Reading texture from the screen and interfacing it to a microcontroller

Gursimran singh
Thapar University
KVPY APP - 982681

Preface

We experience nature, develop theories or engineering techniques, build and evaluate these ideas which lead to future advances. These ideas are improved, worked on, more worked on, and refined. The window of refinement never closes, work never stops and every time what we have are better design, more powerful and practical interfaces and better experience with technology.

I feel I have tried to explore one such open window in my previous report, tried my level best to demonstrate some applications that I could achieve using this idea. But still a lot could be possible when the technique will be available for public use and then much application will emerge.

Last time when I did my project in KVPY, I did not get enough time to implement ‘Stage4’ of my project. I had exams that month and I was also working on another project titled ‘Freescale Smart Car Racing’. So I tried to implement it now.

As, another new and original concept is involved in ‘Stage4’ of reading texture from a mobile phone to communicate information from a mobile phone to a MCU, I believe it was worth documenting some experiences and conclusions here, that I got while trying to implement the idea in reality.

Once again recalling the possible ways of communicating a mobile phone to MCU, like Bluetooth, IR, etc are obviously possible but they are not as simple as this one. The simplicity I’m talking is in terms of the basic concept and overhead that occurs while implementing and running such communications. As we need to transfer only some information we should not use ‘Consumer IR’ or Bluetooth protocols that are designed and tuned to transfer more information like a complete file. In such cases protocols such as this one could be implemented and used easily.

Introduction

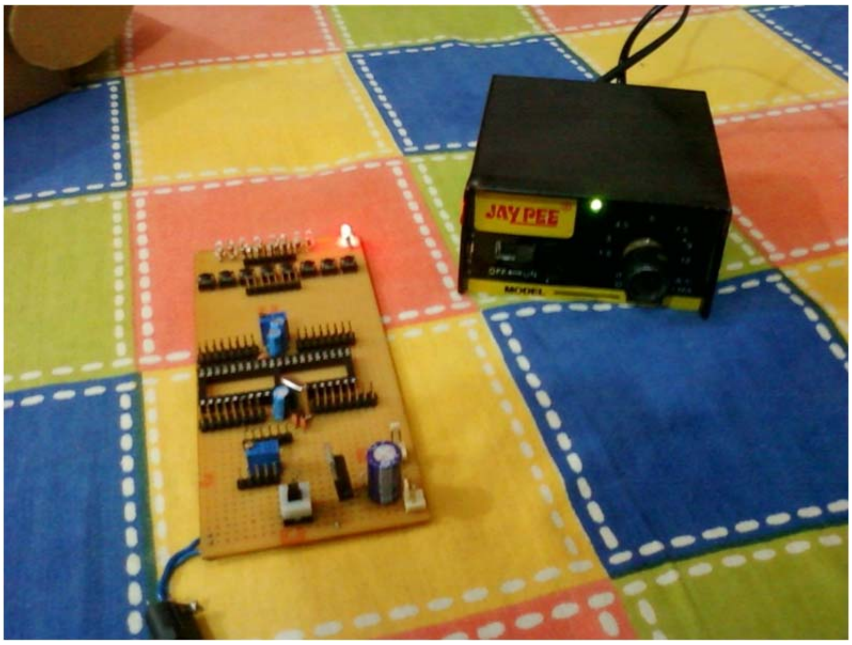
In the last report I mentioned about how we can use MCU and IR sensor to read texture from a mobile phone screen. This texture can then be used to inform MCU about different information, that we could transfer using the ‘miss-call’ concept in my previous report.

Here in this report I have tried to achieve it practically, that aims to TURN-ON or TURN-OFF a simple electric appliance from a remote location using a mobile phone, free of cost. For that I tried to use the same components that I had and tried to demonstrate a basic version of the idea.

As the mobile part is already implemented in my previous report, in this report I will only focus on how we can read texture from the mobile using a IR sensor and further use a relay to to control an electrical appliance. Because I have already used a relay to control CFL light in my previous laser project, for the sake of simplicity, I used a small LED in place of an electrical appliance.

Components

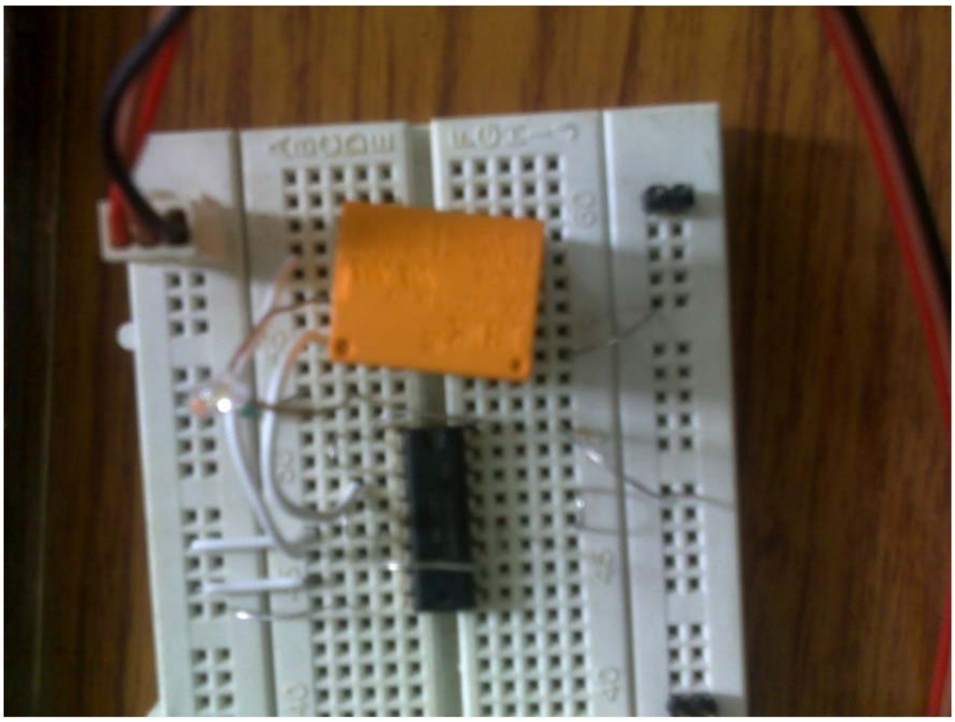
MCU KIT
To make my work easy, I used a kit that I soldered somewhere around summer last year. The main reason of making one such was that there weren’t such kits available around Patiala, so I decided to make it one for myself. This provided me a lot of experience in the field of hardware and circuit design as well. However I was a bit worried about the black fumes of soldering, so I used this mask that my dad got me from the hospital.



