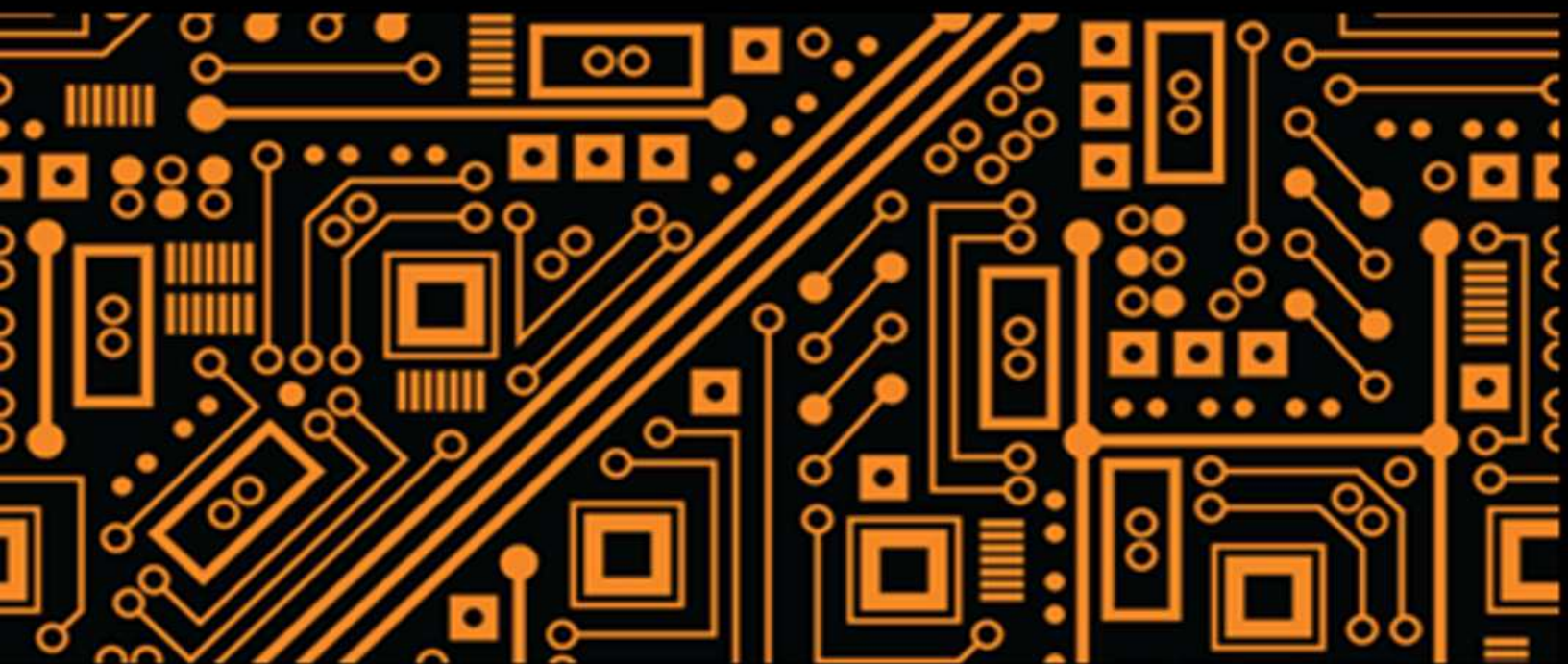




# Arduino Projects

Vol-I

Manoj R.Thakur



**With Proteus Simulation Files, Don't just read it Try it.....**

Circuits4you.com

# Arduino Projects Vol-I

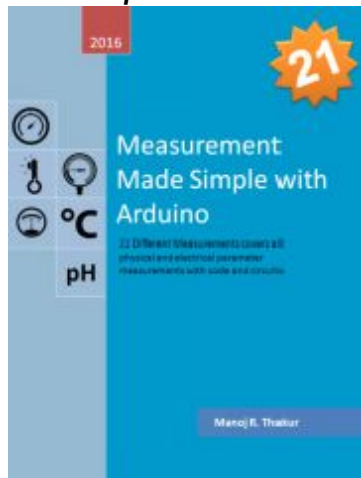
## **With Practical Exercise**

Manoj R. Thakur  
6/22/2016



## Preface

I completed Master Degree in Electronics year 2008, I worked 5 years in MIT as Assistant Professor in Electronics Department and found that many students are struggling for knowing basic things and they are many times get bombarded with lot of information that makes them confuse, so I decided to write a book that focus more on practical approach and keep it like read less and experiment more. My first e-book [Measurement Made Simple with Arduino](#) is the result of my experiences that got great success. Many found that its basis for all measurement needs.



This book idea is to give concept behind making use of measurement and control knowledge to apply in day to day practical application. Here we have provided all the necessary data to make Arduino projects with keeping in mind read less do more. **The best thing about this book is it's not just read or make, you can actually simulate before you make it. Most of the projects are provided with "Proteus" simulation files that are available on my website [circuits4you.com](http://circuits4you.com).**

The book layout is kept very simple like experiment notes

1. Discuss the idea of project
2. Sensor discussion

3. *Circuit Design with simulation file link*
4. *Programming with hex file link*
5. *Conclusions*

*PCB Layout and other resources are given in last chapter. Do not forget to see it.*

*This book is intended to focus on making various day to day life problem solving projects using Arduino. It is kept short and specific to the title of the book. We are using only Arduino Uno for the examples other boards can be used with same code in most cases. This is first book of Arduino Project series.*

# Contents

## 1. GETTING STARTED WITH ARDUINO

1.1 ARDUINO INTRODUCTION

1.2 ARDUINO IDE BASICS

1.3 ARDUINO PROGRAMMING

1.4 ARDUINO PIN-OUTS

## 2. ARDUINO BASED DIGITAL CODE LOCK

2.1 INTRODUCTION

2.2 DIGITAL CODE LOCK CIRCUIT

2.3 DIGITAL CODE LOCK ARDUINO CODE

## 3. ARDUINO TEMPERATURE CONTROLLER

3.1 INTRODUCTION

3.2 TEMPERATURE CONTROLLER CIRCUIT

3.3 TEMPERATURE CONTROLLER ARDUINO CODE

## 4. ARDUINO OBJECT COUNTER

4.1 INTRODUCTION

4.2 OBJECT COUNTER CIRCUIT

4.3 OBJECT COUNTER ARDUINO CODE

## 5. ARDUINO DC DIGITAL VOLTMETER

5.1 INTRODUCTION

5.2 DIGITAL VOLTMETER CIRCUIT

5.3 DIGITAL VOLTMETER ARDUINO CODE

## 6. ARDUINO WATER LEVEL CONTROLLER

- [6.1 INTRODUCTION](#)
- [6.2 WATER LEVEL CONTROLLER CIRCUIT](#)
- [6.3 WATER LEVEL CONTROLLER ARDUINO CODE](#)

## [7. AUTOMATIC LIGHT CONTROLLER](#)

- [7.1 INTRODUCTION](#)
- [7.2 AUTOMATIC LIGHT CONTROLLER CIRCUIT](#)
- [7.3 AUTOMATIC LIGHT CONTROLLER ARDUINO CODE](#)

## [8. SOLAR POWER MONITOR](#)

- [8.1 INTRODUCTION](#)
- [8.2 SOLAR POWER MONITOR CIRCUIT](#)
- [8.3 SOLAR POWER MONITOR CODE](#)

## [9. ULTRASONIC DISTANCE METER](#)

- [9.1 INTRODUCTION](#)
- [9.2 ULTRASONIC DISTANCE METER CIRCUIT](#)
- [9.3 ULTRASONIC DISTANCE METER ARDUINO CODE](#)

## [10. DIGITAL TIMER](#)

- [10.1 INTRODUCTION](#)
- [10.2 DIGITAL TIMER CIRCUIT](#)
- [10.3 DIGITAL TIMER ARDUINO CODE](#)

## [11. AUTOMATIC IRRIGATION SYSTEM](#)

- [11. 1 INTRODUCTION](#)
- [11.2 AUTOMATIC IRRIGATION SYSTEM CIRCUIT](#)
- [11.3 AUTOMATIC IRRIGATION SYSTEM ARDUINO CODE](#)

## [12. MOOD LAMP](#)

[12.1 INTRODUCTION](#)  
[12.2 MOOD LAMP CIRCUIT](#)  
[12.3 MOOD LAMP ARDUINO CODE](#)

## **13. BLUETOOTH BASED HOME AUTOMATION**

[13.1 INTRODUCTION](#)  
[13.2 BLUETOOTH BASED HOME AUTOMATION CIRCUIT](#)  
[13.3 BLUETOOTH BASED HOME AUTOMATION ARDUINO CODE](#)

## **14. TRAFFIC LIGHT CONTROLLER**

[14.1 INTRODUCTION](#)  
[14.2 TRAFFIC LIGHT CONTROLLER CIRCUIT](#)  
[14.3 TRAFFIC LIGHT CONTROLLER CODE](#)

## **15. RPM METER**

[15.1 INTRODUCTION](#)  
[15.2 RPM METER CIRCUIT](#)  
[15.3 RPM METER ARDUINO CODE](#)

## **15. REFERENCES**                      **67**

[15.1 SIMULATION AND HEX FILES](#)                      **67**



## Disclaimer

All do-it-yourself activities involve risk, and your safety is your own responsibility, including proper use of equipments and safety gear, and determining whether you have adequate skill and experience, Some of the resources used for these projects are dangerous unless used properly and with adequate precautions, including safety gear. Some illustrative photos do not depict safety precautions or equipment, in order to show the project steps more clearly. The projects are not intended for use by children.

Use of the instructions and suggestions is at your own risk. Circuits4you.com and Author of this book (Manoj R. Thakur) disclaims all responsibility for any damage, injury, or expense. It is your responsibility to make sure that your activities comply with all applicable laws.

---

# 1. Getting Started with Arduino

## 1.1 Arduino Introduction

Arduino is an open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control the physical world.

