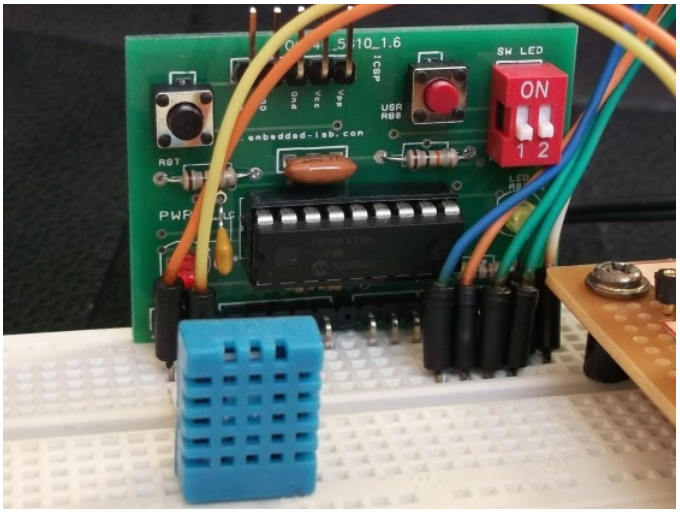
Measurement of temperature and relative humidity using DHT11 sensor and PIC microcontroller

- R-B
- Jan 10th, 2012
-

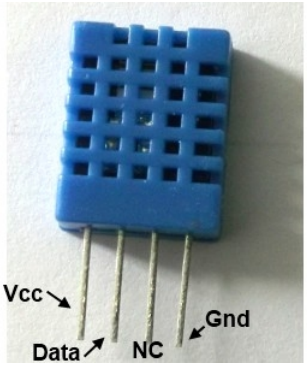
Measurement and control of temperature and relative humidity finds applications in numerous areas. These days devices are available which have both temperature and humidity sensors with signal conditioning, ADC, calibration and communication interface all built inside them. The use of such smart sensors greatly simplify the design and reduces the overall cost. We discussed in past about Humidity and temperature measurements with Sensirion's SHT1x/SHT7x sensors. These sensors are capable of measuring both temperature and relative humidity and provide fully calibrated digital outputs. While SHT1x/SHT7x are very accurate sensors, they are still expensive for hobbyists use. This articles discusses the DHT11 sensor which also provides calibrated digital outputs for temperature and humidity but is relatively lot cheaper than the Sensirion sensors. The DHT11 sensor uses a proprietary 1-wire protocol which we will be exploring here and implementing with the PIC16F628A microcontroller that will receive the temperature and humidity values from the sensor and display them on a 16×2 character LCD.



Interfacing DHT11 sensor with PIC16F628A

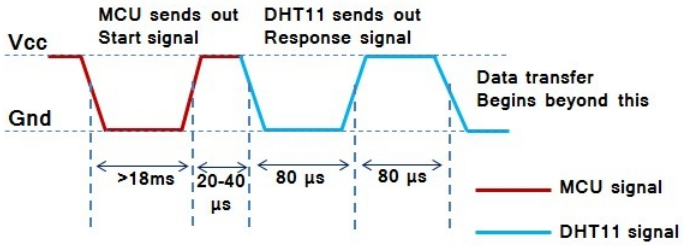
About DHT11 sensor

The DHT11 sensor comes in a single row 4-pin package and operates from 3.5 to 5.5V power supply. It can measure temperature from 0-50 °C with an accuracy of ±2°C and relative humidity ranging from 20-95% with an accuracy of ±5%. The sensor provides fully calibrated digital outputs for the two measurements. It has got its own proprietary 1-wire protocol, and therefore, the communication between the sensor and a microcontroller is not possible through a direct interface with any of its peripherals. The protocol must be implemented in the firmware of the MCU with precise timing required by the sensor.