This kit has an ATMega16 microcontroller and well interfaced male pins to all its I/O ports, an ADC interface, some LEDs and push down switches to test things out on the same kit without any problem. Power output in both regulated and unregulated form can also be taken out from the kit for interfacing with other circuits on separate breadboards.

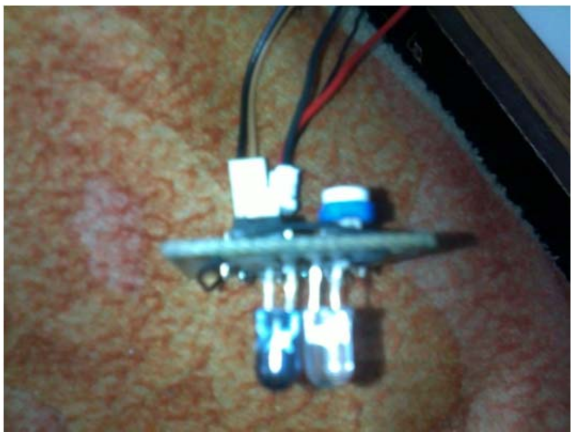
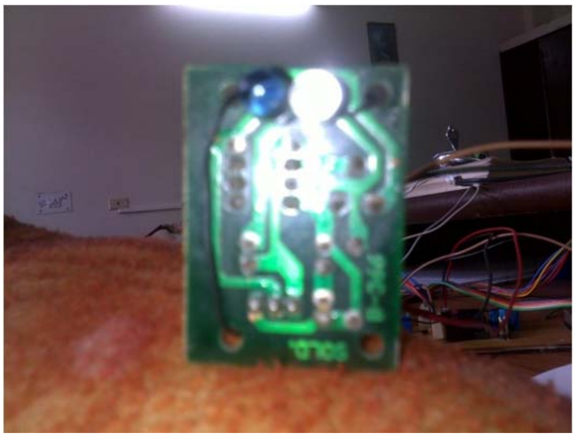
Relay interfaced to L293D

I put this relay on a breadboard and interfaced it to L293D to cater its need of high current. Power was taken from the unregulated supply from the above kit and a 5V signal from the I/O pin of the MCU controls the state of the relay. The relay has been tested in a CFL light and it works perfectly fine without any problem in my last project, so this time for ease of implementation, I put a simple LED instead of CLF light in the output terminals of the relay.



IR sensors

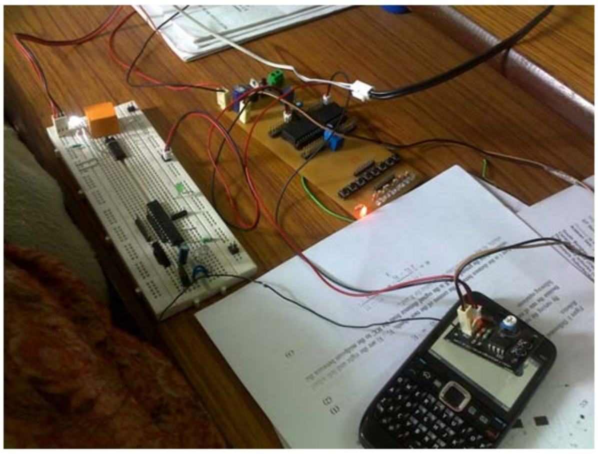
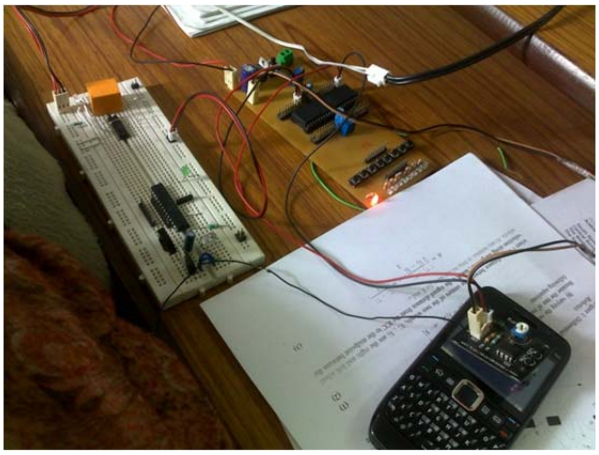
Again the same IR sensors that I got from the robotics workshop solved my purpose. After some calibration they worked perfectly fine. The only issue was that they are a bit sensitive to external ambient light that come inside from a windows beside my bed during the day. So I had to use them carefully during the day to prevent recalibration and errors during texture read from mobile screen.

The output of these IR sensors from the standard male pin connector was a 0/1 from a comparator attached to TX/RX pair. Still the output that I recorded from a multimeter was 3.56, it was probably due to some voltage drop somewhere, and fortunately there wasn't any problem as read a logical 1 at the MCU's I/O pin.