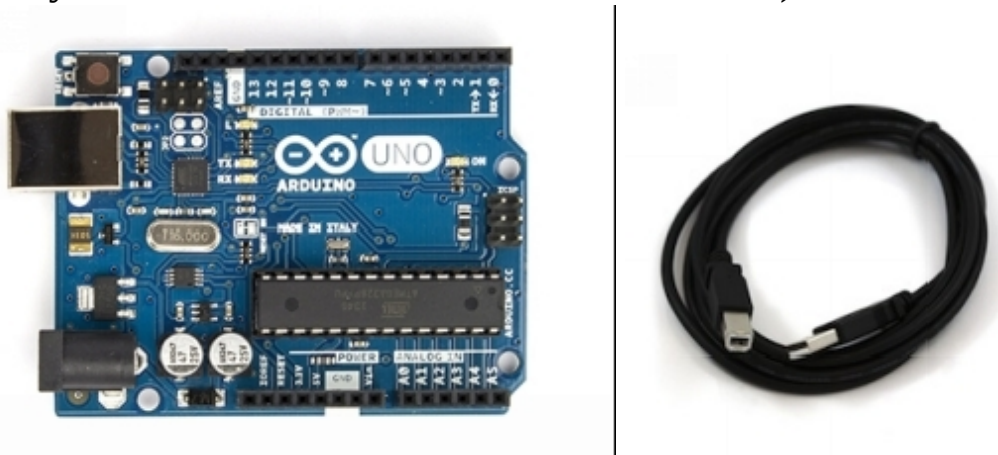
## 1.2 Arduino IDE basics

Let's see how to connect your Arduino board to the computer and upload your first sketch.

### Get an Arduino board and USB cable

In this book, we assume you're using an Arduino Uno. For other Arduino boards most of the things are same.

You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example. (For the Arduino Nano, you'll need an A to Mini-B cable instead.)



**Figure 1.1: Arduino Uno and USB Cable**

### Download the Arduino Software (IDE)

Get the latest version from website [www.arduino.cc](http://www.arduino.cc) download page. When the download finishes, unzip the downloaded file.

### Connect the board

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either the USB connection to the computer or an external power supply. If you're using an Arduino Diecimila, you'll need to make sure that the board is configured to

draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it's on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should go on. For new board (not programmed yet) Yellow LED (Near Pin 13) will blink.

### **Install the drivers**

Installing drivers for the Arduino Uno or Arduino Mega 2560 with Windows 7, Vista, or XP:

Plug in your board and wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts, Driver files are located in Folder "Arduino\Drivers" select appropriate file and install the drivers.

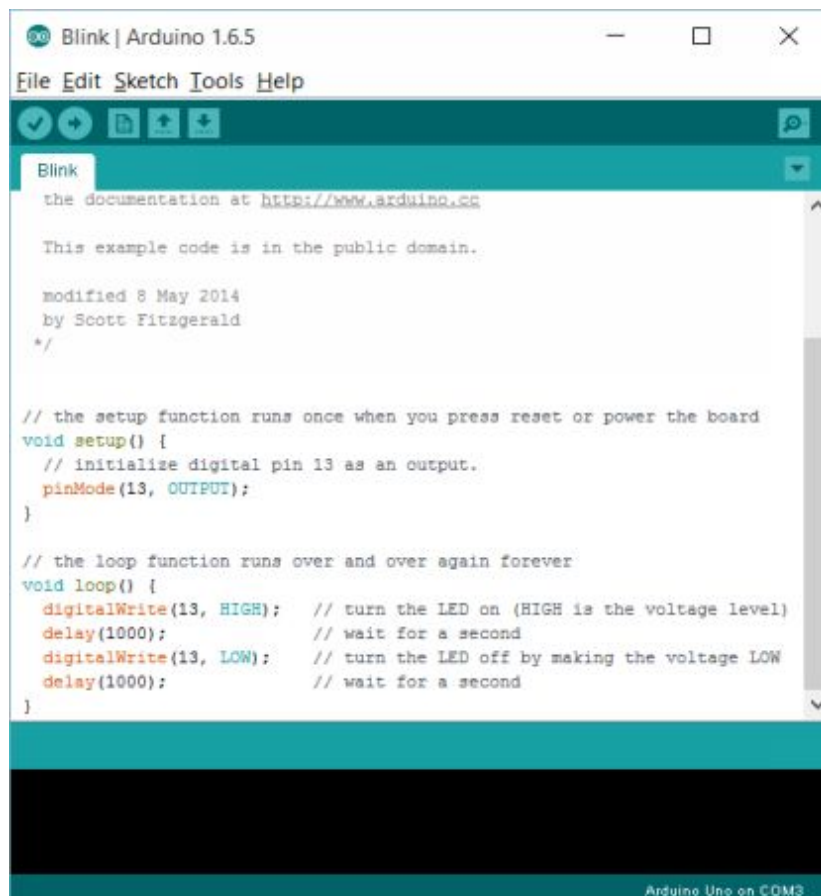
After successful driver installation you will find new com port. You can check correct com port number from device manager.

## 1.3 Arduino Programming

Double-click the Arduino application (arduino.exe) you have previously downloaded. (Note: if the Arduino Software loads in the wrong language, you can change it in the preferences dialog.)

