clusters, and the centroids of these groupings are then used as anchor boxes. Better detection performance is obtained as a result of the anchor boxes being able to line more precisely with the size and shape of the objects.

Both variants employ a comparable loss function in addition to the anchor boxes for model training. A new word, "GHM loss," which is a variant of the focal loss function, is nonetheless included in YOLO v4. It improves data quality and is especially helpful in resolving skewed datasets. the general performance of the model. The FPN architecture utilised in YOLO v3, which is intended to identify objects at various sizes, is also enhanced in YOLO v4.

DOI: https://doi-ds.org/doilink/05.2024-51388889

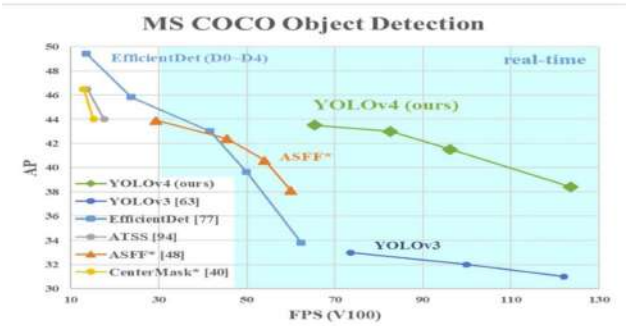


Figure 5.4.5 : Comparative performance of YOLO v4

YOLO v5

The creators of the original YOLO algorithm unveiled YOLO v5 in 2020 as an opensource initiative run by Ultralytics with the goal of enhancing object recognition performance. The fact that YOLO v5 uses the more complex EfficientDet design, which is based on the EfficientNet architecture, is one of the main differences between YOLO and YOLO v5. In comparison to its predecessors, YOLO v5 is able to achieve higher accuracy and better generalisation to a larger range of object categories by utilising a more complicated architecture.

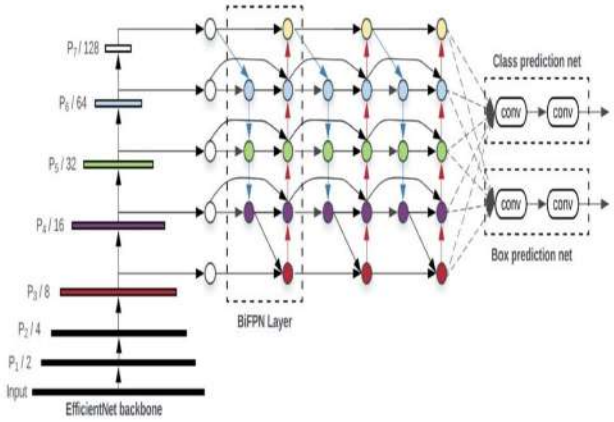


Figure 5.4.6 : Architecture of the EfficientDet model.

There are numerous differences between YOLO and YOLO v5, including their training data and the process for creating anchor boxes. YOLO was trained using the 20-category PASCAL VOC dataset, but YOLO v5 was trained using the 600-category D5 dataset, which is a bigger and more varied dataset. In addition, YOLO v5 makes use of a more complex architecture called EfficientDet, which is based on the EfficientNet design. This architecture enables higher accuracy and better generalisation to a larger variety of object categories.

Additionally, "dynamic anchor boxes" is a brand-new technique for creating anchor boxes that is included in YOLO v5. By clustering the ground truth bounding boxes and using their centroids as anchor boxes, this technique produces anchor boxes that are more closely matched in size and form to the identified objects. The pooling layer known as Spatial Pyramid Pooling (SPP), which enhances detection efficiency on tiny objects, is also included in YOLO v5.

Both YOLO v4 and YOLO v5 train the model using a similar loss function, but YOLO v5 adds a new concept

DOI: https://doi-ds.org/doilink/05.2024-51388889

termed "CIoU loss" that enhances the model's performance on unbalanced datasets. Despite these modifications, YOLO v4 and v5 continue to be significant object identification algorithms with cutting-edge outcomes.

YOLO v6

In 2022, Li et al. suggested YOLO v6 as an improvement over earlier iterations. The CNN architecture used in YOLO v6 is different from that in YOLO v5. EfficientNetL2, a more effective form of the EfficientNet design with fewer parameters and improved computing efficiency, is used by YOLO v6 in place of EfficientDet. On a variety of object detection benchmarks, it can produce good results. Below is a diagram of the YOLO v6 architecture framework.