DHT11 sensor comes in a single row 4-pin package

The following timing diagrams describe the data transfer protocol between a MCU and the DHT11 sensor. The MCU initiates data transmission by issuing a "***Start***" signal. The MCU pin must be configured as output for this purpose. The MCU first pulls the data line low for at least 18 ms and then pulls it high for next 20-40 µs before it releases it. Next, the sensor responds to the MCU "***Start***" signal by pulling the line low for 80 µs followed by a logic high signal that also lasts for 80 µs. Remember that the MCU pin must be configured to input after finishing the "***Start***" signal. Once detecting the response signal from the sensor, the MCU should be ready to receive data from the sensor. The sensor then sends 40 bits (5 bytes) of data continuously in the data line. Note that while transmitting bytes, the sensor sends the most significant bit first.



"Start" and "Response" signals

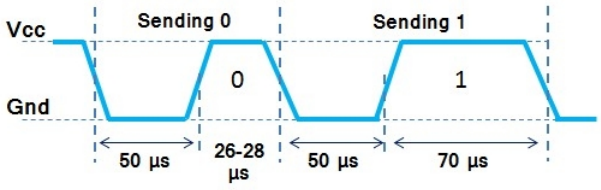
The 40-bit data from the sensor has the following structure.

Data (40-bit) = Integer Byte of RH + Decimal Byte of RH + Integer Byte of Temp. + Decimal Byte of Temp. + Checksum Byte

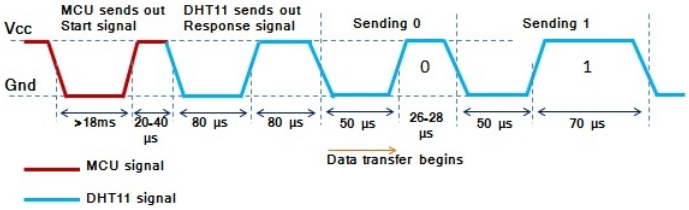
For DHT11 sensor, the decimal bytes of temperature and humidity measurements are always zero. Therefore, the first and third bytes of received data actually give the numeric values of the measured relative humidity (%) and temperature (°C). The last byte is the checksum byte which is used to make sure that the data transfer has happened without any error. If all the five bytes are transferred successfully then the checksum byte must be equal to the last 8 bits of the sum of the first four bytes, i.e.,

***Checksum = Last 8 bits of** (Integer Byte of RH + Decimal Byte of RH + Integer Byte of Temp. + Decimal Byte of Temp.)*

Now lets talk about the most important thing, which is signalling for transmitting "0" and "1". In order to send a bit of data, the sensor first pulls the line low for 50 µs. Then it raises the line to high for 26-28 µs if it has to send "0", or for 70 µs if the bit to be transmitted is "1". So it is the width of the positive pulse that carries information about 1 and 0.



Timing difference for transmitting "1s" and "0s"