### Open the blink example

Open the LED blink example sketch: *File > Examples > 01.Basics > Blink.*

A screenshot of the Arduino IDE window titled "Blink | Arduino 1.6.5". The window shows the "Blink" sketch loaded. The code is as follows:

```
the documentation at http://www.arduino.cc

This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

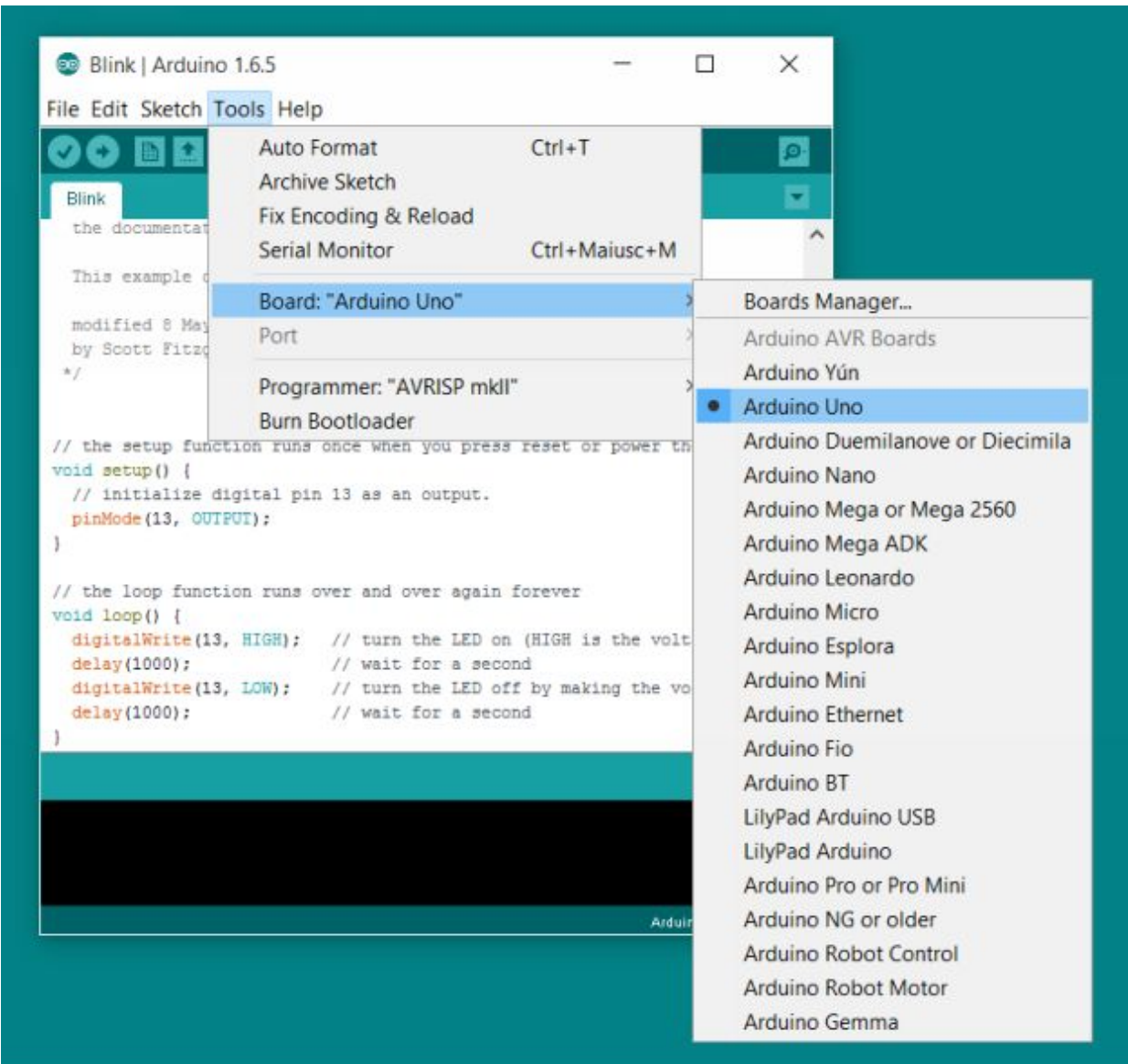
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

The status bar at the bottom right indicates "Arduino Uno on COM3".

**Figure 1.2: Arduino Blink Example**

### Select your board

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino.



**Figure 1.3: Arduino Board Selection**

### Select your serial port

Select the serial device of the Arduino board from the Tools >> Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port. Or look into device manager.

## Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX led's on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.



**Figure 1.4: Arduino Upload Button Location**

A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino up-and-running.



# 1.4 Arduino Pin-outs

## Arduino Nano

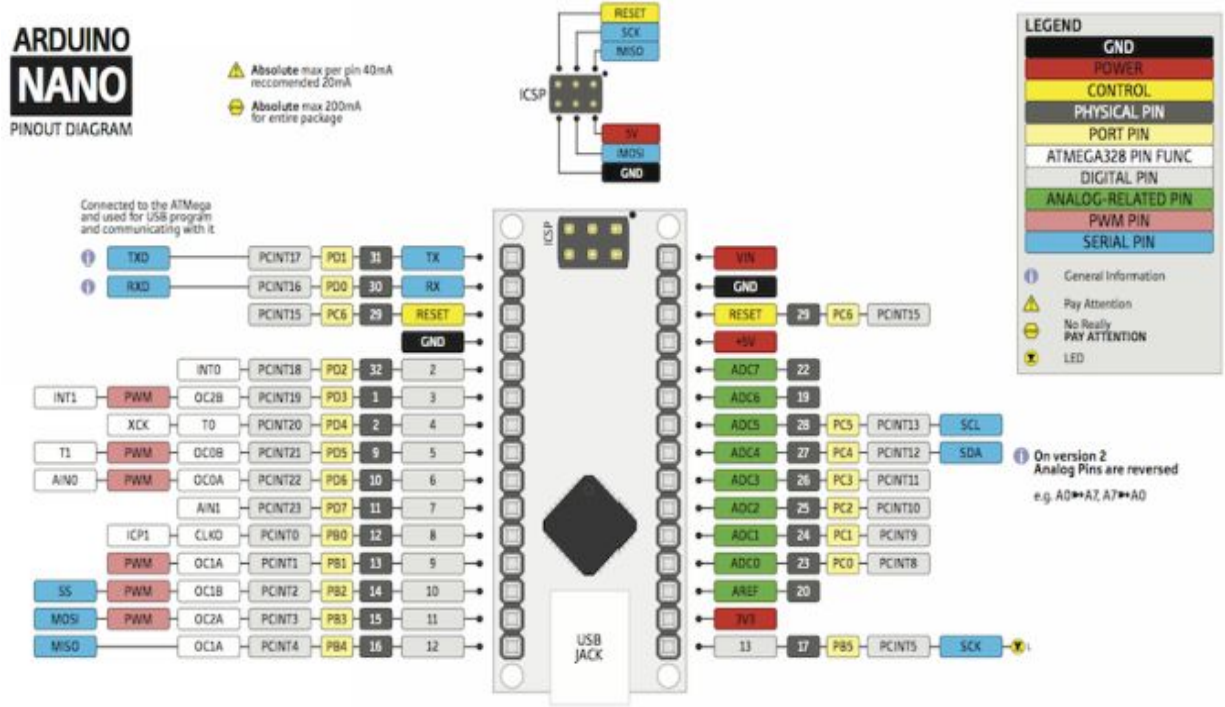


Figure 1.5: Arduino Nano Pinout diagram

## Arduino Uno



# Arduino Pro Mini

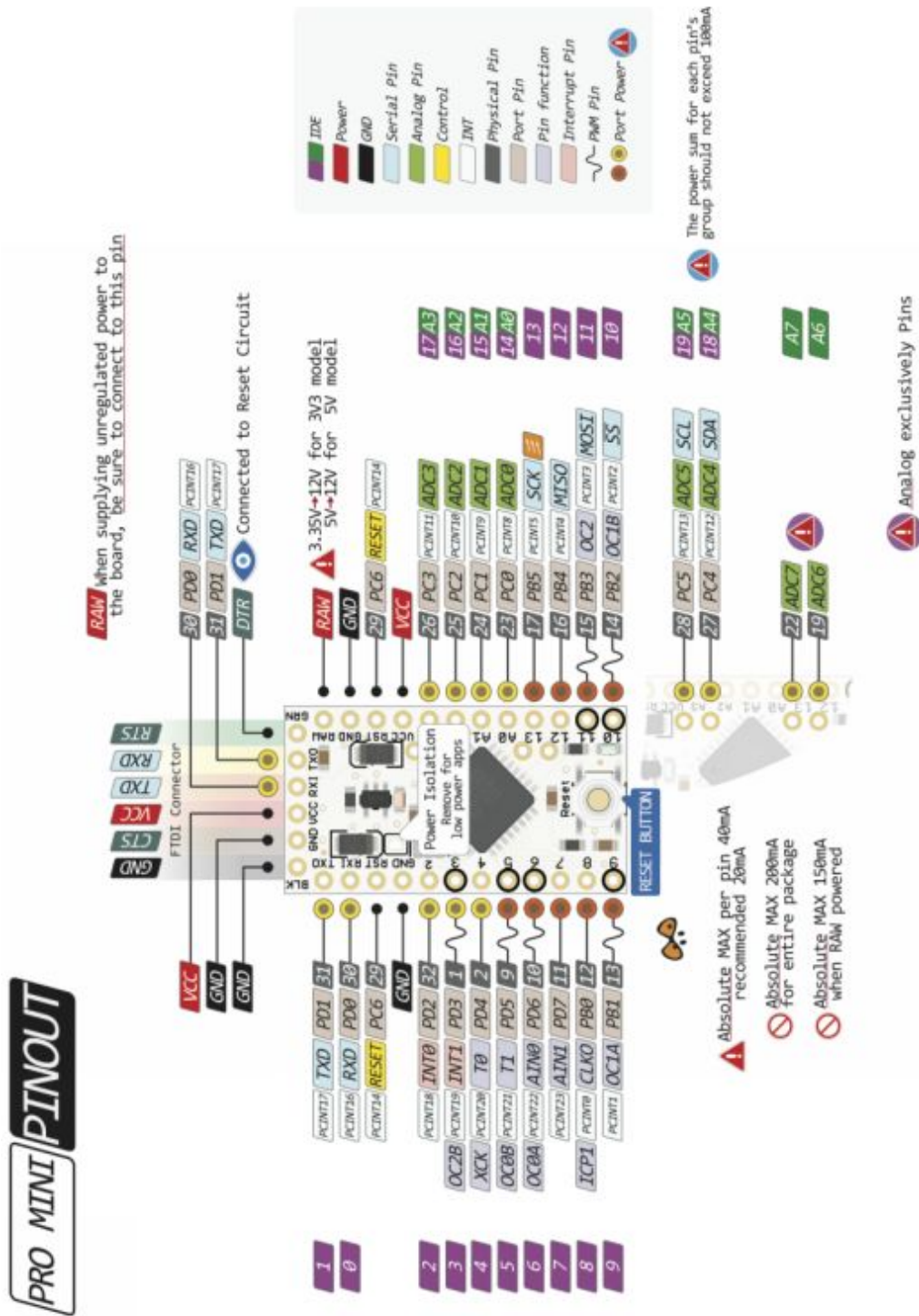


Figure 1.7: Arduino Pro Mini Pinout diagram

## 2. Arduino based digital code lock

---

## 2.1 Introduction

Digital code locks are most common on security systems. An electronic lock or digital lock is a device which has an electronic control assembly attached to it. They are provided with an access control system. This system allows the user to unlock the device with a password. The password is entered by making use of a keypad. The user can also set his password to ensure better protection.

In this project major components include a keypad, LCD and the controller Arduino. This article describes the making of an electronic code lock using arduino.

## What you will learn?

1. How to connect keypad and LCD to arduino?
2. How to take keypad input?
3. Comparing keypad input?
4. Limiting the input?
5. How to make complete digital code lock application?

## Components Required

1. Arduino Uno
2. 16x2 LCD Display
3. 4x4 keypad
4. Relay
5. 1K Resistors Qty. 3
6. BC548
7. LEDs

## 2.2 Digital Code Lock Circuit

Code lock circuit is constructed around Arduino Uno, using LCD and keypad. LCD and keypad forms the user interface for entering the password and displaying related messages such as “Invalid password”, “Door open”, etc. Two LEDs are provided to indicate the status of door whether it is locked or open. To operate latch/lock we are using Relay which can be connected to the electronic actuator or solenoid.

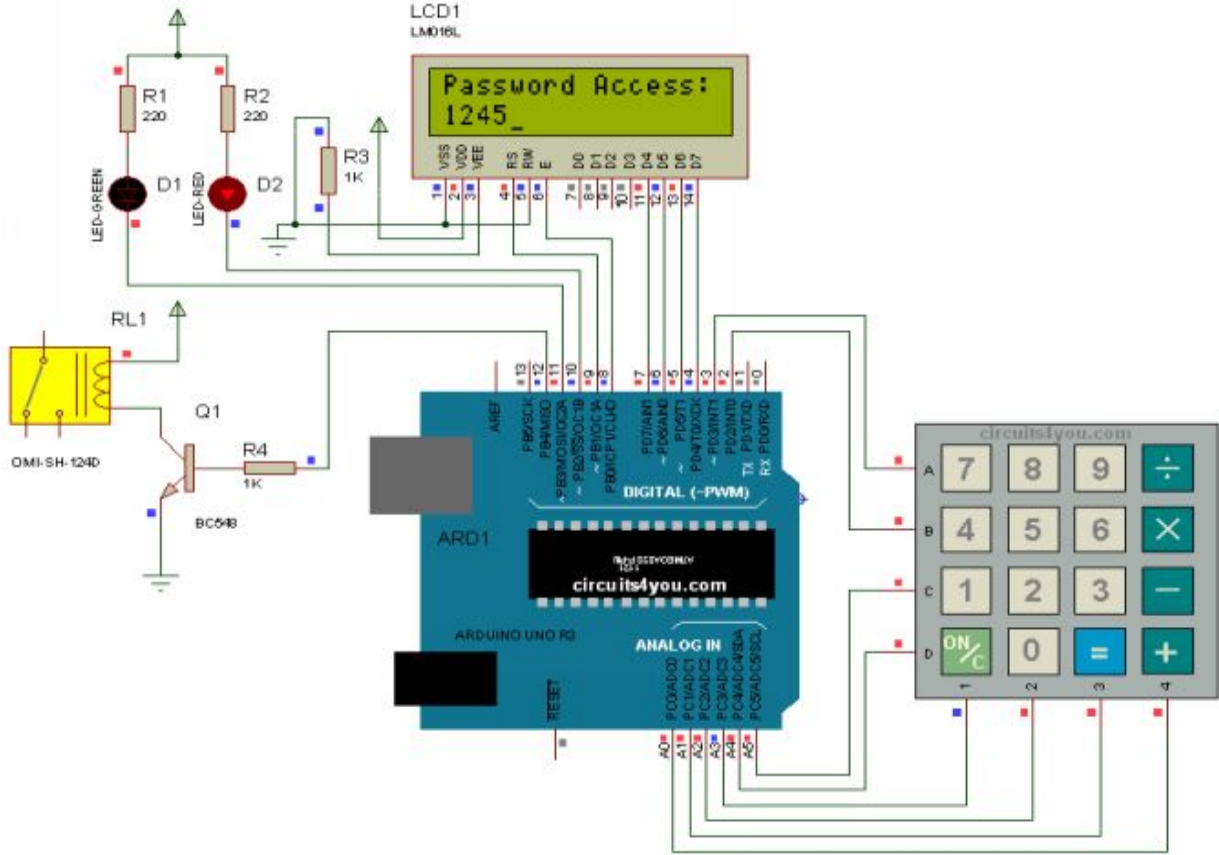


Figure 2.1: Digital Code Lock Circuit



## 2.3 Digital Code Lock Arduino Code

Program is constructed using two libraries “LiquidCrystal” and “Keypad”. Program have different modules, Setup, Loop, Lock. In setup we initialize all the IO connections and LCD, Keypad. In main loop we are taking pressed keys in array “code[]”, Once the four digits are entered we stop accepting keys. We are using numeric keys and ‘C’ , “=” key. ‘C’ key is used to lock or clear the display incase wrong password is entered. We can hide the entered password by putting Star character ‘\*’.

After entering password ‘=’ key acts as ok. If password is correct door is kept unlocked for few seconds. If it is incorrect message will be displayed.

```
/*
  circuits4you.com
  Digital Code Lock Demo
*/

#include <Keypad.h>
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd (9, 8, 7, 6, 5, 4);

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the cymbols on the buttons of the keypads
char hexaKeys [ ROWS ][ COLS ] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'C','0','=','+'}
};
```

```

byte rowPins [ ROWS ] = {3, 2, 19, 18}; //connect to the row pinouts of
the keypad
byte colPins [ COLS ] = {17, 16, 15, 14}; //connect to the column pinouts
of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad ( makeKeymap ( hexaKeys ), rowPins ,
colPins , ROWS , COLS );

const int LED_RED =10; //Red LED
const int LED_GREEN =11; //Green LED
const int RELAY =12; //Lock Relay or motor

char keycount =0;
char code [4]; //Hold pressed keys
//=====
=====
//          SETUP
//=====
=====
void setup (){
  pinMode ( LED_RED , OUTPUT );
  pinMode ( LED_GREEN , OUTPUT );
  pinMode ( RELAY , OUTPUT );

  // set up the LCD's number of columns and rows:
  lcd . begin (16, 2);
  // Print a message to the LCD.
  lcd . print ("Password Access:");
  lcd . setCursor (0,1); //Move cursor to second Line
// Turn on the cursor
  lcd . cursor ();
  digitalWrite ( LED_GREEN , HIGH ); //Green LED Off
  digitalWrite ( LED_RED , LOW ); //Red LED On
  digitalWrite ( RELAY , LOW ); //Turn off Relay (Locked)
}

```

```

//=====
=====
//          LOOP
//=====
=====
void loop (){
  char customKey = customKeypad . getKey ();

  if ( customKey && ( keycount <4) && ( customKey !=' ') &&
( customKey !='C')){
    //lcd.print(customKey); //To display entered keys
    lcd . print (*); //Do not display entered keys
    code [ keycount ]= customKey ;
    keycount ++;
  }

  if( customKey == 'C') //Cancel/Lock Key is pressed clear display and
lock
  {
    Lock (); //Lock and clear display
  }

  if( customKey == '=') //Check Password and Unlock
  {
    if(( code [0]== '1') && ( code [1]== '2') && ( code [2]== '3') &&
( code [3]== '4')) //Match the password. Default password is "1234"
    {
      digitalWrite ( LED_GREEN , LOW ); //Green LED Off
      digitalWrite ( LED_RED , HIGH ); //Red LED On
      digitalWrite ( RELAY , HIGH ); //Turn on Relay (Unlocked)
      lcd . setCursor (0,1);
      lcd . print ("Door Open ");
      delay (4000); //Keep Door open for 4 Seconds
      Lock ();
    }
  }
  else
  {

```

```

    lcd . setCursor (0,1);
    lcd . print ("Invalid Password"); //Display Error Message
    delay (1500); //Message delay
    Lock ();
  }
}
}

//=====
//=====
//          LOCK and Update Display
//=====
//=====

void Lock ()
{
  lcd . setCursor (0,1);
  lcd . print ("Door Locked  ");
  delay (1500);
  lcd . setCursor (0,1);
  lcd . print ("          "); //Clear Password
  lcd . setCursor (0,1);
  keycount =0;
  digitalWrite ( LED_GREEN , HIGH ); //Green LED Off
  digitalWrite ( LED_RED , LOW ); //Red LED On
  digitalWrite ( RELAY , LOW ); //Turn off Relay (Locked)
}

```

## Conclusion

**This code demonstrates how to construct digital code lock and its application using arduino.**

We have used almost all the IO lines of arduino, now you know that analog lines have digital numbers from 14 to 19.

# 3. Arduino Temperature Controller

---

## 3.1 Introduction

Digital Temperature Controller using arduino, here we are using arduino as main controller, this temperature controller controls the temperature of any heating device with given set points. It displays state of the heating element either on or off and current temperature on LCD.

## **What you will learn?**

1. How to connect keys and LCD to arduino?
2. How to take key input?
3. How to Read Temperature sensor LM35?
4. Controlling device as per set point.

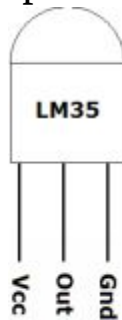


## Components Required

1. Arduino Uno
2. 16x2 LCD Display
3. Keys
4. Relay
5. 1K, 230 Ohm Resistors
6. BC548
7. LEDs
8. LM35 Temperature Sensor

## 3.2 Temperature Controller Circuit

Circuit is constructed using Arduino Uno and LM35 temperature sensor and other components. We are using 16x2 LCD to display current temperature and set points. LM35 gives analog output proportional to the temperature which is given to Arduino analog input A0. Which is then compared with set points if it is more than set point, It means the temperature is more so we turn off the heating element such as heater which is connected to relay output. If temperature is less we turn on the relay (heater). We are displaying status of heater on off on the LED and LCD also. Two tactile switches are used to set the temperature set point.



**Figure 3.1: LM35 Pin Diagram**



### 3.3 Temperature Controller Arduino Code

Program is constructed using one library “LiquidCrystal”. Program have different modules, Setup, Loop. In setup we initialize all the IO connections and LCD, Keypad. In main loop we are taking set point inputs and constantly measure current temperature and compare it with set points. If it is more than set point turn off heater, else turn on heater. You can add some hysteresis.

```
/*
  circuits4you.com
  Digital Temperature Controller
*/
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd (9, 8, 7, 6, 5, 4);

const int LED_RED =10; //Red LED
const int LED_GREEN =11; //Green LED
const int RELAY =12; //Lock Relay or motor

//Key connections with arduino
const int up_key =3;
const int down_key =2;

int SetPoint =30;
//=====
=====
//          SETUP
//=====
=====
void setup (){
  pinMode ( LED_RED , OUTPUT );
  pinMode ( LED_GREEN , OUTPUT );
  pinMode ( RELAY , OUTPUT );
  pinMode ( up_key , INPUT );
```

```

pinMode ( down_key , INPUT );

//Pull up for setpoint keys
digitalWrite ( up_key , HIGH );
digitalWrite ( down_key , HIGH );

// set up the LCD's number of columns and rows:
lcd . begin (16, 2);
// Print a message to the LCD.
lcd . print ("circuits4you.com");
lcd . setCursor (0,1); //Move cursor to second Line
lcd . print ("Temp. Controller");
digitalWrite ( LED_GREEN , HIGH ); //Green LED Off
digitalWrite ( LED_RED , LOW ); //Red LED On
digitalWrite ( RELAY , LOW ); //Turn off Relay
delay (2000);
}
//=====
=====
//          LOOP
//=====
=====
void loop (){
  double Temperature = ((5.0/1024.0) * analogRead ( A0 )) * 100;
//10mV per degree 0.01V/C. Scalling

  lcd . setCursor (0,0);
  lcd . print ("Temperature:"); //Do not display entered keys
  lcd . print ( Temperature );

//Get user input for setpoints
if( digitalRead ( down_key )== LOW )
{
  if( SetPoint >0)
  {
    SetPoint --;
  }
}

```

```

}
if( digitalRead ( up_key )== LOW )
{
  if( SetPoint <150)
  {
    SetPoint ++;
  }
}

//Display Set point on LCD
lcd . setCursor (0,1);
lcd . print ("Set Point:");
lcd . print ( SetPoint );
lcd . print ("C ");

