HOME AUTOMATION SYSTEM USING HAND GESTURES:-

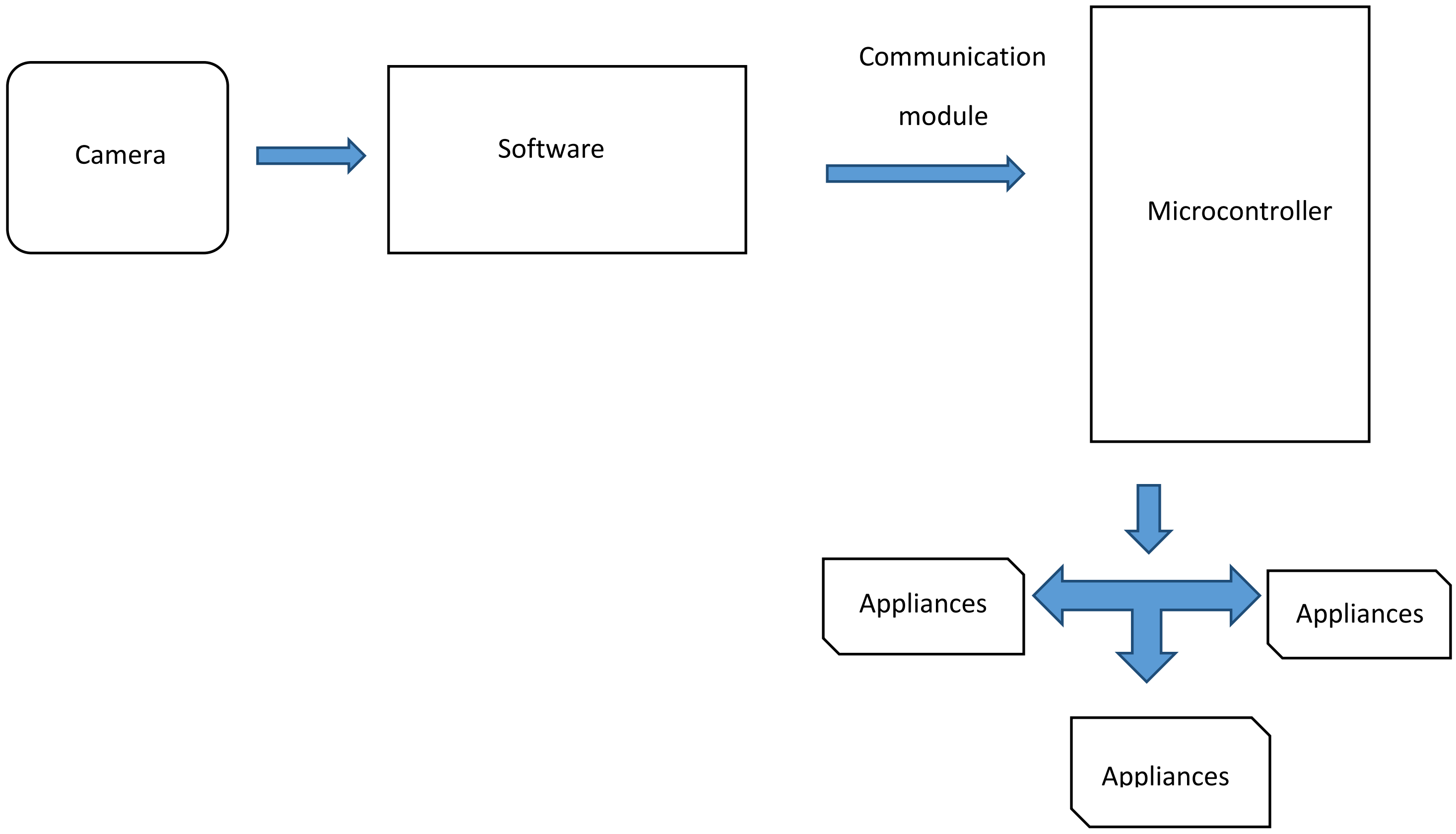
NAME	SECTION	ENROLLMENT NUMBER
RISHABH KUMAR KANDOI	S6	U101115FCS283
SHAILESH MOHTA	S6	U101115FCS305
SHIVARAMA DEVERASETTY	S1	U101115FCS298
SAKSHI DWIVEDI	S1	U101115FEC273

ABSTRACT:-

Gestures play a major role in the daily activities of human life, especially during communication, providing an easy understanding. Gesture recognition refers to recognizing meaningful expressions of motion by a human, involving arms, hands, head/body. Between all the gestures hand gestures helps us to express more in less time. And moreover in today's developed Era the Human-machine interface has developed a lot mainly employing hand gestures. As in present controlling the home appliances using an infrared remote has been common and moreover it's not so different from using a remote to operate the appliances.

Here, in our project we propose an application for hand gesture recognition, of a limited set of hand gestures, for operating low voltage appliances used as a replacement for the actual home appliances. As of now the hand gesture recognition is tough in its own form. We have considered a fixed number of gestures and a reasonable environment in order to achieve the gesture detection and tried to produce a compelling way for gesture detection. Our approach contains steps for recognizing the hand region, contour extraction, locating the edges and counting the number of edges to recognize the gesture and finally implementing the corresponding action on the hardware. When we come to the hardware part which consists of a microcontroller which reads the data given by the hand gesture detection software through a communication module and the microcontroller takes the necessary action on the appliances.

BLOCK DIAGRAM:-



WHAT IS HOME AUTOMATION SYSTEM?

Home automation system is one of the automation systems, which is used for controlling home appliances automatically(sometimes remotely) with the help of various control systems. The home automation systems are used for controlling the indoor & outdoor lights, heat, ventilation, air conditioning in the house, to lock or open the doors & gates, to control electrical & electronic appliances and so on using various control systems with appropriate sensors.

WHAT IS OPEN CV?

As explained in wiki,

***OpenCV** (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing.*

WHAT IS PYTHON ?

Python is a widely used general-purpose, high-level programming language.Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. The language provides constructs intended to enable clear programs on both a small and large scale.

GESTURE RECOGNITION: -

It can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse.

How can we realize hand gesture recognition with opencv and python ?

- *First we will extract ROI(region of interest) from the input frames in our case hand palm.*
- *Then find the contour, draw the convex hulland find the convexity defects depending upon the number of defects find the gestures.*

To many technical terms we will go step by step.

Platform: *Python 2.7*

Libraries: *OpenCV (any version), Numpy, math, serial (from pyserial module)*

Hardware Requirements: *Camera/Webcam, ………………….*

Installation procedure **:**

1) First install Python 2.7. Leave all settings as default. In that case, Python will be installed in default folder C:\Python27\

2) 2) Download Pycharm ide for running python programs. It is not a compulsory step, but it is one of the most easy way to work with python. This is because you can download modules directly from Pycharm>settings>Project >Project Interpreter>'plus sign'>install required module or package.

2) Now install Numpy module. Again leave everything default. Numpy will find Python directory and will be installed to most appropriate folder. Or else you can directly download it inside Pycharm.

3) Now double-click OpenCV.exe. It will ask for extraction folder. Give it as just C:\. It will extract all files to C:\opencv\ . Wait until everything is extracted.