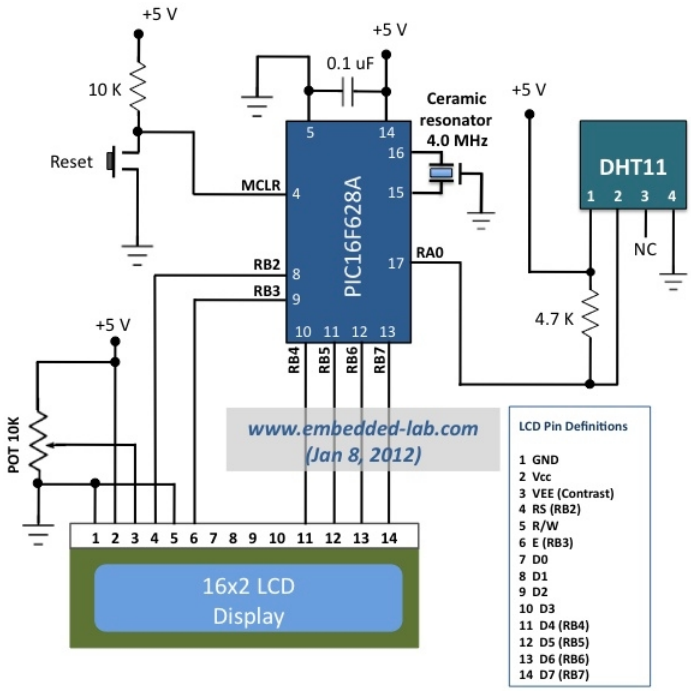


Start, Response and Data signals in sequence

At the end of the last transmitted bit, the sensor pulls the data line low for 50 µs and then releases it. The DHT11 sensor requires an external pull-up resistor to be connected between its Vcc and the data line so that under idle condition, the data line is always pulled high. After finishing the data transmission and releasing the data line, the DHT11 sensor goes to the low-power consumption mode until a new "***Start***" signal arrives from the MCU.

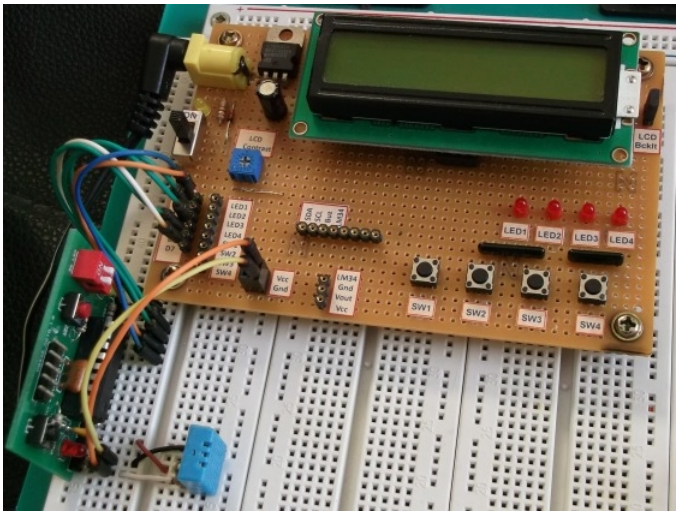
Circuit diagram

Here is the circuit diagram showing the DHT11 sensor and a HD44780-based character LCD interfaced to the PIC16F628A microcontroller. The microcontroller runs at 4.0 MHz clock using an external resonator connected between OSC1 (16) and OSC2 (15) pins. The use of 4.0 MHz clock makes the timing calculation easier as 1 machine cycle becomes 1 µs. The timing information will be used to calculate the width of the received data pulse from the sensor so that we could identify if it is carrying a 1 or 0.

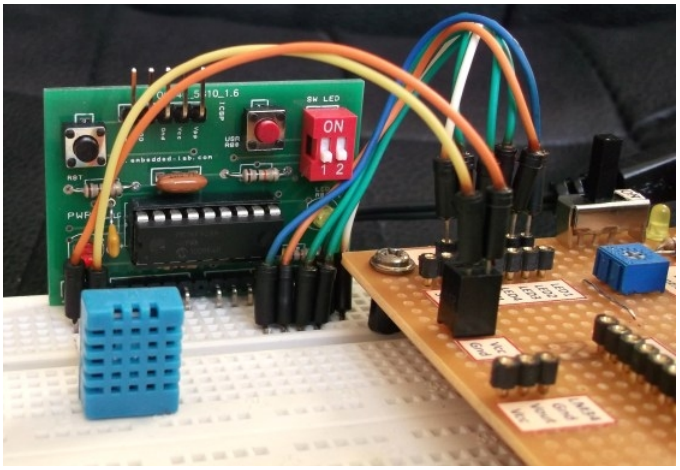


Circuit connections for PIC16F628A and DHT11 sensor

The following pictures show the circuit setup on a breadboard. Don't get confused with the four LEDs and tact switches shown on the perforated board. They have nothing to do with this project. They are there because I am using my **DIY Experimenter's I/O board** for the LCD part of this project. Similarly, I am using my **18-pin PIC16F board** for easy prototyping with the PIC16F628A microcontroller.



