International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.8 (2014), pp.25-40
http://dx.doi.org/10.14257/ijmue.2014.9.8.03

Continues Blood Pressure Measurement and Data Logging Device with SMS Alert

M. K. Chaithanya, K. V. K. Kishore and Avireni Srinivasulu†

VFSTR University, Vadlamudi, Guntur, A.P, India.
Corresponding author: †avireni_s@yahoo.com (or) avireni@ieee.org

Abstract

This paper presents a new blood pressure (BPs) measurement and data logging device which can be extensively used for ICU patients who are intended for a period of observation. The main purpose of this paper is to know about the changes in BP with respect to time and changes in the food taken by patient who are under observation. If any abnormal changes occur in the BP then the system alerts using a buzzer and it also sends a message to a predefined number (i.e., a physician number) using GSM. If not for each and every second it saves the data into SDcard in a file format, therefore appropriate steps can be taken in accordance with the changes in BP. The paper also took care of layman towards BP inorder to understand everyone even without the help of physician. This paper follows a Non-invasive type of BP measuring technique i.e., volume oscillometric method. Through the developed algorithm the BP measured is more accurate which is compared to the readings measured by sphygmomanometer. This system could be made with the features like cost effective, less complex, more reliable and easy to access.

Keywords: *Blood pressure; PPG sensor; Photoplethysmography; Data logging; RTC; SD card module; GSM module*

1. Introduction

Blood pressure which is acronym of BP refers to the variations in the heart beat or heart rate. During each heart beat, BP varies between a systolic (maximum) and diastolic (minimum) pressures in blood circulation, which is due to pumping action of heart. Blood pressure measurement is an important task, whenever a person or a patient is treated or diagnosed. This is essential because it reflects effective functioning of heart. Blood pressure measurement is an important task in the medical field especially during operations. It not, only gives the information about heart but also gives the information about various important organs like kidneys, liver, brain etc. Therefore, accurate BP measurement is a major task that is examined for every individual.

There are two different mechanisms to measure BP. They are invasive [1] and non invasive [1] BP measurement mechanisms. Invasive mechanism involves by inserting a catheter into vascular system. Therefore it hurts the patient or a person to whom BP is to be measured. On the other hand non-invasive mechanism measures BP indirectly which is easier and safer.

Non-invasive mechanism involves in measuring BP by five different methods namely, electronic palpation method [2], pulse wave velocity method [3], volume compensation method [4], volume oscillometric method [5] and arterial tonometry method [6]. Each of these methods has its own advantages as well as disadvantages. Despite of these pros and cons volume oscillometric method is more efficient and easiest one with less complexity. Also Author in reference [5] suggested that volume oscillometric method shows good results when compared to practically using instruments nowadays. But in [5] amplifier calibration is required depending on different volumes of finger.

ISSN: 1975-0080 IJMUE

The present devices like pulse oximeters, sphygmomanometers [8] are operating with a complex operating mechanism and even with the help of stethoscope which follows detection of korotkoff sounds [7] and determination of blood pressures. Among all the above methods volume oscillometric method is simpler and easier which is adopted in the present paper through photoplethysmography [8]. It is the rate of change of blood volume in finger that has a linear relationship with blood pressure, which is measured by an optical sensor. Thus the pressure levels are calculated in an effective manner so as to determine BP.

For the patients or victims in ICUs it is necessary to study about the variations of BP with respect to time. Inspite of this, their BP is examined for regular intervals of time and the appropriate treatment is done. But this is not to be done since their BP will vary with time and food habits. Hence there is a need to measure BP continuously with respect to changes in time. On the other hand indication of BP is not understood by layman of the medical know how which requires determination of character of BP. Along with this, an SMS alert is sent when BP goes abnormal *i.e.*, high/low to a physician.

2. Proposed System

Figure 1 shows the block diagram of developed system which has different components like PPG sensor, real time clock, SD card module and other necessary elements like power supply, oscillators, LCD *etc*.