Setup

The components are interconnected in the following way. I used a 9V power source and took the power to the breadboard and to the MCU kit. Then the regulated power was further taken from the kit to the sensor and the output of the sensor is fed to the ADC0 port of the MCU. Finally one wire carrying signal from the PORTD0 is taken to breadboard and fed to the L293D input pin. The other input pin of L293D is grounded. IR sensor is placed on the screen of the mobile phone in tilted position so that it rests easily there and the 'Turn display off after _ minutes' setting is set to none so that the display never turns OFF.

Code

Version 1

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main (void)
{

  DDRD=0xFF;        //used as output pins
  DDRB=0x00;        //used as input pins

  for(;;)  // Loop Forever
  {
     PORTD = PINB&0b00000001; // Turn on LED1
  }
}
```

Problems

When I run this code, the response of the LED was flickering when the screen was WHITE and OFF when the screen was black. This is due to the fact that the due to ambient and reflective conditions on the mobile screen the output of the IR sensor was varying at a very fast rate from logic LOW to logic HIGH. The rate was however captured by high response MCU and fed to LED so the LED was flickering and dimming.

Version 2

The above problem was tackled by using an ADC. When the fast varying signal of the IR sensor is fed to an ADC, the ADC interprets this fast varying signal as the average value of the time period (as the response of ADC is generally less that response of I/O pins). This is similar to the scenario when we input a PWM signal to an ADC. I tested it using an ADC with the following code and it works fine.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

{
  ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Set ADC prescalar to 128 - 125KHz sample rate @ 16MHz

  ADMUX |= (1 << REFS0); // Set ADC reference to AVCC
  ADMUX |= (1 << ADLAR); // Left adjust ADC result to allow easy 8 bit reading

 // No MUX values needed to be changed to use ADC0

 //   ADCSRA |= (1 << ADATE);  // Set ADC auto trigger enable
  ADCSRA |= (1 << ADEN);  // Enable ADC

 /* The ADSC bit is 1 when the conversion is going on and
    cleared by harware when done */
  ADCSRA |= (1 << ADSC);  // Start A2D Conversions

}
```

```
int main (void)
{

  DDRD=0xFF;        //used as output pins
  DDRB=0x00;        //used as output pins

  initADC();

  for(;;)  // Loop Forever
  {
             //start conversion...
              ADCSRA  |= (1 << ADSC);  // Start A2D Conversions

             // This one waits until the conversion completes..
             while((ADCSRA&(1<<ADSC))==1);

             // Turns ON the relevant LED
            if(ADCH < 150)
            {
                PORTD = 0b00000000; // Turn OFF LED1
            }


            else
            {
                   PORTD = 0b00000001; // Turn ON LED1
            }

  }
}
```

Problems

This code runs fine a LED and put it OFF and ON without any problem but when I interfaced to a relay, the relay was tikkering (or flickering) very fast. I found the reason of this problem in the reading of the IR sensor which itself is switching at a very fast rate. So there was a frequency component in the signal. As there was a frequency component right in the input, the output also contains a frequency component. A relay having mechanical inertia is not able to respond to the frequency component very fast, so producing a buzzing sound.

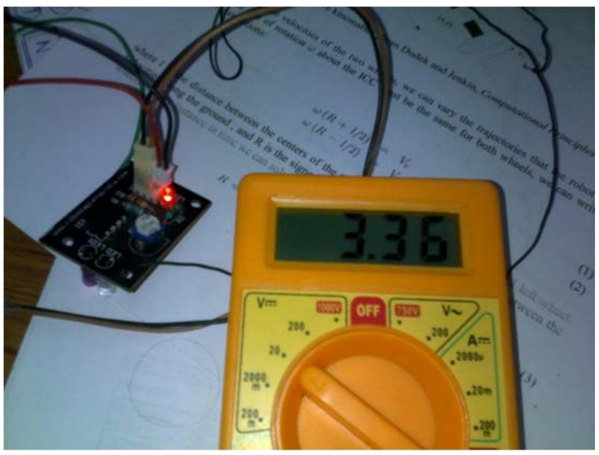
Version 3

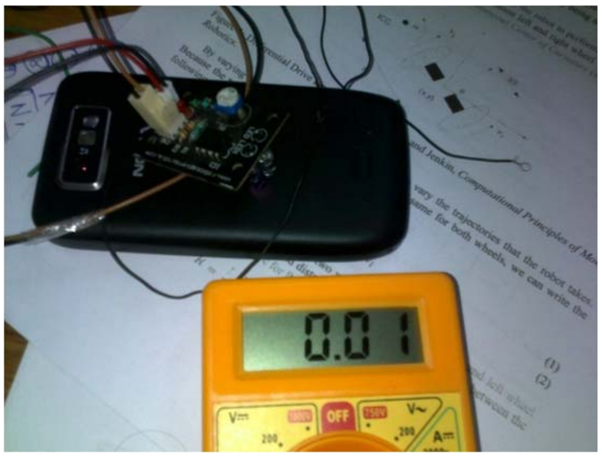
This is the final version of the code. I tried some some hack in the code to kill the frequency component that was in the output signal of the previous code. Using some trial, error, linear approximation and extrapolation I was able to get the tune-value in the code to completely kill the frequency component in the output.