4) Now copy everything in the folder C:\opencv\build\python\x86\2.7\ (most probably, there will be only one file cv2.pyd) and paste it in the folder C:\Python27\Lib\site-packages\

5) Now open your "Python IDLE" (from Start > All Programmes > Python 2.7 > Python IDLE) and just type following :

```
import cv2
```

If everything OK, it will import cv2 module, otherwise an error message will be shown.

Code :-

```
import cv2
import numpy as np
import math
import serial

j=0

cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
    crop_img = img[100:300, 100:300]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    cv2.imshow('Thresholded', thresh1)

    (version, _, _) = cv2.__version__.split('.')

    if version is '3':
        image, contours, hierarchy = cv2.findContours(thresh1.copy(), \
               cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    elif version is '2':
        contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
               cv2.CHAIN_APPROX_NONE)
    cnt = max(contours, key = lambda x: cv2.contourArea(x))


    x,y,w,h = cv2.boundingRect(cnt)

    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)

    hull = cv2.convexHull(cnt)

    drawing = np.zeros(crop_img.shape,np.uint8)

    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)

    cv2.drawContours(drawing,[hull],0,(0,0,255),0)

    hull = cv2.convexHull(cnt,returnPoints = False)

    defects = cv2.convexityDefects(cnt,hull)

    count_defects = 0

    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)



    #########################################to transfer output through serial
port to hardware

    #import serial

    ser = serial.Serial(port='COM4', baudrate=9600)     # port name is visible

                                  # in device manager, take baudrate

                                            # same everywhere

 #########################################




    for i in range(defects.shape[0]):

        s,e,f,d = defects[i,0]

        start = tuple(cnt[s][0])

        end = tuple(cnt[e][0])

        far = tuple(cnt[f][0])

        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)

        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
        #dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        #cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects == 3:
        cv2.putText(img,"3", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        j=3;
        # write the data
        #ser.write(b'1')
    elif count_defects == 2:
        cv2.putText(img, "2", (5,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        if j==3:
            print('2')
            j=j+1
        # write the data
        ser.write(b'2')
    elif count_defects == 1:
        cv2.putText(img,"1", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        if j==3:
            print('1')
            j=j+1
        # write the data
        ser.write(b'1')
    elif count_defects == 4:
        cv2.putText(img,"4", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        if j == 3:
            print('4')
            j = j + 1
            # write the data
        ser.write(b'4')
    else:
        cv2.putText(img,"5", (50,50),\
                    cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
        if j == 3:
            print('5')
            j = j + 1
            # write the data
        ser.write(b'5')

    #cv2.imshow('drawing', drawing)
    #cv2.imshow('end', crop_img)
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
    k = cv2.waitKey(10)
    if k == 27:
        break
```

About Python Code: -

1. Capture frames and convert to grayscale

Our ROI is the the hand region, so we capture the images of the hand and convert them to grayscale.

Q. Why grayscale ?

A. *We convert an image from RGB to grayscale and then to binary in order to find the ROI i.e. the portion of the image we are further interested for image processing. By doing this our decision becomes binary: "yes the pixel is of interest" or "no the pixel is not of interest".*

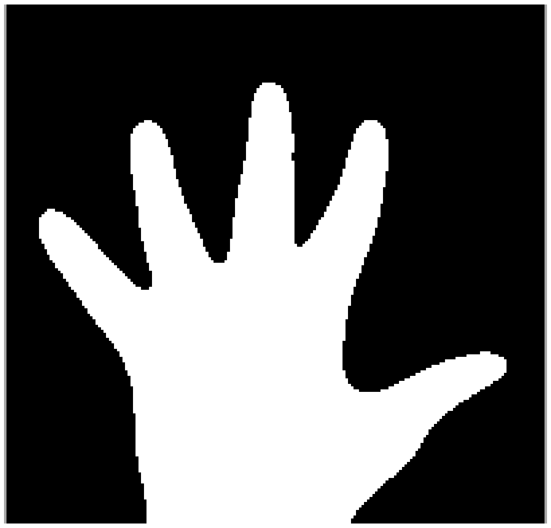
2. Blur image

- *We've used Gaussian Blurring on the original image. We blur the image for smoothing and to reduce noise and details from the image. We are not interested in the details of the image but in the shape of the object to track.*
- *By blurring, we create smooth transition from one color to another and reduce the edge content. We use thresholding for image segmentation, to create binary images from grayscale images.*

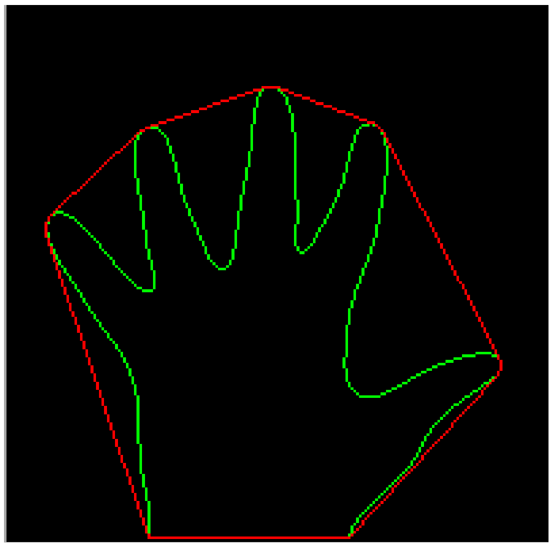
2. *Thresholding*

- *In very basic terms, thresholding is like a Low Pass Filter by allowing only particular color ranges to be highlighted as white while the other colors are suppressed by showing them as black.*

- *I've used Otsu's Binarization method. In this method, OpenCV automatically calculates/approximates the threshold value of a bimodal image from its image histogram. But for optimal results, we may need a clear background in front of the webcam which sometimes may not be possible.*

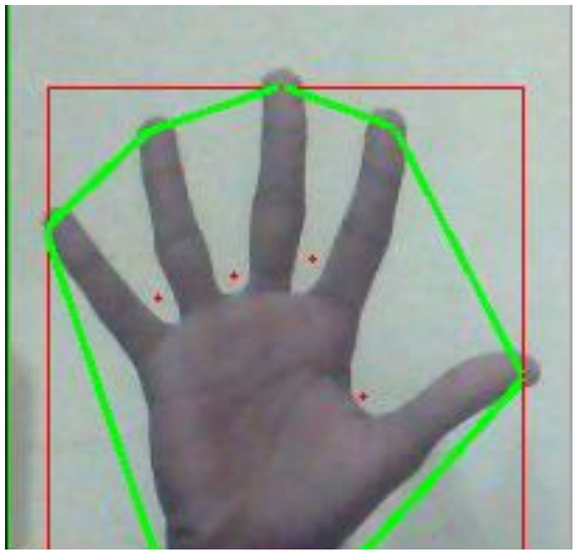


3. *Draw contours*



4. Find convex hull and convexity defects

- *We now find the convex points and the defect points. The convex points are generally, the tip of the fingers. But there are other convex point too. So, we find convexity defects, which is the deepest point of deviation on the contour. By this we can find the number of fingers extended and then we can perform different functions according to the number of fingers extended.*



Some Important Functions Used **:**

cvtColor

Converts an image from one color space to another.