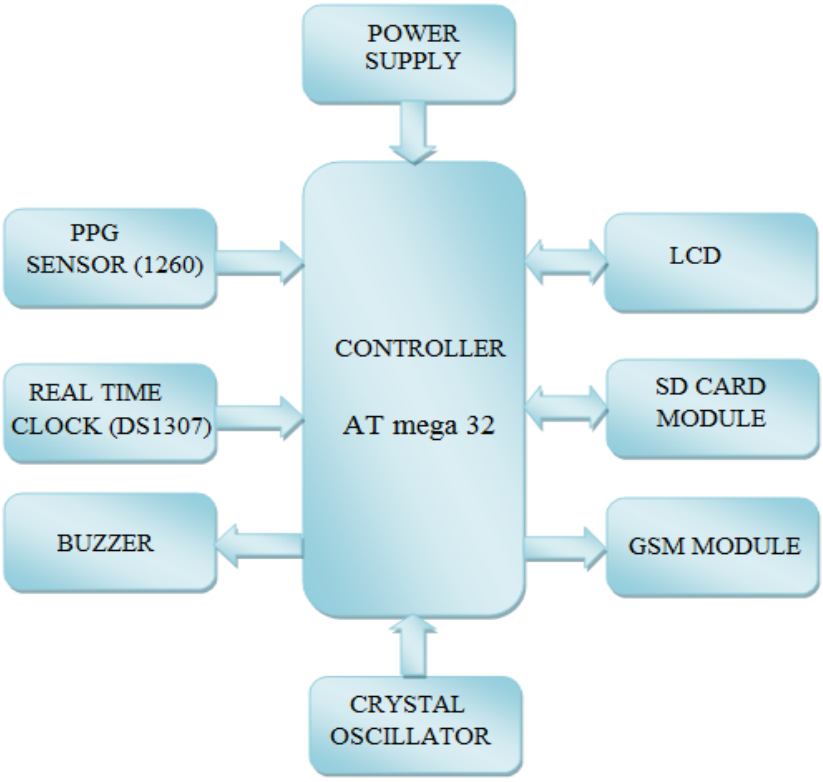

Figure 1. Block Diagram of Proposed System

In the paper, the controller selected is ATmega32 that supports the required peripherals for the required objective. It is programmed using "AVR studio 4.18" and "AT PROG" for dumping the code into the controller. This controller is supported by SPI and I2C communications.

Power supply and crystal oscillators are the two units that are essential for controller operations effectively which provide necessary clock frequency, ambient voltage and

currents. Real Time Clock (RTC) plays a major role in the paper that determines date and time in correspondence to the BP measured. Here, an external RTC (DS1307) is used that gives continuous date and time values after initialization. It can also work with a battery which is avoided here to decrease the hardware complexity.

The another important module in the project is SD card module that has "microSD card" provision which stores the BP with date and time provided by RTC in a file format thus the objective of paper can be achieved. Along with this, we use buzzer and GSM for alerting nearby person and a remote person so that effective steps can be taken as soon as possible.

In order to store the data in SD card FAT32 implementation is done in it. The FAT file system offers good performance and robustness even in light-weight implementations. Therefore, it is widely adopted and supported by many operating systems. Thus, it makes a useful format for memory cards and a flexible way to share data among the operating systems. FAT32 implementation partitions the memory with a unit cluster size of 32 KB.

Figure 2 shows the PPG sensor (1260) that works on Photoplethysmography principle whose output is analog signal (optical signal) which is detected by photo diode that varies with respect to changes in blood volume where the input is taken from a 660 nm Red LED.

Figure 2. PPG Sensor

Figure 3 shows the process of photoplethysmography where a bright LED light is used as an input which is allowed to pass through the finger (since light can easily pass through the finger) and the transferred light is detected by a light detector like photo diode etc. As per the principle, light absorbed at other end varies according to the variations of blood volumes in the finger. The sensor used here is simple to use and accurate in results.

Here we define the character of BP from obtained systole and diastole values. After determining the character of blood pressure we are sending an SMS alert regarding patient number or ID so that one can suggest appropriate diagnosis. For this we employed SIM900 GSM module which operates at 900 MHz that is as shown in Figure 4. In order to initiate or send SMS the appropriate "AT (Attention)-commands" are sent to the GSM module where a prescribed number is already kept in program.