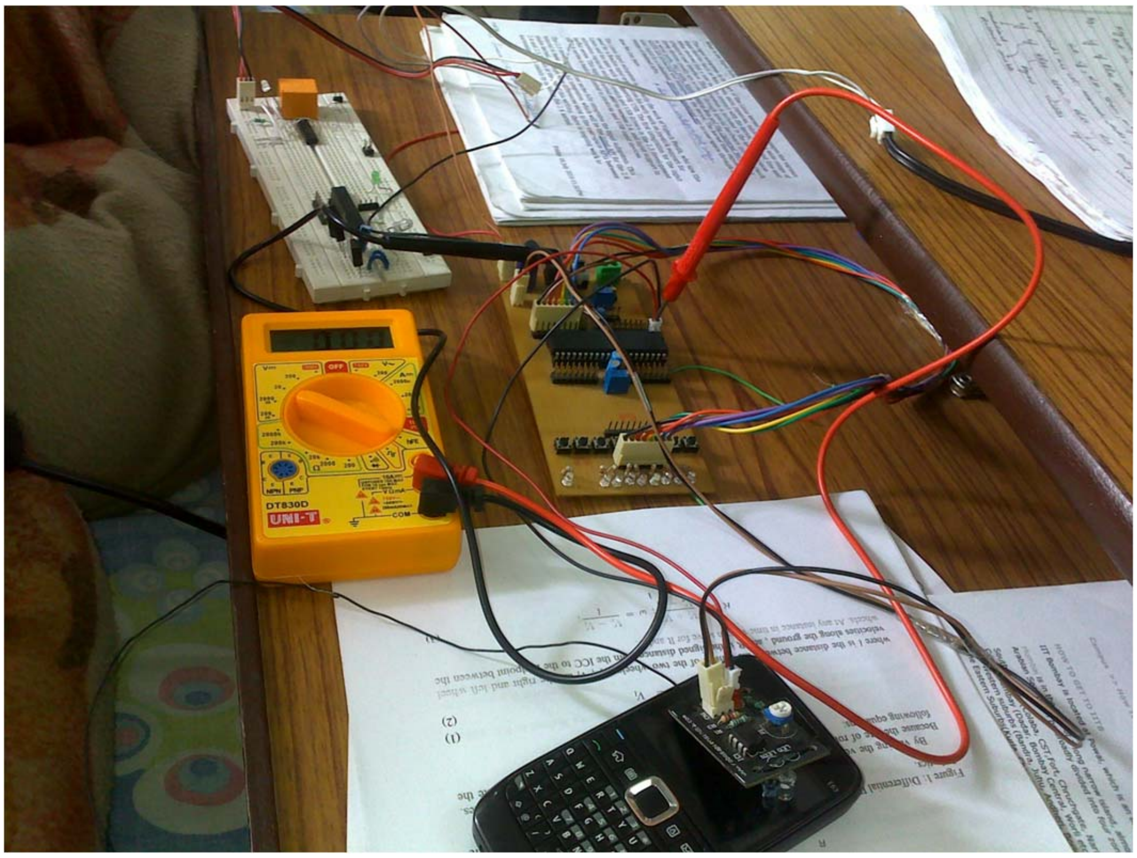
```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

void initADC()
{
  ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Set ADC prescalar to 128 - 125KHz sample rate @ 16MHz

  ADMUX |= (1 << REFS0); // Set ADC reference to AVCC
  ADMUX |= (1 << ADLAR); // Left adjust ADC result to allow easy 8 bit reading

 // No MUX values needed to be changed to use ADC0
 //   ADCSRA |= (1 << ADATE);  // Set ADC auto trigger enable
  ADCSRA |= (1 << ADEN);  // Enable ADC

 /* The ADSC bit is 1 when the conversion is going on and
    cleared by harware when done */
  ADCSRA |= (1 << ADSC);  // Start A2D Conversions

}



int main (void)
{

  DDRD=0xFF;        //used as output pins
  DDRB=0x00;        //used as output pins

  initADC();

  for(;;)  // Loop Forever
  {
          //start conversion...
            ADCSRA |= (1 << ADSC);  // Start A2D Conversions

          // This one waits until the conversion completes..
            while((ADCSRA&(1<<ADSC))==1);

            if(ADCH < 165)
           {
                    i++;

                    if(i>1000)
                    {
                         PORTD = 0b00000000; // Turn OFF LED1
                       i=0;

                    }

          }


          else
          {
                    PORTD = 0b00000001; // Turn ON LED1
                   i=0;

           }

  }
}
```

Testing and debugging

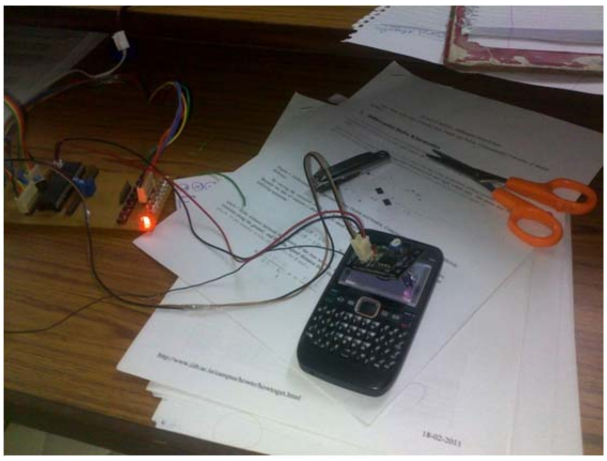
The only testing instrument that I have is a multimeter so I tried my level best to diagnose every problem only with the information that it can give. It’s a cheap multimeter and its response time is fairly low, it averages the values over a time and then displays the average as a stable signal. Every time I took different values from the IR sensor based on different conditions and these were used to tune so that the software can differentiate between 'logic BLACK' and 'logic WHITE'.