C++: void **cvtColor**(InputArray **src**, OutputArray **dst**, int **code**, int **dstCn**=0)

Python: cv2.**cvtColor**(src, code[, dst[, dstCn]]) → dst

Parameters:

src – input image: 8-bit unsigned, 16-bit unsigned (`CV_16UC...`), or single-precision floating-point.
dst – output image of the same size and depth as `src`.
code – color space conversion code (see the description below).
dstCn – number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from `src` and `code` .

The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

The conventional ranges for R, G, and B channel values are:

- 0 to 255 for `CV_8U` images
- 0 to 65535 for `CV_16U` images
- 0 to 1 for `CV_32F` images

threshold

Applies a fixed-level threshold to each array element.

C++: double **threshold**(InputArray **src**, OutputArray **dst**, double **thresh**, double **maxval**, int **type**)

Python: cv2.**threshold**(src, thresh, maxval, type[, dst]) → retval, dst

Parameters:

src – input array (single-channel, 8-bit or 32-bit floating point).
dst – output array of the same size and type as `src`.
thresh – threshold value.
maxval – maximum value to use with the `THRESH_BINARY` and `THRESH_BINARY_INV` thresholding types.
type – thresholding type (see the details below).

The function applies fixed-level thresholding to a single-channel array. The function is typically used to get a bi-level (binary) image out of a grayscale image (**`compare()`** could be also used for this purpose) or for removing a noise, that is, filtering out pixels with too small or too large values. There are several types of thresholding supported by the function.

Gaussian Blur

Blurs an image using a Gaussian filter.

C++: void GaussianBlur(InputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int **borderType**=BORDER_DEFAULT)

Python:cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]]) → dst

Parameters:

src – input image; the image can have any number of channels, which are processed independently, but the depth should be `CV_8U`, `CV_16U`, `CV_16S`, `CV_32F` or `CV_64F`.
dst – output image of the same size and type as `src`.
ksize – Gaussian kernel size. `ksize.width` and `ksize.height` can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from `sigma*` .
sigmaX – Gaussian kernel standard deviation in X direction.

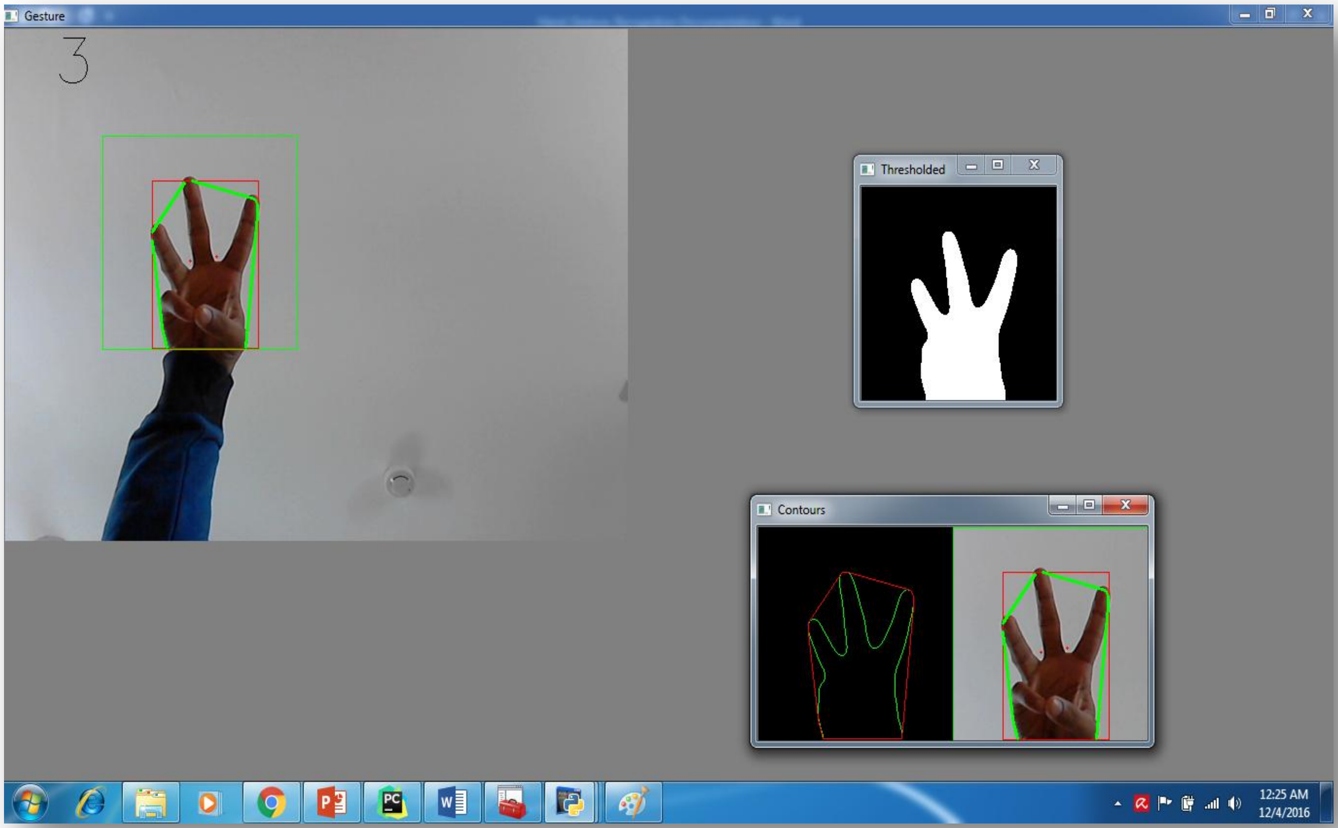
sigmaY – Gaussian kernel standard deviation in Y direction;
if `sigmaY` is zero, it is set to be equal to `sigmaX`, if both sigmas are zeros, they are computed
from `ksize.width` and `ksize.height` , respectively
(see **`getGaussianKernel()`** for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of `ksize`, `sigmaX`, and `sigmaY`.
borderType – pixel extrapolation method
(see **`borderInterpolate()`** for details).

