Design and Simulation of a Wind Speed Measuring System for Cup Anemometer using an 8-Bit Processor

Akinkuade S. T
Science Technology Department,
Federal Polytechnic,
Ado-Ekiti, Nigeria

Oni S. A
Physics Department,
Ekiti State College of Education,
Ikere-Ekiti, Nigeria

***Abstract*—Wind is of high importance man; accurate measurement of the speed of wind at different locations can provide vital information to those that are concerned. A method of converting the number of pulses produced per second by the rotary encoder of a three-cup anemometer to instantaneous speed of wind is designed; the design is based on division of a 16-bit binary number by an 8-bit number and the conversion of 2-byte Hexadecimal number to binary coded decimal using an 8-bit microcontroller. Results of simulation shows that wind speed can be measured accurately with a resolution of 0.05 m/s.**

***Keywords*—Wind;Speed;Encoder:nemometer;Hexadecimal; Decimal; Resolution.**

I. INTRODUCTION

A meteorological device for measuring the speed of wind is an anemometer. [1] is of the opinion that anemometers can be used to measure wind pressure. The simplest type of anemometer is the cup anemometer which consists of a vertical shaft with a number of horizontal arms on which cups are attached, the cups catch the wind and cause the shaft to rotate.Shortcomings of this type of anemometer such as relative insensitivity in low wind velocities and large inertia of cups which leads to overestimation of velocity in a gusty windhad been removed by careful design [2].This type of anemometer continues to be relevant in wind speed measurement because they are the only instruments accepted in power performance measurements in international standards [3]. Several designs of cup, shaft, bearing and encoder of cup anemometers as well as combination of parts for whole anemometer system is given by [1].Anemometers can provide useful information to man, knowledge of wind velocity is required both in air and sea transportation, in weather forecasting, in agriculture to monitor soil erosion as well as evapotranspiration from plants, in pollution and wildfire control and in determination of safety of suspension bridges and site workers in tall buildings. It is a major factor that must be considered if a site will be suitable for wind farm.Since the power obtained from the wind is proportional to the cube of the wind speed, a small error in themeasurement will results in a much larger error in thepredictedwind power.

Major sources of uncertainties in power performance of wind turbinesare due to errors associated with the measurement of wind speed [4]. For easy reading and recording, the output of an anemometer should be electrical, this can be in form of voltage, current or pulse signal that can be digitized internally, displayed on readout or sent to a data acquisition system.In a cup-anemometer where electrical pulses are generated in proportion to the wind speed, the number of pulses generated per revolution depends on the rotary encoder of the anemometer, the number can vary between one to three for a magnet-based encoder and six to forty-four for an optoelectronics-based system [5].The use of a microelectronics system for processing the number of pulses generated in an anemometer, in order to produce the output signal, will result in better accuracy and reliability because of its ability to perform many operational tasks over and over automatically.However an arithmetic operation on a 16-bit data which may be produced in the system using 8-bit microprocessors is a challenge. In this paper, methods that were employed to process a 16-bit binary number in order to obtain a result in Binary Coded Decimal (BCD) are presented.

II. THEORY OF OPERATION OF CUP ANEMOMETER

The cups of the anemometer undergo a circular motion in a horizontal plane as they catch the wind this is shown in figure 1, the vertical shaft to which they are attached rotates about a its central axis.The tangential velocity 'v' of a cup is a measure of the average velocity of the wind; it is related to the angular velocity ω of the rotating shaft by:

$$v = \omega r \qquad (1)$$

Where r is the radius of the circular motion of the cups measured in metres, it is the distance between the centre of each cup to the centre of rotation of the shaft, v is measured in metres per second (m/s) while ω is measured in radians per second (rad/s). The angular displacement θ of the shaft in a period of time t depends on ω as shown below:

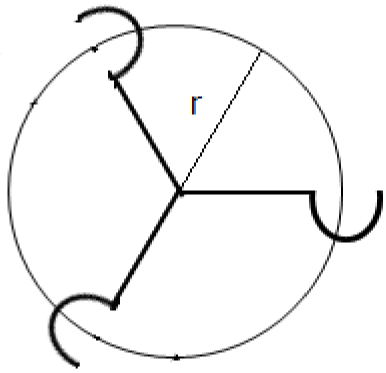

Fig. 1 Circular path of rotating cups of an anemometer.