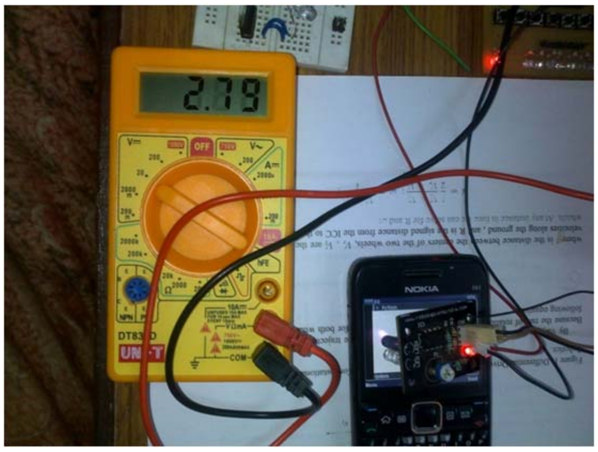
Using the technique to control a LED worked flawlessly as shown in the picture below. I used the application that I made in stage4 software in the previous report and used ON/OFF buttons and controlled the LED and transfer information free of cost.

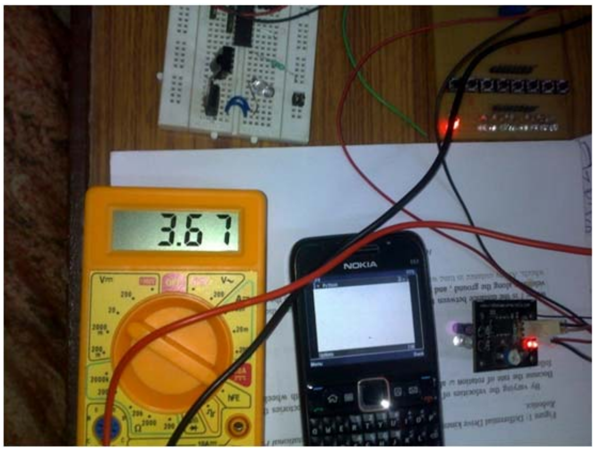


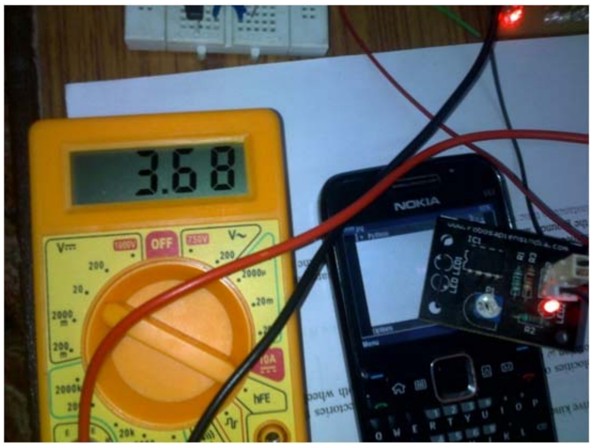
However when I used the same signal that fed to the LED, to control a relay, the scenario was totally different and contrary to what I thought it did not work. When the phone was in BLACK logic, the relay was in OFF state but when the phone got in WHITE logic, relay instead of triggering started producing a buzzing sound. When I tested the voltage that was being fed to the relay it was only 5.5V instead of a 9V signal because the signal from the output port of the MCU is itself only 3.3V (as measured from the multimeter) and not full 5V. Obviously the simple I/O port which is designed to give a 5/0 V signal can't produce a 3.3V, it's a fast varying signal and the averaging effect of my cheap multimeter look it to be as 3.3V

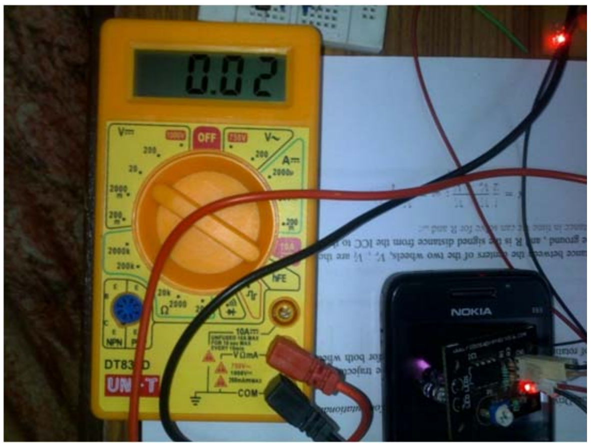
In the rest of the section I have tried my level best to diagnose the problem with the information that I have from this multimeter as no other more responsive instrument is available to me.

With the code that I used in version 2, the only reason of the problem could be that the input itself is flickering which made the output flicker. So the source of problem is the IR sensor. I tested the output voltage from the IR sensor and found something fishy happening there. When phone was in logic WHITE, the output on the multimeter was only 2.79V instead of full 3.68V, the maximum voltage output of the IR sensor, which proves that the output of the sensor is flickering.







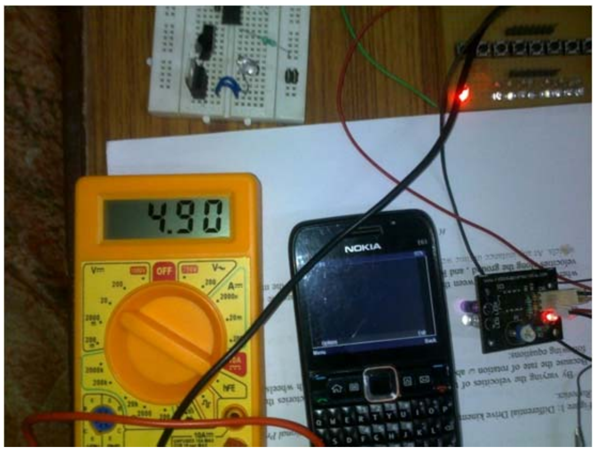


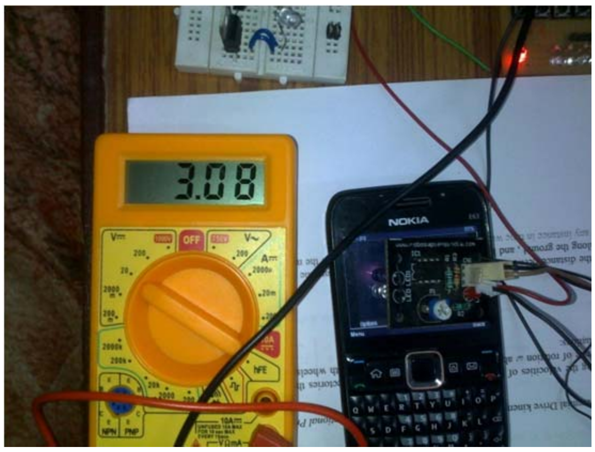
As a comparator is designed to give only two logic voltages that correspond to 0 and 1 and in this case are 0 and 3.68V, this voltage of 2.79 could only be possible due to flickering of the comparator output and the rest is the magic of the my cheap multimeter.

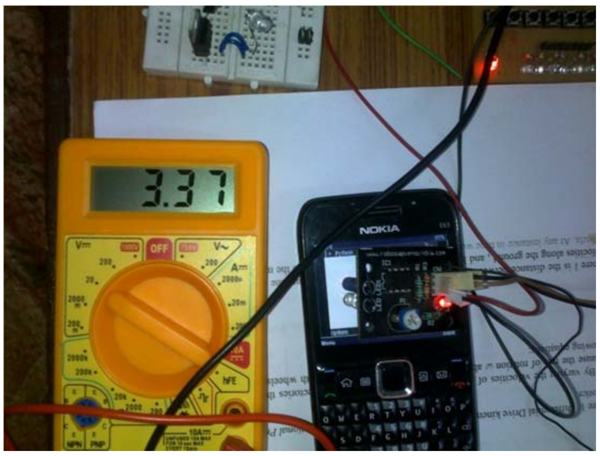
By doing some calculations, I got some results about the voltage that could be obtained if there is logic 1 (3.68V) for around 75% of the time and rest 25% is logic 0.

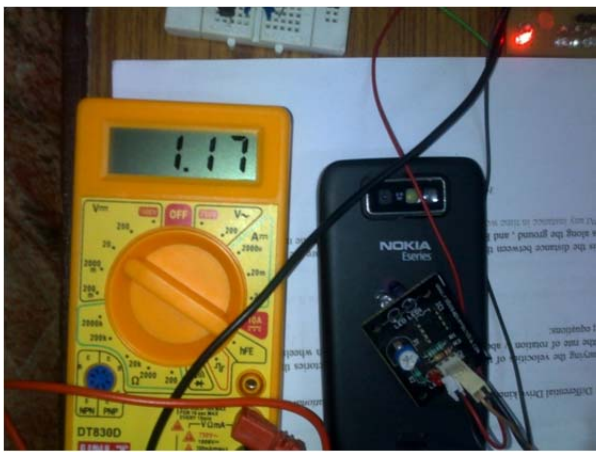
$$3.68x + 0.02(1-x) = 2.79$$
$$X=0.7459$$

Further the flickering the the comparator output itself may be due to two reasons. First the voltage or signal from the photodiode corresponding to BLACK logic and WHITE logic is pretty close which confuses the comparator, so it could not resolve it and outputs 0 (0.02) as well as 1 (3.68) randomly. Second the IR radiation that comes from the screen in itself is flickering. One possible reason that comes to my mind is 'refresh rate' of the screen.
This problem can be investigated by measuring voltage directly from the leg of the photodiode and feeding it to ADC of the MCU. Before that I get an idea by measuring values from the photodiode leg with the help of a multimeter.









These are the various values of the voltage that come under different conditions of the sensor. The two values which are of particular importance to us is 3.08 (logic BLACK) and 3.37 (logic WHITE). These values are close enough but were pretty stable on the multimeter. But these may itself be flickering, so I fed these to the version 2 code with the setting.

$$3.08/5*256=157.696$$
$$3.37/5*256=172.544$$

So I set the value somewhere between the middle of the two. The basic idea is that if the values of these voltages are actually stable (amount if IR radiation is uniform) and not flickering, the MCU should be able to distinguish between the two and the output on the relay side will also not flicker.

We will perform this test on the assumption that MCU will be able to distinguish these values. There may be plenty of reasons why MCU may not be able to give a stable output for a value of voltage in small range [3.08 3.37]. The circuit that we design is one primary condition. The one in which special emphasis is given on the ADC part of the MCU and has been tested for stability can bring us out of doubt but the one I have is poorly constructed and is not aimed to do such precision work. So if the MCU kit that I have, gives us a stable value on the output, it means

(Values of 'sensor leg' output are stable) **AND** (microcontroller is able to resolve *precisely*)

But if either of the condition fails, present test is not sufficient to say which condition on either side of AND has failed. The following code is changed in the 'version 2' for the test.

```
if(ADCH < 165)
```

Unfortunately the relay again produced that buzzing sound so the test is inconclusive. But the most likely reason is screen flickering and not ADC precision. To test this I will have to feed same values to ADC from a stable source and see if it can resolve that precisely or not. The other test that I could perform is to study the signal of the IR sensor with the help of a more responsive instrument like oscilloscope, which is available in the lab in our college.

In case the problem is IR flickering, most appropriate idea to make it work is to make the IR sensors less responsive like our eyes. I have no idea how this can be done but I believe that it can be done or it might be available in market as making sensors more responsive sensors is generally made by hours of research on less responsive sensors. Still it’s not always true.

At this point I thought, I was about to finish the report and say only relevant hardware can solve the problem. However I have a habit of thinking every problem in terms of software and the cause of this problem is clear, and is of frequency.

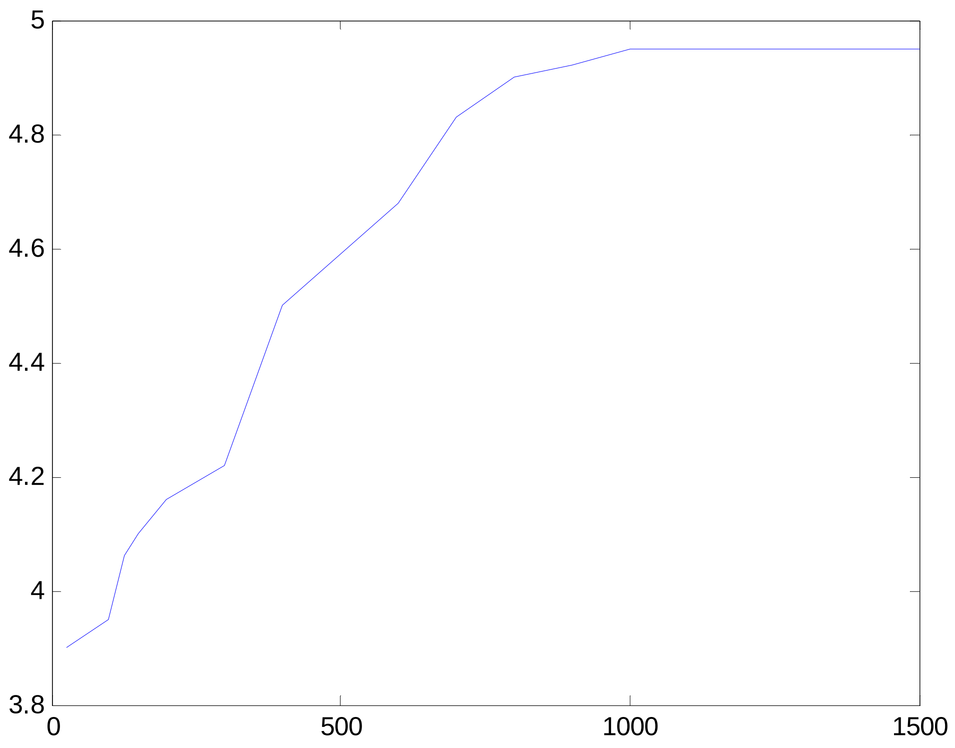
```
If (i>25)
```

I tried to solve the problem in code and took some values using code of ‘version3’ and after the first two values 3.90 and 3.95, I was very sure that the problem is solved. I took some more values and exported the data to matlab. There I draw the data using plot but relationship was not good enough, so I tried some more values and plotted the data once again.

S.No	Loop parameter (to remove frequency)	Output of I/O pin	Remarks
1	25	3.90	Buzzing sound but becoming more discrete as we go down.
2	100	3.95	
3	125	4.06	
4	150	4.10	
5	200	4.16	
6	300	4.22	Sound of triggering very fast is noticeable, proving that frequency is decreased.
7	400	4.50	
8	600	4.68	
9	700	4.83	Relay can be triggered now and works really fine.
10	800	4.90	
11	900	4.92	
12	1000	4.95	
13	1300	4.95	
14	1500	4.95	

(this data is taken with my cheap multimeter, so these may not be very accurate and precise)

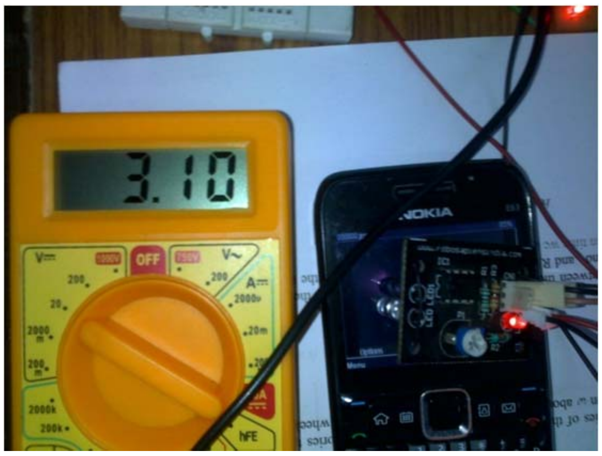
In the following figure, I plotted the data. It’s saturating after 1000 and no significant increase is noticed, in voltage at the I/O port of the MCU after that. The curve is not very smooth because the values that I took from the multimeter are not so accurate and precise and also the values are not enough, for the curve to be smooth. As the multimeter is not very accurate, so I dropped the idea of idea of taking more data and would like to do that using a better multimeter in future.

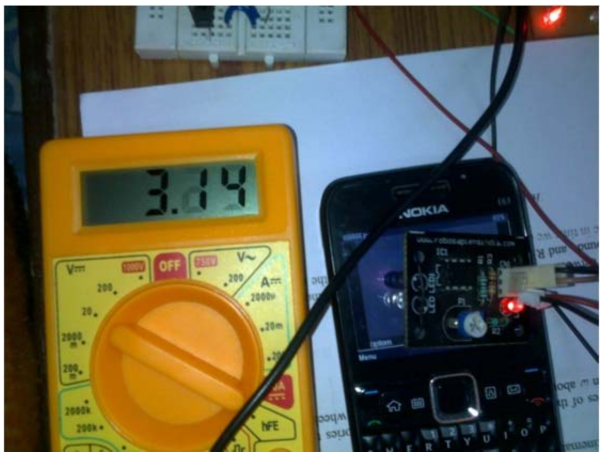


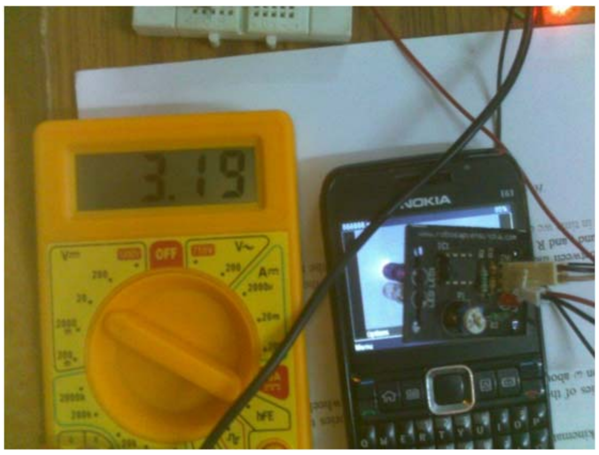
(Curve generated in matlab)

