Measurement of Frequency Deviation By Using Arduino Uno.

Widad Jasim Hamody AL-Rawee
Instructor Training/Baghdad.
Nadheerah Haseeb Tawfeeq AL-Dabbagh
Instructor Training/Baghdad.
Email:saeed.tech2015@yahoo.com

Received on: 3/9/2014 & Accepted on:2/4/2015

Abstract

This study aimed at constructing and implementing a new technique in calculating and correcting the frequency deviation in power systems. Several techniques where used to measure frequency deviation. One of them was Arduino Uno card which uses microcontroller ATmega328. This microcontroller board is an easy to use, accurate, economic and flexible solutions for measurement problems.

The main objective of the work was to measure frequency deviation by microcontroller and displays it on (PC) monitor or small liquid crystal display (LCD) or both. Also, to correct the error if occurred by moving stepper motor with accurate steps which control the fuel gate in the generator, by reading the frequency at periodical time such as per 0.3 second and check the changes, if the frequency is more than required value. The controller and its suggested program (code) in C language will move the stepper motor clockwise (CW) until frequency reaches the required frequency (F_0), i.e.50.00 H_Z, if the reading gets lower than (F_0), the stepper motor will move counter clockwise (CCW) to get the required frequency. The system is applied practically by implementing the suggested design. This search achieved established control frequency of 99% along any variations in loads with high frequency reaches to accuracy 0.01H_Z with low coast components.

Keywords- Frequency deviation, Flip - Flop, Arduino Microcontroller

قياس انحراف التردد بأستخدام لوح الارينو

الخلاصة

هدفت هذه الدراسة الى بناء وتطبيق تقنية جديدة في حساب وتصحيح إنحراف التردد في منظومة توليد القدرة
.تقنيات عديدة أستخدمت لقياس هذا الانحراف. واحدة منها لوح الآردوينو_الذي يستخدم المتحكم الدقيق .
ATmega328هذا اللوح ذو المتحكم الدقيق سهل الأستخدام, دقيق, مع جدوى أقتصادية وذو حلول مرنة لمشاكل
القياسات.

https://doi.org/10.30684/etj.33.4A.16
2412-0758/University of Technology-Iraq, Baghdad, Iraq

الهدف الرئيسي لهذه الدراسة هو قياس إنحراف التردد بواسطة المسيطر الدقيق وعرضه على شاشة حاسوب أو شاشة (LCD) أو كلاهما. كذلك, تصحيح الخطأ إن وجد وذلك بتحريك المحرك الخطوي بخطوات دقيقة والذي يتحكم ببوابة مسيطرةعلى الوقود الخاص بالمولد, وذلك بقراءة التردد بفترات زمنية دورية مثل كل 0.3 ثانية وفحص التغيرات, إذا كان التردد أكبر من القيمة المطلوبة . هذا المسيطر والبرنامج المقترح (كود) بلغة (C) سوف يحرك المحرك الخطوي بأْ تجاه عقرب الساعة حتى وصول التردد الى التردد المطلوب (F_0), اى بكلام اخر الى 50.00 هيرتز , وإذا القراءة يحصل عليها نقصان من التردد (F_o) , فان المحرك الخطوي سوف يتحرك عكس عقرب الساعة للحصول على التردد المطلوب . إن النظام يطبق عمليا بواسطة تنفيذ التصميم المقترح . هذا البحث أنجز سيطرة وتحكم مستقر لغاية 99% رغم تغير الاحمال وبدقة تردد تصل الى 0.01H_Z وبمكونات ذات كلفة بسيطة

INTRODUCTION

A very simple frequency-deviation measuring instrument that works on the principle of a two-arm bridge is described. The circuit was fabricated and tested in the laboratory. They indicate good linearity between frequency deviation and voltage or pulse count over the normal range of power system frequencies [1]. Many papers have been written on the measurement of power system frequency deviation. In a previous work, the measurement of power system frequency deviation is useful for the design of power system stabilizers, and the monitoring of power station frequency. Here a simple scheme using a 90° phase shifter is suggested. The scheme is suitable for implementation using a microprocessor [2]. In another previous work, large frequency deviations due to a number of disturbances are in small isolated power systems. The maximum frequency deviation in the system is limited to prevent other generator tripping. It is important to have an accurate model to calculate it, both for system planning and operation. A new simplified model to calculate the maximum frequency deviation when either a generator or load-related disturbance occurs in these systems is presented. This model takes into account the response of governor-prime mover even when different technologies are presented in the power system. [3].

Frequency measurement is an important issue in the field of electrical engineering Electric power system has become complex over the last decade. The use of distributed generation, the connection of non-linear loads and the presence of some unexpected system faults are the main causes of frequency variations. In addition, power quality includes frequency as the most important index. This previous work deals with digital measurement of power frequency i.e. 50Hz using microcontroller based system. [4].

In this work, a method for measuring the power frequency deviations has been presented. This technique is based on concept of counting the known train of pulses for the period time. [5].This work describes a new technique in calculating the frequency deviation in power systems. In order to get a high accuracy in measurement, microcontroller is used instead of the conventional electronic components. Hence a high reliability is attained in measurements according to any deviation in the frequency of the power line because it is very easy to change the data given to the Microcontroller [6]. However, in this work the microcontroller ATmega328 is used to measure and correct the generator frequency [which is available with its main components at Arduino Uno Cards].

Principle of Operation

The method utilizes microcontroller for measuring and correcting the frequency deviation of the power system, it receives the logic signals from Flip - Flop and divide by two then converts it to a data depending on the program stored in the memory. The output of microcontroller card is coupled to the stepper motor.

The optocoupler is a buffer stage between the generator and the microcontroller. The block diagram of the system is shown in figure (1).