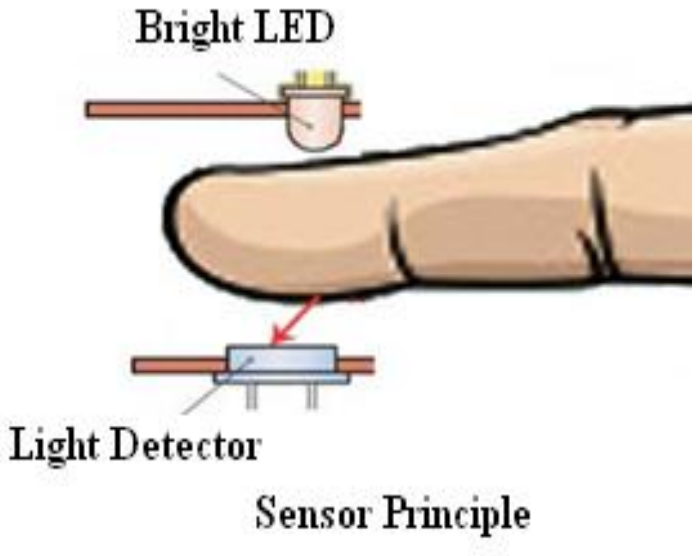

Figure 3. Principle of Photoplethysmography

Figure 4. GSM Module

3. Algorithm Developed

The developed algorithm is as follows,

1. Initialize SD card and check for FAT format.
2. Initialize RTC *i.e.*, updating date and time.
3. Define threshold value *i.e.*, minimum voltage when finger is inserted in hexadecimal equivalent (512 here).
4. Determine the maximum and minimum value among sample when compared to the threshold value which are systolic and diastolic pressures respectively.
5. Take average of some samples for a time period.
6. Time period when divided by average value gives "Beats Per Minute (BPM)."
7. Difference between systolic and diastolic pressure gives "Pulse Pressure [5]."

8. Apply the same formulae as in the case of [5] we calculate “Mean Arterial Pressure (MAP)”.
9. Determine character of BP
10. Take and send the data from RTC and determined blood pressure to SD card.
11. Depending on character of BP send SMS alert to a specified number.

The programming structure is as follows,

```
Main()
{
ADC_ROUTINES ();
UART_ROUTINES ();
RTC_ROUTINES ();
I2C_ROUTINES ();
FAT32_ROUTINES ();
SD_ROUTINES ();
SPI_ROUTINES ();
GSM_ROUTINES ();
}
```