Complete setup of the circuit



PIC16F628A module and the DHT11 sensor are plugged into the breadboard

Software

Writing a software for DHT11 sensor is little more challenging than the hardware part because of the timing conditions for 1s and 0s. I have written sub-routines in mikroC Pro for

PIC for initializing the DHT11 sensor and reading the 40-bit of data in sequence. I have used Timer2 module to keep track of the width of the received data pulse, which is required to identify if the received bit is 1 or 0. When a low-to-high pulse is detected at the beginning of any data bit, TMR2 is cleared and turned ON. Since the clock frequency used here is 4.0 MHz, the TMR2 increments by 1 in every 1 µs. The TMR2 is stopped whenever the data pulse is low again. The value of the TMR2 register gives you the the width of the data pulse in µs. I am using 40 µs as the threshold for identifying 0 and 1. If the TMR2 is greater than 40, it means the received bit is 1, else it is 0. Here is the complete source code written in mikroC Pro for PIC. It can be easily adapted to any other platform, but remember that if you are using a different clock frequency you should have to modify the timer operation accordingly.

```
// LCD module connections
sbit LCD_RS at RB2_bit;
sbit LCD_EN at RB3_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;
sbit LCD_RS_Direction at TRISB2_bit;
sbit LCD_EN_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;

sbit Data at RA0_bit;
sbit DataDir at TRISA0_bit;
char message1[] = "Temp = 00.0 C";
char message2[] = "RH   = 00.0 %";
unsigned short TOUT = 0, CheckSum, i;
unsigned short T_Byte1, T_Byte2, RH_Byte1, RH_Byte2;

void StartSignal(){
  DataDir = 0;     // Data port is output
  Data    = 0;
  Delay_ms(25);    // Low for at least 18us
  Data    = 1;
  Delay_us(30);    // High for 20-40 us
  DataDir = 1;     // Data port is input

}

unsigned short CheckResponse(){
  TOUT = 0;
  TMR2 = 0;
  T2CON.TMR2ON = 1;      // Start TMR2 while waiting for sensor response
  while(!Data && !TOUT); // If there's no response within 256us, the Timer2 overflows
  if (TOUT) return 0;    // and exit
  else {
   TMR2 = 0;
   while(Data && !TOUT);
   if (TOUT) return 0;
   else {
    T2CON.TMR2ON = 0;
    return 1;
   }
  }
}

unsigned short ReadByte(){
  unsigned short num = 0, t;
  DataDir = 1;
  for (i=0; i<8; i++){
   while(!Data);
   TMR2 = 0;
   T2CON.TMR2ON = 1;  // Start TMR2 from 0 when a low to high data pulse
   while(Data);       // is detected, and wait until it falls low again.
   T2CON.TMR2ON = 0;  // Stop the TMR2 when the data pulse falls low.
   if(TMR2 > 40) num |= 1<<(7-i);  // If time > 40us, Data is 1
  }
  return num;
}

void interrupt(){
  if(PIR1.TMR2IF){
   TOUT = 1;
   T2CON.TMR2ON = 0; // stop timer
   PIR1.TMR2IF  = 0; // Clear TMR0 interrupt flag
  }
}

void main() {
  unsigned short check;
  TRISB = 0b00000000;
  PORTB = 0;
  TRISA = 0b00100001;
  CMCON = 7;
  INTCON.GIE = 1;    //Enable global interrupt
  INTCON.PEIE = 1;   //Enable peripheral interrupt
  // Configure Timer2 module
  PIE1.TMR2IE = 1;  // Enable Timer2 interrupt
  T2CON = 0;        // Prescaler 1:1, and Timer2 is off initially
  PIR1.TMR2IF =0;   // Clear TMR INT Flag bit
  TMR2 = 0;
  Lcd_Init();
  Lcd_Cmd(_Lcd_Clear);
  Lcd_Cmd(_LCD_CURSOR_OFF);

  do {
    Delay_ms(1000);
    StartSignal();
    check = CheckResponse();
    if (!check) {
     Lcd_Cmd(_Lcd_Clear);
     Lcd_Out(1, 1, "No response");
     Lcd_Out(2, 1, "from the sensor");
    }
    else{

     RH_Byte1 = ReadByte();
     RH_Byte2 = ReadByte();
```

```
     T_Byte1 = ReadByte();
     T_Byte2 = ReadByte();
     CheckSum = ReadByte();
     // Check for error in Data reception
     if (CheckSum == ((RH_Byte1 + RH_Byte2 + T_Byte1 + T_Byte2) & 0xFF))
     {
      message1[7]  = T_Byte1/10 + 48;
      message1[8]  = T_Byte1%10 + 48;
      message1[10] = T_Byte2/10 + 48;
      message2[7]  = RH_Byte1/10 + 48;
      message2[8]  = RH_Byte1%10 + 48;
      message2[10] = RH_Byte2/10 + 48;
      message1[11] = 223;     // Degree symbol
      Lcd_Cmd(_Lcd_Clear);
      Lcd_Out(1, 1, message1);
      Lcd_Out(2, 1, message2);
      }

      else{
       Lcd_Cmd(_Lcd_Clear);
       Lcd_Out(1, 1, "Checksum Error!");
       Lcd_Out(2, 1, "Trying Again ...");
      }
    }

  }while(1);
}
```

