A

Mini Project Report

On

# Swept Frequency Capacitive Touch Sensing using Micro-Controller

**Naman Kalkhuria (IEC2010065)**

**Prathak Rastogi (IEC2010093)**

**Prashant Mishra (IEC2010096)**

**B.Tech. - 6th Semester (E.C.)**

Under the Guidance of:

**Dr. Ajay Singh Raghuvanshi**

**I.I.I.T.-Allahabad**

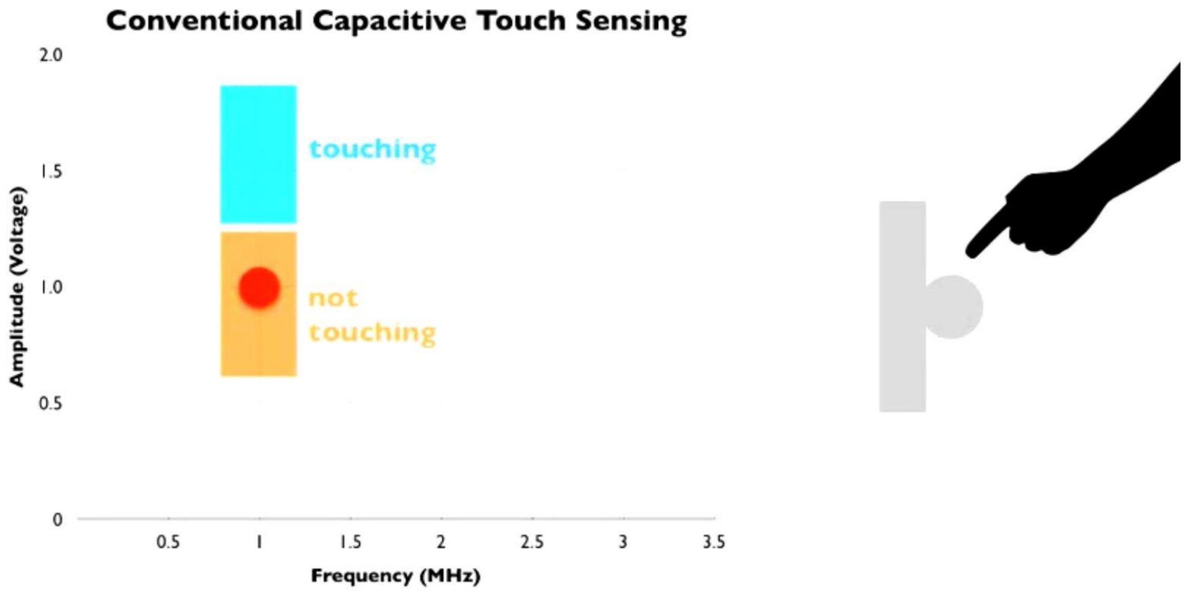
# Indian Institute of Information Technology Allahabad

## TABLE OF CONTENTS:

# LITERATURE SURVEY

In a typical capacitive touch sensor, a conductive object is excited by an electrical signal at a fixed frequency. The sensing circuit monitors the return signal and determines touch events by identifying changes in this signal caused by the electrical properties of the human hand touching the object.