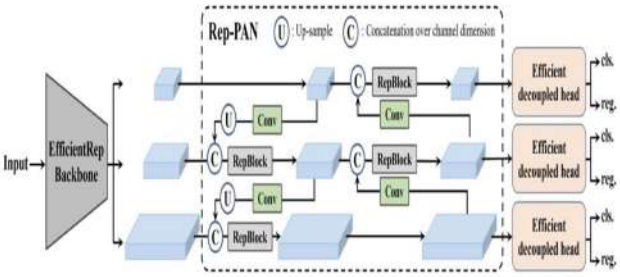
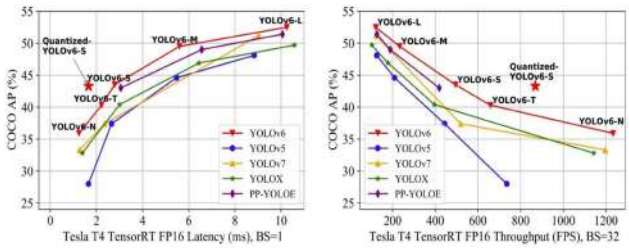


Figure 5.4.7 : Architecture of YOLOv6 model.

OVERVIEW OF YOLO v6.

In YOLO v6, "dense anchor boxes," a new technique for creating the anchor boxes, was included. Below is a comparison of how well YOLO v6 performs in comparison to other cutting-edge techniques.



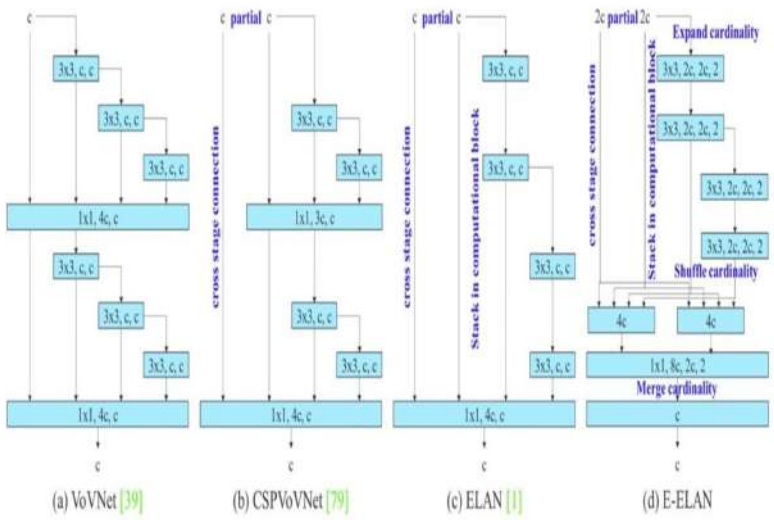
WHAT'S NEW ABOUT YOLO v7?

YOLO v7 is the latest version and better in many ways. Anchor boxes are a major addition in YOLO v7.Items of different shapes are found using anchor boxes with specific aspect ratios. YOLO v7 uses nine anchor boxes to recognize more object types and sizes than previous versions. Fewer false positives improve the algorithm's accuracy.

In addition to anchor boxes, YOLO v7 uses a novel loss algorithm called "focal loss." Earlier YOLO iterations had trouble detecting small objects using the cross-entropy loss function. Focal loss reduces loss for well-classified instances and accentuates hard-to-detect things. More photo analysis resolution is a big improvement

DOI: https://doi-ds.org/doilink/05.2024-51388889

in YOLO v7. YOLO v7 processes photographs at 608 by 608 pixels, unlike v3. Because of its higher resolution, YOLO v7 can recognize microscopic items and is more accurate overall. Focus loss and anchor boxes improve model accuracy in YOLO v7, particularly for small objects. Due to these advances, YOLO v7 has shown cutting-edge results on various object identification benchmarks, making it a viable technique. The YOLO algorithm's evolution has improved object detecting performance and accuracy. Anchor boxes, focus loss, and higher photo resolutions in YOLO v7 have revolutionized object detection. The YOLO technique may lead to greater object detection advancements as technology advances.