$\omega = \theta/t$(2)

The total angular displacement in a complete revolution is 2π radians, while the time to complete the revolution is the period *T*, according to (2),

$\omega = 2\pi/T$(3)

The number of complete revolutions per second is the frequency *f* of the shaft, it is measured in Hertz, and it is equal to the inverse of the period.Therefore (3) can be expressed as:

$$\omega = 2\pi f \quad (4)$$

The expression that relates the tangential velocity of the wind to the frequency of rotation of the shaft of an anemometer can be obtained from (1) and (4) as:

$v = 2\pi f r$(5)

The shaft of a cup anemometer is normally attached to a rotary encoder which rotates with the shaft and ensure that a number of electrical pulses are produced per revolution, the encoder is expected to generate ten pulses per revolution, it is to be made of a circular plate on which ten equally spaced circular holes are to be drilled as depicted in figure 2. As theencoderrotates with the shaft it will interrupt a beam of light incident on a photo sensor; so for each complete revolution of the shaft, ten electrical pulses will be generated.

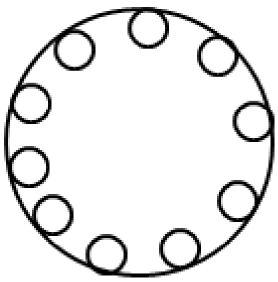

Fig. 2 Rotary encoder of the anemometer

If the shaft is rotating at a constant speed, the number of pulses *n*produced per period of revolution is given as:

$$n = 10/T = 10f \quad (7)$$

Therefore

$$f = n/10 \quad (8)$$

Using (8) in (5),

$$v = 2\pi rn/10 \quad (9)$$

From (9), velocity of the wind is proportional to the number of pulses generated per unit time, the constant of proportionality depends on the *r*; making *r* to be 0.0795 m, makes *v*and*n* to be related as:

$$v = n/20 \quad (10)$$

According to [5], the relationship between the frequency and the wind speed can be expressed as:

$$v = Af + B \quad (11)$$

Where *A* and*B* are coefficients which are obtained through calibration in a wind tunnel.

III. COMPUTATION OF THEWIND SPEED

A. *Division of 2-byte number by a byte*

The count generated in the anemometer can be represented as $D_3D_2D_1D_0(Hex)$ was divided by 0×14, byte by byte as follow, register A was used to hold $D_3D_2$and register B is used to hold the divisor 0×14.Using the DIV A,B instruction,D_3D_2 was divided by 0×14, after the execution of the instruction, the quotient Q_3 in register A was kept in a memory location while the remainder R_1in register B was multiplied by 0×10 and added to the third digit D_1, this resulted in $10R_1 + D_1$. R_1 cannot be more than 0×13, while the maximum value of D_1 cannot be greater than 0×0F, i.e.

$$10R_1 + D_1 \leq 13F \quad (11)$$

The sum $10R_1 + D_1$ is checked if the most significant bit (MSB) is 1 or 0, If the MSB is 0, the result is a byte, it is divided by 0×14andthe quotient Q_2 is kept in another memory location, but If MSB is 1, the division is performed as:

$$(10R_1 + D_1)/0{\times}14 = 0{\times}100/0{\times}14 + LM/0{\times}14 \quad (12)$$

LM being the last two digits of $(10R_1 + D_1)$ in hexadecimal, division of 0×100 by 0×14 results in 0×0C with 0×10 as its remainder. The remainder is added to *LM* prior to division by 0×14, the quotient of (0×10 + LM)/0×14 is added to 0×0C and kept as Q_2in a location in the memory. Remainder of this division is multiplied by 0×10 and added to the last digit *Z*, the result is divided by 0×14 as explained. Therefore,

$$WXYZ(Hex)/0{\times}14 = Q_3Q_2Q_1 + R \quad (13)$$

Where $Q_3Q_2Q_1$is the quotient of the division, and R is the remainder. Q_3, Q_2, Q_1, and *R*, are 8-bit binary numbers. Q_2 and $Q_1$are combined as a byte.