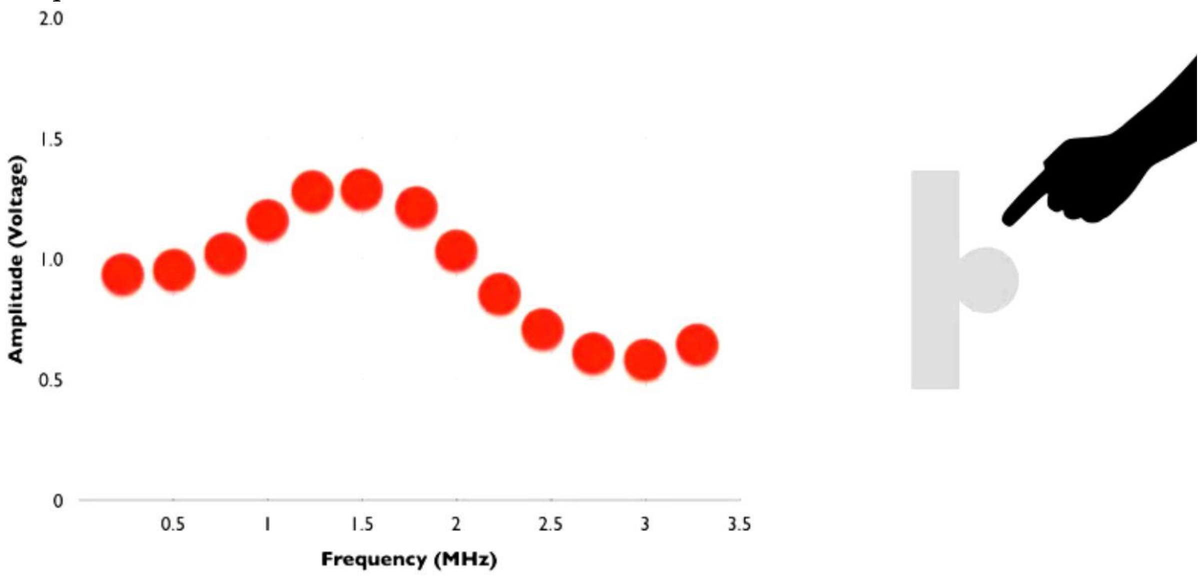
In ***Swept Frequency Capacitive Touch Sensing***, on the other hand, we monitor the response to capacitive human touch over a range of frequencies. Objects excited by an electrical signal respond differently at different frequencies, therefore, the changes in the return signal will also be frequency dependent. Thus, instead of measuring a single data point for each touch event, we measure a multitude of data points at different frequencies. Not only can we determine that a touch event occurred, we can also determine how it occurred. Importantly, this contextual touch information is captured through a single electrode, which could be simply the object itself.



The basic principles of operation in most common capacitive sensing techniques are quite similar: A periodic electrical signal is injected into an electrode forming an oscillating electrical field. As the user’s hand approaches the electrode, a weak capacitive link is formed between the electrode and conductive physiological fluids inside the human hand, altering the signal supplied by the electrode. This happens because the user body introduces an additional path for flow of charges, acting as a charge “sink”. By measuring the degree of this signal change, touch events can be detected.

There is a wide variety of capacitive touch sensing techniques. One important design variable is the choice of signal property that is used to detect touch events, e.g., changes in signal phase or signal amplitude can be used for touch detection. The signal excitation technique is another important design variable. The choice of topology of electrode layouts, the materials used for electrodes and substrates and the specifics of signal measurement resulted in a multitude of capacitive techniques, including charge transfer, surface and projective capacitive, among others.

Capacitive sensing is a malleable and inexpensive technology – all it requires is a simple conductive element that is easy to manufacture and integrate into devices or environments. Consequently, today we find capacitive touch in millions of consumer device controls and touch screens. It has, however, a number of limitations. One important limitation is that capacitive sensing is not particularly expressive – it can only detect when a finger is touching the device and sometimes infer finger proximity. To increase the expressiveness, matrices of electrodes are scanned to create a 2D capacitive image. Such space multiplexing allows the device to capture spatial gestures, hand profiles or even rough 3D shapes. However, this comes at the cost of increased engineering complexity, limiting its applications and precluding ad hoc instrumentation of our living and working spaces. Current capacitive sensors are also limited in materials they can be used with. Typically they cannot be used on the human body or liquids.