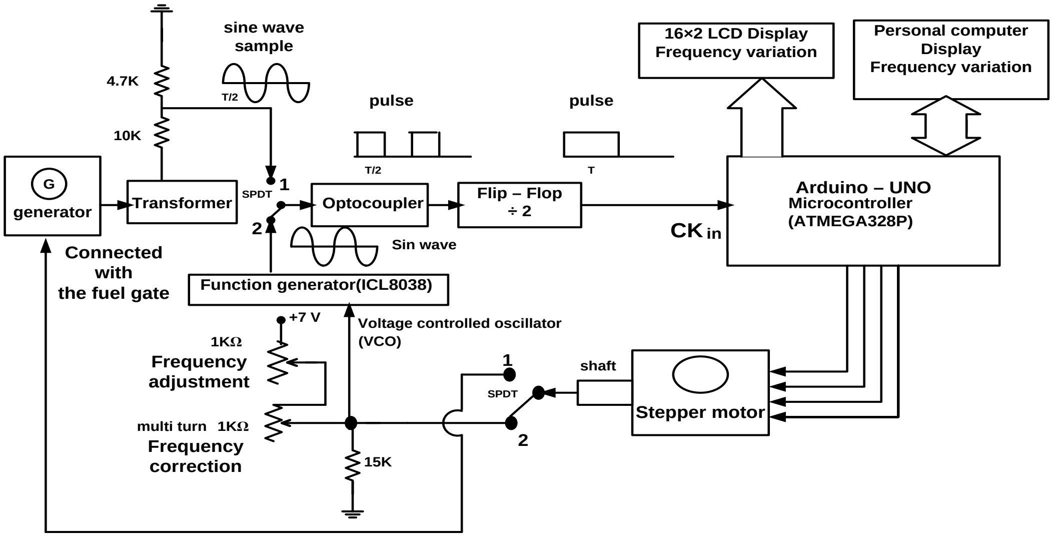

sine wave sample
4.7K
T/2
10K
pulse
T/2
pulse
T
16×2 LCD Display Frequency variation
Personal computer Display Frequency variation
G
generator
Transformer
SPDT
1
2
Optocoupler
Flip – Flop ÷ 2
CK in
Arduino – UNO Microcontroller (ATMEGA328P)
Sin wave
Connected with the fuel gate
Function generator(ICL8038)
+7 V
Voltage controlled oscillator (VCO)
1KΩ
Frequency adjustment
multi turn 1KΩ
Frequency correction
15K
1
SPDT
2
shaft
Stepper motor

Figure (1): Block diagram of the circuit

Practical Circuit Diagram

As shown in figure (2),this unit includes the main operating voltages to simulator function generator which need $\pm 9V$, and provides +5V to the logic circuits (HEF4040 and the controller card) and +7V as a voltage control oscillator (VCO) to change the frequency of the integrated circuit type ICL8038 function generator (F.G) and +5V to drive the stepper motor, these voltages are regulated by type LM7805 and type LM7909 series and got from two transformers (12V,0,12V) And (6V,0,6V) which are operated on 220V. The green led indicates the$+9V$, the red led indicates the $-9V$, the varistor type V471 is to protect the circuit from the over voltage more than 275V.

The ICL8038 is a function generator used as a simulator for electrical generator, provide sine wave signal (A) supplied to optocoupler type 4N35 for safety isolation (B) and then this signal is supplied to Flip-flop type HEF 4040 to get a logic signal (C) goes to pin 13 of Arduino Uno Card to measure the input signal time (T), the controller by the program will read time (T) and then measure frequency (Fo) H_Z and will read out results on 2*16 LCD and LCD of PC and handle the stepper motor to correct frequency if there are variations on its setting value (50.00 H_Z). It will move motor clockwise (CW) or counter clockwise (CCW) which control on a multi turn potentiometer (which simulates the fuel gate) 1KΩ **of** ICL8038 to correct frequency. The other potentiometer of the same integrated circuit ICL8038 is used to change frequency by hand.

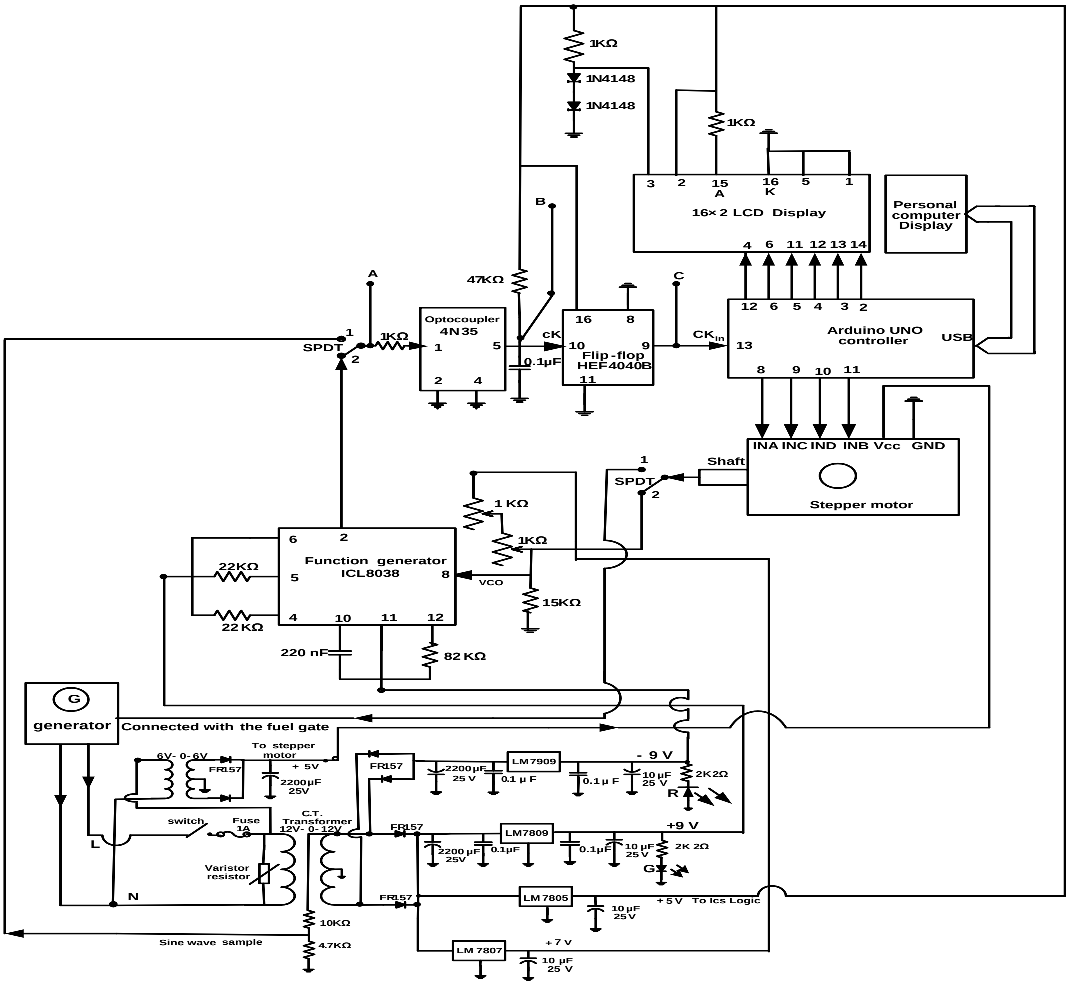


Figure (2): Practical Circuit Diagram

Program Structure

Fig (3) represents the flow chart explain the program structure on which microcontroller depends

*LCD, Pins, serial port, stepper motor initializing.

