

# GESTURE CONTROLLED TOUCHSCREEN BASED AUTOMATION SYSTEM

**Jatin Gaur**

*Department of Electronics and Communication Engineering,*

*PEC University of Technology, Chandigarh, (India)*

*Mentored By: Dr. Sukhwinder Singh*

## **ABSTRACT**

*In the proposed paper, graphical programming is used to create an automation system using a resistive touchscreen, basically a handy remote to control the electric and electronic appliances of the considered over apartment and office. A basic comparison algorithm is used to achieve the same. By means of serial communication, the above mentioned technology is implemented and Labview provides the GUI.*

***Keywords: Resistive, Touchscreen, Automation, Serial Communication, Labview, GUI.***

## **I. INTRODUCTION**

An enhanced, low-cost user interface using touch is a valuable feature for a variety of consumer, medical, automotive, and industrial devices. In many consumer applications, designers prefer expensive capacitive touch screens to resistive technologies because they can track a large number of fingers and seem to offer a friendlier interaction with the user. At present, low-cost resistive technologies fill a market niche where only a single touch is required, extremely accurate spatial resolution is obtained. Hence, through this project, our objective is to create a gesture recognition system and henceafter feeding it to a home automation system so as to match the increasing digitalization of the world.

## **II. EXISTING TECHNOLOGIES**

Every work earlier was operated through switches that had to be turned on and off manually. As the technology grows, everything has gone embedded. To bring down the whole of the electrical system of a place, to a screen is an example of embedded system.

## **III. WORKING**

### **3.1 Description**

Basically the project converts the analog voltage coming from the resistive touch screen into a two co-ordinate integer value and sends it to the PC through the microcontroller. The processing code takes these co-ordinates as inputs and draws a white dot for each co-ordinate, on the output screen.

So when you write continuously on the touch screen, the dots would be plotted close enough to make it look like a line or curve.



### 3.2 ATmega16 Code Explanation

-Map the 4 pins of PORTA in the following way using “#define”

```
#define y1 PA1
#define x2 PA2
#define y2 PA3
#define x1 PA4
```

-Initialize the USART and ADC functions of the microcontroller.

-Enter into an infinite while loop

-Configure x1 (PA4) & x2 (PA2) as outputs. Set x1 (PA4) to high (+5V) state and x2 (PA2) to low (GND) state.

-Read the analog voltage at y2 (PA3) using ReadADC(3) command. Store the discrete value in the variable “x”

-Configure y1 (PA1) & y2 (PA3) as outputs. Set y1 (PA1) to high (+5V) state and y2 (PA3) to low (GND) state.

-Read the analog voltage at x1 (PA4) using ReadADC(4) command. Store the discrete value in the variable “y”

-Transmit the co-ordinates in “x, y” format to the PC using WtCoord(x,y) function.

A software ‘Processing’ was used to analyse the output when block diagrams in LabVIEW were in progress.

### 3.3 Processing Code Explanation

1. First we define the output screen size and also fill the background with some color using *size(width,height)* and *background(value)* functions.

Note: I have taken width=690 and height=540. You can take any values but make sure it’s aspect ratio is same as that of the touch screen dimensions.

2. Next we need to create a serial connection, defining the COM port number where the board is connected and also the baud rate which is done by the following lines

```
Serial myport;
```

```
myport = new Serial(this,"COM9",57600);
```

COM9 is where my board is connected to and I have used baud rate=57600 bps since the program should keep up with my speed of writing.

3. Next we need to call a function whenever a data is available at the serial port. Then we need to read the data and store it in a string type variable.

```
void serialEvent(Serial p){
```

```
String stringData=myport.readStringUntil(10);
if(stringData!=null){
    stringData=trim(stringData);
    int data[]=int(split(stringData,''));
    if(data.length==2){
        x=data[0];
        y=data[1];
    }
}
```

Since our ATmega16 is programmed to continuously send the data (line after line), two set of co-ordinates may get into the “read” function same time causing errors. To avoid that we use “*readStringUntil(10)*” (where 10 is the ASCII value for a new line) instead of plain read. This is would help in setting a mark between two different co-ordinate by skipping every time after a new line occurs.

The *trim()* function is used to remove standard whitespace characters such as space, carriage return, and tab.

Example: “1009,1024/r/n” will be converted into “1009,1024”

4. Next we split the string to extract the x and y co-ordinate separately. For this we use the *split(stringData, ',')* function and store the co-ordinates in two different address locations of a integer type array.

Example: “1009,1024” is split into and stored as x=1009 and y=1024

5. Now that we have the raw co-ordinate value, we need to convert them into a sensible range of values corresponding to the screen size we are going to plot our sketch onto. For this we use the *map(value, start1, stop1, start2, stop2)* function. Where “value” is the incoming value to be converted

“start1” is the lower bound of the value's current range

“stop1” is the upper bound of the value's current range

“start2” is the lower bound of the value's target range

“stop2” is the upper bound of the value's target range

Then the converted co-ordinates are stored in variables “xcord” and “ycord”.

6. Lastly we draw a solid circle (with small radius) at the co-ordinate given by xcord, ycord variables using *ellipse()* function.

7. At any point of time you can clear the drawing screen by pressing ‘c’ on the keyboard. This is achieved by using *keyPressed()* function.

### 3.4 Setup Instructions

1. You can either compile the code given using a suitable compiler like WINAVR with ATMEL STUDIO/AVR STUDIO or simply burn the “slate.hex” file given below.

2. Plug in the converter using an USB cable. If you are connecting it for the first time then get the drivers installed and note down the COM port to which the converter is connected. Go to System Properties> Device Manager and look out for the converter by its name to see the COM port number.