The main code referring to this is as follows,

```
int main(void)
{
 _delay_ms (100);  //delay for VCC stabilization
  init_devices ();
  GREEN_LED_ON;  //turn on green LED to indicate power on
  RED_LED_OFF;      //keep red LED off for now
  transmitString_F
(PSTR("\n\r\n\r****************************************"));
  transmitString_F (PSTR("\n\r   Heart Beat Monitor  Loggger "));
  transmitString_F
(PSTR("\n\r****************************************\n\r"));
  LCD__Clear ();        //"1234567890123456"
  LCD__WriteStringXY (0,0,"---HB Monitor---");
  LCD__WriteStringXY (0,1,"Checking SDCard");
  _delay_ms (2000);
 LCD__Clear ();
 cardType = 0;
  for (i=0; i<10; i++)
  {
       error = SD_init ();
       if (!error) break;
  }
  if(error)
  {
       If (error == 1) transmitString_F(PSTR("SD card not detected.."));
       If (error == 2) transmitString_F(PSTR("Card Initialization failed.."));
       blinkRedLED();
  }
  switch (cardType)
  {
       case 1:transmitString_F(PSTR("SD Card  Detected!"));
       case 2:
       case 3: transmitString_F(PSTR("  SD Card Detected!"));
       LCD__WriteStringXY(0,0,"SD Card Detected");
```

```
        break;
        default:transmitString_F(PSTR("Unknown SD Card Detected!"));
        break;
    }
    error = getBootSectorData (); //read boot sector and keep necessary data in global
variables
    if(error)
    {
        transmitString_F (PSTR("\n\rFAT32 not found!"));  //FAT32 incompatible drive
        LCD__WriteStringXY(0,0,"FAT32 NOT Found");
                //"0123456789012345"
     LCD__WriteStringXY(0,1,"Wait for Input:  ");
        blinkRedLED ();
    }
    else
    {
         LCD__WriteStringXY(0,0,"FAT32 Found          ");
      LCD__WriteStringXY(0,1,"Wait for Input:  ");
    }
    SPI_HIGH_SPEED;            //SCK - 4 MHz
    _delay_ms(1);         //some delay for settling new spi speed
    if (KEY_PRESSED)
    while (1)
    {
        transmitString_F(PSTR("\n\r\n\r> 0 : Exit the Menu"));
        transmitString_F(PSTR("\n\r> 1 : Display current Date/Time"));
        transmitString_F(PSTR("\n\r> 2 : Update Date"));
        transmitString_F(PSTR("\n\r> 3 : Update Time"));
        transmitString_F(PSTR("\n\r> 4 : Get file list"));
        transmitString_F(PSTR("\n\r> 5 : Read File"));
        transmitString_F(PSTR("\n\r> 6 : Delete File"));
        transmitString_F(PSTR("\n\r\n\r> Enter the option:"));
        option = receiveByte();
        transmitByte(option);
        switch (option)
        {
                case '0':transmitString_F(PSTR("\n\rNormal operation started.."));
                                heartbeat ();
                 case '1':RTC_displayDate ();
                                RTC_displayTime ();
                                break;
                case '2':RTC_updateDate ();
                                break;
                case '3':RTC_updateTime ();
                        break;
                case '4':TX_NEWLINE;
                        findFiles (GET_LIST,0);
                        break;
                case '5':
                case '6':transmitString_F(PSTR("\n\rEnter file name: "));
                        for(i=0; i<13; i++)
                        fileName[i] = 0x00;   //clearing any previously stored file name
                        i=0;
```



```
                        while (1)
                        {
                        data = receiveByte ();
                        if (data == 0x0d) break;  //'ENTER' key pressed
                                        if (data == 0x08)          //'Back Space' key
pressed
                                        {
                                                if(i != 0)
                                                {
                                                transmitByte (data);
                                                        transmitByte (' ');
                                                transmitByte (data);
                                                i--;
                                                }
                                                continue;
                                        }
                                        if (data <0x20 || data > 0x7e) continue;
                transmitByte (data);
                        filename [i++] = data;
                        if(i==13){transmitString_F(PSTR(" file name too long.."));
break;}
                        }
                        if (i>12) break;
                        TX_NEWLINE;
                                if(option == '5')
                                {
                                error = readFile ( READ, fileName);
                                if (error == 1) transmitString_F (PSTR("File does not
exist.."));
                                }
                                If (option == '6') deleteFile(fileName);
                        break;
        default:transmitString_F(PSTR("\n\r\n\r Invalid option!\n\r"));
        }  }
    return 0;
   }//end of main
   unsigned char Array_Int [10];
   void inttoarray (unsigned int intval)
   {
           Array_Int [4] = (intval % 10) | 0x30;
           intval = intval / 10;
           Array_Int [3] = (intval % 10) | 0x30;
           intval = intval / 10;
           Array_Int[2] = (intval % 10) | 0x30;
           intval = intval / 10;
           Array_Int [1] = intval | 0x30;
           Array_Int [0] =' ';
   }
    void Logger (void)
    {
        RED_LED_ON;  //turn on red LED to indicate that recording has started
      transmitString_F(PSTR("\n\r\n\r Started Recording Data \n\r"));
                error = RTC_getDate ();
```

```
                if (error)
                {
                   transmitString_F (PSTR("\n\r\n\r RTC_getDate failed \n\r"));
                   blinkRedLED ();
                }
                j=0;
                for (i=0; i<8; i++)
                {
                        filename [i] = date[j++];
                        if (j==2) j++;    //excluding the '/' character from date in the
fileName
                        if (j==5) j=8;
                }
                If (create_newFile_F && (tmp_cnt <= 'z' ))
        {
                   create_newFile_F  = 0;
      fileName[6] = '_';
                        filename [7] = tmp_cnt;
                        tmp_cnt++;
                }