*When the pulse going high the controllers measures its length in μs (=T) (see Appendix A) and then

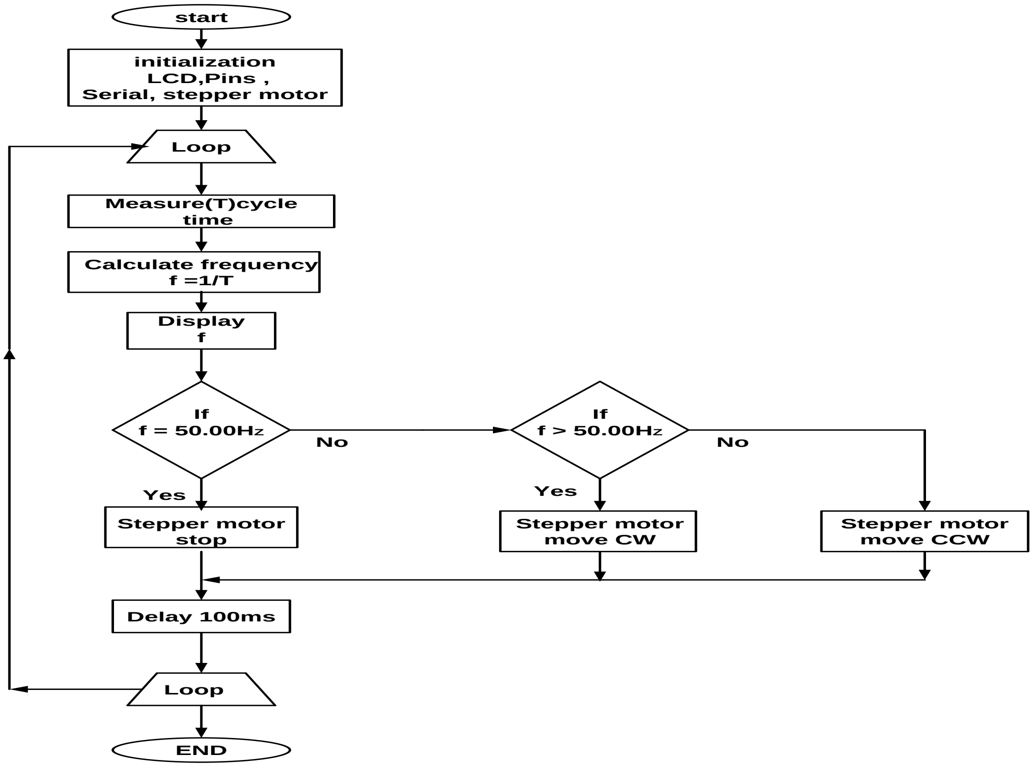


Figure (3): Flow Chart of the Program

Calculate (f) as

$$f = \frac{\mathbf{1}}{\boldsymbol{T}}$$

*The controller shows the frequency on (LCD) & serial (PC) monitor and compares it with the selected frequency value; it measures the deviation if it is equal to 50.00 H_Z the program makes motor off.

If it is greater than 50.00 H_Z the program will move motor (CW) until reaches 50.00 H_Z .

If it is less than 50.00 H_Z the program will move motor (CCW) until reaches 50.00 H_Z .

* delay time for new reading (100ms) (see Appendix B). And then the loop will continue.

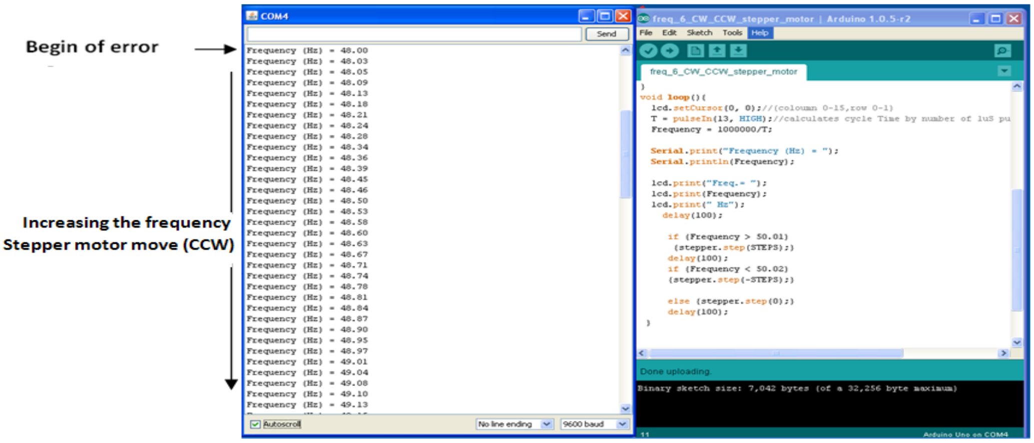
Experimental Result

Code in the Arduino IDE for calculating frequency to show selected practical obtained results for frequency-deviation measuring instrument is used to evaluate the performance of the proposed system. Measurement frequency by microcontroller and show it on (PC) monitor or small (LCD) then correct the frequency by controlling on the fuel gate of the generator using accurate stepper motor and by calculating the excess or reduction frequency, then move that fuel gate open or closed by movement of the motor clockwise (CW) or counter clock wise (CCW) according to variation type using a program (code) in C.

The results presented here for frequency of 48, 49, and 52, 51, 50 H_Z are used from tests carried out on a stepper motor. The practical results are depicted in the illustrated codes:

Since the Arduino kit is interfaced with the computer, we can run the serial monitor of the Arduino software and see the calculated frequency. When frequency reduces to 48 or 49 H_Z as shown in figures (4,5) , The controller reads the access in frequency and compares it with the main stored frequency value and it will begin to give instruction to stepper motor to move counter clockwise (CCW) until the controller reads the correct frequency which is here 50.00H_Z .

The stepper motor will hand the shaft of potentiometer in the direction of increasing the frequency and keep on until reaching 50.00 H_Z as shown in figure (5), this done in a perfect steps , quite and accurate.

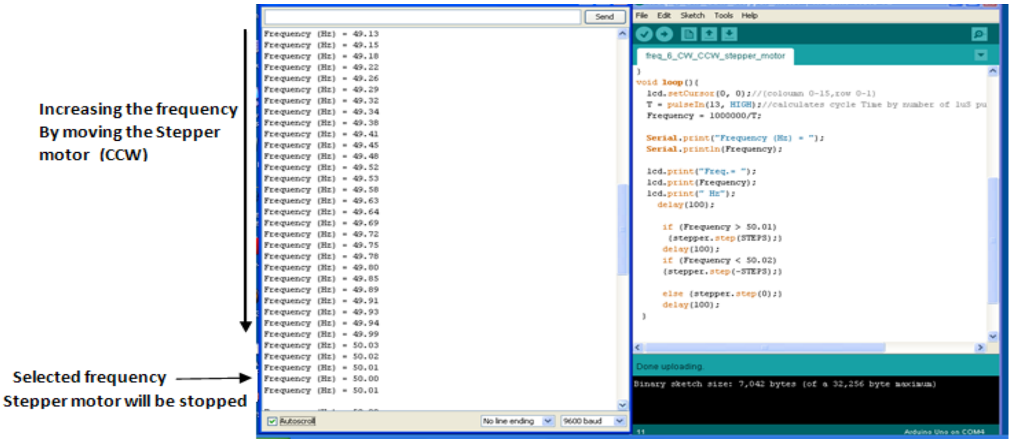


```
Frequency (Hz) = 48.00
Frequency (Hz) = 48.03
Frequency (Hz) = 48.05
Frequency (Hz) = 48.09
Frequency (Hz) = 48.13
Frequency (Hz) = 48.18
Frequency (Hz) = 48.21
Frequency (Hz) = 48.24
Frequency (Hz) = 48.28
Frequency (Hz) = 48.34
Frequency (Hz) = 48.36
Frequency (Hz) = 48.39
Frequency (Hz) = 48.45
Frequency (Hz) = 48.46
Frequency (Hz) = 48.50
Frequency (Hz) = 48.53
Frequency (Hz) = 48.58
Frequency (Hz) = 48.60
Frequency (Hz) = 48.63
Frequency (Hz) = 48.67
Frequency (Hz) = 48.71
Frequency (Hz) = 48.74
Frequency (Hz) = 48.78
Frequency (Hz) = 48.81
Frequency (Hz) = 48.84
Frequency (Hz) = 48.87
Frequency (Hz) = 48.90
Frequency (Hz) = 48.95
Frequency (Hz) = 48.97
Frequency (Hz) = 49.01
Frequency (Hz) = 49.04
Frequency (Hz) = 49.08
Frequency (Hz) = 49.10
Frequency (Hz) = 49.13
```

```
}
void loop(){
  lcd.setCursor(0, 0);//(coloumn 0-15,row 0-1)
  T = pulseIn(13, HIGH);//calculates cycle Time by number of 1uS pu
  Frequency = 1000000/T;

  Serial.print("Frequency (Hz) = ");
  Serial.println(Frequency);

  lcd.print("Freq.= ");
  lcd.print(Frequency);
  lcd.print(" Hz");
    delay(100);

     if (Frequency > 50.01)
      {stepper.step(STEPS);}
     delay(100);
     if (Frequency < 50.02)
     {stepper.step(-STEPS);}

     else {stepper.step(0);}
     delay(100);
 }
```

```
Done uploading.
Binary sketch size: 7,042 bytes (of a 32,256 byte maximum)
```

Figure (4): Code for calculating frequency at 48 H_Z



```
Frequency (Hz) = 49.13
Frequency (Hz) = 49.15
Frequency (Hz) = 49.18
Frequency (Hz) = 49.22
Frequency (Hz) = 49.26
Frequency (Hz) = 49.29
Frequency (Hz) = 49.32
Frequency (Hz) = 49.34
Frequency (Hz) = 49.38
Frequency (Hz) = 49.41
Frequency (Hz) = 49.45
Frequency (Hz) = 49.48
Frequency (Hz) = 49.52
Frequency (Hz) = 49.53
Frequency (Hz) = 49.58
Frequency (Hz) = 49.63
Frequency (Hz) = 49.64
Frequency (Hz) = 49.69
Frequency (Hz) = 49.72
Frequency (Hz) = 49.75
Frequency (Hz) = 49.78
Frequency (Hz) = 49.80
Frequency (Hz) = 49.85
Frequency (Hz) = 49.89
Frequency (Hz) = 49.91
Frequency (Hz) = 49.93
Frequency (Hz) = 49.94
Frequency (Hz) = 49.99
Frequency (Hz) = 50.03
Frequency (Hz) = 50.02
Frequency (Hz) = 50.01
Frequency (Hz) = 50.00
Frequency (Hz) = 50.01
```

```
}
void loop(){
  lcd.setCursor(0, 0);//(coloumn 0-15,row 0-1)
  T = pulseIn(13, HIGH);//calculates cycle Time by number of 1uS pu
  Frequency = 1000000/T;

  Serial.print("Frequency (Hz) = ");
  Serial.println(Frequency);

  lcd.print("Freq.= ");
  lcd.print(Frequency);
  lcd.print(" Hz");
    delay(100);

     if (Frequency > 50.01)
      {stepper.step(STEPS);}
     delay(100);
     if (Frequency < 50.02)
     {stepper.step(-STEPS);}

     else {stepper.step(0);}
     delay(100);
 }
```

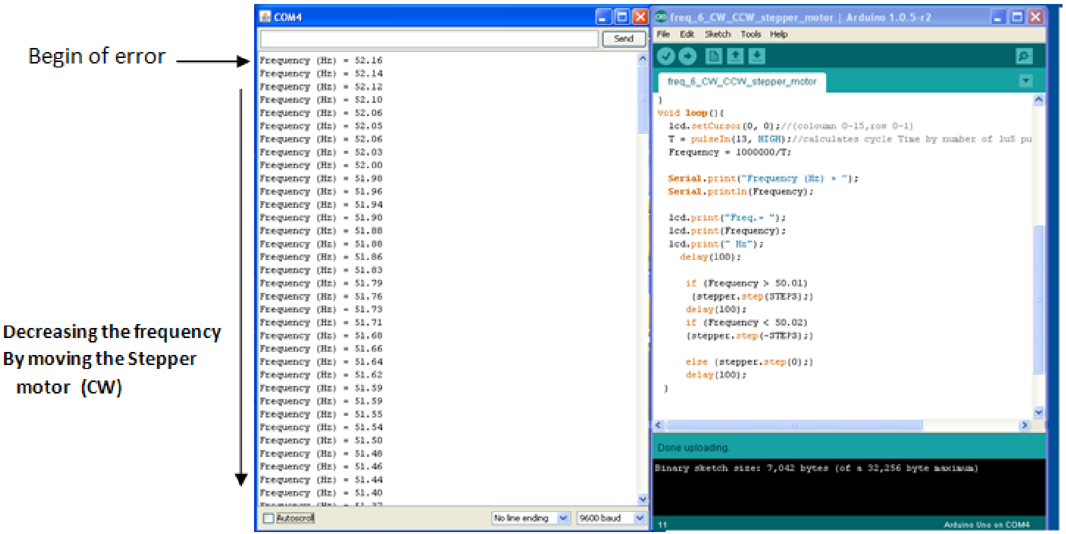
```
Done uploading.
Binary sketch size: 7,042 bytes (of a 32,256 byte maximum)
```

Figure (5): Code for calculating frequency at 49 H_Z and 50Hz

When frequency increases to 52 or 51 H_Z as shown in figures (6,7) , The controller reads the access in frequency and compares it with the main stored frequency value and it will be begin to give instruction to stepper motor to move clockwise (CW) until the controller reads the correct frequency which is here 50.00H_Z .

The stepper motor will handle the shaft of potentiometer in direction of reducing the frequency and keep on until reaching 50.00 H_Z as shown in figure (8), this done in steps and gradually .

The results show the deviations in frequency and its correction to the desired frequency (50.00Hz) by moving motor to increase frequency or decrease frequency [CW, CCW] to 50.00 Hz with accuracy of 1%.