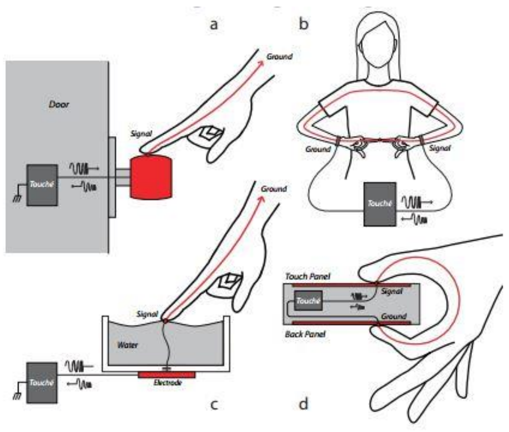


We advocate a different approach to enhance the expressivity of capacitive sensing – by using frequency multiplexing. Instead of using a single, pre-determined frequency, we sense touch by sweeping through a range of frequencies. We refer to the resulting curve as a capacitive profile and demonstrate its ability to expand the vocabulary of interactive touch without increasing the number of electrodes or the complexity of the sensor itself.

## MOTIVATION

In our day-to-day task we come in contact with various types of machines and we always yearn that these machines work according to our will without even telling them to do so. So to make this happen we need a type of interaction with these types of machines.

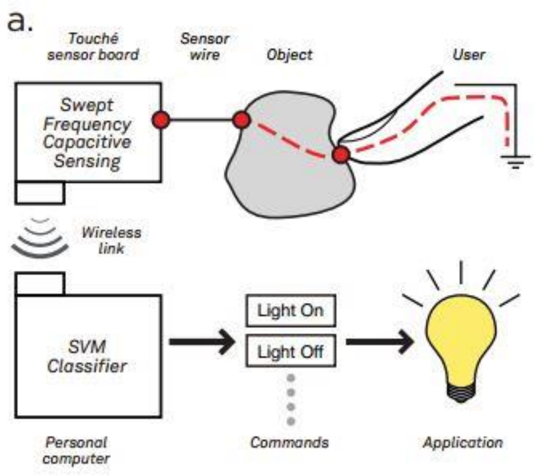
The most common type of interaction we make with objects or machines is touch. So if we can easily make an object sensitive to our touch and make them understand the way we touch; this could help in automating various tasks. By this thought this project came into our mind.



Our project can be implemented in following:

- For detecting the touch on a door knob.
- For detecting configuration of fingers touching to each other.
- For detecting touch on liquid surfaces.
- For detecting the touch and configuration of touch on normal touch sensitive screens.

## PROBLEM FORMULATION



Generally, *swept frequency capacitive sensing* is implemented by Digital Signal Processing and machine learning.

But we are using Micro-controller for implementing this technique of sensing the touch.

## PROPOSED SOLUTION

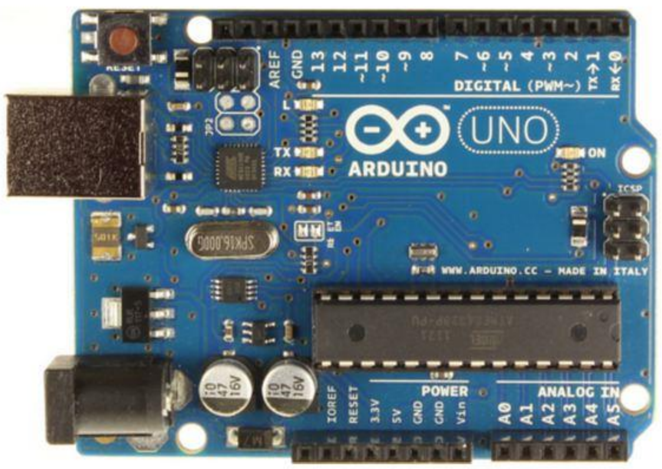
## Methodology:

For implementing this technique of *Swept Frequency Capacitive Sensing*, we are taking the analog input of the sensor into the Arduino board and analyzing it using the processing language in the computer.