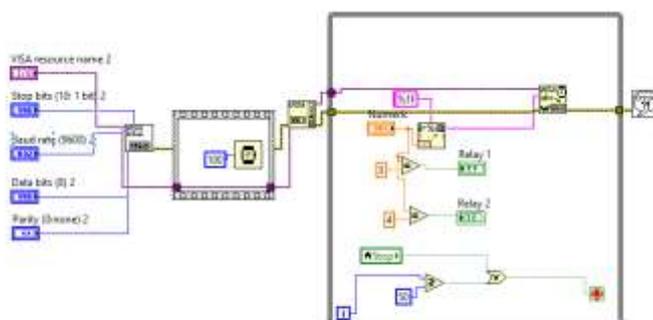
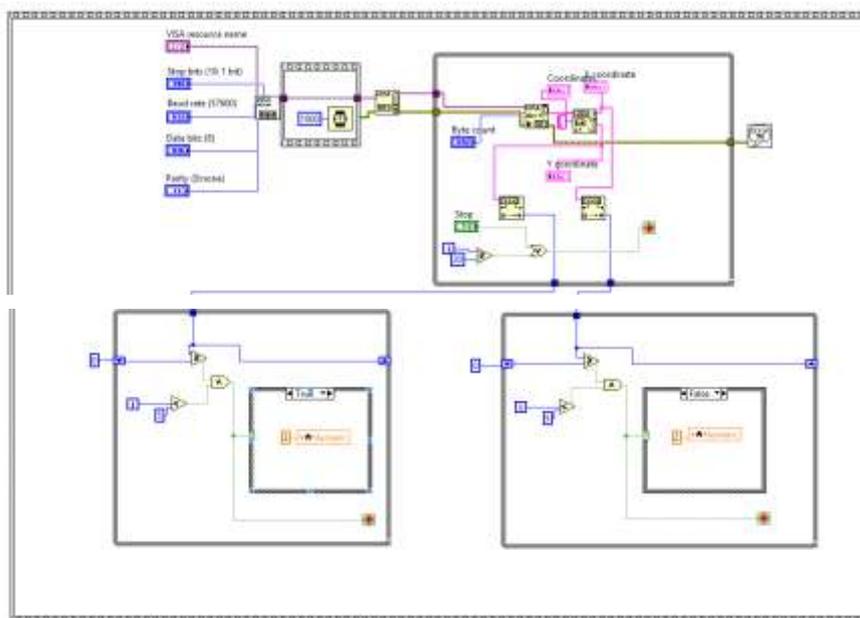
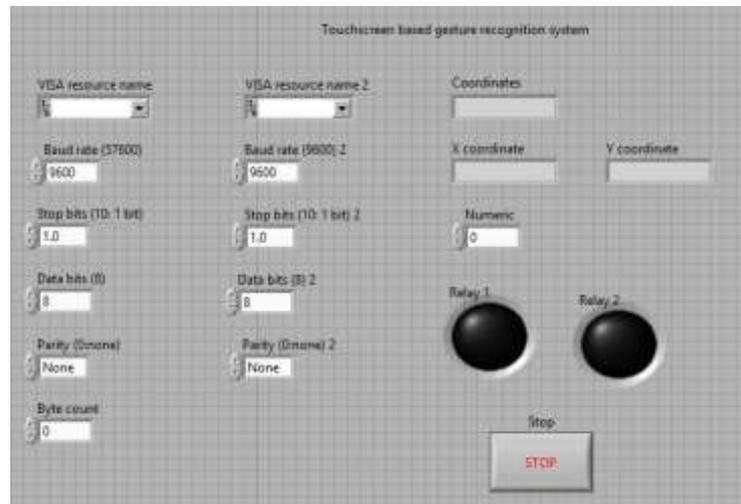
3. Open the processing software and copy paste the code given below. Remember to edit the COM port number.

4. Click on the  button and also turn ON the development board.

5. Now using your finger or a stylus, try drawing something onto the touch screen. If everything is done

properly then you should see something on the output screen window of the processing. The data can in the same way is fed to Labview and henceafter another controller to supply the power to the appliance.

#### IV. INTERFACING



## V. CONCLUSION

The conclusion hence obtained is that the resistive touchscreen on interfacing with the ADC ports of a microcontroller can serve as a input to any software of our choice and henceafter be modified according to application of requirement.

## VI. FUTURE SCOPE AND APPLICATIONS

1. Proper tactile feedback once established would increase sensitivity of touchscreen enormously making its applications suitable in security systems and biometric systems.
2. Technology will become a more organic and natural part of our daily lives, beginning through eyewear, in which we are already seeing in development or eventually, through our bodies becoming more intertwined with technology and hence the widened application in medical field.
3. Interaction with a host of touch sensors wrapped around the body of the phone.
4. interactions will be through the use of voice, gaze, gesture, or spatio-temporal intelligence — the device will know what user wants, before you even have the inkling that one wants to know it.

## REFERENCES

- [1]. [http://www.engr.sjsu.edu/bjfurman/courses/ME106/ME106pdf/A2D\\_ATmega16.pdf](http://www.engr.sjsu.edu/bjfurman/courses/ME106/ME106pdf/A2D_ATmega16.pdf)
- [2]. <http://extremeelectronics.co.in/avr-tutorials/rs232-communication-the-basics/>
- [3]. <http://files.spogel.com/abstracts/p-9385--smart-home-labview.pdf>
- [4]. <http://www.atmel.com/devices/atmega16.aspx>