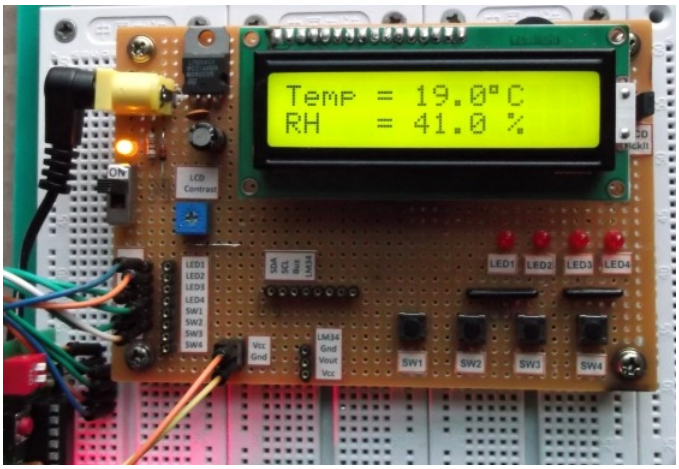
You can also simplify the *ReadByte* subroutine without using the Timer2 module. The following version of *ReadByte* subroutine works equally well. Once the data pin is detected high, wait for 40 µs and check the data line again. If it is still high, it is 1, else 0.

```
unsigned short ReadByte(){
  unsigned short num = 0, t;
  DataDir = 1;
  for (i=0; i<8; i++){
   while(!Data);
   Delay_us(40);
   if(Data) num |= 1<<(7-i);
   while(Data);
   }
  return num;
}
```

Download complete source code and HEX files

Output

The accuracy of DHT11 is not as good as Sensirion's SHT1X/7X series sensors, but it provides an easy and cheap solution to hobbyists for measuring relative humidity and temperature in parallel using a single device, which is sometime required in certain applications such as calculating the **dew point**.



Temperature and relative humidity measured by DHT11 sensor

Related Posts

- **Lab 19: Play musical notes**
  We have discussed in the past experiments how to use a PIC microcontroller to do a variety of things from flashing an LE...
- **Lab 21: Servo motor control**
  A servo motor is a special geared DC motor equipped with an electronic circuit for controlling the direction of rotation...
- **Making a digital capacitance meter using microcontroller**
  Capacitors are one of the most common passive electrical components that are extensively used in all kinds of electronic...