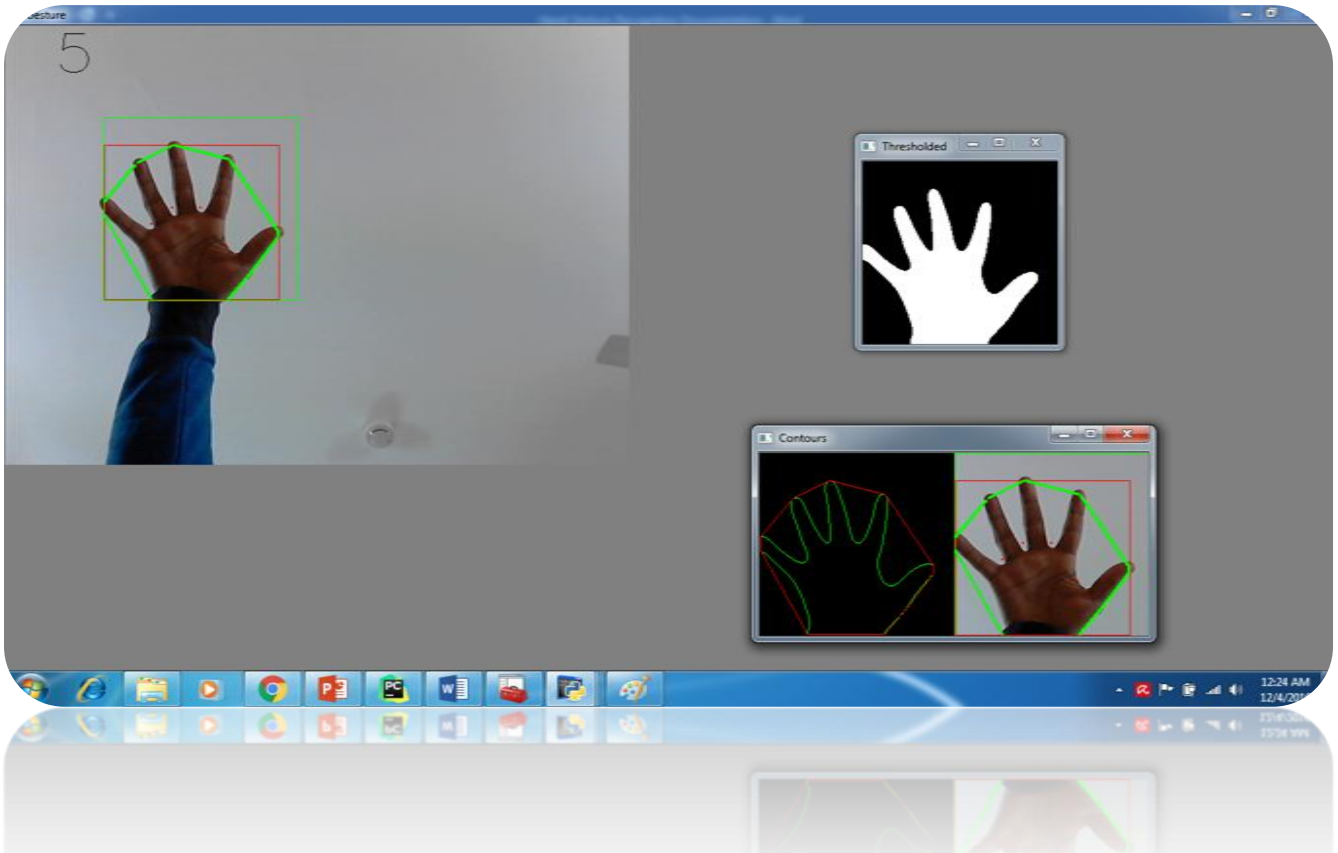
The function convolves the source image with the specified Gaussian kernel. In-place filtering is supported

.>>**At the end, number of count_defects tells the number of fingers giving the feature of finger counting in real-time.**

RESULTS :-

Illustrated in the next page are some of the screenshots of hand recognition and gesture being detected as required.


Gesture
3
Thresholded
Contours
12:25 AM
12/4/2016


Gesture
5
Thresholded
Contours
12:24 AM

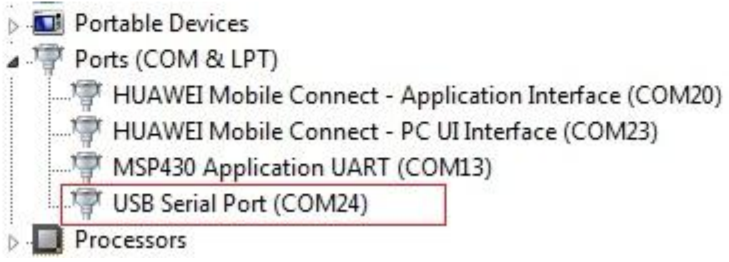
Python serial port communication

Although serial port has been around for a while, it is still very common as a way to communicate between computers and various electronic devices. In Windows XP, there is a tool called HyperTerminal, which allows you to connect to a device through a serial port. However, you do not have HyperTerminal in Windows 7 or newer version. There are many alternatives for doing the same job, such as Terminal.exe (I used this to check serial communication is working or not). We can also do the same thing using very simple Python code.

Here we will write a Python code that allows us to manually enter serial command to the device and show the response from it. Using Python gives us more flexibility because we can always modify the code if the serial command is to be sent repetitively. Thus, it is easier to use Python than HyperTerminal if you want to automatize the task.

Install pySerial module before we start.In the Python code, we start with importing the essential pySerial module. The port name will be something like COM1 on Windows, if you use a Mac, it might be something like /dev/tty.name, where 'name' can be different for different devices.

*Please give the **COM number corresponding to your Serial port** or USB to Serial Converter instead of COM24.*


Portable Devices
Ports (COM & LPT)
HUAWEI Mobile Connect - Application Interface (COM20)
HUAWEI Mobile Connect - PC UI Interface (COM23)
MSP430 Application UART (COM13)
USB Serial Port (COM24)
Processors

We need to setup the baud rate, which is the transmission speed (baudrate should be same as you may have written in your microcontroller code). Serial port has many other parameters, depending on what device we use. In this example we are communicating with a PIC16F877A microcontroller via RS232 port. I will just assign the baud rate and leave everything else as default.

The last part of the code takes user's input and convert if to an ascii string, and the 'write()' method sends it to the device. Note that carriage return characters '\r' and '\n' are appended to the end of the string such that the device knows when to perform the action.

Class `serial.Serial`

`__init__`(*port=None*, *baudrate=9600*, *bytesize=EIGHTBITS*, *parity=PARITY_NONE*, *stopbits=STOPBITS_ONE*, *timeout=None*, *xonxoff=False*, *rtscts=False*, *write_timeout=None*, *dsrdtr=False*, *inter_byte_timeout=None*)

Parameters:

