MINI PROJECT

5th semester

Name – RAJARSHI BANERJEE

Class Roll - 423

Roll no. – T91/OEE/184119

Reg no. – D01-1111-0178-17

Year - 3rd yr.

Semester – 5th

Name – NILADRI DAS

Class Roll – 401

Roll no. - T91/OOE/184110

Reg no. – D01-1112-0049-18

Year – 3rd yr.

Semester – 5th

UNDER THE GUIDANCE OF

DR. KANIK PALODHI

UNIVERSITY OF CALCUTTA

DEPARTMENT OF APPLIED OPTICS AND PHOTONICS

Contents

Transmission of data through laser with wired clock pulse

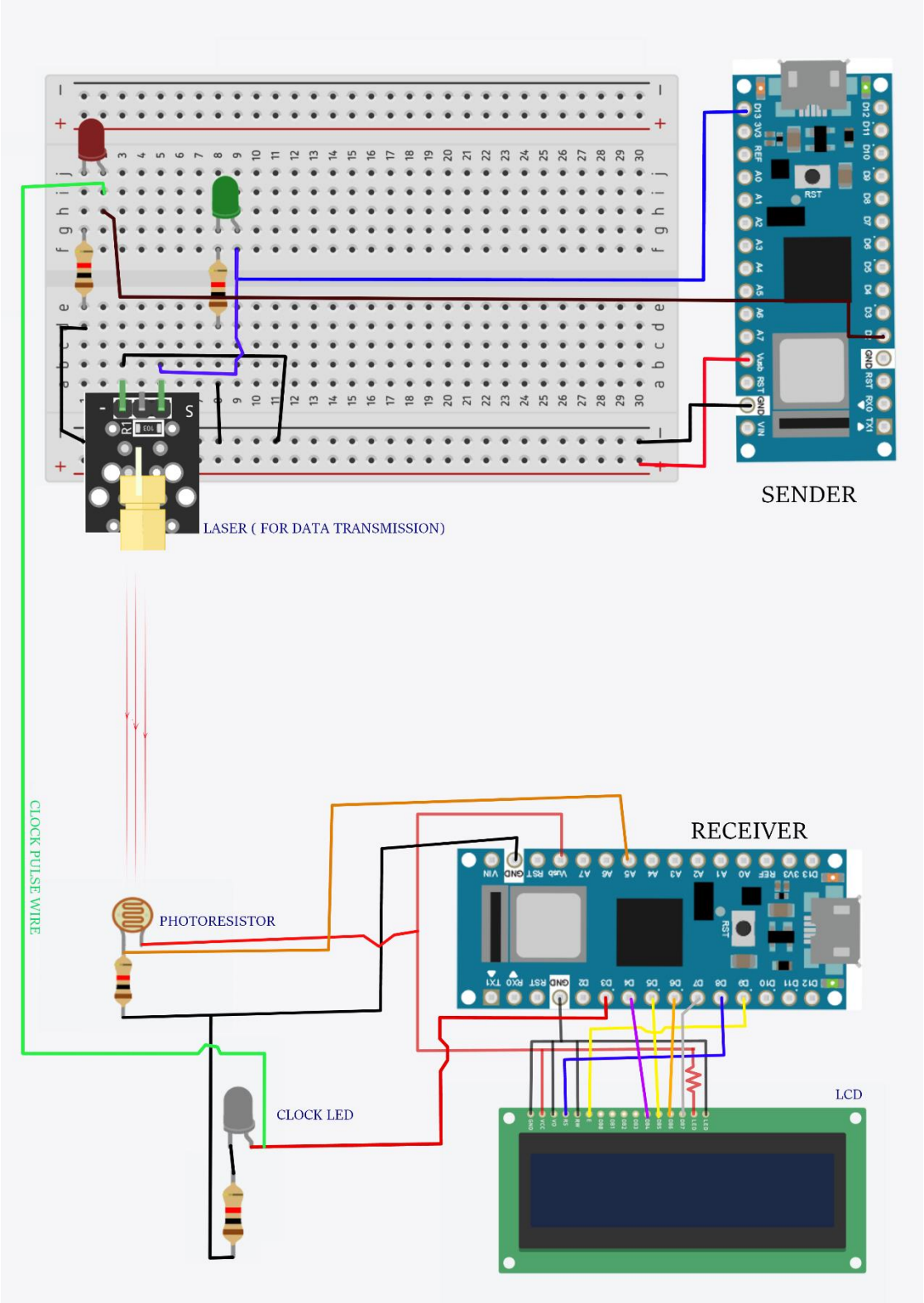
This project is to demonstrate the transmission of digital data signal in the visible spectrum. That is transmission of digital data using visible light.

Why visible light?

It is because of the following reasons:

1. Light waves are able to carry far more information and is faster than radio waves used in Wi-Fi technology since the visible spectrum is 10000 times larger than spectrum occupied by the radio waves.
2. With the modern led bulbs technology, it is easier and more energy efficient.
3. Radio waves can be intercepted by people outside your network since they can pass through walls compromising the security of your data. But light is stopped by opaque objects making Light significantly more secure than other wireless technologies.
4. Availability – this means every electronic source of light can be used to transmit data.

Here in this project we used two Arduino nano to demonstrate how data can be transmitted using a laser with a wired clock pulse.



CIRCUIT DIAGRAM

CIRCUIT DESCRIPTION

- APPARATUS REUIRED ARE
  - Arduino Nano - 2
  - Liquid Crystal Display - 1
  - Resistors (1kΩ) - 5
  - Jumper wires (male-to-male) - 20
  - LEDs - 3
  - Laser - 1
  - Photoresistor (LDR) - 1
- SENDER
  - Data pin 13 – via which the 8 bit data will be transferred. This pin is connected to a green LED and a laser.
  - Pin 2 of arduino nano is the clock pin to send continuous pulse. This pin is connected to red LED which is directly connected to the receiver.
- RECIEVER
  - Clock pin 3 is connected to a white LED.
  - LDR or the photoresistor is connected to analog pin A5.

Each LED is being connected with a 1K ohm resistor at cathode.

Circuit Construction and Details

Two microcontrollers (Arduino nanos) are used as a transmitter and a receiver.

Transmitter module:

The positive end of the laser is connected to the pin number 13 and the other end is connected to ground. LED has been attached in parallel with the laser just to indicate whether there is signal, since the laser is inserter at one end of a hollow pipe to reduce the ambient light as much as possible.

Pin number 2 of the Arduino nano has been used to send a constant clock pulse. The significance of this clock pulse if given in the theory section and challenges faced section later. This pin is connected to a red LED to get a visual representation of the pulse.