//Check Temperature is in limit
if( Temperature > SetPoint )
{
  digitalWrite ( RELAY , LOW ); //Turn off heater
  digitalWrite ( LED_RED , HIGH );
  digitalWrite ( LED_GREEN , LOW ); //Turn on Green LED
}
else
{
  digitalWrite ( RELAY , HIGH ); //Turn on heater
  digitalWrite ( LED_GREEN , HIGH );
  digitalWrite ( LED_RED , LOW ); //Turn on RED LED
}

delay (100); //Update at every 100mSeconds
}
//=====
=====

```

## Conclusion

**This code demonstrates how to construct digital temperature controller using arduino.**

We have used combination of LCD and Temperature sensor LM35 to make simple temperature controller using Arduino.

## 4. Arduino Object Counter

---



## 4.1 Introduction

Object counting is required in many applications. Industrial production counters, part counters and many more. This project is constructed using Arduino Uno, 7-Segment Display, 74HC595 shift register. We are using two switches to show the up and down count.

## What you will learn?

1. How to connect 7-Segment Display to Arduino?
2. How to take key input?
3. Use of shift register to reduce IOs.

## Components Required

1. Arduino Uno
2. 4 digit 7-segment Display Common Cathode.
3. Keys
4. 1K Resistors
5. 75HC595

# 4.2 Object Counter Circuit

Main components of object counter circuit are 4 digit 7-Segment display, Arduino Uno, 74HC595.

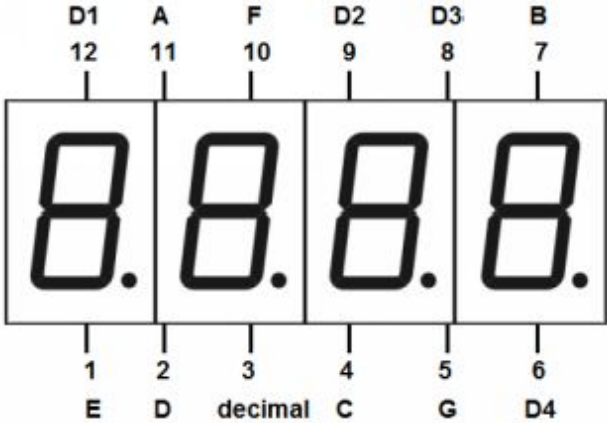


Figure 4.1: 4 Digit 7-Segment Display Pin Diagram

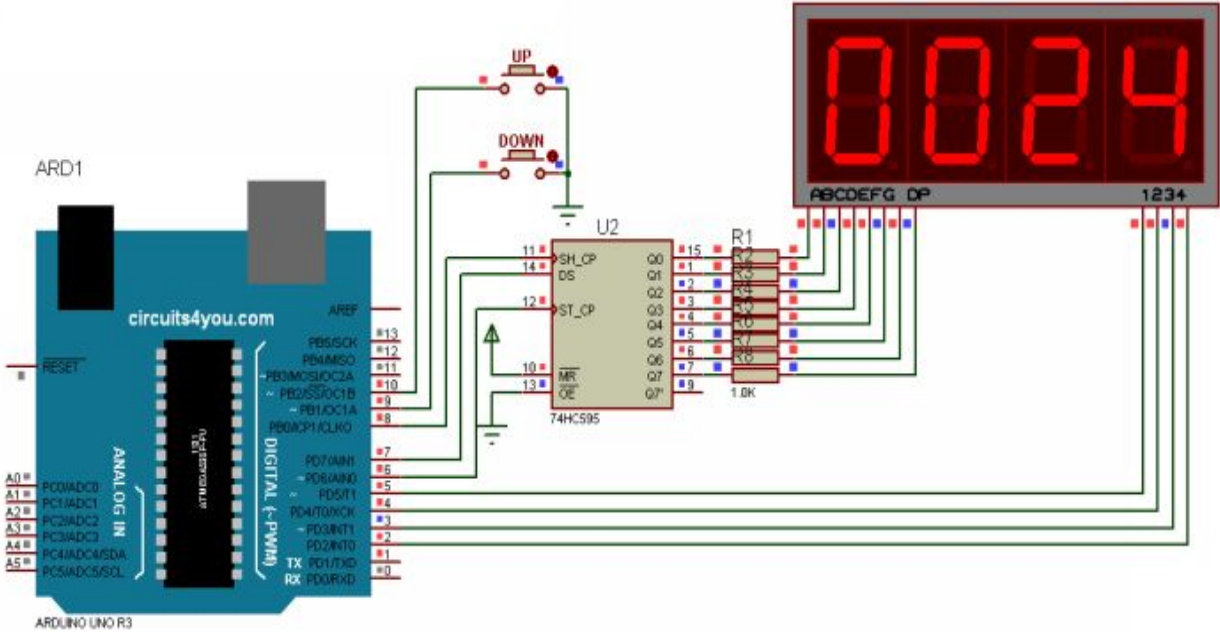


Figure 4.2: Object Counter Circuit

## 4.3 Object Counter Arduino Code

Program is constructed using “TimerOne” library. Program have different modules, Setup, Loop. In setup we initialize all the IO connections and Timer, Display. In main loop we are taking key inputs and constantly updating display to show counter value.

```
/*  
  Digital Object Counter using 4-Digit 7-segment Display  
  www.circuits4you.com  
*/  
  
#include <TimerOne.h>  
  
//Define 74HC595 Connections with arduino  
const int Data =7;  
const int Clock =8;  
const int Latch =6;  
  
const int SEG0 =5;  
const int SEG1 =4;  
const int SEG2 =3;  
const int SEG3 =2;  
  
//Up down keys connection  
const int up =10;  
const int down =9;  
  
int cc =0;  
char Value [4];  
const char SegData []=  
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};  
  
int Count =0;  
//=====
```

```

//          Setup
//=====
=====
void setup () {
  // initialize the digital pin as an output.
  Serial . begin (9600);
  pinMode ( Data , OUTPUT );
  pinMode ( Clock , OUTPUT );
  pinMode ( Latch , OUTPUT );
  pinMode ( SEG0 , OUTPUT );
  pinMode ( SEG1 , OUTPUT );
  pinMode ( SEG2 , OUTPUT );
  pinMode ( SEG3 , OUTPUT );

  pinMode ( up , INPUT ); //Keys or proximity sensor
  pinMode ( down , INPUT );

  digitalWrite ( up , HIGH ); //Pull up for keys
  digitalWrite ( down , HIGH );

      //Initialize Display Scanner
      cc =0;
      Timer1 . initialize (50000); // set a timer of length 100000
      microseconds (or 0.1 sec - or 10Hz => the led will blink 5 times, 5 cycles
      of on-and-off, per second)
      Timer1 . attachInterrupt ( timerIsr ); // attach the service routine here
}
//=====
=====
//          Loop
//=====
=====
void loop () {
  char cnt [4];

  if( digitalRead ( down )== LOW )
  {

```

```

while( digitalRead ( down )== LOW ); //Wait until low
if( Count >0) //Minimum Counting to zero
{
    Count --;
}
}

if( digitalRead ( up )== LOW )
{
    while( digitalRead ( up )== LOW ); //Wait until low
    if( Count <10000) //Max counting 9999
    {
        Count ++;
    }
}
//Display Count on Segments
sprintf ( cnt ,"%04d", Count ); //We get ASCII array in Volt
Serial . println ( Count ); //Print Count on Serial for debug

Value [0]= cnt [0] & 0x0F; //Anding with 0x0F to remove upper nibble
Value [1]= cnt [1] & 0x0F; //Ex. number 2 in ASCII is 0x32 we want
only 2
Value [2]= cnt [2] & 0x0F;
Value [3]= cnt [3] & 0x0F;
delay (50);
}

//=====
//=====
//          Generates Digit
//=====
//=====

void DisplayDigit (char d )
{
    int i ;

for( i =0; i <8; i ++ ) //Shift bit by bit data in shift register

```

```

{
    if(( d & 0x80)==0x80)
    {
        digitalWrite ( Data , HIGH );
    }
    else
    {
        digitalWrite ( Data , LOW );
    }
    d = d <<1;

    //Give Clock pulse
    digitalWrite ( Clock , LOW );
    digitalWrite ( Clock , HIGH );
}

//Latch the data
digitalWrite ( Latch , LOW );
digitalWrite ( Latch , HIGH );
}

//=====
//
//                               TIMER 1 OVERFLOW INTTERRUPT FOR
DISPALY
//=====
//=====

void timerIsr ()
{
    cc ++;
    if( cc ==5) //We have only 4 digits
    { cc =1;}
    Scanner ();
    TCNT0 =0xCC;
}

//=====
//
//                               SCAN DISPLAY FUNCTION

```



```
//=====
=====
void Scanner ()
{
  switch ( cc ) //Depending on which digit is selcted give output
  {
    case 1:
      digitalWrite ( SEG3 , HIGH );
      DisplayDigit ( SegData [ Value [0]]);
      digitalWrite ( SEG0 , LOW );
    break;
    case 2:
      digitalWrite ( SEG0 , HIGH );
      DisplayDigit ( SegData [ Value [1]]);
      digitalWrite ( SEG1 , LOW );
    break;
    case 3:
      digitalWrite ( SEG1 , HIGH );
      DisplayDigit ( SegData [ Value [2]]);
      digitalWrite ( SEG2 , LOW );
    break;
    case 4:
      digitalWrite ( SEG2 , HIGH );
      DisplayDigit ( SegData [ Value [3]]);
      digitalWrite ( SEG3 , LOW );
    break;
  }
}
//=====
=====
```

## Conclusion

**This code demonstrates how to construct digital object counter using arduino.**

You can try with different sensor such as IR proximity to make counting of object passing in front of it. Here we used shift register to reduce IO requirement. You can try 74LS48 7-segment decoder also. Shift register gives benefit of control of each segment. You can display some alphanumeric characters and negative sign.

## 5. Arduino DC Digital Voltmeter

---

## 5.1 Introduction

A voltmeter is an instrument used for measuring electrical potential difference between two points in an electric circuit. Analog voltmeters move a pointer across a scale in proportion to the voltage of the circuit; digital voltmeters give a numerical display of voltage by use of an analog to digital converter. We are using internal ADC of Arduino to make Digital Voltmeter capable to display 0 to 5V. You can increase its input voltage capacity by using voltage divider circuit.

### What you will learn?

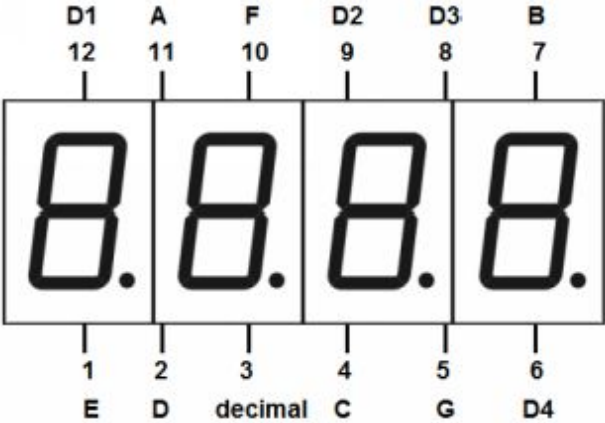
1. How to connect 7-Segment Display to Arduino?
2. How to read analog input?
3. Use of shift register to reduce IOs.
4. Measurement of DC voltage using Arduino.

## Components Required

1. Arduino Uno
2. 4 digit 7-segment Display Common Cathode.
3. Variable Resistor.
4. 1K Resistors
5. 75HC595

## 5.2 Digital Voltmeter Circuit

Main components of object counter circuit are 4 digit 7-Segment display, Arduino Uno, 74HC595.



**Figure 5.1: 4 Digit 7-Segment Display Pin Diagram**

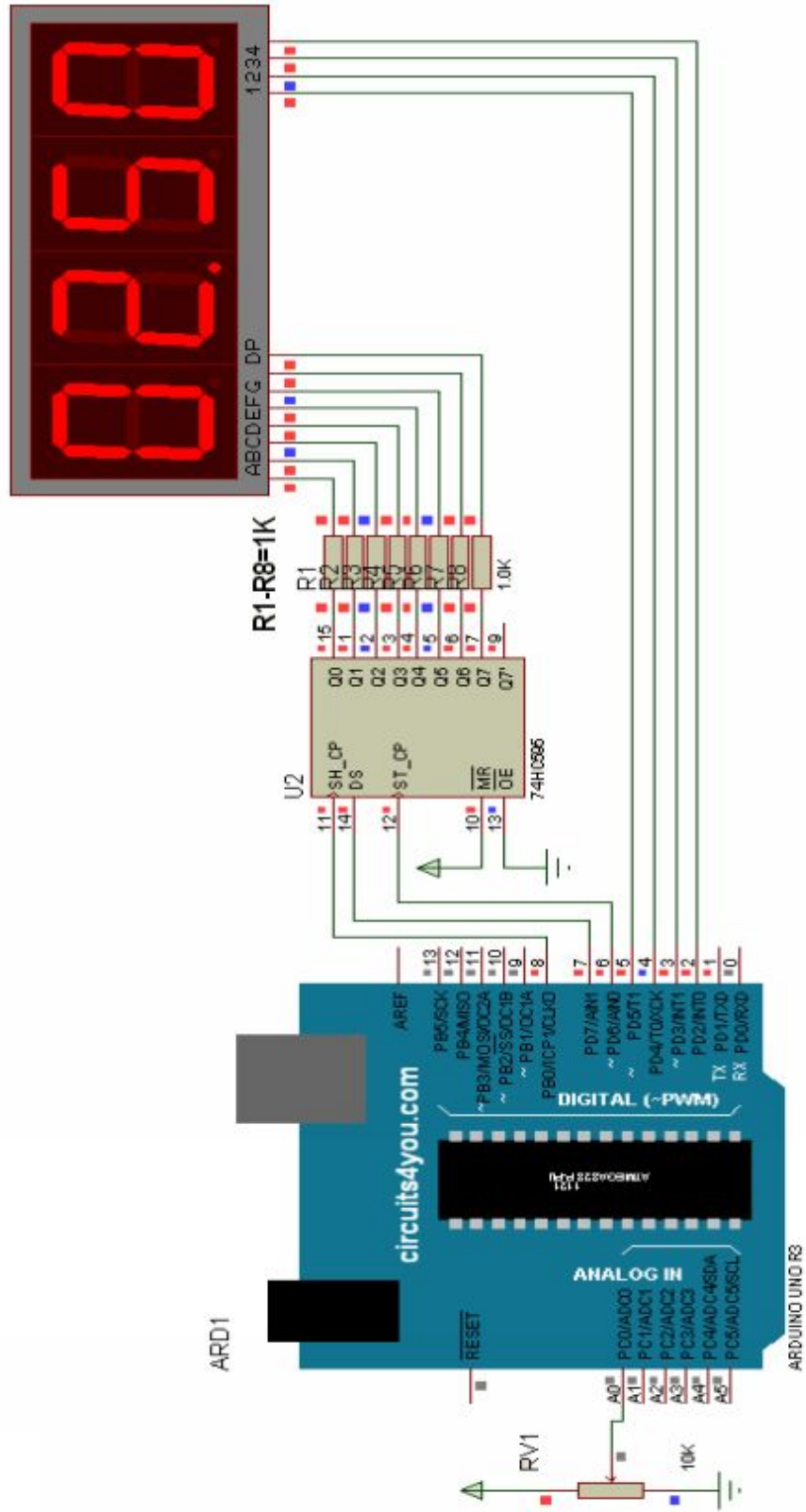


Figure 5.2: Digital Voltmeter Circuit

## 5.3 Digital Voltmeter Arduino Code

Program is constructed using “TimerOne” library. Program have different modules, Setup, Loop. In setup we initialize all the IO connections and Timer, Display. In main loop we are taking Analog input and constantly updating display to show voltage value.

```
/*
  Digital Voltmeter using 4-Digit 7-segment Display
  www.circuits4you.com
*/

#include <TimerOne.h>

//Define 74HC595 Connections with arduino
const int Data =7;
const int Clock =8;
const int Latch =6;

const int SEG0 =5;
const int SEG1 =4;
const int SEG2 =3;
const int SEG3 =2;

int cc =0;
char Value [4];
const char SegData []=
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

//=====
//
//          Setup
//=====
//=====

void setup () {
  // initialize the digital pin as an output.
```



```

Serial . begin (9600);
pinMode ( Data , OUTPUT );
pinMode ( Clock , OUTPUT );
pinMode ( Latch , OUTPUT );
pinMode ( SEG0 , OUTPUT );
pinMode ( SEG1 , OUTPUT );
pinMode ( SEG2 , OUTPUT );
pinMode ( SEG3 , OUTPUT );

    //Initialize Display Scanner
    cc =0;
    Timer1 . initialize (10000); // set a timer of length 100000
microseconds (or 0.1 sec - or 10Hz => the led will blink 5 times, 5 cycles
of on-and-off, per second)
    Timer1 . attachInterrupt ( timerIsr ); // attach the service routine here
}
//=====
=====
//      Loop
//=====
=====
void loop () {
    char Volt [4];
    int Voltage = analogRead ( A0 );
    //To get fixed point decimal point we multiply it by 100
    Voltage = (500/1024.0) * Voltage ; //Scaling of 0 to 5V i.e. 0 to 1023 to
0 to 10 (in 10 steps)

    //Display Voltage on Segments
    sprintf ( Volt , "%04d", Voltage ); //We get ASCII array in Volt
    Serial . println ( Volt );

    Value [0]= Volt [0] & 0x0F; //Anding with 0x0F to remove upper nibble
    Value [1]= Volt [1] & 0x0F; //Ex. number 2 in ASCII is 0x32 we want
only 2
    Value [2]= Volt [2] & 0x0F;
    Value [3]= Volt [3] & 0x0F;

```



```

//=====
=====
void timerIsr ()
{
    cc ++;
    if( cc ==5) //We have only 4 digits
    { cc =1;}
    Scanner ();
    TCNT0 =0xCC;
}

//=====
=====
//          SCAN DISPLAY FUNCTION
//=====
=====
void Scanner ()
{
    switch ( cc ) //Depending on which digit is selcted give output
    {
        case 1:
            digitalWrite ( SEG3 , HIGH );
            DisplayDigit ( SegData [ Value [0]]);
            digitalWrite ( SEG0 , LOW );
            break;
        case 2:
            digitalWrite ( SEG0 , HIGH );
            DisplayDigit ( SegData [ Value [1]] | 0x80); //0x80 to turn on decimal
point
            digitalWrite ( SEG1 , LOW );
            break;
        case 3:
            digitalWrite ( SEG1 , HIGH );
            DisplayDigit ( SegData [ Value [2]]);
            digitalWrite ( SEG2 , LOW );
            break;
        case 4:

```

```
digitalWrite ( SEG2 , HIGH );  
DisplayDigit ( SegData [ Value [3]]);  
digitalWrite ( SEG3 , LOW );  
break;  
}  
}  
//=====
```

## Conclusion

**This code demonstrates how to construct digital voltmeter using arduino.**

You can try with different sensor and measurement with this code for more on AC, DC voltage, current measurement you can refer “[Measurement Made Simple with Arduino e-Book](#)” available on [circuits4you.com](#) and Amazon.

## 6. Arduino Water Level Controller

---

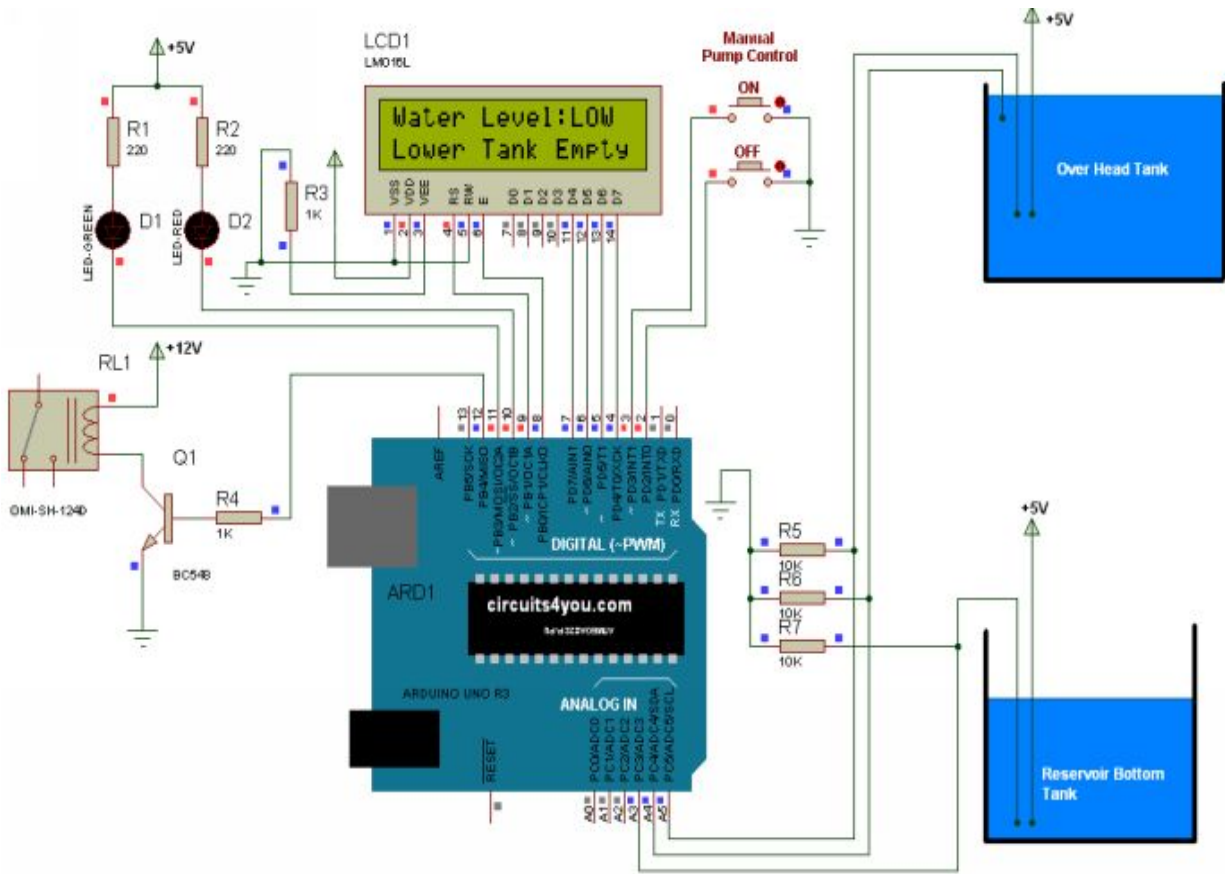
## 6.1 Introduction

Water Level controllers do not need operator for performing start and stop operations of water pump. This automatic water level controller switches ON the motor when the water level in the tank becomes low (desired prefixed lower limit). It switches OFF the motor once the tank becomes full.

### Components Required

1. Arduino Uno
2. LCD 16x2.
3. Float Sensor.
4. 1K,220E,10K Resistors
5. 12V Relay
6. BC548 Transistor
7. Switches.

## 6.2 Water Level Controller Circuit



**Figure 6.1: Water Level Controller Circuit**

Main components of water level controller are Relay and level sensors. Level sensors are build using analog in placed close with +5V. Water acts as conductor when water touches the contact some part of voltage goes to analog in pins, this way it detects water level. Float sensors can be used instead of contacts.

Pump can be controlled manually using switches. Pump automatically turns off when it detects the upper tank level full or bottom sump tank is empty. In automatic operation pump turns on when upper tank level goes below the sensor level.

Water pump is connected to the relay. LCD display shows the tank level and pump status.



## 6.3 Water Level Controller Arduino Code

```
/*
  circuits4you.com
  Water Level Controller
*/
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd (9, 8, 7, 6, 5, 4);

const int LED_RED =10; //Red LED
const int LED_GREEN =11; //Green LED
const int RELAY =12; //Lock Relay or motor

//Key connections with arduino
const int on_key =3;
const int off_key =2;
char Pump =0;
//=====
=====
//          SETUP
//=====
=====

void setup (){
  pinMode ( LED_RED , OUTPUT );
  pinMode ( LED_GREEN , OUTPUT );
  pinMode ( RELAY , OUTPUT );
  pinMode ( on_key , INPUT );
  pinMode ( off_key , INPUT );

  //Pull up for setpoint keys
  digitalWrite ( on_key , HIGH );
  digitalWrite ( off_key , HIGH );

  // set up the LCD's number of columns and rows:
```

```

lcd . begin (16, 2);
// Print a message to the LCD.
lcd . print ("circuits4you.com");
lcd . setCursor (0,1); //Move cursor to second Line
lcd . print ("Pump Controller");
digitalWrite ( LED_GREEN , HIGH ); //Green LED Off
digitalWrite ( LED_RED , HIGH ); //Red LED Off
digitalWrite ( RELAY , LOW ); //Turn off Relay
lcd . setCursor (0,1);
lcd . print ("Pump : OFF ");
delay (2000);
}
//=====
=====
// LOOP
//=====
=====
void loop (){

lcd . setCursor (0,0);
lcd . print ("Water Level:"); //Do not display entered keys
if( analogRead ( A4 )>512) //Check High Level
{
lcd . print ("FULL ");
digitalWrite ( LED_GREEN , LOW ); //Green LED On
Pump =0; //Pump off
}

if( analogRead ( A5 )<400) //Check Low Level
{
lcd . print ("LOW ");
digitalWrite ( LED_GREEN , HIGH ); //Tank Level Low turn off Green
LED
}

if( analogRead ( A3 ) < 400) //Lower Tank Empty
{

```

```

Pump =0;
lcd . setCursor (0,1);
lcd . print ("Lower Tank Empty ");
delay (1000);
}
else
{
  if( analogRead ( A5 )<400) //Upper tank water level low turn on pump
  {
    Pump =1;
  }
}

if( Pump ==1)
{
  digitalWrite ( LED_RED , LOW ); //Turn on pump indication and
pump
  digitalWrite ( RELAY , HIGH );
  lcd . setCursor (0,1);
  lcd . print ("Pump : ON  ");
}
else
{
  digitalWrite ( LED_RED , HIGH ); //Turn off pump indication and
pump
  digitalWrite ( RELAY , LOW );
  lcd . setCursor (0,1);
  lcd . print ("Pump : OFF  ");
  delay (1000);
}

//Get user input for setpoints
if( digitalRead ( on_key )== LOW )
{
  while( digitalRead ( on_key )== LOW ); //Wait until low
  Pump =1; // Turn on pump
}

```

```
if( digitalRead ( off_key )== LOW )
{
  Pump =0; //Turn off Pump
}

delay (100); //Update at every 100mSeconds
}
//=====
=====
```

## Conclusion

Test circuit with water and observe LED and Relay on/off with simulating test conditions.

For continuous precise water level measurement you can refer “[Measurement Made Simple with Arduino e-Book](#)” available on [circuits4you.com](http://circuits4you.com) and Amazon.

## 7. Automatic Light Controller

---

## 7.1 Introduction

Automatic light controller offers energy saving and convenience in the areas with a photo sensor (LDR). This senses the ambient light conditions in the surrounding area and switches ON-OFF the lighting load. The darkness level in the surrounding is settable. It is in-built with an additional PIR Sensor which TURNS ON Light in the presence of human and switches OFF after 10 seconds if no human detected for energy saving operation. Thus it provides artificial light only when it is needed. This reduces the large amount of energy wastage and helps in making the most energy efficient lightings.

### Components Required

1. Arduino Uno
2. PIR Sensor.
3. LDR Sensor.
4. 1K, 10K Resistors
5. 12V Relay
6. BC548 Transistor
7. Switches.

## 7.2 Automatic Light Controller Circuit

Circuit is constructed with PIR sensor, LDR and Arduino. Light Load is connected to Relay. Manual on off is possible with given switches.

### PIR Sensor



**Figure 7.1: PIR Sensor**

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

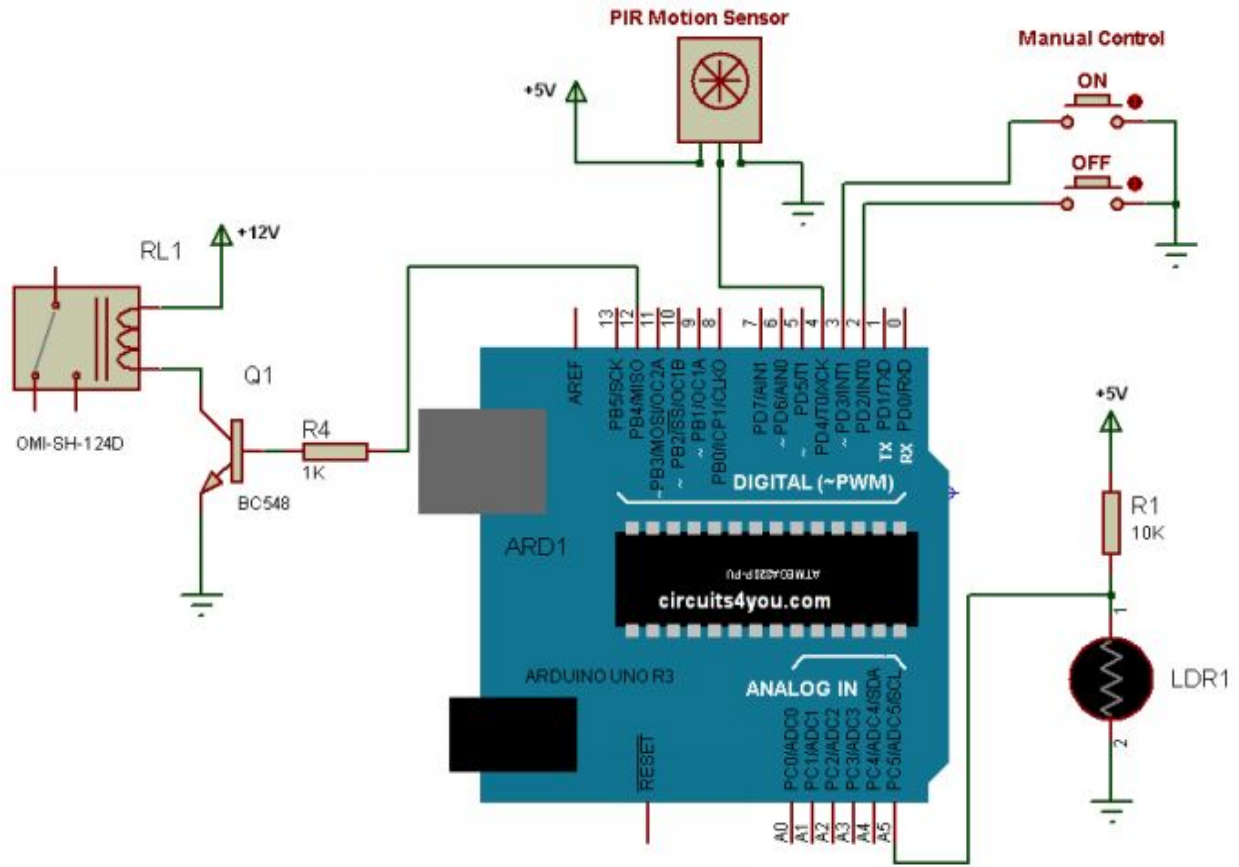
### LDR



**Figure 7.2: LDR Sensor**

A photoresistor (or light-dependent resistor, LDR, or photocell) is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity.





**Figure 7.3: Automatic Light Controller Circuit**

## 7.3 Automatic Light Controller Arduino Code