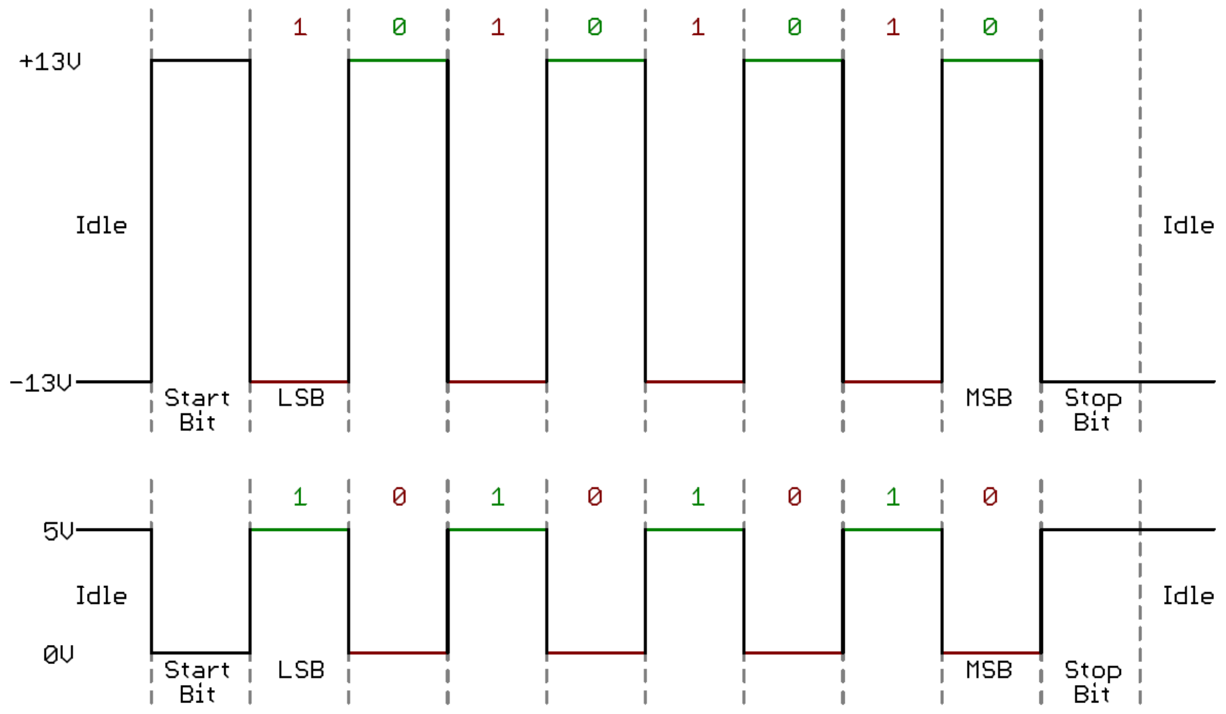
- **port** – Device name or `None`.
- **baudrate** (*int*) – Baud rate such as 9600 or 115200 etc.
- **bytesize** – Number of data bits. Possible values: `FIVEBITS`, `SIXBITS`, `SEVENBITS`, `EIGHTBITS`
- **parity** – Enable parity checking. Possible values: `PARITY_NONE`, `PARITY_EVEN`, `PARITY_ODD` `PARITY_MARK`, `PARITY_SPACE`
- **stopbits** – Number of stop bits. Possible values: `STOPBITS_ONE`, `STOPBITS_ONE_POINT_FIVE`, `STOPBITS_TWO`
- **timeout** (*float*) – Set a read timeout value.
- **xonxoff** (*bool*) – Enable software flow control.
- **rtscts** (*bool*) – Enable hardware (RTS/CTS) flow control.
- **dsrdtr** (*bool*) – Enable hardware (DSR/DTR) flow control.
- **write_timeout** (*float*) – Set a write timeout value.
- **inter_byte_timeout** (*float*) – Inter-character timeout, `None` to disable (default).

Connecting Pc/Laptop With Your Application (Hardware):-

*This method of serial communication is sometimes referred to as **TTL serial** (transistor-transistor logic). Serial communication at a TTL level will always remain between the limits of **0V and Vcc**, which is often 5V or 3.3V. A logic high ('1') is represented by Vcc, while a logic low ('0') is 0V.*

*The serial port on your computer (if it's lucky enough to have one, they're quickly becoming a relic) complies with the **RS-232** (Recommended Standard 232) telecommunications standard. RS-232 signals are similar to your microcontroller's serial signals in that they transmit one bit at a time, at a specific baud rate, with or without parity and/or stop bits. The two differ solely at a hardware level. By the RS-232 standard a logic high ('1') is* *represented by a negative voltage – anywhere from -3 to -25V – while a logic low ('0') transmits a positive voltage that can be anywhere from +3 to +25V. On most PCs these signals swing from* ***-13 to +13V****.*

The more extreme voltages of an RS-232 signal help to make it less susceptible to noise, interference, and degradation. This means that an RS-232 signal can generally travel longer physical distances than their TTL counterparts, while still providing a reliable data transmission.



This timing diagram shows both a TTL (bottom) and RS-232 signal sending 0b01010101

Solutions

So, you may see where the problem lies in interfacing these two signals. To connect these two ports you not only have to ***invert*** *the signals, but you also have to deal with regulating the potentially harmful RS-232 voltages to something that won't destroy a microcontroller's serial pins. There are a handful of solutions to this problem of voltage converting and inverting. The most common, and easiest solution is just plugging a MAX-232 in between the two devices:There are many generic derivatives of the MAX-232. Maxim IC just happened to be the first to market with this neato device (decades ago!) so out of habit, we call all ICs that do similar jobs 'MAX-232s'.*

MicroController PIC16F877A programming :

Programming is done simply in Embedded C. It is similar to C-language. The programme can be written in any editor like text file, Dev cpp, Codeblocks, etc, but the file extension should be .c in any case.

Requirements:

1) PCW software (or any other compiler) to convert your .c file to .hex so that it can be machine readable.

2) MPLAB software to burn (or transfer) your programme (.hex file) to your hardware by connecting a programmer (is specific for different microcontrollers).

Code :-

```
#include<16f877a.h>                          //Header file of the pic microcontroller chosen
#include <stdlib.h>
#use delay(clock=20000000)                   //use clock frequency here used with your pic device
#fuses HS, NOWDT, NOPROTECT, NOBROWNOUT, NOPUT
#zero_ram
#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)    //same baudrate to be used in python
void main(void)                                 //code and put Tx and Rx pin in xmit and rcv
{       char ch;
   while(1)
  {     ch=getch();
      if (ch == '5')
          continue;
       if (ch == '1')
          {
          output_toggle(pin_b1);
          }
       if (ch == '2')
          {
          output_toggle(pin_b2);
          }
         if (ch == '3')
          {
          output_toggle(pin_b3);
          }
          if (ch == '4')
          {
          output_toggle(pin_b4);
          }
          delay_ms(2000);                          //2sec delay to take the next input
       }
  }
```

According to this code, whenever an input of 1,2,3 or 4 is given to the microcontroller, output to the required pin gets low or high according to the previous output (i.e. output gets toggled).

About Output-Low And Output-High:-

When working with ICs and microcontrollers, you'll likely encounter pins that are active-low and pins that are active-high. Simply put, this just describes how the pin is activated. If it's an active-low pin, you must "pull" that pin LOW by connecting it to ground. For an active high pin, you connect it to your HIGH voltage (usually 3.3V/5V).