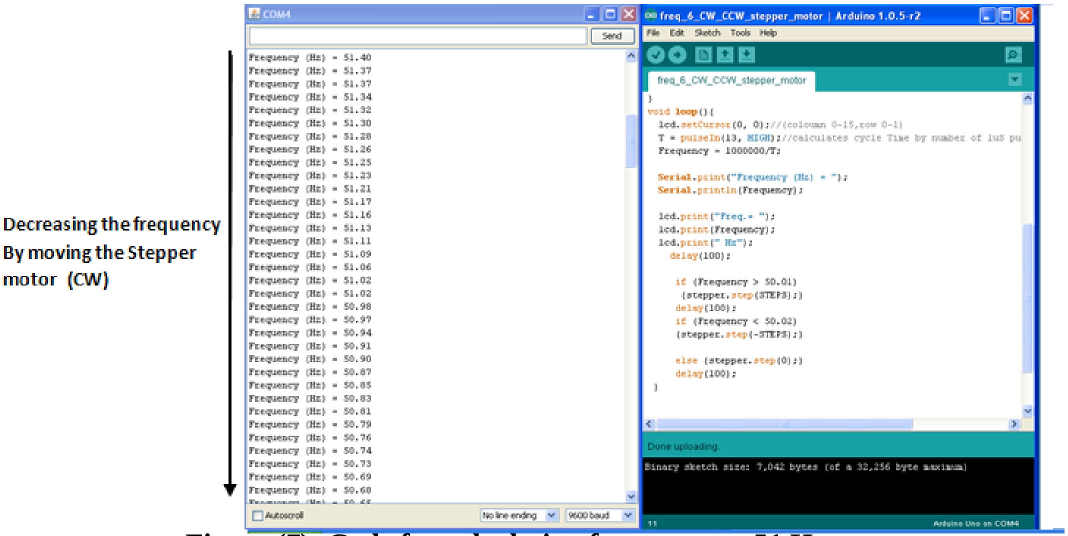


```
Frequency (Hz) = 52.16
Frequency (Hz) = 52.14
Frequency (Hz) = 52.12
Frequency (Hz) = 52.10
Frequency (Hz) = 52.06
Frequency (Hz) = 52.05
Frequency (Hz) = 52.06
Frequency (Hz) = 52.03
Frequency (Hz) = 52.00
Frequency (Hz) = 51.98
Frequency (Hz) = 51.96
Frequency (Hz) = 51.94
Frequency (Hz) = 51.90
Frequency (Hz) = 51.88
Frequency (Hz) = 51.88
Frequency (Hz) = 51.86
Frequency (Hz) = 51.83
Frequency (Hz) = 51.79
Frequency (Hz) = 51.76
Frequency (Hz) = 51.73
Frequency (Hz) = 51.71
Frequency (Hz) = 51.68
Frequency (Hz) = 51.66
Frequency (Hz) = 51.64
Frequency (Hz) = 51.62
Frequency (Hz) = 51.59
Frequency (Hz) = 51.59
Frequency (Hz) = 51.55
Frequency (Hz) = 51.54
Frequency (Hz) = 51.50
Frequency (Hz) = 51.48
Frequency (Hz) = 51.46
Frequency (Hz) = 51.44
Frequency (Hz) = 51.40
Frequency (Hz) = 51.37
```

```
}
void loop(){
  lcd.setCursor(0, 0);//(coloumn 0-15,row 0-1)
  T = pulseIn(13, HIGH);//calculates cycle Time by number of 1uS pu
  Frequency = 1000000/T;

  Serial.print("Frequency (Hz) = ");
  Serial.println(Frequency);

  lcd.print("Freq.= ");
  lcd.print(Frequency);
  lcd.print(" Hz");
    delay(100);

     if (Frequency > 50.01)
      {stepper.step(STEPS);}
     delay(100);
     if (Frequency < 50.02)
     {stepper.step(-STEPS);}

     else {stepper.step(0);}
     delay(100);
 }
```

```
Binary sketch size: 7,042 bytes (of a 32,256 byte maximum)
```

Figure (6): Code for calculating frequency at 52 H_Z

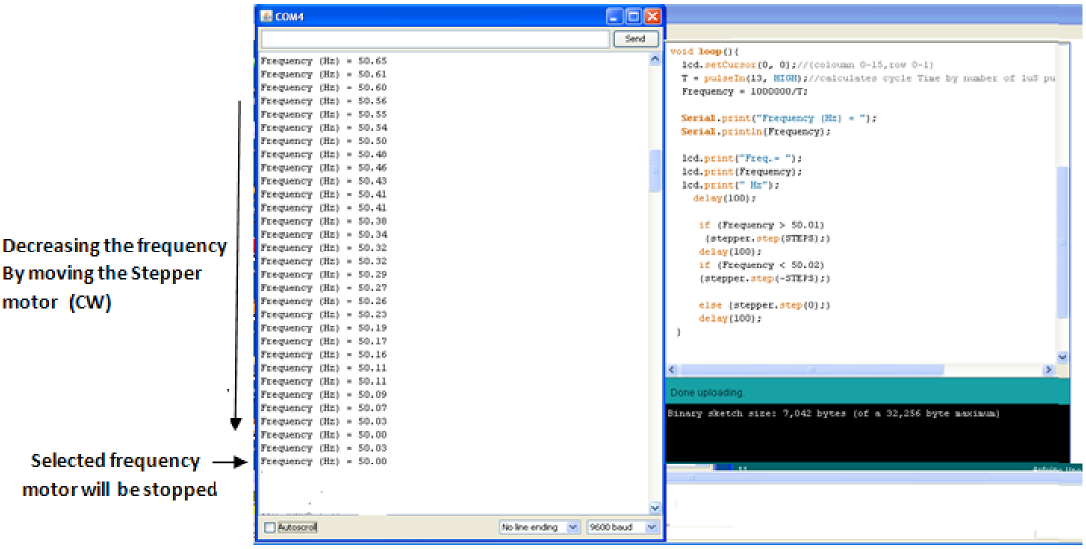


```
Frequency (Hz) = 51.40
Frequency (Hz) = 51.37
Frequency (Hz) = 51.37
Frequency (Hz) = 51.34
Frequency (Hz) = 51.32
Frequency (Hz) = 51.30
Frequency (Hz) = 51.28
Frequency (Hz) = 51.26
Frequency (Hz) = 51.25
Frequency (Hz) = 51.23
Frequency (Hz) = 51.21
Frequency (Hz) = 51.17
Frequency (Hz) = 51.16
Frequency (Hz) = 51.13
Frequency (Hz) = 51.11
Frequency (Hz) = 51.09
Frequency (Hz) = 51.06
Frequency (Hz) = 51.02
Frequency (Hz) = 51.02
Frequency (Hz) = 50.98
Frequency (Hz) = 50.97
Frequency (Hz) = 50.94
Frequency (Hz) = 50.91
Frequency (Hz) = 50.90
Frequency (Hz) = 50.87
Frequency (Hz) = 50.85
Frequency (Hz) = 50.83
Frequency (Hz) = 50.81
Frequency (Hz) = 50.79
Frequency (Hz) = 50.76
Frequency (Hz) = 50.74
Frequency (Hz) = 50.73
Frequency (Hz) = 50.69
Frequency (Hz) = 50.68
Frequency (Hz) = 50.65
```

```
}
void loop(){
  lcd.setCursor(0, 0);//(coloumn 0-15,row 0-1)
  T = pulseIn(13, HIGH);//calculates cycle Time by number of 1uS pu
  Frequency = 1000000/T;

  Serial.print("Frequency (Hz) = ");
  Serial.println(Frequency);

  lcd.print("Freq.= ");
  lcd.print(Frequency);
  lcd.print(" Hz");
    delay(100);

     if (Frequency > 50.01)
      {stepper.step(STEPS);}
     delay(100);
     if (Frequency < 50.02)
     {stepper.step(-STEPS);}

     else {stepper.step(0);}
     delay(100);
 }
```

```
Binary sketch size: 7,042 bytes (of a 32,256 byte maximum)
```