Receiver module:

Here a photo resistor has been used as the main element for receiving the laser signal. It has two terminals, one terminal is connected with a resistor of 1Kohm in series making it 3 terminal device as shown in the figure. The two extreme ends are connected to Vcc and Ground respectively while the middle terminal is connected to the analog pin A5 of the Arduino to receive analog input.

A 16x2 LCD is connected to the Arduino to see what data has been transferred along with each 8 bit ascii value in binary of every character.

Pins of LCD are connected as follows:

D4,D5,D6,D7,RS(Register select) and E(enable) of the lcd is connected to the digital pin numbers 4,5,6,7,8,9 respectively.

VSS is connected to ground and VDD is connected to Vcc or 5v.

V0 or the contrast pin, RW pin of lcd is connected to ground.

A(anode) and K(cathode) pins are connected to Vcc and ground respectively.

The clock pin is pin number 3 since it is the interrupt pin. A white LED is connected with the pin 3, and a wire from the transmitter is connected to the positive terminal of the LED pin.

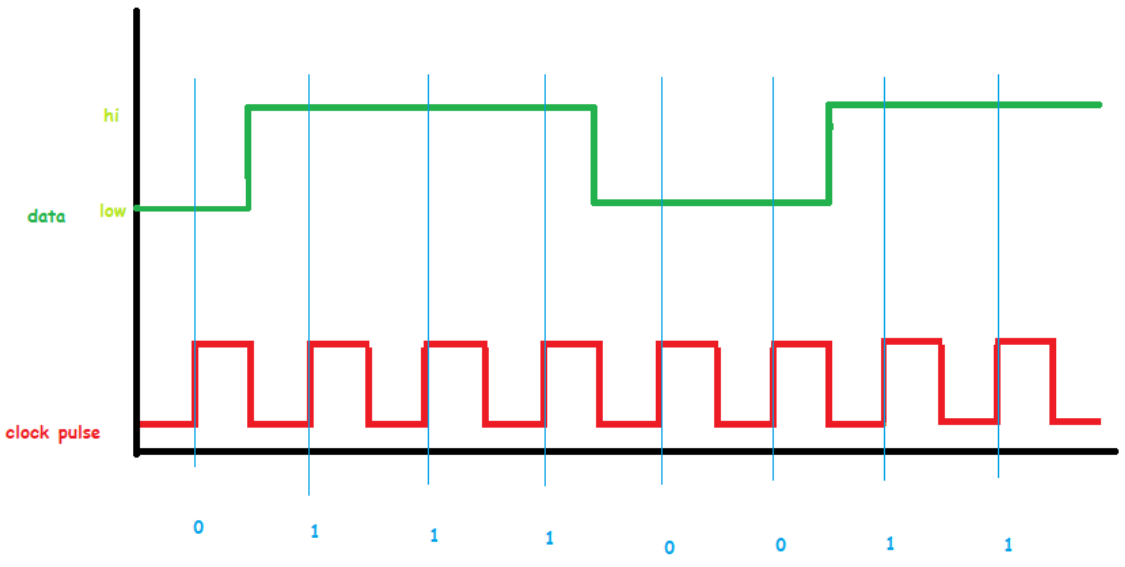
THEORY

What is data? Computer data or data in general means information. In the world of computing data is the information processed or store by a computer. This information may be in the form of text documents, images, audio clips, software programs or other types of data. Data are represented in binary form. That is 0s and 1s are used to represent digital data. These are two sates 1 being the HIGH or the ON state and 0 being the LOW or OFF state. From this we get the word bit which basically means binary digits.

Since machine cannot understand human level language. Inside a machine everything is the binary format. We can use this knowledge to transmit data using light. LED as we all know also has 2 states which is the ON and OFF state. Using this we can assign 1 and 0 to the respective state, and this will represent a one bit. Sequence of 8 bit will give us different characters. Including all the English alphabets to numbers and general purpose symbols. To represent an 8 bit data, we can use 8 LEDs to represent every single bit.

Now in order to transmit data we can use the LED or LASER as a sender to send data signal. Since every 8 bit represents a character, then several 8 bits data will give rise to a word or even sentence.

But to interpret the signal rather the bits correctly we need a reference, a clock which will change its state at a fixed rate. With respect to this the receiver can decode the bits accurately. Otherwise any delay or even missing a single bit will give rise to a different result. To give a clear picture here is an example:



In the picture above shows how data is decoded with respect to the clock. Here 01110011 is the ascii value for the lower case letter ‘s’. the data signal represents this character. The rising edge of the clock pulse is used to determine the corresponding value of the data signal.

The clock pulse will be sent via a wire and the data via a laser.

The communication is based on synchronous communication. Where the message is exchanged in real time. It requires that the transmitter and a receiver are present in the same time and space.

A message is nothing but a string. String is an array of characters. Each character is of size 8 bit means 8 bits ascii value can be used to represent a single character. So, the bits are to be transferred bit by bit (which is the case in serial communication).

Here in the code the the interrupt pin 3 has been used as the clock pin to implement the interrupt function. This is the only way the Arduino processor can do two task simultaneously.