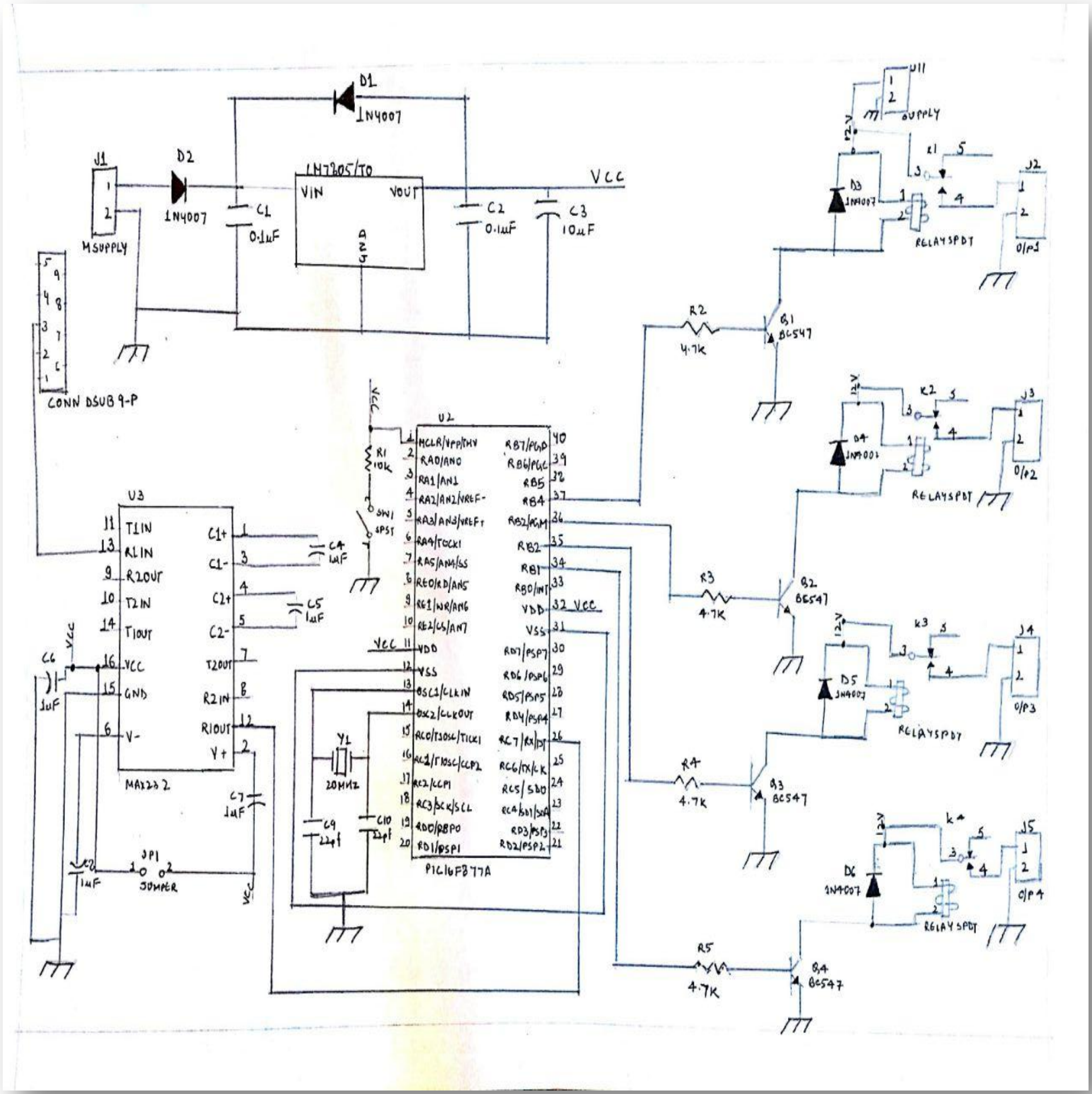
For example, let's say you have a shift register that has a chip enable pin, CE. If you see the CE pin anywhere in the datasheet with a line over it like this, CE, then that pin is active-low. The CE pin would need to be pulled to GND in order for the chip to become enabled. If, however, the CE pin doesn't have a line over it, then it is active high, and it needs to be pulled HIGH in order to enable the pin.

Many ICs will have both active-low and active-high pins intermingled. Just be sure to double check for pin names that have a line over them. The line is used to represent NOT (also known as bar). When something is NOTTED, it changes to the opposite state. So if an active-high input is NOTTED, then it is now active-low. Simple as that!

Since PIC16F877A has EEPROM (almost all microcontrollers has it), i.e the memory can be erased again and again, so you can burn your programme as many times as you require.

HARDWARE

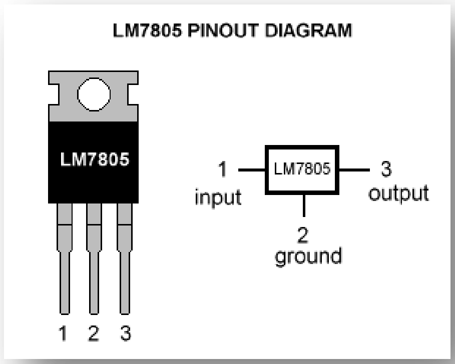
SCHEMATIC



SUPPLY SECTION:-

A 12v is supplied to a CON 2 which goes to LM7805 and gets dropped to 5v also some capacitors have been connected to work as a filter and providing a pure DC voltage without any ripples.

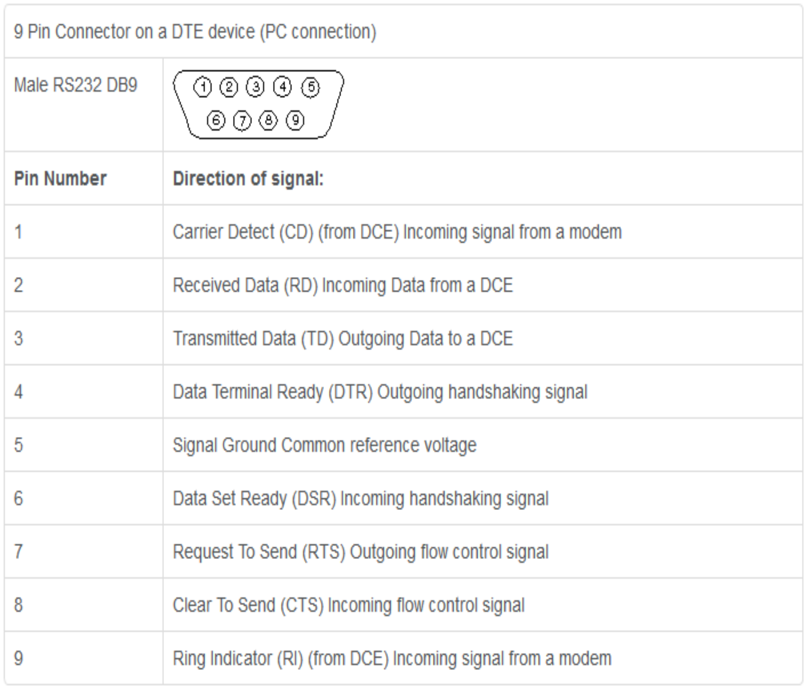
LM7805 belong to a series of fixed voltage integrated circuit voltage regulators designed for elimination of noises. The internal current limiting and thermal shut down features of these regulators make them immune to overloads, also these devices can be used with external components to obtain adjustable output voltages and currents. Each regulator can deliver up to 1.5A of output current.



Diodes are also connected in parallel for providing safety to LM7805 in reverse condition i.e. If the output voltage is greater than 7v, the emitter base junction of the series pass element could breakdown and be damaged so to prevent this a diode shunt is used. This 5v is given to the places where required (MAX232 & PIC). Also the 12v given to the CON has also been given to the relays.

CONTROL SECTION:-

When the gesture is detected through the web cam, it sends the character through USB to RS232 Converter. RS232 is a standard for serial communication transmission of data. It defines the signal connecting between DTE(Data Terminal Equipment) i.e. PC and DCE(Data Circuit- terminating Equipment) i.e. Microcontroller. A serial port sends and receives data *one bit at a time over one wire. While it takes eight times as long to transfer each byte of data this way but only a few wires are required. For full duplex communication only three wires one to send, one to receive and a common signal ground wire is required. Once the start bit has been sent, then the transmitter sends the actual data bits. Both the receiver and transmitter must agree on the number of data bits and baud rate ().The **baud rate** is the **rate** at which information is transferred in a communication channel. In the serial port context, "9600 **baud**" means that the serial port is capable of transferring a maximum of 9600 bits per second.After the data bit has been transmitted, a stop bit is sent which has a value of 1-.   The data comes at the Tx (RIN pin 3) of the DB 9 and is received first by the IC MAX232 at is pin 13(RIN).*