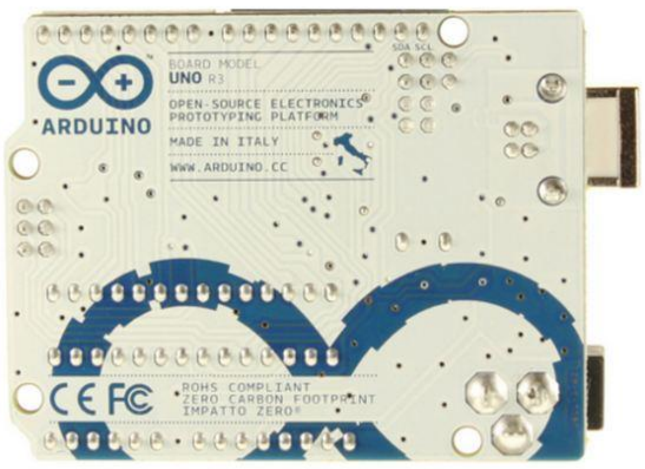
## Various hardware and software components to be use:

1. **MICRO-CONTROLLER BOARD:-** ARDUINO UNO R3

   The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.



Arduino Uno R3 Front



Arduino Uno R3 Back

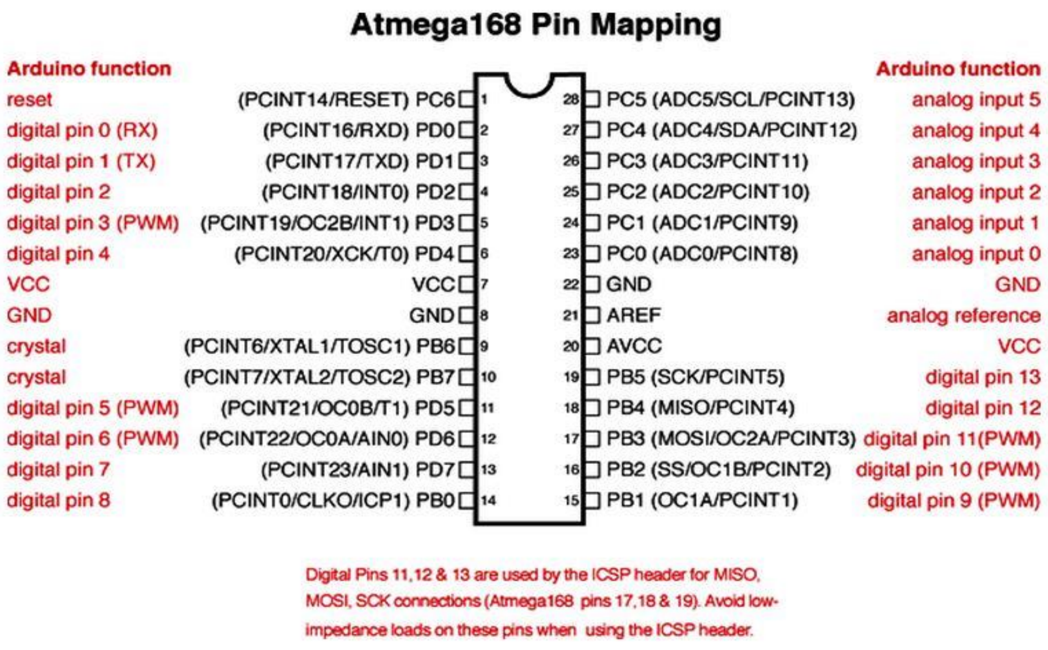
### Summary

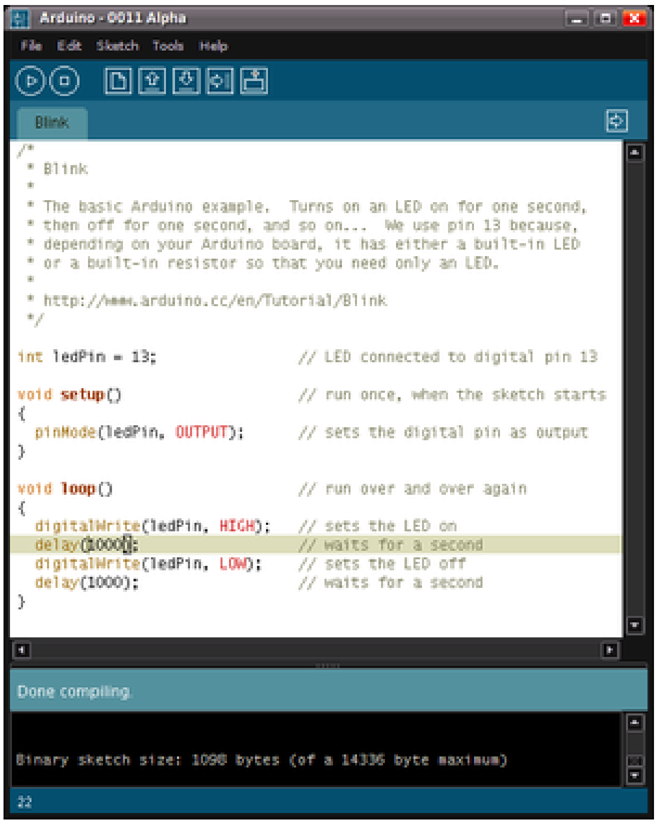
| | |
|---|---|
| Microcontroller | ATmega328 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |

| | |
|---|---|
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328) of which 0.5 KB used by boot loader |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |
| Clock Speed | 16 MHz |

## ATmega168/328-Arduino Pin Mapping

Note that this chart is for the DIP-package chip. The Arduino Mini is based upon a smaller physical IC package that includes two extra ADC pins, which are not available in the DIP-package Arduino implementations.


Atmega168 Pin Mapping
Arduino function
reset (PCINT14/RESET) PC6 1
digital pin 0 (RX) (PCINT16/RXD) PD0 2
digital pin 1 (TX) (PCINT17/TXD) PD1 3
digital pin 2 (PCINT18/INT0) PD2 4
digital pin 3 (PWM) (PCINT19/OC2B/INT1) PD3 5
digital pin 4 (PCINT20/XCK/T0) PD4 6
VCC VCC 7
GND GND 8
crystal (PCINT6/XTAL1/TOSC1) PB6 9
crystal (PCINT7/XTAL2/TOSC2) PB7 10
digital pin 5 (PWM) (PCINT21/OC0B/T1) PD5 11
digital pin 6 (PWM) (PCINT22/OC0A/AIN0) PD6 12
digital pin 7 (PCINT23/AIN1) PD7 13
digital pin 8 (PCINT0/CLKO/ICP1) PB0 14
Arduino function
28 PC5 (ADC5/SCL/PCINT13) analog input 5
27 PC4 (ADC4/SDA/PCINT12) analog input 4
26 PC3 (ADC3/PCINT11) analog input 3
25 PC2 (ADC2/PCINT10) analog input 2
24 PC1 (ADC1/PCINT9) analog input 1
23 PC0 (ADC0/PCINT8) analog input 0
22 GND GND
21 AREF analog reference
20 AVCC VCC
19 PB5 (SCK/PCINT5) digital pin 13
18 PB4 (MISO/PCINT4) digital pin 12
17 PB3 (MOSI/OC2A/PCINT3) digital pin 11(PWM)
16 PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
15 PB1 (OC1A/PCINT1) digital pin 9 (PWM)
Digital Pins 11,12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

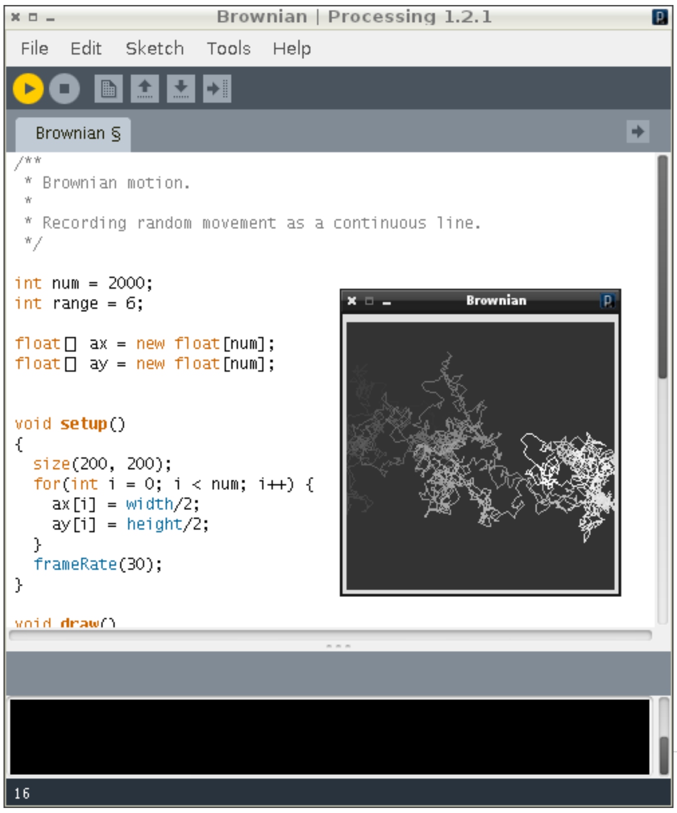
```
Binary sketch size: 1098 bytes (of a 14336 byte maximum)
```