        else
        {
      filename [6] = '_';
                        filename [7] = '_';
                }
                fileName[8] = '.';
                fileName[9] = 'C';
                fileName[10] = 'S';
                fileName[11] = 'V';
                error = RTC_getTime();
        if(error)
                {
                   transmitString_F(PSTR("\n\r\n\r RTC_getTime failed \n\r"));
                   blinkRedLED();
                }
                for (i=0; i<10; i++) dataString [i] = date [i];
                dataString [i++] = ',';
                for (j=0;j<8; j++)  dataString [i++] = time[j];
                dataString [i++] = ',';
                inttoarray (final_bpm);
                Array_Int [5]='B';
                Array_Int [6]='P';
        for (j=0;j<7; j++)
        {
              dataString [i++] = Array_Int [j];
        }; //loading BPM
        dataString [i++] = ',';
                if (pulse_pressor > 160)
                        pulse_pressor -= 40;
                inttoarray (pulse_pressor);
                if (pulse_pressor <100)
                {
                   Array_Int [9]='W'      ;
```



```
                Array_Int [8]='O'       ;
                Array_Int [7]='L'       ;
            }
        else if (pulse_pressor >100 && pulse_pressor <130  )
        {
                Array_Int [9]='R'       ;
                Array_Int [8]='O'       ;
                Array_Int [7]='N'       ;
        }
          else
                {
                        Array_Int [9]='H'          ;
                        Array_Int [8]='I'          ;
                        Array_Int [7]='H'          ;
                        }
                Array_Int [5]='P';
                Array_Int [6]='P';
            For (j=0;j<10; j++)
        {
                 dataString [i++] = Array_Int[j];
                }
                dataString [i++] = ',';
                inttoarray (Map);
                Array_Int [5]='M';
                Array_Int [6]='P';
        for (j=0;j<7; j++)
        {
                        dataString [i++] = Array_Int[j];
                }//loading pulse_pressor
                dataString [i++] = ',';
        dataString [i++] = '\r';
                dataString [i++] = '\n';
                dataString [i] = '*'
                error = writeFile (fileName);
            if (error)
        {
                LCD_WriteFail();
                }
            else
        {
                        transmitString_F(PSTR("\n\r\n\r  Write Done\n\r"));
                }  }
   void TimerISr(void){                    // triggered when Timer2 counts to 124
     cli();                                // disable interrupts while we do this
     Signal = ADC__Read(7);              // read the Pulse Sensor
     sampleCounter += 2;                     // keep track of the time in mS with this variable
     int N = sampleCounter - lastBeatTime;        // monitor the time since the last beat to
avoid noise
     if(Signal < thresh && N > (IBI/5)*3){        // avoid dichrotic noise by waiting 3/5 of
last IBI
       if(Signal < T)
          {                    // T is the trough
         T = Signal;                     // keep track of lowest point in pulse wave
```

```
    } }
  if(Signal > thresh && Signal > P)
  {        // thresh condition helps avoid noise
      P = Signal;                        // P is the peak
  }                                  // keep track of highest point in pulse wave
  if(N > 250){                           // avoid high frequency noise
    if( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
      Pulse = true;                      // set the Pulse flag when we think there is a pulse
        GREEN_LED_ON;
      IBI = sampleCounter - lastBeatTime;       // measure time between beats in mS
      lastBeatTime = sampleCounter;             // keep track of time for next pulse
      if(secondBeat){                    // if this is the second beat, if secondBeat == TRUE
        secondBeat = false;              // clear secondBeat flag
        for(int i=0; i<=9; i++){         // seed the running total to get a realisitic BPM at
startup
          rate[i] = IBI;
        }     }
      If (firstBeat){                    // if it's the first time we found a beat, if firstBeat ==
TRUE
        firstBeat = false;               // clear firstBeat flag
        secondBeat = true;               // set the second beat flag
        sei();                           // enable interrupts again
      }
      unsigned long  int  runningTotal = 0;              // clear the runningTotal variable
      for(int i=0; i<=8; i++){           // shift data in the rate array
        rate [i] = rate[i+1];            // and drop the oldest IBI value
        runningTotal += rate[i];         // add up the 9 oldest IBI values
      }
      rate[9] = IBI;                     // add the latest IBI to the rate array
      runningTotal += rate[9];           // add the latest IBI to runningTotal
      runningTotal /= 10;                // average the last 10 IBI values
      BPM = 60000/runningTotal;          // how many beats can fit into a minute? that's
BPM!
        SyP =P;
        DyP=T;
      QS = true;                         // set Quantified Self flag
       }     }
  if (Signal < thresh && Pulse == true){   // when the values are going down, the beat is
over
    GREEN_LED_OFF;
    Pulse = false;                       // reset the Pulse flag so we can do it again
    amp = P - T;                         // get amplitude of the pulse wave
    thresh = amp/2 + T;                  // set thresh at 50% of the amplitude
    P = thresh;                          // reset these for next time
    T = thresh;
  }
  if(N > 2500){                          // if 2.5 seconds go by without a beat
    thresh = PEAK_VAL;// 512;                  // set thresh default
    P = PEAK_VAL;// 512;                       // set P default
    T = PEAK_VAL;//512;                        // set T default
    lastBeatTime = sampleCounter;        // bring the lastBeatTime up to date
    firstBeat = true;                    // set these to avoid noise
    secondBeat = false;                  // when we get the heartbeat back
```



```
}
sei ();                    // enable interrupts when you are done!
}
```