The quickness of YOLO v7 is one of its key benefits. 155 frames per second is substantially quicker than other cutting-edge object identification systems in processing pictures. Even the baseline YOLO model could process at a top speed of 45 frames per second. This qualifies it for delicate real-time applications like surveillance and autonomous vehicles, where faster processing rates are essential. Yolo v7 features a modification to the layer aggregation approach for effective object feature learning in order to reach this high processing speed. This entails more effectively aggregating the feature maps from the prior layers, enabling quicker and more precise object recognition. Additionally, YOLO v7 trains with a higher batch sizeIt enables improved GPU use and quicker training durations. YOLO v7's architecture and training procedure have been improved, and the end result is an extremely effective and precise object identification system that can be used in practical applications.

DOI: https://doi-ds.org/doilink/05.2024-51388889

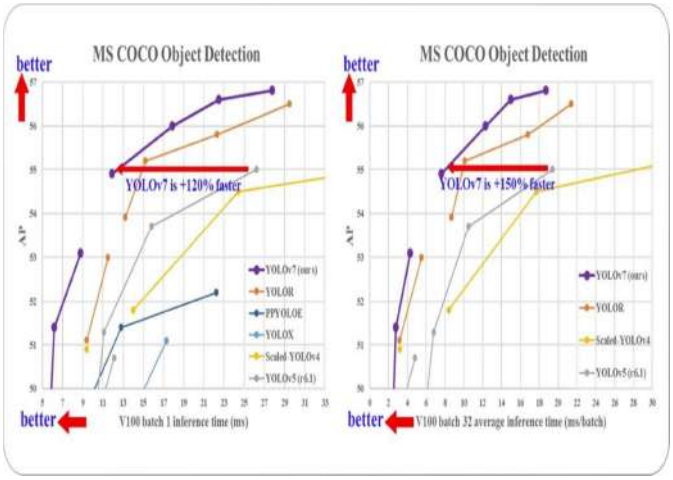


Figure 5.4.8 : Performance of YOLOv7 model

A comparison of YOLO v7's performance and inference speed with modern, cutting-edge real-time object detectors. Average Precision (AP).

YOLO v7 is competitive with other object detection algorithms in terms of accuracy. At an IoU (intersection over union) threshold of 0.5, it obtains an average accuracy of 37.2% on the widely used COCO dataset. Similar to other cutting-edge object detection techniques, this performance. The graph below shows a quantitative evaluation of YOLO v7's performance in comparison to other algorithms.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Figure5.4.9: Performance comparision of all versions

On the COCO dataset, the accuracy of two-stage detectors like Faster R-CNN and Mask RCNN is

DOI: https://doi-ds.org/doilink/05.2024-51388889

substantially lower than that of YOLO v7. Even while these models provide better average accuracy, they often need longer inference durations.

LIMITATIONS OF YOLO v7

1. The difficulty of properly recognizing tiny things is one of YOLO v7's drawbacks, which is shared by many object identification systems. It might have trouble accurately identifying these things in cluttered environments or when they are far from the camera.
2. In addition, YOLO v7 could have trouble recognizing items of various sizes, especially those that are noticeably bigger or smaller than other things in the scene.
3. YOLO v7 could also be sensitive to changes in lighting or other environmental factors, making it less practical to use in practical situations where lighting conditions might fluctuate.
4. The computational complexity of YOLO v7 is still another drawback. This may restrict the method's potential for usage in some applications by making it difficult to run the algorithm in real-time on devices with limited resources, such as smartphones or other edge devices.

YOLO v8

Ultralytics has confirmed the release of YOLO v8, which is expected to offer new features and improved performance over previous versions. One of the most notable improvements in YOLO v8 is the introduction of a new API that is designed to simplify both training and inference on both CPU and GPU devices. In addition, the framework is expected to support previous versions of YOLO. While the developers have not yet released a scientific paper describing the architecture and performance of YOLO v8, it is anticipated that the new version will offer significant enhancements. These enhancements may include improvements in accuracy, speed, and the ability to detect smaller and larger objects, among other things. The development of YOLO v8 is an exciting development for the field of object detection, as it promises to offer new capabilities and performance improvements. With the new API and support for previous versions, YOLO v8 is expected to be accessible to a wider range of users, including those who may not have extensive experience with deep learning or computer vision. As with any new technology, there may be challenges and limitations associated with YOLO v8, but it is anticipated that the benefits will outweigh any drawbacks. As more information becomes available about the performance and capabilities of YOLO v8, it will be interesting to see how it is adopted by researchers and practitioners in the field of computer vision.