## Programming

The Arduino Uno can be programmed with the Arduino software.
The ATmega328 on the Arduino Uno comes pre-burned with a boot loader that allows us to upload new code to it without the use of an external hardware programmer.

## 2. SOFTWARE USED:

### Processing Programming Language:

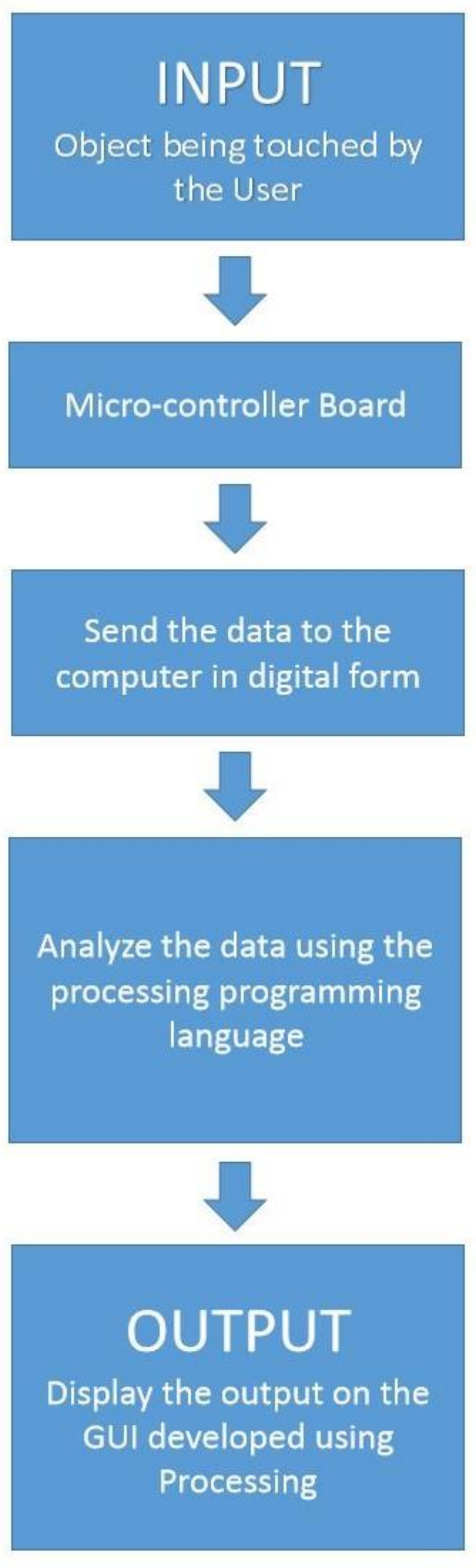


```
/**
 * Brownian motion.
 *
 * Recording random movement as a continuous line.
 */

int num = 2000;
int range = 6;

float[] ax = new float[num];
float[] ay = new float[num];


void setup()
{
  size(200, 200);
  for(int i = 0; i < num; i++) {
    ax[i] = width/2;
    ay[i] = height/2;
  }
  frameRate(30);
}

void draw()
```