B. *Fractional part of the calculation*

The remainder *R* in (13), being a fraction of 0×14and a fraction of 20 in decimal, *R* is multiplied by 5 to make a fraction of 10 in decimal, the result is converted to BCD then ASCII, and displayed after a decimal point.

C. *Conversion of 2-byte hexadecimal number to binary coded decimal (BCD)*

A 2-byte hexadecimal number; $A_3A_2A_1A_0$ is converted to BCD by starting with the least byte A_1A_0, this is divided by 0×0A successively until the quotient is zero, the remainder of the last division is the Most Significant Bit (MSB), if the remainders are D_2, D_1 and D_0 respectively, they are arranged as packed BCD numbers;$D_2 D_1D_0$. The hexadecimal number N in the upper byte A_3A_2 is taken as N(FF+01) since it is generated each time a carry is generated from lower byte, this is equivalent to 256N in Decimal. 56 is therefore added to D_1D_0in decimal, the result is kept in a register and 2 is added with carry to $D_2$the result is kept in another register, any carry bit generated is taken care of, This procedure is repeated N times, the contents of the two registers and the carry is the BCD equivalent of $A_3A_2A_1A_0$.

IV. METHODOLOGY

Timer 1 of a microcontroller AT 80C52 was used in mode 01 i.e. as a 16-bit counter. It was made to count the number of pulses in Hexadecimal system, for one second; the count was held in two registers TH1 and TL1 of the timer, it was divided by 0×14i.e. 20 in decimal, according to (10), the integral and fractional parts of the result were converted to Binary coded decimal, then to American Standard Code For Information Exchange (ASCII) for it to be displayed on a dot matrix liquid crystal display (LCD) as the speed in m/s. The source code was written in assembly language, compiled and simulated using the diagram shown in figure 3, clock pulses of different frequencies were fed to the microcontroller and the results were shown in table 1.