In this project, we used YOLOv4 for object detection. We trained on the COCO dataset which has nearly 33000 training images. We used a dataset from ROBOFLOW for emergency vehicle detection training

VI. IMPLEMENTATION

6.1 SOFTWARE IMPLEMENTATION IN PC

Firstly we have done our project on our personal PCs. We downloaded the python environment Anaconda 3 and we used Jupyter Notebook as an editor. We downloaded all the necessary packages like OpenCV, Numpy, etc.

DOI: https://doi-ds.org/doilink/05.2024-51388889

After that we deployed the code in Raspberry Pi. Before that, you have to follow some steps.They are mentioned below.

6.2.SOFTWARE INSTALLATION OF RASBERRY PI

The Raspberry Pi's operating system (OS), which may be downloaded from the website's Downloads section https://www.raspberrypi.org/downloads/

Figure 6.2.1: Raspberry Pi software installation

6.3.Connecting motion capture camera with Raspberry Pi

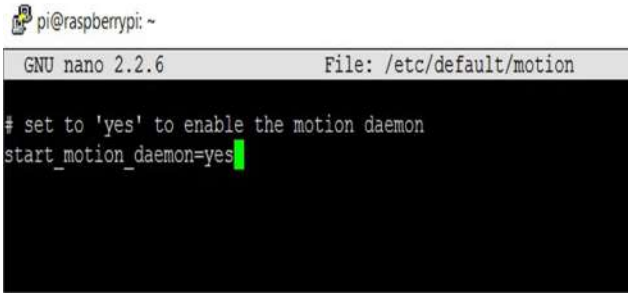


Figure 6.3.1: Connecting camera to Raspberry Pi

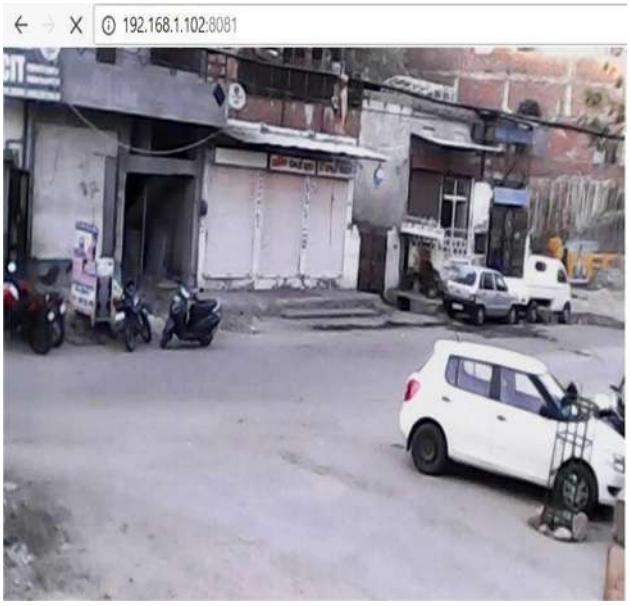


Figure 6.3.2: Live Stream from Camera

DOI: https://doi-ds.org/doilink/05.2024-51388889

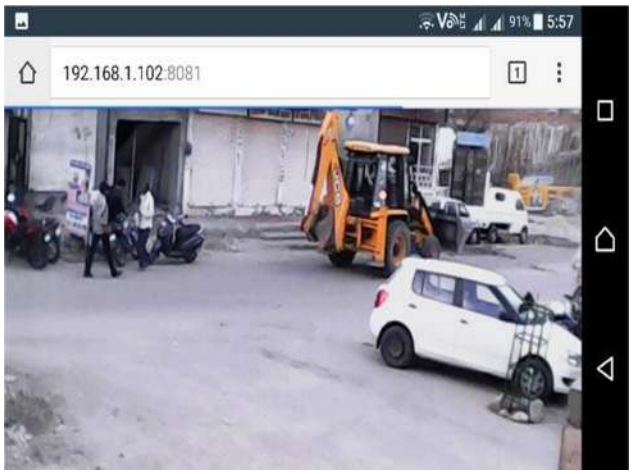


Figure 6.3.2: Live Stream from Camera

VII. SIMULATION RESULTS

Figure 7.1: Results-vehicles are identified

Figure 7.2: Results – emergency vehicles are identified

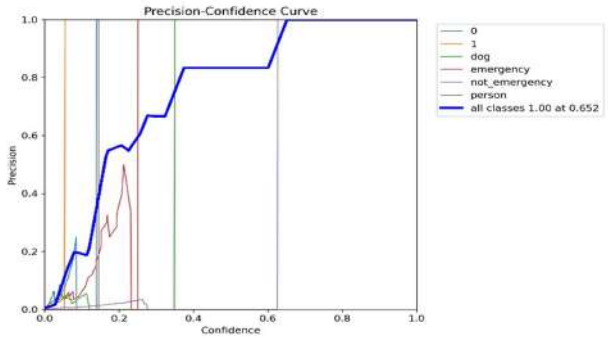


Figure 7.3: Results of our trained model

DOI: https://doi-ds.org/doilink/05.2024-51388889

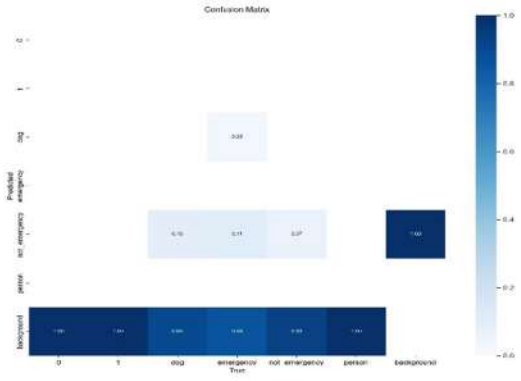


Figure 7.4: Confusion matrix of our trained model

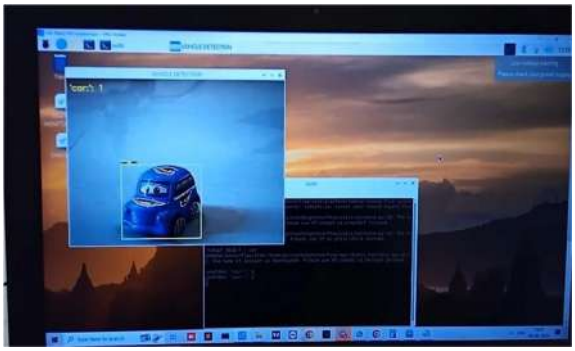


Figure 7.5 : Results from Raspberry pi

VIII. CONCLUSION AND FUTURE SCOPE

8.1.Conclusion

A cutting-edge system combining IoT and machine learning algorithms was created to develop an automatic traffic light system. This system aimed to determine the appropriate duration for the green traffic light based on the density of traffic in real time. Each lane was equipped with a webcam to capture images, which were then processed using the YOLO detection algorithm to accurately identify the number of vehicles present. To perform the detection, a convolutional neural network (CNN) with pretrained YOLO coco weights was employed. By analyzing the vehicle count, the green time for the traffic light was set optimally, reducing overall waiting times for motorists.

The master controller for this innovative system was based on Raspberry Pi, a versatile and powerful microcomputer. The traffic signal systems were equipped with LEDs, allowing for seamless integration and control. The prototype underwent rigorous testing in real-world scenarios, demonstrating a remarkable accuracy of 64% in accurately determining the number of vehicles present and setting an optimized green time.

DOI: https://doi-ds.org/doilink/05.2024-51388889

This groundbreaking solution addresses the challenges faced by traditional traffic light systems by leveraging the power of IoT and machine learning. By dynamically adjusting the green time based on the current traffic density, the system aims to achieve an optimal traffic flow and enhance overall efficiency on the roads. The utilization of the YOLO detection algorithm with pre-trained weights ensures high precision and reliable vehicle identification. But The accuracy of emergency vehicles was less as we trained with our custom data set. We could improve the accuracy by training with huge datasets, and by increasing the number of epoch layers.

With further advancements and refinements, this automatic traffic light system has the potential to revolutionize traffic management and improve the commuting experience for drivers. The combination of IoT, machine learning, and Raspberry Pi offers a scalable and adaptable platform that can be deployed in various traffic scenarios, contributing to smarter and more efficient cities.

8.2.Future Scope

In future iterations, the utilization of Raspberry Pi microcontrollers presents a promising opportunity. We can improve this by extending this to a network instead of one signal. We could train with a huge dataset and by changing some other properties of layers and improve accuracy. By integrating the OpenCV software directly into the Raspberry Pi, there will be no need to install OpenCV separately on the system. This streamlined approach ensures efficient and hassle-free implementation. Moreover, leveraging Raspberry Pi's capabilities, the traffic view can be transmitted to the traffic controller room, enabling them to make informed decisions regarding green signal duration. This intelligent system would strategically allocate longer green signal times in areas with high traffic density, effectively minimizing unnecessary waiting times at signals. The integration of Raspberry Pi in traffic management systems holds immense potential for optimizing signal timings. By providing real-time traffic monitoring and analysis, the system can dynamically adjust green signal durations to alleviate congestion and enhance traffic flow. This technology-driven solution aims to reduce overall waiting times and improve the commuting experience for drivers. The use of Raspberry Pi not only simplifies software integration but also enables seamless communication between traffic control rooms and signal infrastructure, enabling efficient traffic management strategies to be implemented.

IX. REFERENCES

[1] 1.Gul Shahzad; Heekwon Yang; Arbab Waheed Ahmad; Chankil Lee, “EnergyEfficient Intelligent Street Lighting System Using Traffic-Adaptive Control”, IEEE 2016, Volume: 16

[2] Nicole Díaz, Jorge Guerra, Juan Nicola, “Smart Traffic Light Control System”, IEEE 2018.

[3] Khushi, “Smart control of Traffic Light System using Image Processing”, International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC-2017)

[4] 4.J. K. and A. Desai, ”IoT: Networking Technologies and Research Challenges”, International Journal of Computer Applications, vol. 154, no. 7, pp. 1-6, 2016. 5.” Application of Raspberry Pi and PIR Sensor for Monitoring of Smart Surveillance System”, International Journal of Science and Research (IJSR), vol. 5, no. 2, pp. 736737, 2016.

DOI: https://doi-ds.org/doilink/05.2024-51388889

[5] 6.K. Choi, "Visible Light Communication with Color and Brightness Control of RGB LEDs", ETRI Journal, vol. 35, no. 5, pp. 927-930, 2013.

[6] 7.P. N R, "Smart pi cam based Internet of things for motion detection using Raspberry pi", International Journal of Engineering and Computer Science, 2016.

[7] Raj Kumar D bhure, K. Saisrikar, Ch. Devayani, D. Shivareddy, Energy Efficiency Routing For Manet Using Residual Energy, International Journal of Multidisciplinary Engineering in Current Research - IJMEC Volume 8, Issue 4, April-2023, http://ijmec.com/, ISSN: 2456-4265.

[8] Mr. Touseef Sumeer, Shahzada Salim, Md Abdul Jabbar, Shaik Rehmath, Home Appliances Control Using Remote Control System, International Journal of Multidisciplinary Engineering in Current Research - IJMEC Volume 8, Issue 4, April-2023, http://ijmec.com/, ISSN: 2456-4265.

[9] Chandra Singh, Ch. Nivas, G. Pranay Kumar, V. Manoj Kumar, Analysis And Impact Of Electric Vehicle Charging Station On Power Quality, International Journal of Multidisciplinary Engineering in Current Research - IJMEC Volume 8, Issue 4, April-2023, http://ijmec.com/, ISSN: 2456-4265.

[10] Ishaq Bin Mohammed Barabood, Mohd Mohsin Uddin, Mohd Faraz Uddin, Mrs. Asra Sultana, Cellar Ventillation System With Auto Detection And Control, International Journal of Multidisciplinary Engineering in Current Research - IJMEC Volume 8, Issue 4, April-2023, http://ijmec.com/, ISSN: 2456-4265.

[11] Ganvi Ranjitha , *Sheguri Kushma,* Boyala Bhavani, Classification And Detection Of Hyperspectral Images, International Journal of Multidisciplinary Engineering in Current Research - IJMEC Volume 8, Issue 4, April-2023, http://ijmec.com/, ISSN: 2456-4265.

DOI: https://doi-ds.org/doilink/05.2024-51388889