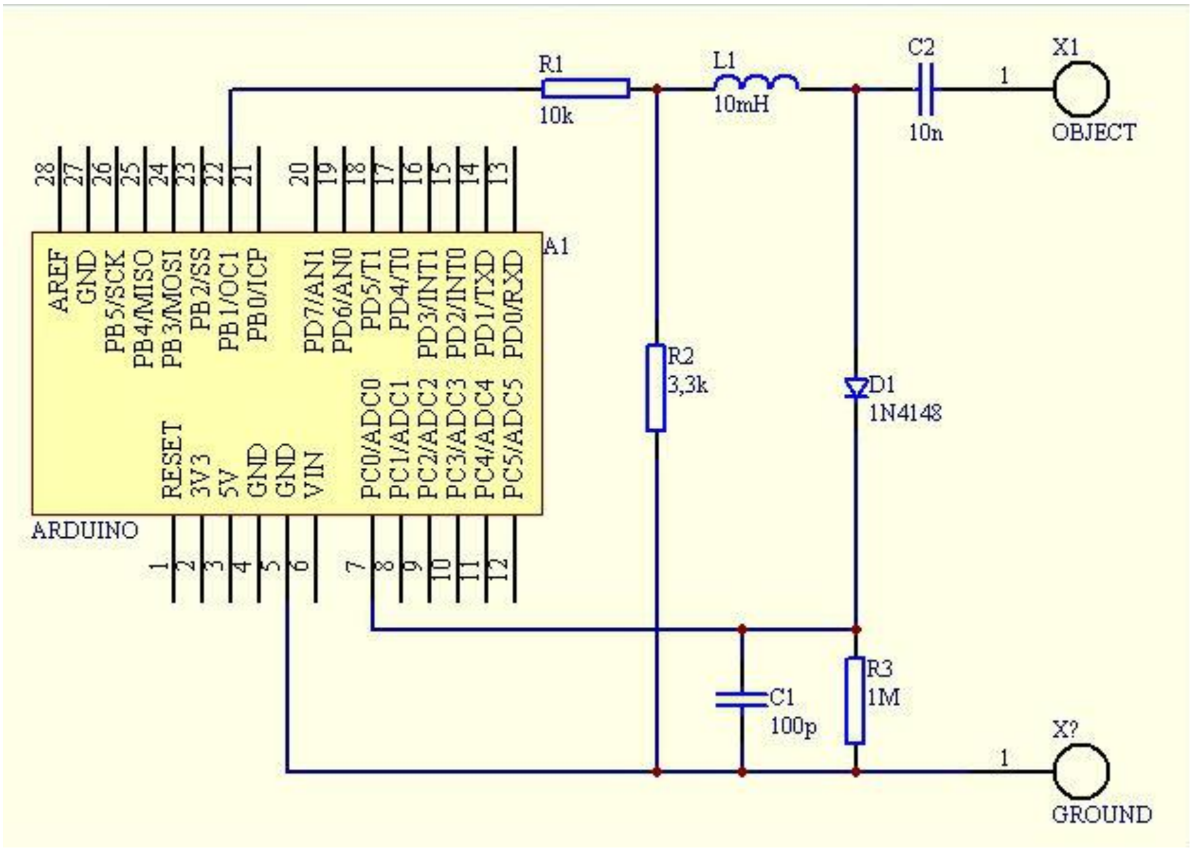
Processing includes a *sketchbook*, a minimal alternative to an integrated development environment (IDE) for organizing projects.

Processing also allows for users to create their own classes within the PApplet sketch. This allows for complex data types that can include any number of arguments and avoids the limitations of solely using standard data types such as: int (integer), char (character), float (real number), and color (RGB, ARGB, hex).