In the above programming, ADC_ROUTINES is used for reading sensor values from PPG sensor, RTC_ROUTINES is used for reading date and time while I2C_ROUTINES is for interfacing it, UART_ROUTINES is for serial communication, FAT32_ROUTINES is for checking file format in the SD card, SD_ROUTINES is for checking for SD card, while SPI_ROUTINES is for interfacing SD card with the controller board, GSM_ROUTINES is for sending message. All the above routines are used as modules in the main part of code.

The system operation begins with checking of SD card and FAT32 formats which is continued by date and time update and follows to measurement of BP, MAP and BPM which is displayed on LCD. When a particular push button is turned ON regarding recording of data will store the whole data in SD card continuously that is done by a timer action. Among this data when an abnormal BP is encountered then a particular message is sent to a desired number.

4. Hardware and Software Description

Figure 5 shows the developed board with SD card module interfaced to it. Development board consists of different modules like ATmega32 controller, RTC, serial connector, buzzer, GSM, LCD, push buttons, some LEDs etc. Almost all the units in the board are used in order to reach the objective.

Here, the controller selected is ATmega32 since it can be effectively interfaced with the SD card module and real time clock (RTC). Inspite of using internal RTC which is in the controller we are using external RTC since any power fluctuations can reset the internal RTC that results re-initialization of clock's time and date to default. Serial connector is used to connect RS232 through which we are updating date, time and making the device to enter into its operation with the help of personal computer or a desktop computer.

Figure 5. Development Board with SD Card Module

Software used here is AVR studio 4.18 which is applied for programming ATmega controllers that supports both assembler and GCC versions of programming. In such a way, it is highly flexible for programming and simplicity of coding. "WinAVR-20100110" is the installer that supports for automatic make file generator. Finally, ATmega Programmer (AT PROG) software is used to dump the code serially into the controller.

5. Results

Figure 6 shows blood pressure value with date and time. All this data is sent to the SD card through SPI mechanism.

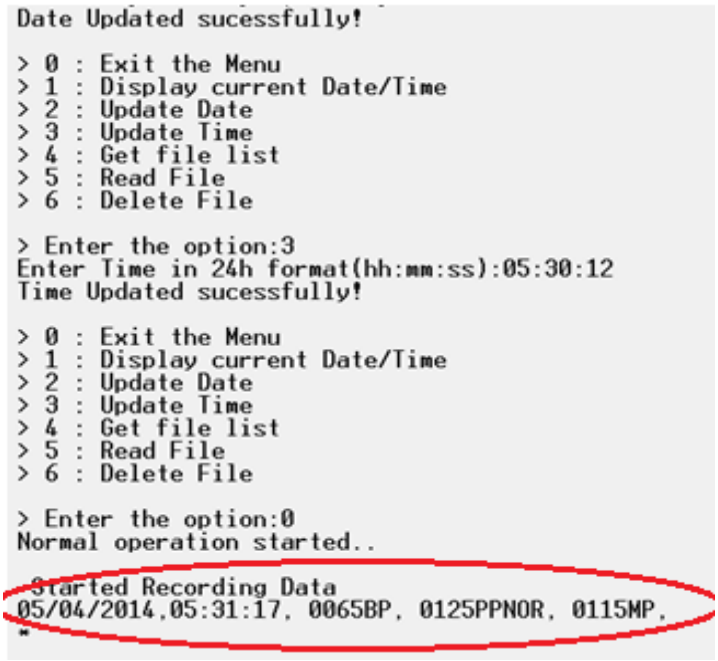

Figure 6. BP with Date and Time

Before we use the SD card, it should be formatted in FAT32 format so as enable the file system in it. Thus it can store the data in it, in a file format. Figure 7 and 8 shows the file that is saved in SD card and data in it which resembles the observed data in hyper terminal. It is not required to specify the file name the programming has been done in such a way that DATE is taken as default file name that can be seen in Figure 6.

Table 1 shows the blood pressure values that are compared with the sphygmomanometer readings where there is a slight difference between them.

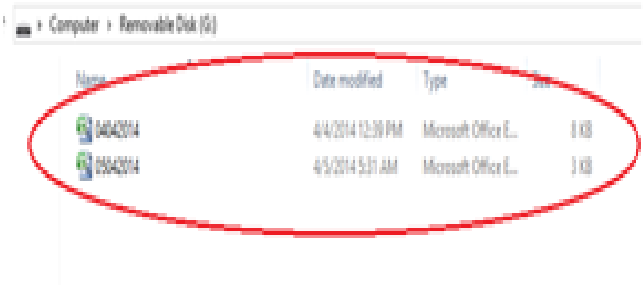

Figure 6. BP with Date and Time

*05/04/2014	10:35:09	0072BP	0170PPHI	0106MP
*05/04/2014	10:35:13	0060BP	0170PPHI	0106MP
*05/04/2014	5:31:17	0065BP	0125PPN(	0115MP
*05/04/2014	5:31:20	0065BP	0170PPHI	0115MP
*05/04/2014	5:31:23	0065BP	0170PPHI	0115MP
*05/04/2014	5:31:26	0065BP	0150PPHI	0115MP
*05/04/2014	5:31:28	0065BP	0150PPHI	0115MP
*05/04/2014	5:31:31	0065BP	0170PPHI	0115MP
*05/04/2014	5:31:34	0065BP	0170PPHI	0115MP
*05/04/2014	5:31:37	0065BP	0170PPHI	0115MP
*05/04/2014	5:31:40	0065BP	0170PPHI	0115MP
*05/04/2014	5:31:43	0089BP	0170PPHI	0115MP
*05/04/2014	5:31:47	0075BP	0170PPHI	0115MP
*05/04/2014	5:31:50	0079BP	0170PPHI	0115MP
*05/04/2014	5:31:53	0071BP	0170PPHI	0115MP
*05/04/2014	5:31:56	0070BP	0170PPHI	0115MP
*05/04/2014	5:31:59	0078BP	0170PPHI	0115MP

Figure 7. List of Files Stored in the SD Card

Table 1. Showing the Difference between Obtained Results and Sphygmomanometer Readings (Systole)

S.No	RESULT OBTAINED (systole)	SPGYHMOMANOMETER READINGS (systole)
1.	125	130 (normal)
2.	170	165 (high)
3.	170	168 (high)
4.	150	146 (high)
5.	150	147 (high)

6. Conclusion

Thus, it can be concluded that the developed system plays a major role in determining continues blood pressure with respect to time. This type of systems is mostly used when a patient is kept in observation for a period of time. By buzzer alert of the system one effectively take action on the patient and also by defining the character of BP a layman can also understand just by going through the file. On the other hand, physician can diagnose a patient from a remote location as system provides SMS alert at critical situations. By this a physician can also take effective steps towards him/her as the change in blood pressure is studied or observed clearly. Another major advantage is that a patient is studied thoroughly and accurately at low cost and less complexity.