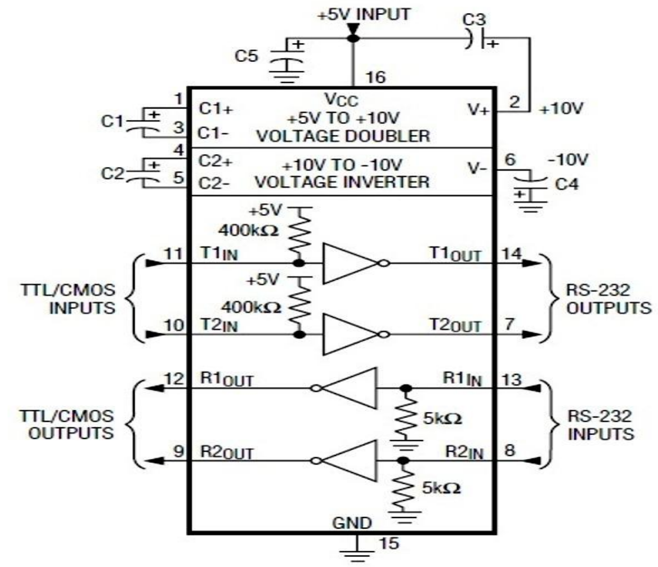


9 Pin Connector on a DTE device (PC connection)	
Male RS232 DB9	① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
Pin Number	**Direction of signal:**
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

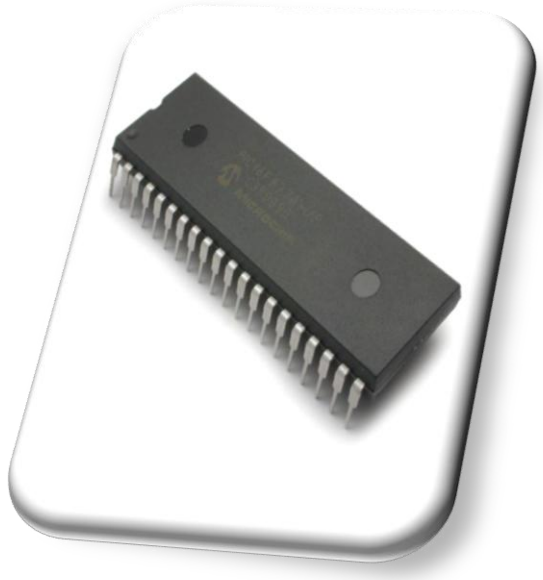
MAX232 DUAL TRANSMITTER/ RECEIVER:-

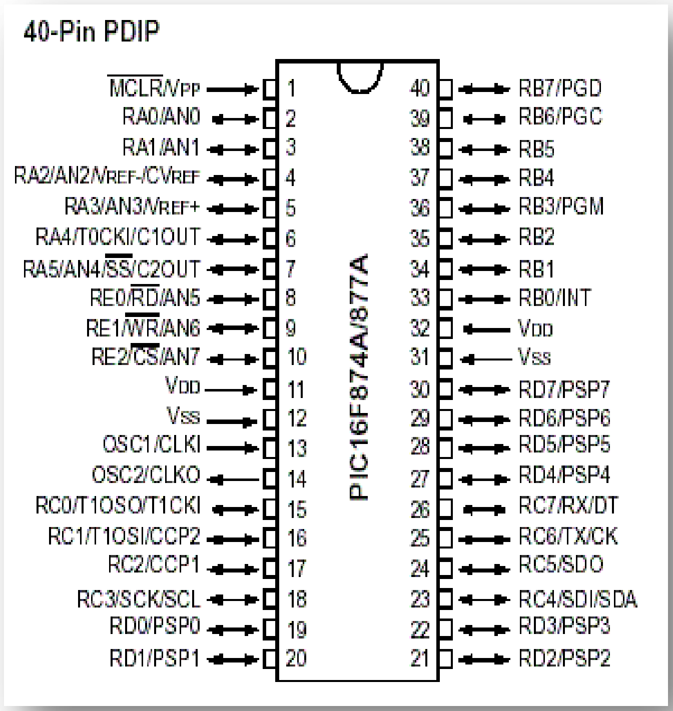
MAX232 is a 16 pin DIP dual transmitter/ receiver integrated circuit that converts signal from serial port (RS232) to signal suitable for use in TTL-compatible digital logic circuit. The receivers reduce RS inputs i.e.-25v to +25v, to standard 5v TTL levels. The chip contains charge pumps which pumps the voltage to the desired voltage with the help of four capacitors of 1uF. It consists of three blocks, first is the voltage doubler which doubles 5v to 10v using switched capacitors, then this voltage in the second block is inverted from +10v to

-10v and last block i.e. the third block consists of 2 transmitters and receivers which actually converts the voltage levels.



After voltage levels shift, the signal is send to PIC16F877a IC.





The PIC is a 40 pin IC which features 256 bytes of EEPROM data memory, self-programming, an ICD, 2 comparators, 8 channels of 10 bit Analog – to – Digital converter, 2 capture/ compare/PWM functions, the synchronous serial port can be configured s either 3-wire Serial Peripheral Interface (SPI) or the 2 wire Inter-Integrated Circuit (I^2C) bus and a Universal Asynchronous Receiver Transmitter (USART). It has an internal clock as well as an external clock also has been provided by a quartz crystal of 20MHz and coupling capacitors at pin no. 13 & 14.

A microcontroller performs mainly four instructions of:

-decoding data

-reading the literals

-processing data

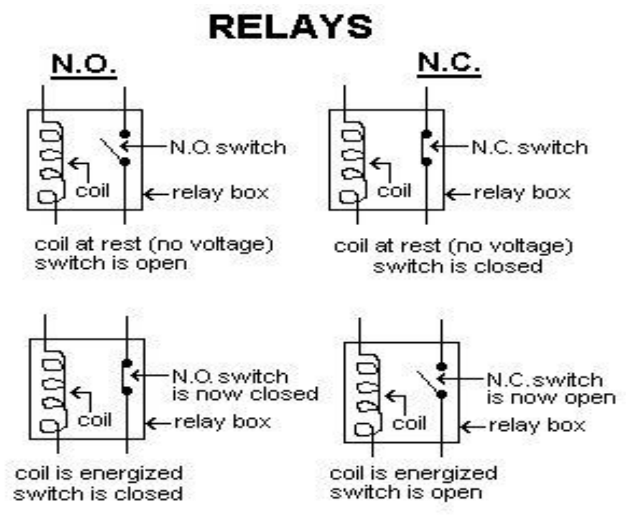
-writing result

*Each instruction cycle takes (Tcy=4/Foc) ns of time i.e. PIC takes four oscillator cycles to complete one instruction cycle. If we take a 20MHz crystal that means period for one osc cycle is (1/20MHz) =50ns. Four of the above instructions make one instruction cycle, so in total it takes 50*4= 200ns.Then from the pins 34,35,36,37 of Port B the loads have been attached.*