Figure (7): Code for calculating frequency at 51 H



Decreasing the frequency By moving the Stepper motor (CW)

Selected frequency motor will be stopped

```
Frequency (Hz) = 50.65
Frequency (Hz) = 50.61
Frequency (Hz) = 50.60
Frequency (Hz) = 50.56
Frequency (Hz) = 50.55
Frequency (Hz) = 50.54
Frequency (Hz) = 50.50
Frequency (Hz) = 50.48
Frequency (Hz) = 50.46
Frequency (Hz) = 50.43
Frequency (Hz) = 50.41
Frequency (Hz) = 50.41
Frequency (Hz) = 50.38
Frequency (Hz) = 50.34
Frequency (Hz) = 50.32
Frequency (Hz) = 50.32
Frequency (Hz) = 50.29
Frequency (Hz) = 50.27
Frequency (Hz) = 50.26
Frequency (Hz) = 50.23
Frequency (Hz) = 50.19
Frequency (Hz) = 50.17
Frequency (Hz) = 50.16
Frequency (Hz) = 50.11
Frequency (Hz) = 50.11
Frequency (Hz) = 50.09
Frequency (Hz) = 50.07
Frequency (Hz) = 50.03
Frequency (Hz) = 50.00
Frequency (Hz) = 50.03
Frequency (Hz) = 50.00
```

```
void loop(){
  lcd.setCursor(0, 0);//(coloumn 0-15,row 0-1)
  T = pulseIn(13, HIGH);//calculates cycle Time by number of 1uS pu
  Frequency = 1000000/T;

  Serial.print("Frequency (Hz) = ");
  Serial.println(Frequency);

  lcd.print("Freq.= ");
  lcd.print(Frequency);
  lcd.print(" Hz");
    delay(100);

     if (Frequency > 50.01)
      {stepper.step(STEPS);}
     delay(100);
     if (Frequency < 50.02)
     {stepper.step(-STEPS);}

     else {stepper.step(0);}
     delay(100);
 }
```

```
Done uploading.
Binary sketch size: 7,042 bytes (of a 32,256 byte maximum)
```

Figure (8) Code for calculating frequency at 50 H_Z

The waveforms obtained practically, figure (9) shows signal shape at point A and B. Figure (10) shows signal shape at point B and C. Interfacing a unipolar stepper motor and LCD with a microcontroller are shown in figures.(11),and (12) respectively, finally figure (13) shows system stage of microcontroller based- deviation in power systems.

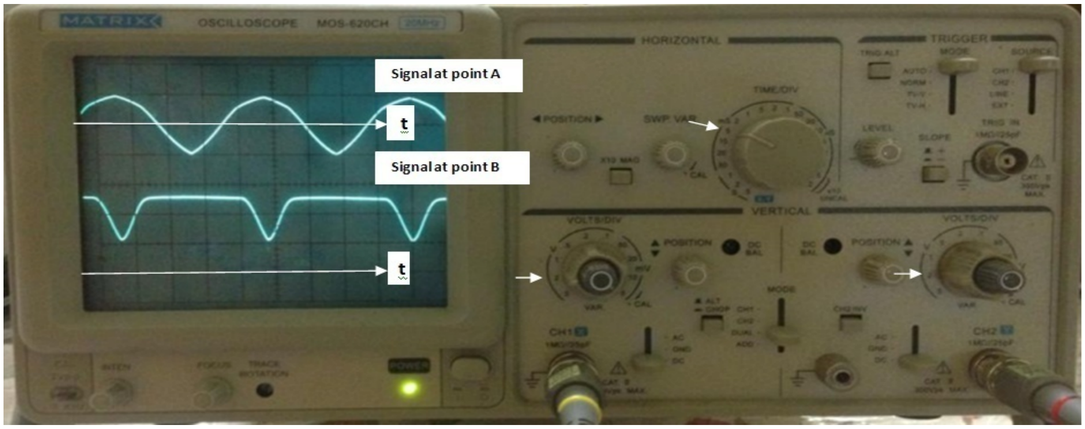


Figure (9): shows signal shape at point A and B
Time: 5 msec / div.A, B: 2 V/ div

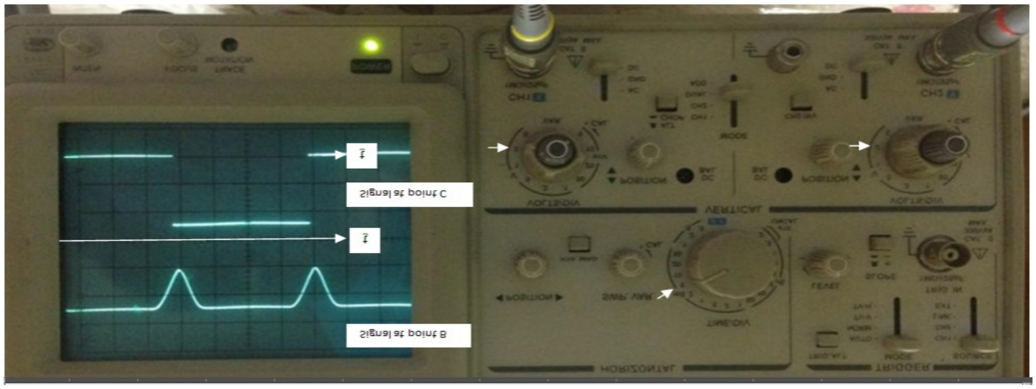
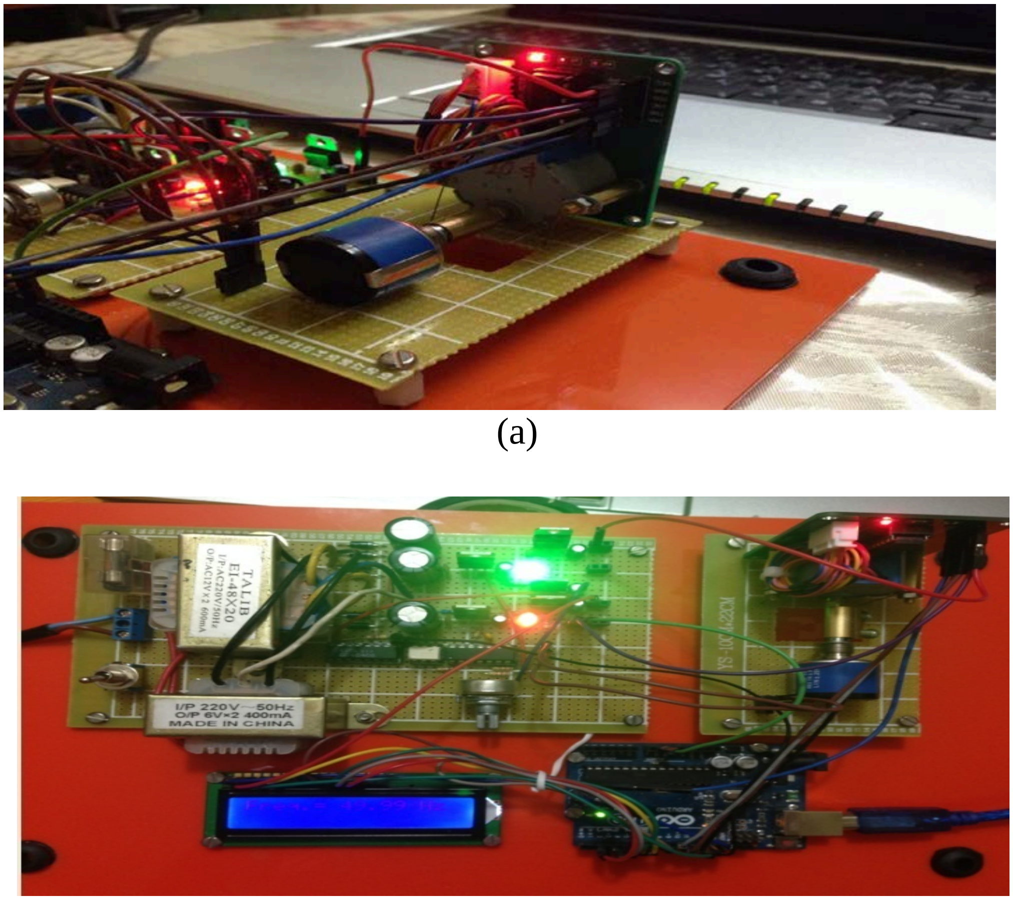


Figure (10): shows signal shape at point B and C
Time: 5 msec / div. B, C: 2 V/ div

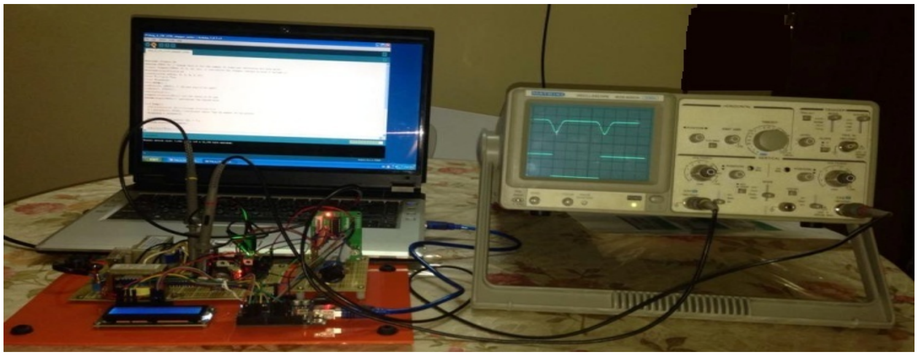


(a)

(b)

Figure(11): show interfacing a unipolar stepper motor with a microcontroller
(a): side view. (b): Top view.

Figure (12): show interfacing LCD with a microcontroller



Figure(13): Microcontroller based- deviation in power systems

Conclusions and suggested work studies

From the work carried out, concise conclusions have been reached regarding the use of simple frequency-deviation measuring instrument. Arduino microcontroller Measure and correct frequency deviation have been designed and tested. Hardware parts are reduced and flexibility is achieved by software implementation. The work deals also with digital measurement of power frequency using Arduino microcontroller based system. The system was tested over frequency range from $48H_Z$ to $52H_Z$ at 1% resolution. The present circuit design is found very simple, reliable, accurate, and works in the reduction of size and weight. Proper operation is verified by 50 H_Z operational prototype which was successful in laboratory tests.

The instrument is able to be developed in future to give it an additional feature as stated below:

1) Use wireless communication like (Blue tooth or XBee) to send the data and making remote

Control for long distance.

2) Use servo motor to get same control with less power and more accurate positions.

References

[1]Ahmad M ., " Power system frequency deviation measurement using an electronic bridge", المكتبة الافتراضية \ IEEE Journal Instrumentation and Measurement, Vol.37, 1 Pages. 147-148, 1988.

[2]Mukhtar Ahmed, "Measurment of power system frequency deviation using microprocessor", IEEE Journal Instrumentation and Measurement, Vol.168, 1 Pages.161-164, 1990.

[3]Eido I., Fernandez –Bernal F., Genteno p., Rouco L., "Maximum frequency deviation calculation in small isolated power system", IEEE Journal Transactions on Power Systems, Vol. 24, 4 Pages. 1731-1738, 2009.

[4] Khairul Alam1, Tanmoy Chakraborty2, Srabana Pramanik (Chaudhury) 3, Debabrata Sarddar4, Satadal Mal5, "Measurement of Power Frequency with Higher Accuracy using PIC Microcontroller", *India* 2013.

[5]Jitendra Dwivedi, M.shukal, k.s.verma, R.K.Singh, " A Novel Technique for Indication of Power Frequency Deviations in Electrical Systems "power Electronics and instrumentation engineering, Vol. 102, pp. 80-82, 2010.

[6] Safaa S., Omran Ali S., Alkhalid Ahmed Saeed Mohamm, " microcotroller based frequency deviation power line meter", Iraq 2004.

Nomenclature

f: frequency
Fo: frequency required value
LCD: liquid crystal display
CW: clockwise
CCW: counter clockwise
ICL: integrated circuit logic
VCO: voltage control oscillator
F.G: function generator
r.p.m: revolution per minute

(Appendix A)
Arduino Software

Using Arduino IDE software to read out frequency on LCD (2*16), this software supports the LCD in this project. Also it supports stepper motor type uni-polar, and firstly it must fix the programs from Arduino library of LCD and stepper motor as:

1. #include < liquid crystal. h> for LCD
2. #include< stepper. h> for stepper
3. Then it must define the LCD terminals to which pins of Arduino Uno control here: suggested pins are: 12, 6, and 5,4,3,2
4. Choose the pins of controller that will be connected to stepper motor which will be here choose; 8, 9, 10, 11
5. Need to define the number of motor steps which is fixed to which means number of steps per revolution.

Set the pin no. 13 of controller to the input signal which its frequency needs to be monitored.
6. Set (T) to present the cycle time as float variable.
7. Set "frequency" for signal frequency as float variable.

The input signal is a sine wave which must be converted to logic and then divide by two to get period time (T) which represents cycle time, this time it will provide microcontroller and the program measures by number of internal generated (1µsec) signal, the total number equal to (T) in (µsec)
Signal, through this equation:

$$f = \frac{1}{number\ of\ pulses \times \mu s} \quad \ldots (1)$$

$$f = \frac{1000000}{T} \quad \ldots (2)$$

Due to the program monitors frequency and any variations in its value in referring to the selected Value of frequency, the program will drive stepper to change the fuel gate to change the speed of the generator (which here is potentiometer). The motor moves clockwise (CW) or Counter clockwise (CCW) according to the changes and it handles pot of oscillator until frequency be as a settled one.

(Appendix B)
Configuration:

- declare pin 13 as input:
Pin mod (13, input):
- operate LCD 16*2:
- set the speed of stepper motor to 60 RPM stepper. Set speed (60);
- To show results on pc by using serial port:
Serial. Begin (9600);
- locate the data row and column; begin at first row and fist column LCD. Set cursor (0, 0);
-use instruction "pulse In" to measure the time cycle in µsec and put it in variable " T " from pin 13 while the input signal " High "
T=pulse in (13, high);-calculate the frequency F= 1/T=1/T in µsec =1,000,000/T in a variable "frequency":
Frequency = 1000000/T;
-show frequency readings on LCD for PC:
Serial. Print in (frequency);-show frequency on the LCD 2*16 by write "Freq. =", follow it by writing the value of variable (Frequency) then write the "Hz"
LCD. print ("Freq. =");
LCD. print (Frequency);
LCD. print ("H_Z");
-to read clearly need to make a delay time for new reading for some time 100 msec
Delay (100):

-Checking the frequency and use the condition statements to manage the stepper motor movement and its direction check, the sign + operates motor counter clockwise direction and the sign – operates motor counter clockwise direction. The selected frequency range is 50.01H_Z to 50.02 H_Z

If (frequency > 50.01)

{Stepper. step (steps) ;} ——— CW

Delay (100);

If (frequency < 50.02)

{Stepper. step (-steps) ;} ——— CCW

-if frequency 50.00 HZ

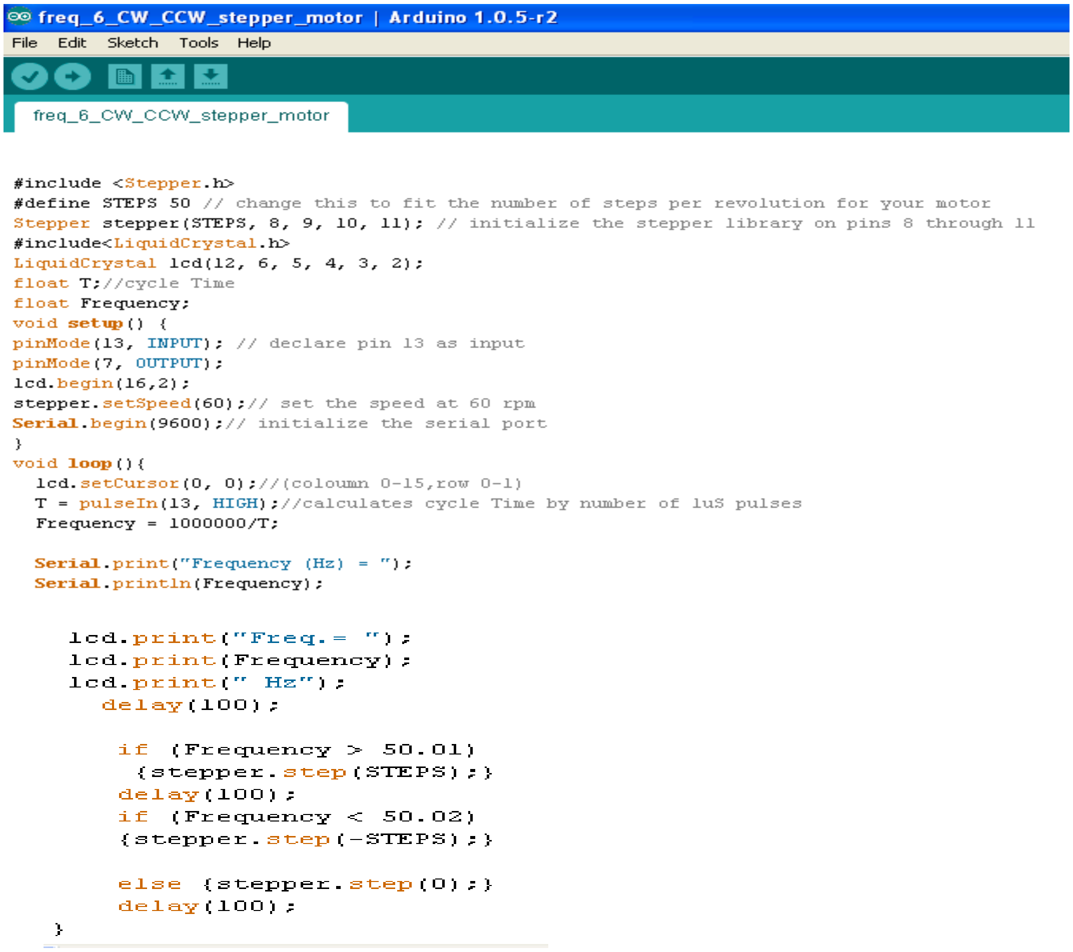
The motor stop and a delay for 100 msec and go on

Else {stepper. step (0) ;}

Delay (100);

-end

Arduino Code to calculate cycle time by number of 1µsec pulses


freq_6_CW_CCW_stepper_motor | Arduino 1.0.5-r2
File
Edit
Sketch
Tools
Help
freq_6_CW_CCW_stepper_motor

```
#include <Stepper.h>
#define STEPS 50 // change this to fit the number of steps per revolution for your motor
Stepper stepper(STEPS, 8, 9, 10, 11); // initialize the stepper library on pins 8 through 11
#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 6, 5, 4, 3, 2);
float T;//cycle Time
float Frequency;
void setup() {
pinMode(13, INPUT); // declare pin 13 as input
pinMode(7, OUTPUT);
lcd.begin(16,2);
stepper.setSpeed(60);// set the speed at 60 rpm
Serial.begin(9600);// initialize the serial port
}
void loop(){
  lcd.setCursor(0, 0);//(coloumn 0-15,row 0-1)
  T = pulseIn(13, HIGH);//calculates cycle Time by number of 1uS pulses
  Frequency = 1000000/T;

  Serial.print("Frequency (Hz) = ");
  Serial.println(Frequency);

     lcd.print("Freq.= ");
     lcd.print(Frequency);
     lcd.print(" Hz");
        delay(100);

          if (Frequency > 50.01)
           {stepper.step(STEPS);}
          delay(100);
          if (Frequency < 50.02)
          {stepper.step(-STEPS);}

          else {stepper.step(0);}
          delay(100);
 }
```

Figure (A): Code inthe Arduino IDE for measuring cycle time