References

[1] M. H. Winter, P. Escourrou, M. D. Goldman, M. Slama, L. Drueu and M. Jaffrin, “Comparison of invasive and noninvasive blood pressure measurements for determining cardiac output in man”, Proceedings of the 14th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Paris, France, **(1992)** October 29-November 1, pp. 744-745.

[2] S. Nissila, M. Sorvisto, H. Sorvoja, E. Vieri-Gashi and R. Myllyla, "Non-invasive blood pressure measurement based on the electronic palpation method", Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Hong Kong SAR, China, **(1998)** October 29-November 1; pp. 1723-1726.
[3] D. Chen and S. Wanqing, "The method for Pulse Wave Velocity measurement Based on Chaotic Oscillator", Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence, Shanghai, China, **(2009)** November 7-8, pp. 340-344.
[4] S. Gao, Y. Song, S. Tanaka and K. Yamakoshi, "Development of Instantaneous Blood Pressure Measuring System at the Wrist Based on Volume-Compensation Method", Proceedings of the 3rd International Conference on Bioinformatics and Biomedical Engineering, Beijing, China, **(2009)** June 11-16, pp. 1-4.
[5] Md. Manirul Islami, Fida Hasan Md. Rafi, A. F. Mitul, M. Ahmad, M. A. Rashid, and M. F Bin Abd Malek, "Development of a Noninvasive Continuous Blood Pressure Measurement and Monitoring System", Proceedings of the International Conference on Informatics, Electronics and Vision, Dhaka, Bangladesh, **(2012)** May 18-19, pp. 1085-1090.
[6] K. Shioya and T. Dohi, "Blood Pressure Measurement device based on the arterial tonometry method with micro triaxial force sensor", Proceedings of the 17th International Conference on Solid-State Sensors, Actuators and Microsystems and Eurosensors XXVII, Barcelona, Spain, **(2013)** June 16-20, pp. 2389-2392.
[7] M. Nitzan, "Automatic Noninvasive Measurement of Arterial Blood Pressure, Instrumentation and Measurement Magazine", IEEE, vol. 14, no. 1, **(2011)**, pp. 32-37.
[8] L. Wang, E. Pickwell-Macpherson and Y. T. Zhang, "Blood Pressure Contour Analysis after Exercise by the Photoplethysmogram Using a Transfer Function Method. Proceedings of the 5th International Summer School and Symposium on Medical Devices and Biosensors, Hong Kong, China, **(2008)** June 1-3, pp. 82-85.

Authors

M.Krishna Chaithanya received his B.Tech degree Electronics and Communication Engineering from JNT University, Ananthapur in 2012. He is now working towards his M.Tech thesis in Embedded Systems at Vignan's Foundation for Science, Technology and Research University, Vadlamudi, Guntur, India. His area of research includes Embedded Systems.

K.V.Krishna Kishore received his M.Tech degree in Computer Science and Engineering from Andhra University, Visakhapatnam. He is now working as Professor of CSE and Dean, IT Services in Vignan University. He got more than 22 years of teaching and research experience. His current research interests include Digital image processing, Biometrics, Face recognition, Facial emotional classifications, Neural Networks, Pattern Recognition and Embedded Systems. He has published more than 15 journal and conference papers in these areas. He has received best teacher award for four times in Vignan's Engineering College. He has Chaired National Youth festival for three times. He has chaired one National and IEEE International Conference.

Avireni Srinivasulu was born in Thurimella, (A.P), India. He received the B.Tech degree in electronics and communication engineering from Sri Venkateswara University, Tirupati in 1986, M.E, degree in power electronics engineering from Gulbarga University, Gulbarga in 1991, M.S, degree in software systems from Birla Institute of Technology and Science (BIT'S), Pilani in 1998 and Ph.D. degree in electronics & communication engineering (VLSI Design) from Birla Institute of Technology, Mesra, India in 2010. He is working as a Dean (R&D), Professor of electronics & communication engineering, VFSTR University, Guntur, India. He has 25 years of teaching and 15 years of research experience in the department of electronics & communication engineering. Dr. Avireni.S is a senior member of IEEE, senior member of IACSIT, life member of I.S.T.E and a member of the Institution of Engineers (India). He has published over 45 articles in international journals and international conference proceedings; his main research areas are microelectronics, VLSI design and analog ASIC.