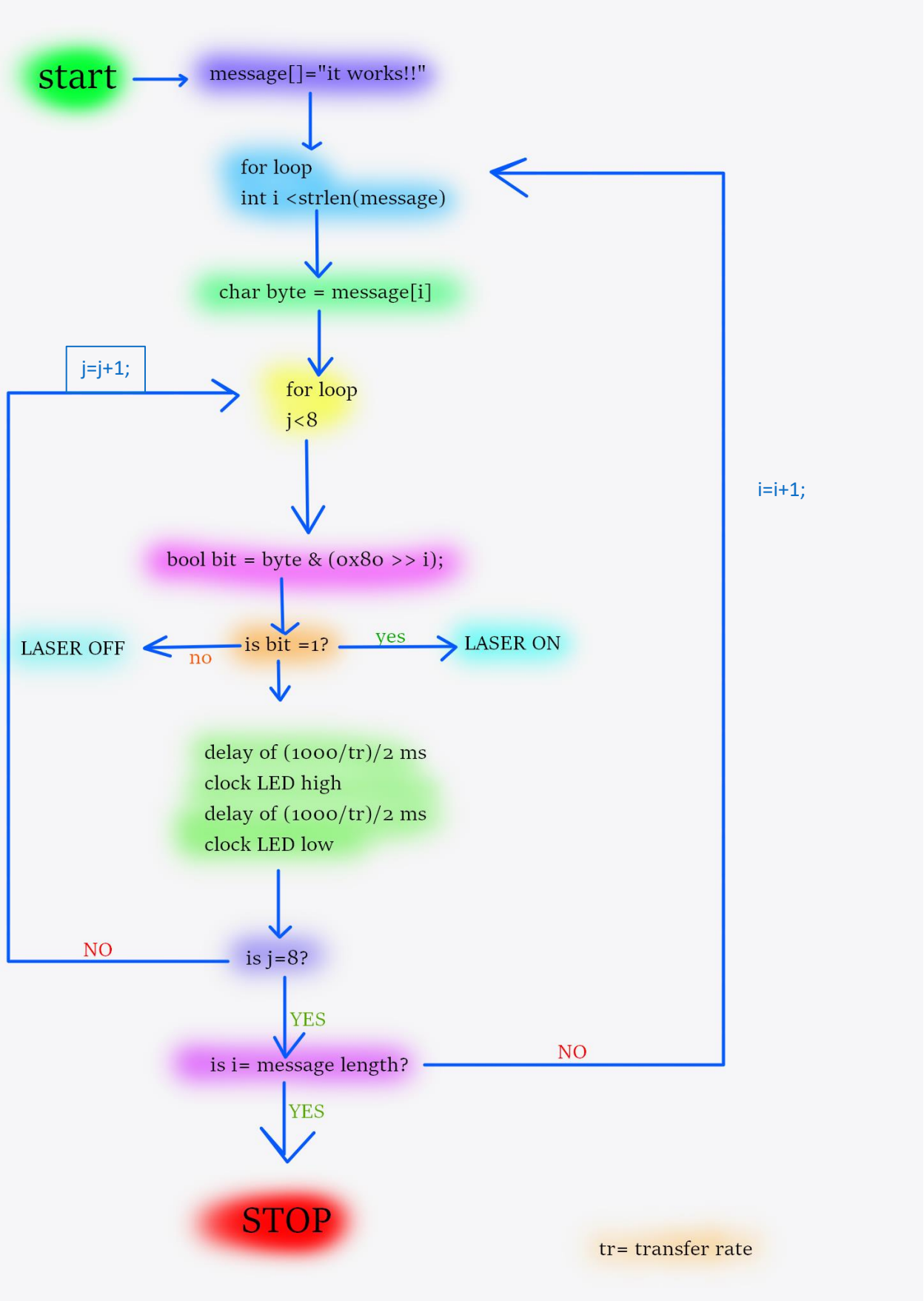
When the clock pin value changes from 0 or low to 1 or high the code will process the onClockPulse() function. Hence, the word RISING in line number 23 on the receiver code.

Working principle of LDR

When light falls i.e. when the photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band. These photons in the incident light should have energy greater than the bandgap of the semiconductor material to make the electrons jump from the valence band to the conduction band.

Hence when light having enough energy strikes on the device, more and more electrons are excited to the conduction band which results in a large number of charge carriers. The result of this process is more and more current starts flowing through the device when the circuit is closed and hence it is said that the resistance of the device has been decreased. This is the most common **working principle of LDR**

ALOGORITHM/FLOWCAHRT FOR TRANSMITTER



Explanation of the line bool bit = byte& (0x80>>i)

& -> bitwise and operator

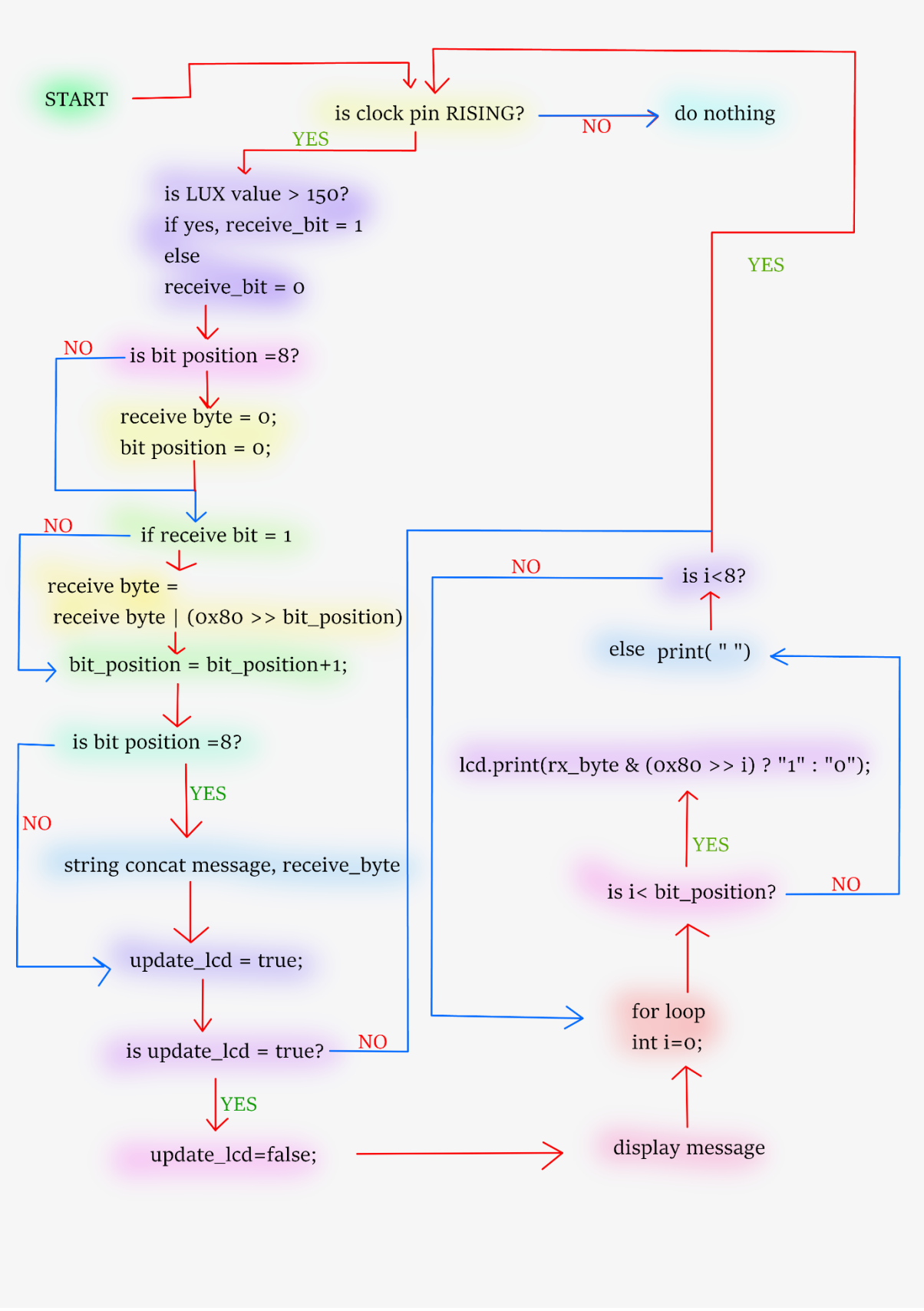
>> -> right shift operator

0x80 -> it is the hexadecimal representation of 1000 0000

The line will extract each and every single bit of the character from the string (message) and turn the laser on according to the extracted bit.

Tr = transfer rate at which the data is being transferred. We have restricted the transfer rate to 8bits/second.

RECEIVER MODULE ALGORITHM/FLOWCHART



CODE FOR TRANSMISSION MODULE

```
int tr = 8; // tr = transfer rate
#define clk 2 // clock
#define data 13 // transmission data
const char *message = "it works!!!";

void setup() {
 Serial.begin(9600);
 pinMode(clk, OUTPUT);
 pinMode(data, OUTPUT);
 for (int j = 0; j < strlen(message); j++) {
  char byte = message[j];

  for (int i = 0; i < 8; i++) {
   bool bit = byte & (0x80 >> i); // reading each bit of a letter
// right shift operator
   digitalWrite(data, bit); // data led
   Serial.print(bit);

   delay((1000 / tr) / 2);

   // clock pulse
   digitalWrite(clk, HIGH);
   delay((1000 / tr) / 2);
   digitalWrite(clk, LOW);
  }Serial.println();

 }

 digitalWrite(data, LOW);
}

void loop(){}
```

CODE FOR RECEIVER MODULE

```
#include <LiquidCrystal.h>

// Pin assignments
#define RX_CLOCK 3
#define RX_DATA A5
#define D4 4
#define D5 5
#define D6 6
#define D7 7
#define RS 8
#define EN 9

LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);

char message[16];
volatile byte rx_byte = 0;
volatile int bit_position = 0;
volatile bool update_lcd = true;

void setup() {
 pinMode(RX_DATA, INPUT);
 strcpy(message, "");
 lcd.begin(16, 2);
 attachInterrupt(digitalPinToInterrupt(RX_CLOCK), onClockPulse, RISING);
}

void onClockPulse() {
 bool rx_bit = analogRead(RX_DATA)>150? true : false;

 if (bit_position == 8) {
  rx_byte = 0;
  bit_position = 0;
 }

 if (rx_bit) {
  rx_byte |= (0x80 >> bit_position);
 }

 bit_position += 1;

 if (bit_position == 8) {
  strncat(message, (const char *)&rx_byte, 1);
 }

 update_lcd = true;
}
```

```
void loop() {
  if (update_lcd) {
    update_lcd = false;

    lcd.noCursor();
    lcd.setCursor(0, 0);
    lcd.print(message);
    lcd.setCursor(0, 1);
    for (int i = 0; i < 8; i += 1) {
      if (i < bit_position) {
        lcd.print(rx_byte & (0x80 >> i) ? "1" : "0");
      } else {
        lcd.print(" ");
      }
    }
    lcd.setCursor(strlen(message), 0);
    lcd.cursor();
  }
}
```

CHALLENGES FACED WHILE MAKING THE PROJECT

- Since these are electronic devices and really small, soldering of each and every pins of both the arduinos and the lcd was difficult. Poor soldering might lead to poor connection and miss interpretation of data.
- Ambient light will affect the laser communication so have to keep threshold greater than the ambient lighting lux.
- Long distance communication is not possible since the clock pulse is been transferred through a wire. In case of asynchronous communication where the bits are only transferred via laser module without the requirement of any clock pulse as such.
- The problem with this type of communication is both the devices are to synced perfectly. Which means they are to reset simultaneously without any fraction of delay. Even then out of 10 only 1 time the data is being interpreted or decoded accurately. So, synchronization is a big problem. The code will be complex.

CONCLUSION

- This demonstration shows that it is possible to transfer data in the visible electromagnetic spectrum.
- It is easier and faster to transmit data and is highly secure.

DISCUSSION

- To get the best and accurate result both the arduinos must be reset at the same time.
- Without the use of clock pulse wire, synchronizing the two device i.e the receiver and the transmitter would have been very difficult. In that case the probability of getting the accurate data on the receiver end is nearly 0.1 i.e one accurate result out of ten case. Otherwise misinterpretation of data is inevitable.
- Both the clock pulse and the data could have been transferred via laser but the problem is there is no analog interrupt pin in Arduino. So it can’t be used .

Real time face detection Using MATLAB

Abstract:-- **The human face is a complicated multidimensional visual model and hence it is very difficult to develop a computational model for recognizing it. The paper presents a methodology for recognizing the human face based on the features derived from the image. The proposed methodology is implemented in two stages. The first stage detects the human face in an image using violaJones algorithm. In the next stage the detected face in the image is recognized using a fusion of Principle Component Analysis and Feed Forward Neural Network. The performance of the proposed method is compared with existing methods. Better accuracy in recognition is realized with the proposed method. The proposed methodology uses Bio ID-Face-Database as standard image database.**

INTRODUCTION :-- **This electronic document is about face detection. In computer literature face detection has been one of the most studied topics. Given an arbitrary image, the goal of this project is to determine whether or not there are any faces in the image and detection of eyes and upper body. While this appears to be a trivial task for human beings, it is very challenging task for computers. The difficulty associated with face detection can be attributed to many variations in scale, location, view point, illumination, occlusions, etc. Although there have been hundreds of reports reported approaches for face detection, if one were asked to name a single face detection algorithm that has most impact in recent decades, it will most likely be the Viola and Jones face detection, which is capable of processing images extremely rapidly and achieve high detection rates. In this we are going to study and understand the Viola Jones algorithm by implementing the detection frame work and based on that implementation, conduct experiment to improve the performance.**

The first contribution of this paper is a new image representation called an integral image that allows for very fast feature evaluation. The integral image can be computed from an image using a few operations per pixel. Once computed, any one of these Har Basis like features can be computed at any scale or location in constant time.

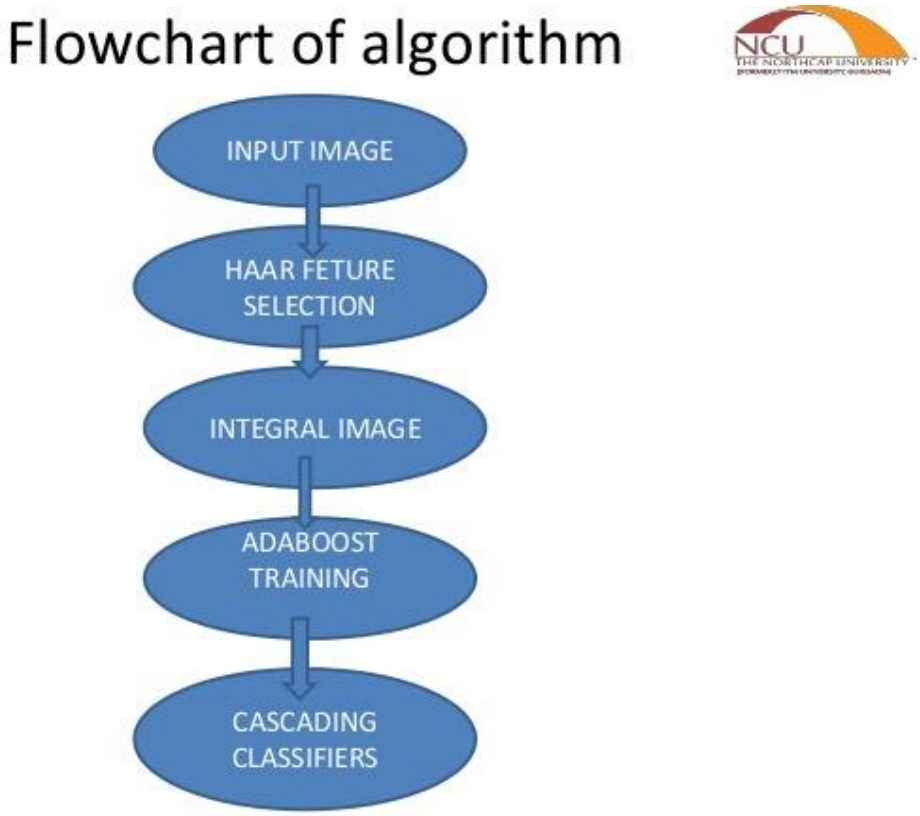
The second contribution of this paper is a simple and efficient classifier that is built by selecting a small number of important features from a huge library of potential features using AdaBoost. . AdaBoost provides an effective learning algorithm and strong bounds on generalization performance.

The third major contribution of this paper is a method for combining successively more complex classifiers in a cascade structure which dramatically increases the speed of the detector by focusing attention on promising regions of the image.

THEORY :-- **The real time face detection method is totally based on viola jones algorithm.**

>>VIOLA JONES APPROACH :---In order to detect an object using a camera, viola and jones algorithm is one of the fast and accurate approaches. All the smart phones and

digital cameras manufacturers are adapting these techniques to enhance the focus. Har and adaboost features help the algorithm to detect the face in a fraction of second. Then, they are subtracted from the total of white boxes. Finally, the result will be compared to the defined threshold and if the criteria are met, the feature considers a hit.



This approach is used to detect the objects combine's four key concepts:

1. Simple rectangular features, Haar-like features.

2. Integral image for rapid feature detection.

3. Adaboost machine- learning method.

4. Cascaded classifier to combine many features efficiently.

Now we have discussed about these four topics in shortly----

1)Haar Basis Function:---Haar like features Haar like features are used todetect variation in the black and light portion of the image. This computation forms a single rectangle around the detected face. Based on the colour shade near nose or forehead a contour is formed. Some commonly used Haar features are:

a. Two rectangle feature.

b. Three rectangle feature.

c. Four rectangle feature.

The value of two rectangle feature is the difference between the sums of the pixels within the two rectangle regions as shown in fig.10, in three rectangles, the value is centre rectangle subtracted by the addition of two surrounding rectangles. Whereas four rectangle features computes the difference between the diagonal pairs of the rectangles.

2)Integral Images:--- They are also known as summed area tables. Integral image is used to felicitate quick feature detection. The meaning of integral image is the outline of the pixel values in the original images.

The integral image at location (x ,y) contains the sum of the pixels above and to the left of (x, y) inclusive II (x, y) =∑ i(x, y).

3) Adaboost machine learning method:---- For combining different classifiers of same data set this algorithm uses concept of bagging .The purpose of this concept is to improve an unstable classifier where a small change in the learning set produces a large change in the classifier. We use adaboost algorithm to select small features from the face which helps to compute the image fast and easily. This method gives desired region of the object discarding unnecessary background. This model can be interpreted by using neural networks. Adaboost learning method is fast and it gives desired data. This data can be classified into classifier which contains small features commonly employed for pattern detection. It has high accuracy and detection speed but requires more time to train.

4) Cascade classifier :---This algorithm (viola and jones) eliminates the face candidates quickly using cascade of stages making stricter requirements in each stage with further stages become difficult to candidate to pass if they fail in any one of the stage. If the candidate passes all the stages then the face is said to be detected.

>>Graphical user interface(GUI):--- The real-time face detection program is developed using any higher version of MATLAB. A graphic user interface allows users to perform tasks interactively by using controls likes switches and sliders. GUI can be created easily and can be run in MATLAB or be used as stand- alone application.

Computer Vision Toolbox

- **Features detection, extraction, matching.**
- **Object Detector and tracking:--- It is detected object using the viola jomes algorithm.**

  Detector = vision.cascadeobjectdetector(model)

In here model means eye, nose, mouth and the upper body.

- ***Step function;---***

BBOXES = step (detector, I)

In here, I means the image , detector operates using vision cascadeobject detector and also the 'BBOXES' means x, y width height of the object.

- **Inserting the Boxes:----**

IFace = insertObjectAnnotion ('I', 'rectangle', 'bboxes', 'face')

In here, the 'I' represents the image , the 'IFace' represents the output image, 'Face' , 'rectangle' , 'bboxes' represent label, shape and positions.

Algorithm and coding using matlab:-----

```
%% imread
im=imread('images.jpg');
imshow(im);
%% using uigetfile
[file,path]=uigetfile('*.*','Select image');
loc=strcat(path,file);
pic=imread(loc);
imshow(pic);%show the image using imshow function from plots tab
%% rgb2gray, bw, warning off
warning('off');
imgray=rgb2gray(pic);
figure,imshow(imgray);
le=graythresh(pic);
imbw=im2bw(pic,le);
imshow(imbw);
imbw2=imbinarize(imgray,le);
% imshow(imbw);
%% imwrite
imwrite(imbw2,'bw.jpg','jpg');
%% crop
cpic=imcrop(pic,[1000 900 500 100]);
imshow(cpic);
%% resize
rpic=imresize(pic,0.1);
rpic=imresize(pic,[500 500]);
imshow(rpic);
```

```
%% flip
fpic=flip(pic,1);
imshow(pic);
figure,imshow(fpic);
%% rotate
ropic=imrotate(pic,30,'crop');
imshow(ropic);


%% face detect
[file,path]=uigetfile('*.*','Select image');
loc=strcat(path,file);
pic=imread(loc);
pic2=rgb2gray(pic);
%face
ff=vision.CascadeObjectDetector();
bbox=step(ff,pic2);
dd=insertObjectAnnotation(pic,'Rectangle',bbox,'Face');
pts=detectMinEigenFeatures(pic2,'ROI',bbox);
dd = insertMarker(dd,pts,'+','color','green');
imshow(dd);hold on
plot(pts.Location(:,1),pts.Location(:,2));
hold off

%% mouth
fm=vision.CascadeObjectDetector('Mouth');
fm.MergeThreshold=110;
bbox=step(fm,pic2);
dd=insertObjectAnnotation(pic,'Rectangle',bbox,'Mouth');
imshow(dd);
%% Nose
fn=vision.CascadeObjectDetector('Nose');
bbox=step(fn,pic2);
dd=insertObjectAnnotation(pic,'Rectangle',bbox,'Nose');
imshow(dd);

%% Eye pair
fe=vision.CascadeObjectDetector('RightEye','MergeThreshold',40);
bbox=step(fe,pic2);
dd=insertObjectAnnotation(pic,'Rectangle',bbox,'Eye');
imshow(dd);

%% Upper body
fb=vision.CascadeObjectDetector('UpperBody','MergeThreshold',5);
```

```
bbox=step(fb,pic2);
dd=insertObjectAnnotation(pic,'Rectangle',bbox,'Body');
imshow(dd);

%% Webcam
web=webcam('HD WebCam');
%preview(web);
% pause(2);
% pp=snapshot(web);
% imshow(pp);
% pause(2);
% clear('web');
while true
   pic=snapshot(web);
   ff=imshow(pic);
   pause(0.01);
end

%% Real-time face detection
clc;close all;
% clear('li');
li=webcam();
im=snapshot(li);
dete=vision.CascadeObjectDetector('Mouth','MergeThreshold',100);
pp=imshow(im);
while true
   im=snapshot(li);
   im2=rgb2gray(im);
   bb=step(dete,im2);
   im2=insertObjectAnnotation(im,'rectangle',bb,'Face');
   imshow(im2);
end
```

Cascade object Detector

```
classdef (StrictDefaults)CascadeObjectDetector < matlab.System
   %CascadeObjectDetector Detect objects using the Viola-Jones algorithm
   %   DETECTOR = vision.CascadeObjectDetector creates a System object
   %   that detects objects using the Viola-Jones algorithm. The DETECTOR
   %   is capable of detecting a variety of objects, including faces and a
   %   person's upper body. The type of object to detect is controlled by
   %   the ClassificationModel property. By default, the DETECTOR is
   %   configured to detect faces.
   %
```

```
%   DETECTOR = vision.CascadeObjectDetector(MODEL) creates a System
%   object, DETECTOR, configured to detect objects defined by MODEL.
%   MODEL is a string describing the type of object to detect. There
%   are several valid MODEL strings. Examples include
%   'FrontalFaceCART', 'UpperBody', and 'ProfileFace'.
%
%   <a
href="matlab:helpview(fullfile(docroot,'toolbox','vision','vision.map'),'vision.Cascade
ObjectDetector.ClassificationModel')">A list of all available models is shown in the
documentation.</a>
%
%   DETECTOR = vision.CascadeObjectDetector(XMLFILE) creates a System
%   object, DETECTOR, and configures it to use the custom classification
%   model specified with the XMLFILE input. XMLFILE can be created using
%   the trainCascadeObjectDetector function or OpenCV training
%   functionality. You must specify a full or relative path to the
%   XMLFILE, if it is not on the MATLAB path.
%
%   DETECTOR = vision.CascadeObjectDetector(...,Name,Value) configures
%   the System object properties, specified as one or more name-value
%   pair arguments. Unspecified properties have default values.
%
%   BBOXES = step(DETECTOR, I) performs multi-scale object detection on
%   the input image, I, and returns, BBOXES, an M-by-4 matrix defining
%   M bounding boxes containing the detected objects. Each row in
%   BBOXES is a four-element vector, [x y width height], that specifies
%   the upper left corner and size of a bounding box in pixels. When no
%   objects are detected, BBOXES is empty. I must be a grayscale or
%   truecolor (RGB) image.
%
%   [...] = step(DETECTOR, I, ROI) detects objects within the
%   rectangular search region specified by ROI. ROI must be a 4-element
%   vector, [x y width height], that defines a rectangular region of
%   interest within image I. The 'UseROI' property must be true to use
%   this syntax.
%
%   System objects may be called directly like a function instead of using
%   the step method. For example, y = step(obj, x) and y = obj(x) are
%   equivalent.
%
%   CascadeObjectDetector methods:
%
%   step     - See above description for use of this method
```

```
%   release  - Allow property value and input characteristics changes
%   clone    - Create cascade object detector object with same property
%            values
%   isLocked - Locked status (logical)
%
%   CascadeObjectDetector properties:
%
%   ClassificationModel   - Name of the classification model
%   MinSize               - Size of the smallest object to detect
%   MaxSize               - Size of the biggest object to detect
%   ScaleFactor           - Scaling for multi-scale object
%                         detection
%   MergeThreshold        - Threshold for merging colocated detections
%   UseROI                - Detect objects within a region of interest
%
%   % Example 1:  Face detection
%   % --------------------------
%   faceDetector = vision.CascadeObjectDetector; % Default: finds faces
%
%   I = imread('visionteam.jpg');
%   bboxes = step(faceDetector, I); % Detect faces
%
%   % Annotate detected faces
%   IFaces = insertObjectAnnotation(I, 'rectangle', bboxes, 'Face');
%   figure, imshow(IFaces), title('Detected faces');
%
%   % Example 2: Upper body detection
%   % -----------------------------------
%   bodyDetector = vision.CascadeObjectDetector('UpperBody');
%   bodyDetector.MinSize = [60 60];
%   bodyDetector.MergeThreshold = 10;
%   bodyDetector.UseROI = true;
%
%   I2 = imread('visionteam.jpg');
%
%   % Search for objects in the top half of the image.
%   [height, width, ~] = size(I2);
%   roi = [1 1 width height/2];
%   bboxBody = step(bodyDetector, I2, roi); % Detect upper bodies
%
%   % Annotate detected upper bodies
%   IBody = insertObjectAnnotation(I2, 'rectangle', ...
%                          bboxBody, 'Upper Body');
```

```
    %   figure, imshow(IBody), title('Detected upper bodies');
    %
    %   See also trainCascadeObjectDetector, vision.PeopleDetector

    %   Copyright 2011-2019 The MathWorks, Inc.

    %   References:
    %   -----------
    %  [1] Paul Viola and Michael J. Jones "Rapid Object Detection using a
    %     Boosted Cascade of Simple Features" IEEE CVPR, 2001
    %
    %  [2] Rainer Lienhart, Alexander Kuranov, Vadim Pisarevsky
    %     "Empirical Analysis of Detection Cascades of Boosted Classifiers
    %     for Rapid Object Detection", DAGM Symposium for Pattern
    %     Recognition, pp. 297-304, 2003

    %#codegen
    %#ok<*EMCLS>
    %#ok<*EMCA>
    properties(Nontunable)
        %ClassificationModel A trained cascade classification model
        %   Specify the name of the model as a string. The value specified
        %   for this property may be one of the valid MODEL strings listed
        %   <a
href="matlab:helpview(fullfile(docroot,'toolbox','vision','vision.map'),'vision.Cascade
ObjectDetector.ClassificationModel')">here</a> or an OpenCV XML file containing
custom classification
        %   model data. When an XML file is specified, a full or relative
        %   path is required if the file is not on the MATLAB path.
        %
        %   Default: 'FrontalFaceCART'
        %
        %   See also <a
href="matlab:helpview(fullfile(docroot,'toolbox','vision','vision.map'),'vision.Cascade
ObjectDetector.ClassificationModel')">Available models</a>
        ClassificationModel = 'FrontalFaceCART';
    end
    properties
        %MinSize Size of the smallest object to detect
        %   Specify the size of the smallest object to detect, in pixels,
        %   as a two-element vector, [height width]. Use this property to
        %   reduce computation time when the minimum object size is known
        %   prior to processing the image. When this property is not
        %   specified, the minimum detectable object size is the image size
```

```
        %   used to train the classification model. This property is
        %   tunable.
        %
        %   Default: []
        MinSize = [];
        %MaxSize Size of the biggest object to detect
        %   Specify the size of the biggest object to detect, in pixels, as
        %   a two-element vector, [height width]. Use this property to
        %   reduce computation time when the maximum object size is known