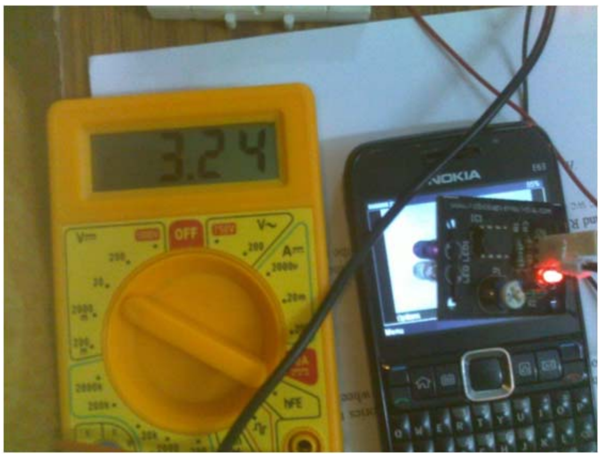
Further improvements

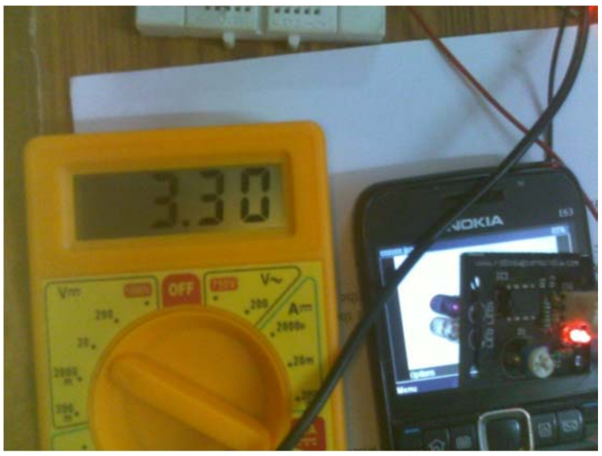
The above idea can also be extended to a read six textures from the mobile screen. For that I used the raw voltage of photodiode which is proportional to amount of IR, it receives. As it is expected that different textures would reflect/emit different amount of IR light, it can in theory detect what is going on in the mobile. For that I read the values using the multimeter.

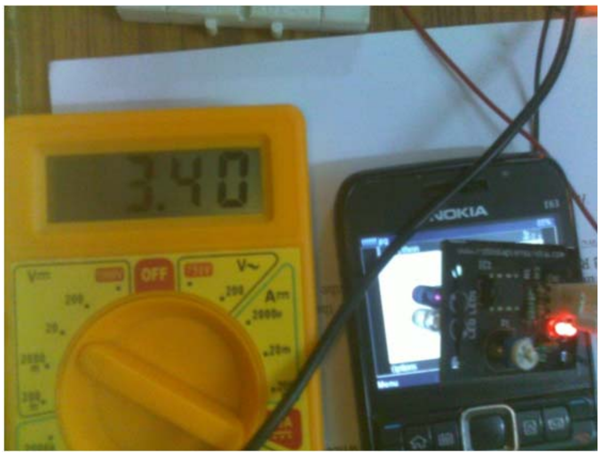












The values that I got were totally impractical because these were too close. I tried to capture these for detecting in the microcontroller but the value was too close and fluctuating so I met no significant success. So in the last I concluded that this could be done using some other better sensor.

As the backbone of the concept is reading texture from the mobile phone, I believe I can study the same using various DAQ toolkits available and studying the kind of spectra a mobile screen emits in MATLAB. I have learnt this excellent tool recently and have found that it's an excellent tool for research and mathematics. When I know what the spectra look like under different distinguishable conditions, I can easily figure out what I must measure. Using this information I can get or design a sensor accordingly.

Conclusion

Reading texture from a mobile screen wasn't that easy using an IR sensor, as I thought previously. The very first and main problem that occurred was that of the reflective screen at the top of the mobile phone display. No matter what the colour of the display is, the transparent screen always reflects enough IR light to output LOGIC 1 from the sensor. I tried various ways to tackle the problem and somehow made it work.

The above problem was most easily tacked by tilting the IR sensor. By tilting, the reflected IR rays from the transparent screen were not received by the photo diode but the emitted and reflected IR from the bottom where the colours are produced was received. It's exactly what we want and so could invert logic by using different screen textures that will emit different amount of IR light.

Another thing that I would like to mention here is that the IR sensor that I got from a robotics workshop is very inaccurate. I used it in the project and it really created a lot of problems. It always displays a 3.36V no matter how much IR the photo diode receives and further apart from using a comparator (that gives logic 0 or 1). The reason of this problem is quite evident now. So next time, I must design a new sensor myself whose hardware I can hack when I need. I have already talked a lot about the features (low refresh rate, averaging effect), my IR sensor must have.

While working on the project I found from the internet that apart from using an IR sensor, a more precise and better sensor can be made using three red, green and blue LEDs and one photodiode arrangement that will detect colours. However the problem of refection will still remain but that could be worked out. However I should not comment on how this can be done because it's all practical things.