V. SOURCE CODE OF THE SYSTEM

```
ORG 0000H
MOV P2,#00H         ;make P2 an output port
MOV P1,#0FFH                ;make P1 an output port
MOV P0,#00H         ;make P0 an input port
MOV P3,#00H         ;make P3 an input port
SETB P3.5
MOV TMOD,#51H       ;make timer 1 a 16-bit counter
MAIN:
MOV TH1,#00H        ;Clear timer 0 register
MOV TL1,#00H
CALL DELAY          ; wait for 0.5 sec
CALL COUNT          ;store count into locations 38&
39
CALL FEND   ;convert count to velocity by dividing by
20
CALL HEXBCD;call the subroutine to ;convert hex to
bcd
CALL DODO           ;call display subroutine
CALL FINAL
JMP MAIN

DELAY:
SETB TR1    ;start counter
BACK:
MOV R3,#14H         ;r3 = 20
WAS: MOV R2,#63H            ;r2 = 100
 DEW: MOV R1,#0FBH          ;r1 = 250DJNZ R1,$
 DJNZ R2,DEW
DJNZ R3,WAS
CLR TR1         ;stop counter
RET

DELAY2:
MOV R3,#00FH ;r3 = 15
DJNZ R3,$
RET

COUNT:
MOV 39,TL1      ;move low byte of the count to A
MOV 38,TH1      ;move high byte of the count to A
RET

FEND:
MOV A,38        ;copy TH1 to accumulator
MOV B,#14H      ;place 14 in register B
DIV AB          ;divide th1 by 14
MOV 30,A        ;store Q1 in location 30
MOV A,B         ;move Rem1 into A
MOV B,#10H      ;pace 10 in B
MUL AB          ;(Rem1 x 10)B= MS BYTE, A= LS
BYTE
MOV 36,A        ;store LS BYTE in 36
MOV A,B         ;copy MS Byte into A
MOV 37,A        ;store ms byte into 37
MOV A,39        ;copy TL1 to A
ANL A,#0F0H     ;clear lower nible of tl1
SWAP A          ;
MOV 42,A        ;copy upper nible of TL1 into 42
ADD A,36        ;LS byte of (Rem1 x 16)+uppernible of
TL1
MOV 43,A        ;store the result in 43
MOV A,37;copy MS byte of (Rem1 x 16)into A
CJNE A,#01H,SEAT        ;if A is not =1,go to SEAT
MOV 40,#00CH            ;100/14= C + 10, place C ;in 40
MOV 41,#10H             ;place 10 in location 41
MOV A,43                ;
ADD A,41        ;rem of 100/14;10+content of 43
SJMP SAT                ;go to SAT
SEAT:
MOV 40,#00H  ;if MS byte of ;Rem1*16)is not=1
MOV A,43
SAT:   MOV B,#14H               ;B=14
DIV AB                          ;A/14
ADD A,40                ;add the quotient to C
MOV 31,A        ;keep the second digit in 31
MOV A,B         ;mov Rem into A
MOV B,#10H
MUL AB          ;A= LS BYTE, B= MS BYTE
MOV 34,A                ;LS BYTE STORED
MOV A,B
MOV 35,A        ;STORE MS BYTE INTO 35
MOV A,39                ;copy TL1 to A
ANL A,#00FH     ;CLEAR UPPER NIBLLE OF TL1
MOV 42,A                ;keep the result in 42
ADD A,34        ;add the result to LS byte of TL1
MOV 43,A                ;keep result in 43
```

```
MOV A,35              ;
CJNE A,#01H,SEAT1
MOV 40,#00CH
MOV 41,#10H
MOV A,43
ADD A,41
SJMP SAT1

SEAT1:
MOV 40,#00H
MOV A,42
ADD A,34

SAT1:
MOV B,#14H
DIV AB
ADD A,40
MOV 32,A      ;3rd digit          MOV A,B
MOV 33,A
MOV A,31      ;combine 2nd & 3rd digits to a byte
SWAP A        ;combine 2nd & 3rd digits to a byte
ORL A,32      ;combine 2nd & 3rd digits to a byte
MOV R0,A      ;keep result in R0
MOV A,30      ;get 1st digit
MOV R7,A      ;keep it in R7
RET

HEXBCD:
MOV A,R0      ;move the low byte of hex no into A
MOV B,#00AH ;put 10 into register B
DIV AB        ;divide content of A by 10
MOV R0,B      ;put the remainder in R0
MOV B,#00AH ;put 10 into register B
DIV AB; divide cont. of A by 10 ;(quotient is in A, rem in B)
MOV R1,B      ;put the remainder in R1
MOV R3,A      ;put the msd of low byte in R3
MOV A,R1
SWAP A
ORL A,R0
MOV R0,A
MOV A,R7      ;move the high byte of the hex no ;into A
MOV R5,A      ;copy the same number in R5
MOV R6,#00H ;set R6 to zero
CJNE R7,#00H,MORE
MOV R4,#00H ;if R7 =0, let R4 =0 and R2 =0
MOV R2,#00H
JMP SUM       ;go to sum
MORE:
MOV R4,#56H ;if R7 is not zero, let R4 =56, R2=2
MOV R2,#02H

SUM:
MOV A,R0      ;put the content of R1 into A
ADD A, R4     ;add the content of R4 to A
DA A          ;convert the result to BCD
MOV R0,A      ;store the result in R1
MOV A,R3      ;move R3 into A
ADDC A,R2     ;add R2 to A Plus any carry earlier ;generated
DA A          ;convert result to BCD
JNC REST      ;if there is no carry, go to REST
INC R6        ;if there is carry, increase R6 by 1
REST:   MOV R3,A      ;store the sum in R3
DJNZ R5,SUM   ;reduce R5 by 1, if it is not zero, go ;to sum

OUT:
MOV A,R6      ;put the content of R6 in A
LAST:
MOV A,33
MOV B,#05H
MUL AB
MOV B,#00AH
DIV AB
MOV 45,A
MOV A,B
SWAP A
ORL A,45
SWAP A
MOV 45,A
RET

DODO:
MOV A,R3      ;mov content of R3 TO ;accumulator
ANL A,#0F0H   ;clear the lower nibble
SWAP A
ORL A,#30H    ;convert BCD to ASCII
MOV 48,A      ;keep first digit in location 48
MOV A,R3      ;mov content of R3 TO ;accumulator
ANL A,#00FH   ;clear the UPPER nibble
ORL A,#30H    ;convert BCD to ASCII
MOV 49,A      ;keep SECOND digit in location 49
MOV A,R0      ;;mov content of R0 TO ;accumulator
ANL A,#0F0H   ;clear the lower nibble
SWAP A
ORL A,#30H    ;convert BCD to ASCII
MOV 4AH,A     ;keep THIRD digit in location 4A
MOV A,R0      ;;mov content of R0 TO ;accumulator
ANL A,#00FH   ;;clear the UPPER nibble
ORL A,#30H    ;convert BCD to ASCII
MOV 4BH,A     ;keep 4th digit in location 4B
MOV A,45      ;;mov content of 45 TO ;accumulator
ANL A,#0F0H   ;clear the lower nibble
SWAP A
ORL A,#30H    ;convert BCD to ASCII
MOV 4CH,A     ;keep FIFTH digit in location 4C
MOV A,45      ;;mov content of 45 TO ;accumulator
ANL A,#00FH   ;;clear the UPPER nibble
ORL A,#30H    ;convert BCD to ASCII
MOV 4DH,A     ;keep SIXTH digit in location 4D
RET
FINAL:
MOV A,#0CH ;shift cursor right
ACALL COMNWRT ;call command subroutine
ACALL DELAY3 ;give LCD some time
MOV A,#80H ;cursor at line 1, pos. 0
ACALL COMNWRT ;call command subroutine
ACALL DELAY3 ;give LCD some time
```

```
MOV A,48; copy the content of location 48 to ;Accumulator
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,49;copy the content of location 49 to ;Accumulator
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,4AH;copy the content of location 4A to ;Accumulator
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,4BH;copy the content of location 4B to ;Accumulator ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,#'.' ;display DECIMAL POINT
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,4CH;copy the content of location 4C to ;Accumulator
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,4DH;copy the content of location 4D to ;Accumulator
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,#' ' ;display a space
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,#'m' ;display letter m
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,#'/' ;display /
ACALL DATAWRT ;call display subroutine
ACALL DELAY3 ;give LCD some time
MOV A,#'s' ;display letter s
ACALL DATAWRT ;call display subroutine
RET

COMNWRT: ;send command to LCD
MOV P1,A ;copy reg A to port 1
CLR P3.0 ;RS=0 for command
CLR P3.1 ;R/W=0 for write
SETB P3.2 ;E=1 for high pulse
ACALL DELAY3 ;give LCD some time
CLR P3.2 ;E=0 for H-to-L pulse
RET
DATAWRT: ;write data to LCD
MOV P1,A ;copy reg A to port 1
SETB P3.0 ;RS=1 for data
CLR P3.1 ;R/W=0 for write
SETB P3.2 ;E=1 for high pulse
ACALL DELAY3 ;give LCD some time
CLR P3.2 ;E=0 for H-to-L pulse
RET
DELAY3: MOV R3,#50 ;50 or higher for fast CPUs
HERE2: MOV R4,#255 ;R4 = 255
HERE:  DJNZ R4,HERE ;stay until R4 becomes 0
       DJNZ R3,HERE2
       RET
       END
```

VI. SHEMATIC OF THE DESIGN

The diagram of the system is as shown in Fig. 3

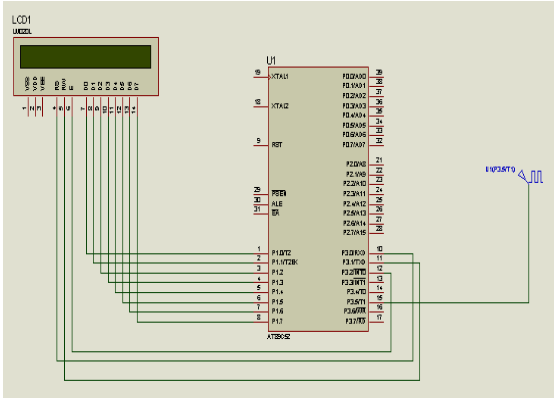

Fig. 3 Schematic of the measuring System

VII. RESULTS OF SIMULATION

The results obtained for different frequencies during simulation and the expected values based on (10) is shown in table 1.

TABLE I.RESULTS OF SIMULATION

Frequency (Hz)	**Displayed results (m/s)**	**Expected results (m/s)**	**Absolute % error**
1.0	0000.05	0.05	0
50.0	0002.50	2.50	0
320.0	0016.00	16.00	0
5600.0	0280.00	280.00	0
65500.0	3274.90	3275.00	0.00305
65536.0	3276.70	3276.80	0.00305
65537.0	3276.75	3276.85	0.00305
65538.0	3276.75	3276.90	0.00457
65539.0	0000.05	3276.95	99.9984
65540.0	0000.10	3277.00	99.9969

VIII. DISCUSSION

For low frequencies, pulses are counted accurately. However at high frequenciesbetween65500.0 Hz and 65537.0 Hz, there is an error of 0.1in the displayed result, this amounted to two pulses short of the expected number, and percentage error of 0.00305; the maximum count that can be held in TH1 and TL1 registers is FFFF in Hexadecimal which is equivalent to 65535 in Decimal, due to shortage of count, acceptable results were obtained up to 65538.0 Hz, above this frequency, the count in the registers roll over and the excess count remains in them. This is computed and shown in the display hence the sudden rise in percentage error at 65539.0 Hz and above.

IX. CONCLUSION

The result of simulation shows that the system can be used to measure the speed of wind in a cup anemometer accurately. The minimum and maximum speeds that can be measured are 0.25 and 3276.75 m/s respectively. The offset *B* and any other necessary corrections have be determined during calibration, after the construction of the anemometer.

X. REFERENCES

[1] Tan Wee Choon, Churia Prakash, Lim EngAik, Teoh TheanHin, Development of Low Wind Speed Anemometer, *International Journal on Advanced Science Engineering Information Technology*,Vol.2 No.3 ISSN 2088-5334, 2012

[2] Sheppard P A,an Improved Design of Cup Anemometer, Imperial College of Science and Technology 1940, http://ualberta.ca 2014

[3] Troels F P, Development of Classification System for Cup Anemometers- CLASSCUP, Pitney Bowes Management Systems, Roskilde, 2003, pg. 5

[4] J. A. Dahlberg, T.F. Pedersen & P. Busch, ACCUWIND-Methods for Classification of Cup Anemometers, Riso National Laboratory, Roskidle, 2006, pg. 68, ISBN 87-550-3514-0

[5] S. Pindado, J. Cubas, & F. Sorribes-Palmer, The Cup Anemometer, a Fundamental Meteorological Instrument for the Wind Energy Industry, International Electronic Conference on Sensors and Applications, www.mdpi.com/journal/sensors, 2014.