        %   prior to processing the image. When this property is not
        %   specified, the maximum detectable object size is SIZE(I). When
        %   'UseROI' is true, the maximum detectable object size is the
        %   defined by the height and width of the ROI. This property is
        %   tunable.
        %
        %   Default: []
        MaxSize = [];
        %ScaleFactor Scaling for multi-scale object detection
        %   Specify the factor used to incrementally scale the detection
        %   scale between MinSize and MaxSize. The ScaleFactor must be
        %   greater than or equal to 1.0001. At each increment, N, the
        %   detection scale is
        %
        %     round(TrainingSize*(ScaleFactor^N))
        %
        %   where TrainingSize is the image size used to train the
        %   classification model. The training size used for each
        %   classification model is shown <a
href="matlab:helpview(fullfile(docroot,'toolbox','vision','vision.map'),'vision.Cascade
ObjectDetector.ClassificationModel')">here</a>. This property is tunable.
        %
        %   Default: 1.1
        ScaleFactor = 1.1;
        %MergeThreshold Threshold for merging colocated detections
        %   Specify a threshold value as a scalar integer. This property
        %   defines the minimum number of colocated detections needed to
        %   declare a final detection. Groups of colocated detections that
        %   meet the threshold are merged to produce one bounding box
        %   around the target object. Increasing this threshold can help
        %   suppress false detections by requiring that the target object
        %   be detected multiple times during the multi-resolution
        %   detection phase. By setting this property to 0, all detections
        %   are returned without merging. This property is tunable.
```

```
    %
    %   Default: 4
    MergeThreshold = 4;
  end

  properties(Nontunable, Logical)
    % UseROI Detect objects within a ROI
    %   Set to true to detect objects within a rectangular region of
    %   interest within I.
    %
    % Default: false
    UseROI = false;
  end

  properties (Transient,Access = private)
    pCascadeClassifier; % OpenCV pCascadeClassifier
  end

  properties(Hidden,Dependent,SetAccess = private)
    %TrainingSize Image size used to train classification model
    %   This is the smallest object that the classification model is
    %   trained to detect. It is the smallest object size that the
    %   model can detect.
    TrainingSize;
  end
  methods
    %--------------------------------------------------------------------
    % Constructor
    %--------------------------------------------------------------------
    function obj = CascadeObjectDetector(varargin)
      if isSimMode()
        obj.pCascadeClassifier = vision.internal.CascadeClassifier;
      else

        obj.pCascadeClassifier = ...

vision.internal.buildable.cascadeClassifierBuildable.cascadeClassifier_construct();
        for i = 1:length(varargin)
          coder.internal.errorIf(~issparse(varargin{i}) && isstring(varargin{i}), ...
            'vision:validation:stringnotSupportedforCodegen');
        end
      end
      setProperties(obj,nargin,varargin{:},'ClassificationModel');
      % we need to load XML before calling getClassifierInfo (called
```

```
        % from obj.TrainingSize in validatePropertiesImpl
        loadXMLFromClassModel(obj);
        validatePropertiesImpl(obj);
    end

    %--------------------------------------------------------------
    % ClassificationModel set method
    %--------------------------------------------------------------
    function set.ClassificationModel(obj,value)
        if isSimMode() && isstring(value)
            value = ...
                convertStringsToChars(value);
        end
        coder.extrinsic('exist');

        validateattributes(value,{'char'},{'nonempty','row'}, 'CascadeObjectDetector');

        % for exist: file must be in path or an absolute path must
        % be provided
        file_exists = coder.internal.const(exist(value,'file')) == 2;
        coder.internal.errorIf(~isSupportedModel(value) && ~file_exists ...
            && ~startsWith(value, '/sdcard'), 'vision:ObjectDetector:modelNotFound',
value);
        obj.ClassificationModel = value;
        loadXMLFromClassModel(obj);
    end

    %--------------------------------------------------------------
    % UseROI set method
    %--------------------------------------------------------------
    function set.UseROI(obj, value)
        obj.UseROI = logical(value);
    end

    %--------------------------------------------------------------
    % ScaleFactor set method
    %--------------------------------------------------------------
    function set.ScaleFactor(obj,value)
        validateattributes( value,{'numeric'},...
            {'scalar', '>=',1.0001,'real', 'nonempty','nonsparse','finite'},...
            '','ScaleFactor');
        obj.ScaleFactor = value;
    end
```

```
    %-----------------------------------------------------------------
    % MinSize set method
    %-----------------------------------------------------------------
    function set.MinSize(obj,value)
      validateSize('MinSize',value);
      obj.MinSize = value;
    end

    %-----------------------------------------------------------------
    % MaxSize set method
    %-----------------------------------------------------------------
    function set.MaxSize(obj,value)
      validateSize('MaxSize',value);
      obj.MaxSize = value;
    end

    %-----------------------------------------------------------------
    % MergeThreshold set method
    %-----------------------------------------------------------------
    function set.MergeThreshold(obj,value)
      validateattributes( value, ...
        {'numeric'}, {'scalar','>=' 0, 'real','integer',...
        'nonempty','nonsparse','finite'},...
        '','MergeThreshold');
      obj.MergeThreshold = value;
    end

    %-----------------------------------------------------------------
    % TrainingSize get method
    %-----------------------------------------------------------------
    function value = get.TrainingSize(obj)
      if (isSimMode())
        info = obj.pCascadeClassifier.getClassifierInfo();
        value = info.originalWindowSize;
      else
        [originalWindowSize,
~]=vision.internal.buildable.cascadeClassifierBuildable.cascadeClassifier_getClassifier
Info(...
            obj.pCascadeClassifier);
        value = originalWindowSize;
      end
    end
  end
```

```
methods(Access = protected)
    %-------------------------------------------------------------
    % Cross validate properties
    %-------------------------------------------------------------