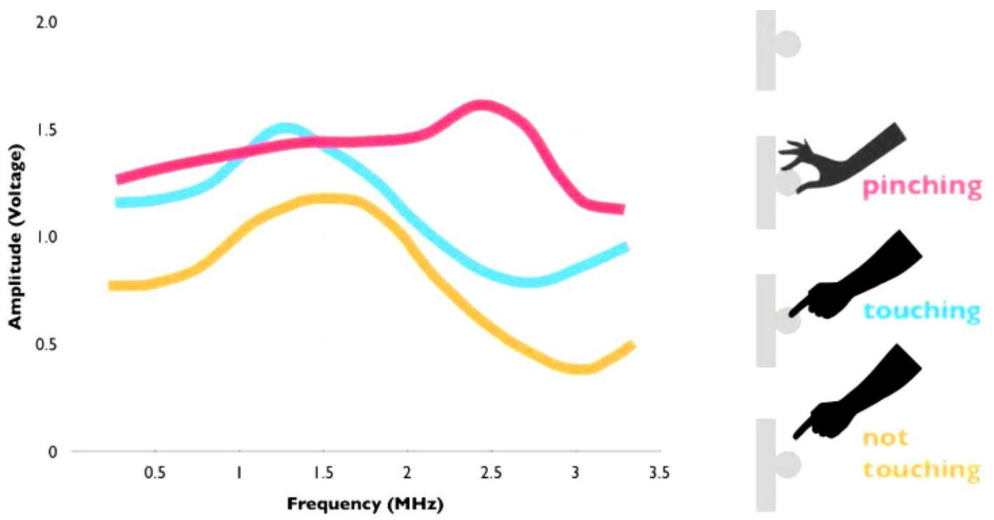
## Block Diagram:


INPUT
Object being touched by the User
Micro-controller Board
Send the data to the computer in digital form
Analyze the data using the processing programming language
OUTPUT
Display the output on the GUI developed using Processing

## CIRCUIT DIAGRAM:



## GRAPH GENERATION AND ANALYSIS USING COMPUTER:-



For generation of the graph we sense the input from the circuit in different frequencies thus and plotted the amplitude at different output frequencies.

For analysing, we will be using the method of storing the values of the graph in 2D array in which we can analyse the peak of the graph i.e. the point where the amplitude is maximum. By storing the range of values for which we can differentiate different configurations we can easily distinguish between different types of configuration of touching.

## WORK SO FAR:

### Code for generation of PWM and ADC:

```
#define SET(x,y) (x |=(1<<y))

#define CLR(x,y) (x &= (~(1<<y)))

#define CHK(x,y) (x & (1<<y))

#define TOG(x,y) (x^=(1<<y))

#define N 160

float results[N];

float freq[N];

int sizeOfArray = N;

void setup()

{

  TCCR1A=0b10000010;

  TCCR1B=0b00011001;

  ICR1=110;

  OCR1A=55;

  pinMode(9,OUTPUT);

  pinMode(8,OUTPUT);

  Serial.begin(115200);

  for(int i=0;i<N;i++)

    results[i]=0;
}

void loop()
{
  unsigned int d;
  int counter = 0;
  for(unsigned int d=0;d<N;d++)
  {
    int v=analogRead(0);
    CLR(TCCR1B,0);
    TCNT1=0;
    ICR1=d;
    OCR1A=d/2;
    SET(TCCR1B,0);
    results[d]=results[d]*0.5+(float)(v)*0.5;
   freq[d] = d;
 //   plot(v,0);
 //   plot(results[d],1);
  // delayMicroseconds(1);
  }
PlottArray(1,freq,results);
  TOG(PORTB,0);
}
```

## Work till now: (*till 5th March, 2013*)

Learned about the working of the circuit

$\downarrow$

Learned about working of timers and ADCs in ATmega

$\downarrow$

Tried to program the PWM generation and ADC program

$\downarrow$

Learned the Processing programming language

$\downarrow$

Tried to work on the GUI for the computer

$\downarrow$

Collected the circuit components

## SCOPE OF WORK:

Will implement the circuit on PCB

$\downarrow$

Will complete the GUI

$\downarrow$

Will complete the task of sensing

$\downarrow$

Will implement it on glass of water

## References:

- ➢ http://www.instructables.com/id/Touche-for-Arduino-Advanced-touch-sensing/?ALLSTEPS
- ➢ http://www.youtube.com/watch?v=E4tYpXVTjxA
- ➢ http://arduino.cc/blog/2012/06/01/touche-with-arduino/

## Suggestions: