CAPACITANCE METER USING 555 TIMER , ARDUINO UNO R3 AND NOKIA 5110 LCD.

THEORY

This project aims to build a simple capacitance meter, a device for measuring capacitance in nanoFarads and microFarads, with a range of about 10 nanoFarad to hundreds of microFarads.
It is based on the characteristics of the well known 555 timer IC, configured as a monostable multivibrator.

The 555 as monostable

In this mode, the 555 is used to produce an output pulse whose duration can be determined by the choice of a resistor and a capacitor using a very simple formula as the Ohm's law itself:

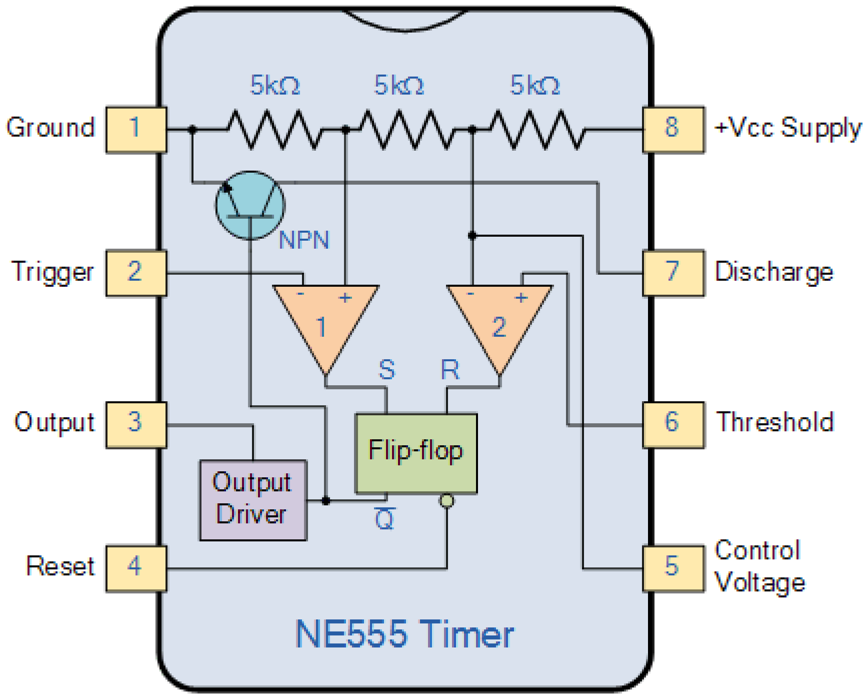
T = 1.1 x R1 x C1

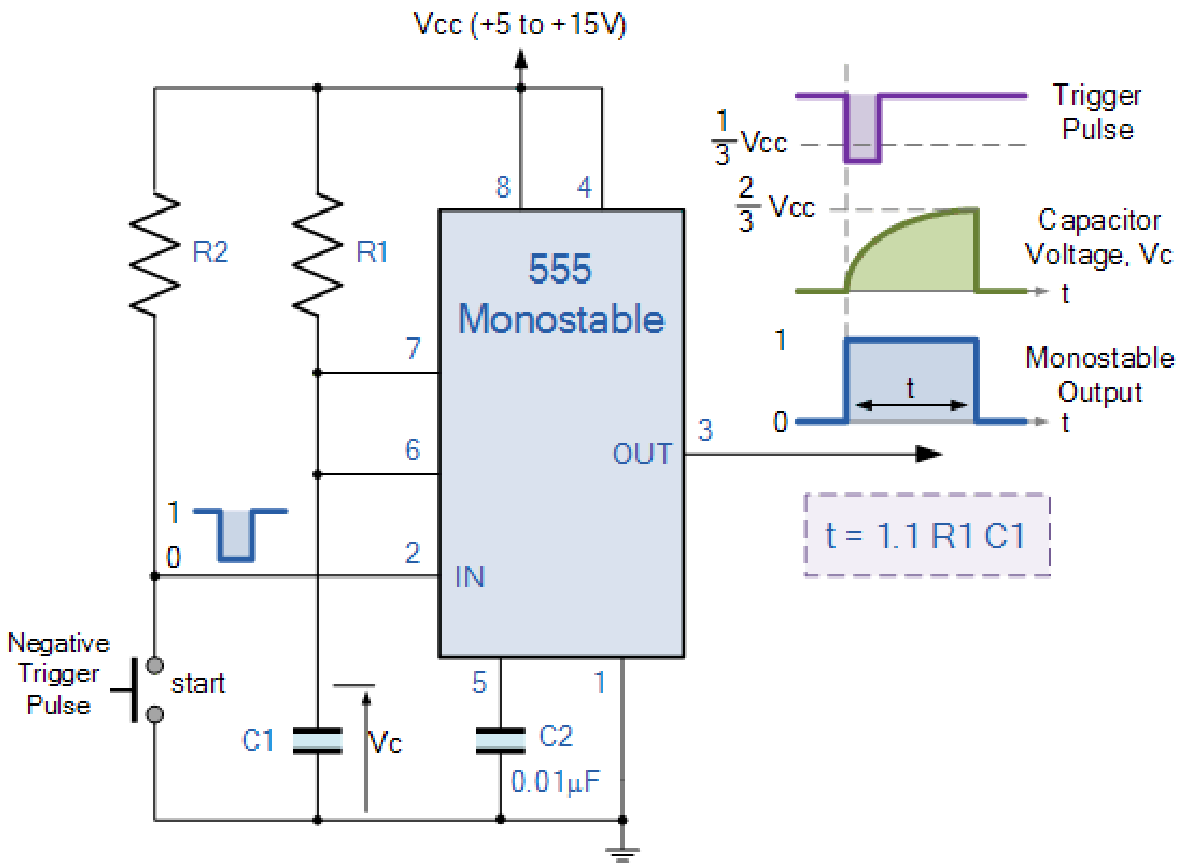
This formula allows us to calculate the capacitance:

C1 = T / (1.1 x R1)

And as R1 is known (we have chosen it), we only need to know the value of T (pulse duration).

This is where the Arduino enters the game, having the mission to determine T, calculate C1 and inform us the results through the serial port and connected NOKIA 5110 LCD Module.





This diagram is just for the explanation for 555 as monostable multivibrator.

When a negative (0V) pulse is applied to the trigger input (pin 2) of the Monostable configured 555 Timer oscillator, the internal comparator, (comparator No1) detects this input and “sets” the state of the flip-flop, changing the output from a “LOW” state to a “HIGH” state. This action in turn turns “OFF” the discharge transistor connected to pin 7, thereby removing the short circuit across the external timing capacitor, C1.

This action allows the timing capacitor to start to charge up through resistor, R1 until the voltage across the capacitor reaches the threshold (pin 6) voltage of 2/3Vcc set up by the internal voltage divider network. At this point the comparators output goes “HIGH” and “resets” the flip-flop back to its original state which in turn turns “ON” the transistor and discharges the capacitor to ground through pin 7. This causes the output to change its state back to the original stable “LOW” value awaiting another trigger pulse to start the timing process over again. Then as before, the Monostable Multivibrator has only “ONE” stable state.

The **Monostable 555 Timer** circuit triggers on a negative-going pulse applied to pin 2 and this trigger pulse must be much shorter than the output pulse width allowing time for the timing capacitor to charge and then discharge fully. Once triggered, the 555 Monostable will remain in this “HIGH” unstable output state until the time period set up by the R_1 x C_1 network has elapsed. The amount of time that the output voltage remains “HIGH” or at a logic “1” level, is given by the following time constant equation.

$$\tau = 1.1 R_1 C_1$$

Where, t is in seconds, R is in Ω’s and C in Farads.

THE FUNCTION OF THE ARDUINO

ARDUINO FOR MEASURING THE PULSE WIDTH

The output (3) of the 555 is connected to the arduino (for this project we used ARDUINO UNO R3) at two of its terminals (2 and 3) which are programmatically attached to the interrupts 0 and 1 to enable them to detect changes of the voltage level at the 555 output terminal.

So the pin 2 which is linked to Interrupt 0 will detect the RISING edge of the output pulse and instantly stop (interrupt) all activity that was taking place at that moment and arduino executes the code in the function Start() that we wrote.

The code in the Start() function just save the time returned by the millis() function in a volatile (global) variable. An led is also turned on here to give a visual sign that the interrupt took place. Then the function ends and so does the interrupt.

Then, when the 555 output pulse ends, the arduino pin 3 which is linked to the interrupt 1 detects its FALLING edge triggering another interruption in the operation of the arduino to execute now the code in another function which we called Stop(). As in the Start() function, the code here just set another volatile variable (t_final) with the value returned by the millis() function and turns the led off, marking the end of the pulse.

The code in the Loop () function is responsible for continuously verifying whether the value of the variable t_final is no longer zero, event that will happen at the end of the pulse.

When the condition in our if block confirms that t_final is effectively no longer 0, the code proceeds to the determination of T (the pulse duration) and with it the capacitance is calculated.

T = T_final - T_initial

C1 = T / (1.1 x R1)

In the schematic of the circuit we can see the values of the components used.

1 M-ohm for R1 is suitable for capacitors in the range of about 10 nF to 1uF.

Higher values of capacitance will cause durations of pulses of several seconds, so in the case of measuring these values it would be advisable to use a 1 kohm resistor.

ARDUINO FOR DISPLAYING THE RESULTS IN NOKIA 5110 LCD

In this project we are using Nokia 5110 Lcd to display the results that ie,the capacitance in nanofarads and in microfarads.

Connections

For the data transmission pins – SCLK and DN(MOSI) – we'll use the Arduino's **hardware SPI pins**, which will help to achieve a faster data transfer. The chip select (SCE), reset (RST), and a data/command (D/C) pins can be connected to **any digital I/O pin**. Finally, the LED pin should be connected to a PWM-capable Arduino pin, so we can dim the backlight as we please.

LCD PINS	ARDUINO PINS	NOTES
PIN_$\overline{\text{SCE}}$	7	chip enable(active low)
PIN_VCC	+5 vcc	LCD supply voltage
PIN_$\overline{\text{RESET}}$	6	external reset input
PIN_D/$\overline{\text{C}}$	5	data/$\overline{\text{command}}$
PIN_SDIN	4	serial data input
PIN_SCLK	13	serial clock input
PIN_GND	gnd	ground
PIN_BKL	+3.3v	Backlight