Filed under: Embedded Lab Projects, PIC Projects, Tips and Tricks

- RSS feed for comments on this post
- TrackBack URI

27 Responses to this post

1. Jack on January 11th, 2012 11:50 am

   i was looking for this project.. Thanks for posting it with a woderful description

2. DHT11 humidity and temperature sensor package - Hack a Day on January 11th, 2012 7:33 pm

   [...] have a price tag that is well above what most hobbiests are willing to spend. He decided to take an in-depth look at the DHT11 sensor; which you can get your hands on for under $3 if you know where to [...]

3. DHT11 humidity and temperature sensor package » Geko Geek on January 11th, 2012 8:00 pm

   [...] have a price tag that is well above what most hobbiests are willing to spend. He decided to take an in-depth look at the DHT11 sensor; which you can get your hands on for under $3 if you know where to [...]

4. DHT11 humidity and temperature sensor package | TechnoFiesta on January 11th, 2012 11:07 pm

   [...] have a price tag that is well above what most hobbiests are willing to spend. He decided to take an in-depth look at the DHT11 sensor; which you can get your hands on for under $ 3 if you know where to [...]

5. José Xavier on January 12th, 2012 7:25 am

   Great info 😃

   Where you bought it?

6. DHT11 humidity and temperature sensor package « Uncategorized « Cool Internet Projects on January 12th, 2012 8:51 am

   [...] have a price tag that is well above what most hobbiests are willing to spend. He decided to take an in-depth look at the DHT11 sensor; which you can get your hands on for under $3 if you know where to [...]

7. Keyster747 on January 12th, 2012 2:29 pm

   it looks like they sell them at http://www.goodluckbuy.com
   $5.81 for 2 of them (free shipping)
   SKU is 74681

   warning: it will probably take you a few weeks to get them depending on where you are in the world.

8. Shafiq on January 12th, 2012 11:10 pm

   Nicely done. Can you please what software/program did you use to draw the diagram.

   Thanks

9. DHT11 humidity and temperature sensor package | CisforComputers on January 13th, 2012 12:04 am

   [...] have a price tag that is well above what most hobbiests are willing to spend. He decided to take an in-depth look at the DHT11 sensor; which you can get your hands on for under $3 if you know where to [...]

10. R-B on January 13th, 2012 1:08 am

    MS Powerpoint!

11. Kevin Smith on February 10th, 2012 1:34 pm

    Hello

    I have built this circuit and when I power it up, the LCD display shows
    “No response from sensor”. What could be the problem.

    Thanks

12. Kevin Smith on February 10th, 2012 1:43 pm

    Problem solved, faulty resonator.

13. vinalice on February 28th, 2012 9:14 pm

    Hi,you use what software to debug n build? isit mplab or microC pro? Does it work in MPLAB?

14. Ayie on March 3rd, 2012 11:10 pm

    Which compiler are u using? This coding compatible for mplab ide c18 or not? and which programmer are u using? i want to program into pic18f4520 using pickit2.

15. R-B on March 4th, 2012 3:20 am

    I used mikroC Pro for PIC compiler.

16. R-B on March 4th, 2012 3:23 am

    MikroC

17. Jasper on March 10th, 2012 9:15 am

    Nice description for using a DHT11 and PIC microcontroller. What do you use with the measurements of temperature, relative humidity and dew point? From the measurement can you read if it’s rainy/dry/cloudy/sunny? Doe this mean that the preferred location to place the sensor is outdoors?

18. Ayie on March 13th, 2012 9:14 am

    What type of programmer are u using? I cnt program into the pic using pickit2.

19. vince on March 13th, 2012 12:27 pm

i followed ur coding and everything correctly,even with the 4Mhz oscillator,and yet i still get black square are 2nd line,any idea?

20. vince on March 13th, 2012 11:06 pm

    why delete my previous comment?

21. R-B on March 25th, 2012 12:39 pm

    PICKit2 should be able to program PIC16F628A.

22. Mr Sosgez on March 26th, 2012 7:32 am

    Great article, very well presented – thanks! I translated the code to compile under CCS C compiler on PIC16C887 on an OLIMEX board and it works fine.

23. Paul on May 9th, 2012 9:54 am

    Can someone explain this section of code to me?

    message1[7] = T_Byte1/10 + 48;

    I understand the T_Byte is the temp in C. Why the divide by 10 adn then add 48?

24. R-B on May 9th, 2012 10:01 am

    Paul,
    You divide T_Byte1 by 10 to get the tens digit, and T_Byte%10 gives the units digit of the temperature. You add 48 to convert it into equivalent ASCII character so that you could display it on character LCD. The ASCII code for 0 is 48, for 1 is 49, for 2 is 50, and so on.

25. Paul on May 9th, 2012 12:32 pm

    Thanks RB. Now I understand

26. Donovan on May 16th, 2012 3:20 am

    hi, im using mplab ccs c compiler. i followed the code correctly but they kept saying “Error 128 “DHT11.c” Line 10(1,1): A #DEVICE required before this line.” i understand its a header file tats missing. how do i solve this problem?

27. MARTIN on June 3rd, 2012 11:11 am

    hi,do u hv any version can be run in Mplab C18, 18F4520. Thanks

Leave a comment

Name (required)

Email Address (required)

Website

XHTML: You can use these tags: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <strike>

Post your comment

.....

Subscribe through email

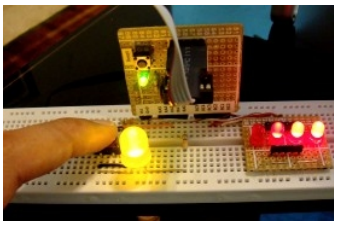
Sign Up

Featured Project

Multi-purpose dual power supply (5.0V and 3.3V) regulator board

All embedded systems require electric power to operate. Most of the electronic components inside them, including the processors, can operate at a wide range of supply voltage. For example, the operating voltage range for the PIC16F1847 microcontroller is 2 to 5.

Experimenting with PIC



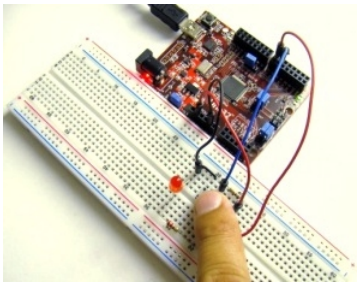
These tutorials are aimed to provide you an introductory level theory and practice of embedded system design through the application of PIC microcontrollers.

Browse our PIC Tutorials

Most Popular Posts

- Heart rate measurement from fingertip
- Programmable digital timer switch using a PIC Microcontroller
- PIC-based Digital Voltmeter (DVM)
- A very simple IR remote control switch for an electrical appliance
- A Digital temperature meter using an LM35 temperature sensor

chipKIT Programming and Interfacing



Browse our chipKIT Tutorials

Categories

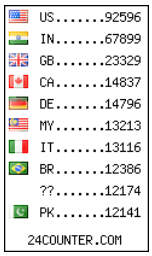
- 555 Timer (6)
- Analog (1)
- Arduino (10)
- AVR Projects (25)
- AVR Tutorials (5)
- chipKIT (9)
- dsPIC (1)
- Embedded Lab Projects (25)
- Embedded Labs (25)
- Embedded Lessons (33)
- MCU develeopment tools (1)
- Microcontroller Programmers (6)
- MSP430 Launchpad (1)
- PIC Projects (51)
- PIC Tutorials (42)
- PIC18F (10)
- Power Supply (6)
- Processing (2)
- Product Review (11)
- Products (2)
- Robotics (4)

- Tech News (28)
- Tips and Tricks (37)
- Uncategorized (1)

Blogroll

- Dangerous Prototypes
- EEWeb
- Electronics-Lab
- Hack A Day

Visitors



flag counter