PIN NO.	NAME	FUNCTION	DESCRIPTION
1	RE3/MCLR/Vpp	RE3	General purpose input Port E
		MCLR	Reset pin. Low logic level on this pin resets microcontroller
		Vpp	Programming voltage
2	RA0/AN0/ULPWU/C12IN0	RA0	General purpose I/O Port A
		AN0	A/D Channel 0 input
		ULPWU	Stand by mode deactivation input
		C12IN0	Comparator C1 or C2 negative input
3	RA1/AN1/C12IN1	RA1	General purpose I/O Port A
		AN1	A/D Channel 1
		C12IN1	Comparator C1 or C2 negative input
4	RA2/AN2/VREF-/CVREF/C2IN+	RA2	General purpose I/O Port A
		AN2	A/D Channel 2
		VREF-	A/D Negative voltage reference input
		CVREF	Comparator voltage reference output
		C2IN+	Comparator C2 Positive Input
5	RA3/AN3/VREF+/C1IN+	RA3	General purpose I/O Port A
		AN3	A/D Channel 3
		VREF+	A/D Positive Voltage reference Input
		C1IN+	Comparator C1 Positive input
6	RA4/T0CKI/C1OUT	RA4	General purpose I/O Port A
		T0CKI	Timer T0 Clock Input
		C1OUT	comparator C1 output
7	RA5/AN4/SS/C2OUT	RA5	General purpose I/O Port A
		AN4	A/D Channel 4
		SS	SPI module input
		C2OUT	Comparator C2 output
8	RE0/AN5	RE0	General pupose I/O port E
		AN5	A/D Channel 5
9	RE1/AN6	RE1	General pupose I/O port E
		AN6	A/D Channel 6
10	RE2/AN7	RE2	General pupose I/O port E
		AN7	A/D Channel 7
11	Vdd	positive	positive supply
12	Vss	negative	ground(GND)
13	RA7/OSC1/CLKIN	RA7	General purpose I/O Port A
		OSC1	Crystal Oscillator input
		CLKIN	External clock input
14	RA6/OSC2/CLKOUT	RA6	General purpose I/O Port A
		OSC2	Crystal Oscillator input
		CLKO	Fosc/4 output
15	RC0/T1OSO/T1CKI	RC0	General purpose I/O Port C
		T1OSO	Timer T1 oscillator output
		T1CKI	Timer T1 clock input
16	RC1/T1OSO/CCP2	RC1	General purpose I/O Port C
		T1OSI	Timer T1 oscillator input
		CCP2	CCP1 and PWM1 module I/O
17	RC2/P1A/CCP1	RC2	General purpose I/O Port C
		PIA	PWM module output
		CCP1	CCP2 and PWM2 module I/O
18	RC23/SCK/SCL	RC3	General purpose I/O Port C
		SCK	MSSP module clock I/O in SPI mode
		SCL	MSSP module clock I/O in i^2C mode
19	RD0	RD0	General purpose I/O Port D
20	RD1	RD1	General purpose I/O Port D
21	RD2	RD2	General purpose I/O Port D
22	RD3	RD3	General purpose I/O Port D
23	RC4/SDI/SDA	RC4	General purpose I/O Port A
		SDI	MSSP module Data input in SPI mode
		SDA	MSSP module data I/O in I^2C mode
24	RC5/SDO	RC5	General purpose I/O Port C
		SDO	MSSP module Data output in SPI mode
25	RC6/TX/CK	RC6	General purpose I/O Port C
		TX	USART Asynchronous output
		CK	USART Synchronous clock
26	RC7/RX/DT	RC7	General purpose I/O Port C
27	RD4	RD4	General purpose I/O Port D
28	RD5/P1B	RD5	General purpose I/O Port D
		P1B	PWM output
29	RD6/P1C	RD6	General purpose I/O Port D
		P1C	PWM output
30	RD7/P1D	RD7	General purpose I/O Port D
		P1D	PWM output
31	Vss	negative	Ground
32	Vdd	positive	Positive supply
33	RB0/AN12/INT	RB0	General purpose I/O Port B
		AN12	A/D channel 12
		INT	External Interrupt
34	RB1/AN10/C12INT3	RB1	General purpose I/O Port B
		AN10	A/D channel 10
		C12INT3	Comparator C1 or C2 negative input
35	RB2/AN8	RB2	General purpose I/O Port B
		AN8	A/D channel 8
36	RB3/AN9/PGM/C12IN2	RB3	General purpose I/O Port B
		AN9	A/D channel 9
		PGM	Programming enable pin
		C12IN2	Comparator C1 or C2 negative input
37	RB4/AN11	RB4	General purpose I/O Port B
		AN11	A/D channel 11
38	RB5/AN13/T1G	RB5	General purpose I/O Port B
		AN13	A/D channel 13
		T1G	Timer T1 External input
39	RB6/ICSPCLK	RB6	General purpose I/O Port B
		ICSPCLK	Serial Programming clock
40	RB7/ICSPDAT	RB7	General purpose I/O Port B
		ICSPDAT	Programming enable pin

LOAD SECTION:-

From the pin of microcontroller, a resistor of 4.7k has been connected to minimize the base current going to the NPN transistor(BC547). This transistor is working as a switch which here keeps on oscillating between saturation and cut off region when on and off respectively. The emitter is ground and the collector goes to the 12v relay. Relay are basically electromagnetic switch which are used where it is necessary to control a circuit by a separate low power signal i.e. to drive high voltage loads from low current-voltage signals. When the supply is given the common terminal switches from normally closed to normally open terminal and the loads get on. It is necessary to have a diode(1N4007) parallel to the relay in reverse biased. Since an inductor cannot change its current instantly, so the diode provides a path for the current when the coil is switched off, or else a voltage spike will occur causing arcing on switch contacts and possibly destroying transistors.


RELAYS
N.O.
N.C.
N.O. switch
relay box
coil
coil at rest (no voltage) switch is open
N.C. switch
relay box
coil
coil at rest (no voltage) switch is closed
N.O. switch is now closed
relay box
coil
coil is energized switch is closed
N.C. switch is now open
relay box
coil
coil is energized switch is open

APPLICATIONS :-

- *To overcome situations where normal cabling is difficult or financially impractical.*
- *It can be used in home theatre system where short distance communication is required.*
-

- *Suitable for physically impaired people to operate the devices within the room.*
- *It is more secure as it reduces the chance of power overloading ,short circuit of the appliances.*
- *Not only in home automation, hand gesture recognition may find use in many other areas like:*
  To change Tv channels, manipulate volume control in any device, door lock system, security purposes, etc.

Appendix

Variables and Parameters:

Some variables are considered for optimization purposes. Also, there are variables that define certain parameters such as capture box area, threshold values, threshold for finger lengths, threshold for hand radius, etc.
All the parameters are named in such a way that it would be easy to understand what they do in a function. Also, all of the important ones are defined at the very beginning of the document.

References & Tutorials:

1. OpenCV documentation: http://docs.opencv.org/2.4.8/
2. HTML book series, full of basic tutorials on Python. This particular article is very useful to understand how to work with classes in Python: http://learnpythonthehardway.org/book/ex40.html
3. Filtering and smoothening images: http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html
4. Contours in OpenCV: http://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html
5. Pyserial documentation: https://pythonhosted.org/pyserial/
6. PIC16F877A program and burn: https://www.elprocus.com/pic-microcontroller-programming-using-c-language/
7.PIC16F877A DESCRIPTION: http://microcontrollerslab.com/pic16f877a-introduction-features/
8.MAX232 DESCRIPTION: http://www.ti.com/lit/ds/symlink/max232.pdf
9. RELAY DESCRIPTION: http://www.circuitstoday.com/working-of-relays

And Google of course. Internet has a solution to every problem. Google is the key to find the right one.

We tried our best to cover each and every aspect of our project: Hand gesture based home automation.