    function validatePropertiesImpl(obj)

        % validate that MinSize is greater than or equal to the minimum
        % object size used to train the classification model

        if ~isempty(obj.MinSize)
            % obj.TrainingSize calls getClassifierInfo in get.TrainingSize
            coder.internal.errorIf(any(obj.MinSize < obj.TrainingSize), ...
                'vision:ObjectDetector:minSizeLTTrainingSize',...
                obj.TrainingSize(1),obj.TrainingSize(2));
        end

        % validate the MaxSize is greater than the
        % pModel.TrainingSize when MinSize is not specified
        if isempty(obj.MinSize) && ~isempty(obj.MaxSize)
            % obj.TrainingSize calls getClassifierInfo in get.TrainingSize
            coder.internal.errorIf(any(obj.TrainingSize >= obj.MaxSize), ...
                'vision:ObjectDetector:modelMinSizeGTMaxSize',...
                obj.TrainingSize(1),obj.TrainingSize(2));
        end

        % validate that MinSize < MaxSize
        if ~isempty(obj.MaxSize) && ~isempty(obj.MinSize)
            coder.internal.errorIf(any(obj.MinSize >= obj.MaxSize), ...
                'vision:ObjectDetector:minSizeGTMaxSize');
        end

    end

    %-------------------------------------------------------------
    % Validate inputs to STEP method
    %-------------------------------------------------------------
    function validateInputsImpl(obj,I,varargin)
        validateattributes(I,...
            {'uint8','uint16','double','single','int16'},...
            {'real','nonsparse'},...
            '','',2);
        coder.internal.errorIf(~any(ndims(I)==[2 3]), ...
            'vision:dims:imageNot2DorRGB');
```

```
    sz = size(I);
    coder.internal.errorIf(ndims(I)==3 && sz(3) ~= 3,...
        'vision:dims:imageNot2DorRGB');

    if obj.UseROI
        vision.internal.detector.checkROI(varargin{1},size(I));
    end
end

%--------------------------------------------------------------------
% STEP method implementation
%--------------------------------------------------------------------
function bboxes = stepImpl(obj,I,varargin)

    if obj.UseROI
        roi = varargin{1};
    else
        roi = zeros(0,4);
    end

    Iroi = vision.internal.detector.cropImageIfRequested(I, roi, obj.UseROI);

    Iu8  = im2uint8(Iroi);

    grayImage  = convertToGrayscale(obj, Iu8);

    if (isSimMode())

        bboxes = double(obj.pCascadeClassifier.detectMultiScale(grayImage, ...
            double(obj.ScaleFactor), ...
            uint32(obj.MergeThreshold), ...
            int32(obj.MinSize), ...
            int32(obj.MaxSize)));
    else
        if isempty(obj.MinSize)
            obj_MinSize = [0 0];
        else
            obj_MinSize = obj.MinSize;
        end
        if isempty(obj.MaxSize)
            obj_MaxSize = [0 0];
        else
            obj_MaxSize = obj.MaxSize;
        end
```

```
            bboxes =
vision.internal.buildable.cascadeClassifierBuildable.cascadeClassifier_detectMultiScal
e(obj.pCascadeClassifier ,...
                grayImage, ...
                double(obj.ScaleFactor), ...
                uint32(obj.MergeThreshold), ...
                int32(obj_MinSize), ...
                int32(obj_MaxSize));
        end

        bboxes(:,1:2) = vision.internal.detector.addOffsetForROI(bboxes(:,1:2), roi,
obj.UseROI);

    end

    %--------------------------------------------------------------------
    % Release method implementation
    %--------------------------------------------------------------------
    function releaseImpl(obj)
        if ~isSimMode()
            % delete cascadeClassifier object

vision.internal.buildable.cascadeClassifierBuildable.cascadeClassifier_deleteObj(obj.p
CascadeClassifier);
        end
    end

    %--------------------------------------------------------------------
    % Custom save/load method
    %--------------------------------------------------------------------
    function s = saveObjectImpl(obj)
        s = saveObjectImpl@matlab.System(obj);
    end

    function loadObjectImpl(obj,s, ~)

        coder.extrinsic('exist');

        obj.ScaleFactor = s.ScaleFactor;
        obj.MinSize = s.MinSize;
        obj.MaxSize = s.MaxSize;
        obj.MergeThreshold = s.MergeThreshold;

        invalidModel = false;
```

```
        if isfield(s,'ClassificationModel') && ischar(s.ClassificationModel)
            if (~isSupportedModel(s.ClassificationModel))

                file_exists = coder.internal.const(exist(s.ClassificationModel,'file')) == 2;
                if (~file_exists)
                    invalidModel = true;
                    % error while setting the ClassificationModel
                    % throw a warning and leave ClassificationModel set to default
                    warning(...
                        message('vision:ObjectDetector:modelNotFoundOnLoad',...
                        s.ClassificationModel,'FrontalFaceCART'));
                end
            end
        end

        if (~invalidModel)
            obj.ClassificationModel = s.ClassificationModel;
        end

        if isfield(s, 'UseROI') % UseROI added in R2015a
            obj.UseROI = s.UseROI;
        else
            obj.UseROI = false;
        end

    end
    %--------------------------------------------------------------------
    % Initialize classification model
    %--------------------------------------------------------------------
    function loadXMLFromClassModel(obj)
        if (isSimMode())
            obj.pCascadeClassifier.load(obj.getModelPath(obj.ClassificationModel));
        else
            ClassificationModelPath =
coder.internal.const(obj.getModelPath(obj.ClassificationModel));

vision.internal.buildable.cascadeClassifierBuildable.cascadeClassifier_load(...
                obj.pCascadeClassifier, [ClassificationModelPath char(0)]);
            % for packNGo, we need to put the model file in the zip
            % folder. 'load' function first looks for the model file in
            % the original path, if it is not found there, it searches
            % for the file in the current folder.

coder.updateBuildInfo('addNonBuildFiles',ClassificationModelPath,'','BlockModules');
```

```
    end
end
```

- *CONCLUSION*:---- In this paper, I have discussed about the commonly used face detection methods. These methods signify the importance and utility for different applications. Since these methods are dynamic, much advancement is made every day to achieve accurate and true face detection. This approach of detecting face was derived by viola and jones to increase computational efficiency. All the smart phones and digital cameras manufacturers are adapting these techniques to enhance the focus. Har and adaboost features help the algorithm to detect the face in a fraction of second.