CODE USED FOR THE PROJECT

```
/* CAPACITANCE METER USING 555 TIMER AND ARDUINO */
/* DISPLAYING THE RESULTS IN NOKIA 5110 LCD (48*84) */

#define PIN_SCE   7
#define PIN_RESET 6
#define PIN_DC    5
#define PIN_SDIN  4
#define PIN_SCLK  13

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48
float R1 = 1000000;
float C1 = 0;
int led_pin = 12;
unsigned long t_inicio = 0;
unsigned long t_final = 0;
long T = 0;

#include<stdlib.h>
/*byte array for each character to pe printed on screen */
static const byte ASCII[][5] =
{
 {0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 "
```

```
,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 23 #
,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 24 $
,{0x23, 0x13, 0x08, 0x64, 0x62} // 25 %
,{0x36, 0x49, 0x55, 0x22, 0x50} // 26 &
,{0x00, 0x05, 0x03, 0x00, 0x00} // 27 '
,{0x00, 0x1c, 0x22, 0x41, 0x00} // 28 (
,{0x00, 0x41, 0x22, 0x1c, 0x00} // 29 )
,{0x14, 0x08, 0x3e, 0x08, 0x14} // 2a *
,{0x08, 0x08, 0x3e, 0x08, 0x08} // 2b +
,{0x00, 0x50, 0x30, 0x00, 0x00} // 2c ,
,{0x08, 0x08, 0x08, 0x08, 0x08} // 2d -
,{0x00, 0x60, 0x60, 0x00, 0x00} // 2e .
,{0x20, 0x10, 0x08, 0x04, 0x02} // 2f /
,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 30 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 31 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 32 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 33 3
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 34 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 35 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 36 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 37 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 38 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 39 9
,{0x00, 0x36, 0x36, 0x00, 0x00} // 3a :
,{0x00, 0x56, 0x36, 0x00, 0x00} // 3b ;
,{0x08, 0x14, 0x22, 0x41, 0x00} // 3c <
,{0x14, 0x14, 0x14, 0x14, 0x14} // 3d =
,{0x00, 0x41, 0x22, 0x14, 0x08} // 3e >
,{0x02, 0x01, 0x51, 0x09, 0x06} // 3f ?
,{0x32, 0x49, 0x79, 0x41, 0x3e} // 40 @
,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 41 A
,{0x7f, 0x49, 0x49, 0x49, 0x36} // 42 B
,{0x3e, 0x41, 0x41, 0x41, 0x22} // 43 C
,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 44 D
,{0x7f, 0x49, 0x49, 0x49, 0x41} // 45 E
,{0x7f, 0x09, 0x09, 0x09, 0x01} // 46 F
,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 47 G
,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 48 H
,{0x00, 0x41, 0x7f, 0x41, 0x00} // 49 I
,{0x20, 0x40, 0x41, 0x3f, 0x01} // 4a J
,{0x7f, 0x08, 0x14, 0x22, 0x41} // 4b K
,{0x7f, 0x40, 0x40, 0x40, 0x40} // 4c L
,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 4d M
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
,{0x7f, 0x09, 0x09, 0x09, 0x06} // 50 P
,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 51 Q
,{0x7f, 0x09, 0x19, 0x29, 0x46} // 52 R
,{0x46, 0x49, 0x49, 0x49, 0x31} // 53 S
,{0x01, 0x01, 0x7f, 0x01, 0x01} // 54 T
,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 55 U
,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 56 V
```

```
,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 57 W
,{0x63, 0x14, 0x08, 0x14, 0x63} // 58 X
,{0x07, 0x08, 0x70, 0x08, 0x07} // 59 Y
,{0x61, 0x51, 0x49, 0x45, 0x43} // 5a Z
,{0x00, 0x7f, 0x41, 0x41, 0x00} // 5b [
,{0x02, 0x04, 0x08, 0x10, 0x20} // 5c ¥
,{0x00, 0x41, 0x41, 0x7f, 0x00} // 5d ]
,{0x04, 0x02, 0x01, 0x02, 0x04} // 5e ^
,{0x40, 0x40, 0x40, 0x40, 0x40} // 5f _
,{0x00, 0x01, 0x02, 0x04, 0x00} // 60 `
,{0x20, 0x54, 0x54, 0x54, 0x78} // 61 a
,{0x7f, 0x48, 0x44, 0x44, 0x38} // 62 b
,{0x38, 0x44, 0x44, 0x44, 0x20} // 63 c
,{0x38, 0x44, 0x44, 0x48, 0x7f} // 64 d
,{0x38, 0x54, 0x54, 0x54, 0x18} // 65 e
,{0x08, 0x7e, 0x09, 0x01, 0x02} // 66 f
,{0x0c, 0x52, 0x52, 0x52, 0x3e} // 67 g
,{0x7f, 0x08, 0x04, 0x04, 0x78} // 68 h
,{0x00, 0x44, 0x7d, 0x40, 0x00} // 69 i
,{0x20, 0x40, 0x44, 0x3d, 0x00} // 6a j
,{0x7f, 0x10, 0x28, 0x44, 0x00} // 6b k
,{0x00, 0x41, 0x7f, 0x40, 0x00} // 6c l
,{0x7c, 0x04, 0x18, 0x04, 0x78} // 6d m
,{0x7c, 0x08, 0x04, 0x04, 0x78} // 6e n
,{0x38, 0x44, 0x44, 0x44, 0x38} // 6f o
,{0x7c, 0x14, 0x14, 0x14, 0x08} // 70 p
,{0x08, 0x14, 0x14, 0x18, 0x7c} // 71 q
,{0x7c, 0x08, 0x04, 0x04, 0x08} // 72 r
,{0x48, 0x54, 0x54, 0x54, 0x20} // 73 s
,{0x04, 0x3f, 0x44, 0x40, 0x20} // 74 t
,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 75 u
,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 76 v
,{0x3c, 0x40, 0x30, 0x40, 0x3c} // 77 w
,{0x44, 0x28, 0x10, 0x28, 0x44} // 78 x
,{0x0c, 0x50, 0x50, 0x50, 0x3c} // 79 y
,{0x44, 0x64, 0x54, 0x4c, 0x44} // 7a z
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ←
,{0x78, 0x46, 0x41, 0x46, 0x78} // 7f →
};
/*FUNCTION TO PRINT CHARACTER IN LCD*/

void LcdCharacter(char character)
{
  LcdWrite(LCD_D, 0x00);
  for (int index = 0; index < 5; index++)
  {
    LcdWrite(LCD_D, ASCII[character - 0x20][index]);
  }
  LcdWrite(LCD_D, 0x00);
```

```
}
/*FUNCTION TO CLEAR LCD*/
void LcdClear(void)
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}
/*INITIALISATION OF THE LCD*/
void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
   pinMode(A2,INPUT);
  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);
  LcdWrite(LCD_C, 0x21 );  // LCD Extended Commands.
  LcdWrite(LCD_C, 0xbf );  // Set LCD Vop (Contrast).
  LcdWrite(LCD_C, 0x04 );  // Set Temp coefficent. //0x04
  LcdWrite(LCD_C, 0x00 );  // LCD bias mode 1:48. //0x13
  LcdWrite(LCD_C, 0x20 );  // LCD Basic Commands
  LcdWrite(LCD_C, 0x0C );  // LCD in normal mode.

}
/*FUNCTION TO PRINT STRING ON SCREEN*/
void LcdString(char *characters)
{
  while (*characters)
  {
    LcdCharacter(*characters++);
  }
}
/*Function to write data into lcd */
void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

void LcdWriteCmd(byte cmd)
{
  digitalWrite(PIN_DC, LOW); //DC pin is low for commands
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, cmd); //transmit serial data
  digitalWrite(PIN_SCE, HIGH);
}
```

```
/*SETTING UP THE ARDUINO*/
void setup(void)
{
  LcdInitialise();
  LcdClear();
  LcdString("CAPACITANCE");
  Serial.begin(9600);
    attachInterrupt(0,start,RISING);

    attachInterrupt(1,stop,FALLING);

    pinMode(led_pin,OUTPUT);
}
/*FUNCTION TO MOVE THE CURSOR TO A DESIRED POSITION*/
void LcdXY(int x, int y)
{
  LcdWriteCmd(0x80 | x);  // Column.
  LcdWriteCmd(0x40 | y);  // Row.
}
char str[4]=" uF";
char str2[4]=" nF";
char string[8];
void loop(void)
{
  delay(100);



    if (t_final > t_inicio && t_inicio >0)
    {
        T = t_final - t_inicio;
        C1 = T *1000/ (1.1 * R1);


        Serial.print("t = ");
        Serial.print(T);
        Serial.println(" microSec");
        Serial.print("C = ");
        Serial.print(C1 );
        Serial.println(" microF");
        Serial.print("C = ");
        Serial.print(C1 * 1000);
        Serial.println(" nanoF");
        LcdXY (0,9);
        LcdString((dtostrf(C1,5,2,string)));
        LcdString(str);
        LcdXY (0,18);
        LcdString((dtostrf(C1*1000,5,2,string)));
        LcdString(str2);
        LcdXY (0,28);
        LcdString("HURRAY");
        LcdXY (0,37);
        LcdString("Done!");
        Serial.println(t_final);
        Serial.println(t_inicio);
        Serial.println();
        t_inicio = 0;
        t_final = 0;
    }

}
void start()
{
    t_inicio = millis();
    digitalWrite(led_pin, HIGH);
}

void stop()
{
    t_final = millis();
    digitalWrite(led_pin, LOW);
}
/* the end of code */
```

********************************CONCLUDED***********************************