```
/*
  circuits4you.com
  Day night Switch with Occupancy Sensor (Automatic Light Controller)
*/
const int RELAY =12; //Lock Relay or motor

//Key connections with arduino
const int on_key =3;
const int off_key =2;
int counter =0, manual =0;

//Sensor Connections
const int LDR = A5 ;
const int PIR =4;
//=====
=====
//          SETUP
//=====
=====
void setup (){
  pinMode ( RELAY , OUTPUT );
  pinMode ( on_key , INPUT );
  pinMode ( off_key , INPUT );
  pinMode ( PIR , INPUT );

  //Pull up for setpoint keys
  digitalWrite ( on_key , HIGH );
  digitalWrite ( off_key , HIGH );
  digitalWrite ( PIR , HIGH );

  digitalWrite ( RELAY , LOW );    //Turn off Relay
}
//=====
=====
```

```

//          LOOP
//=====
=====
void loop (){

//Turn on Lights if Motion is detected and Light intensity is low
if( digitalRead ( PIR )== HIGH )
{
  counter =1000; //Set 10 Seconds time out counter
  if( counter >15) //Motion detected for 1.5 Seconds
  {
    if( analogRead ( LDR )>512) //Light intensity is low
    {
      digitalWrite ( RELAY , HIGH ); //Turn on Lights
    }
  }
}

counter --;
if( counter ==0)
{
  if( manual ==0) //Check that it is not manually turned on
  {
    digitalWrite ( RELAY , LOW );
  }
}
//Get user input for setpoints
if( digitalRead ( on_key )== LOW )
{
  digitalWrite ( RELAY , HIGH ); //Turn on Lights
  manual =1; //Manually it is turned on
}
if( digitalRead ( off_key )== LOW )
{
  digitalWrite ( RELAY , LOW ); //Turn off Lights
  manual =0;
}
}

```

```
delay (10); //Update at every 10mSeconds
}
//=====
=====
```

## **Conclusion**

Circuit is build with very few components and it saves lot of energy.  
Test circuit.

## 8. Solar Power Monitor

---

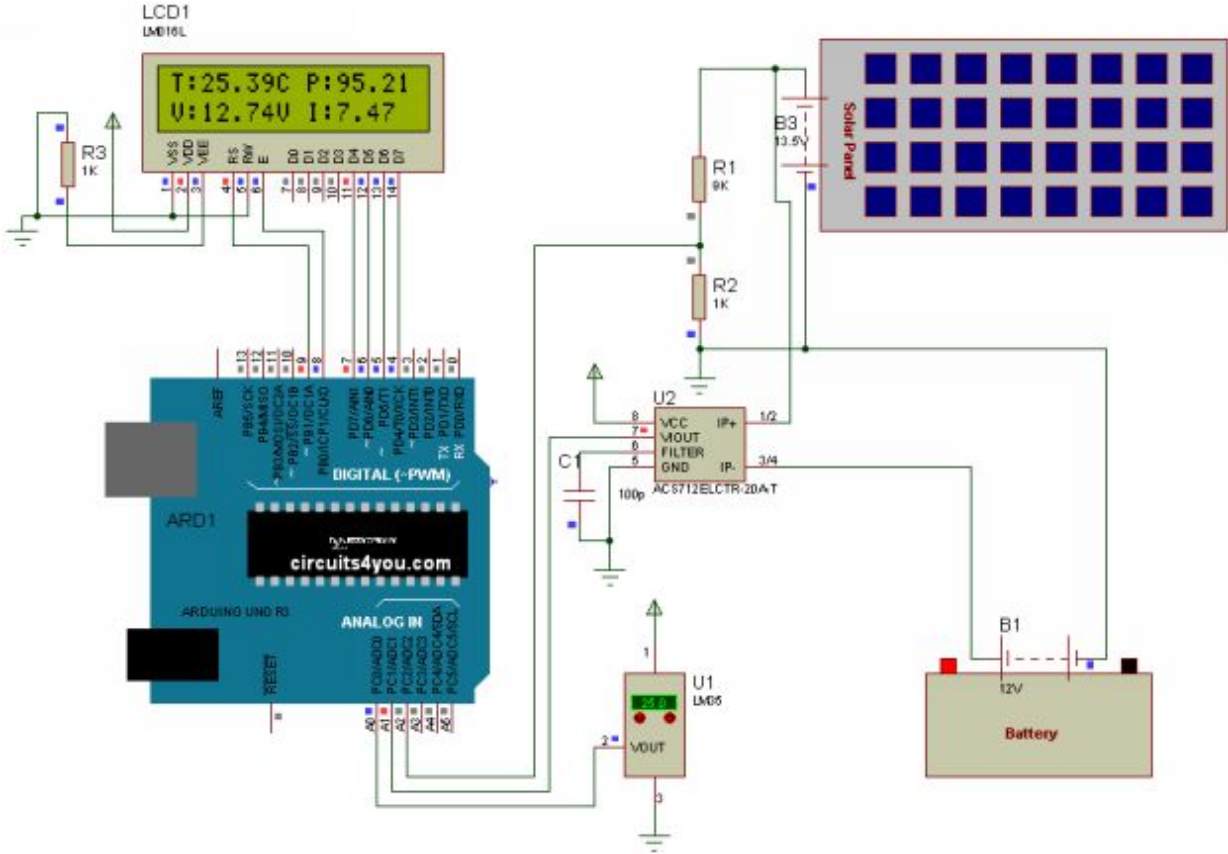
## 8.1 Introduction

Solar Power Monitor is circuit to monitor current power generation of solar power. It uses ACS712 Current sensor, Voltage sensor (use voltage divider), Temperature sensor LM35. The circuit will display instantaneous current, voltage and power on LCD. Maximum current we can measure here is 20Amps with 12v solar system.

### Components Required

1. Arduino Uno
2. ACS712-20A
3. 16x2 LCD
4. 1K, 9K Resistors
5. LM35 Temperature sensor

## 8.2 Solar Power Monitor Circuit



**Figure 8.1: Solar Power Monitor Circuit**

LCD displays Current measured using ASC712-20Amp sensor, voltage when there is sunlight it displays voltage across solar panel is equal to the battery voltage at night we get correct battery voltage level.



## 8.3 Solar Power Monitor Code

```
/*
  circuits4you.com
  Solar Power Generation Monitoring System
*/
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd (9, 8, 7, 6, 5, 4);

//Sensor Connections
const int Temp = A0 ;
const int Current = A1 ;
const int VoltDiv = A2 ;

double mVperAmp = 100; // use 100 for 20A Module and 66 for 30A
Module
double ACSoffset = 2500;
double RawValue = 0;
double Volt = 0;
double Amps = 0;
//=====
=====
//          SETUP
//=====
=====
void setup (){
  // set up the LCD's number of columns and rows:
  lcd . begin (16, 2);
  // Print a message to the LCD.
  lcd . print ("circuits4you.com");
  lcd . setCursor (0,1); //Move cursor to second Line
  lcd . print (" Solar Power ");
  delay (2000);
}
//=====
=====
```

```

//          LOOP
//=====
=====
void loop (){
  double Temperature = ((5.0/1024.0) * analogRead ( Temp )) * 100;
//10mV per degree 0.01V/C. Scalling
  double Voltage = ((5.0 / 1024.0) * analogRead ( VoltDiv ))* 10;
//Voltage divider /10 can measure upto 50V

//ACS712 Current Measurement
RawValue = analogRead ( Current );
Volt = ( RawValue / 1024.0) * 5000.0; // Gets you mV
Amps = (( Volt - ACSoffset ) / mVperAmp );

//Display Line 1 Temperature and Power
lcd . setCursor (0,0); //Move cursor to line 1
lcd . print ("T:");
lcd . print ( Temperature );
lcd . print ("C ");
lcd . print ("P:");
lcd . print ( Voltage * Amps ); //Calculate power
lcd . print ("W");

//Display Line 2 Voltage and Current
lcd . setCursor (0,1); //Move cursor to line 2
lcd . print ("V:"); //Display Voltage
lcd . print ( Voltage );
lcd . print ("V ");
lcd . print ("I:"); //Current Current
lcd . print ( Amps );
lcd . print (" A");

delay (100); //Update at every 100mSeconds
}
//=====
=====

```



## Conclusion

Here we have learned how to measure voltage, current and power using arduino. You can use ACS712 current sensor module.

## 9. Ultrasonic Distance Meter

---

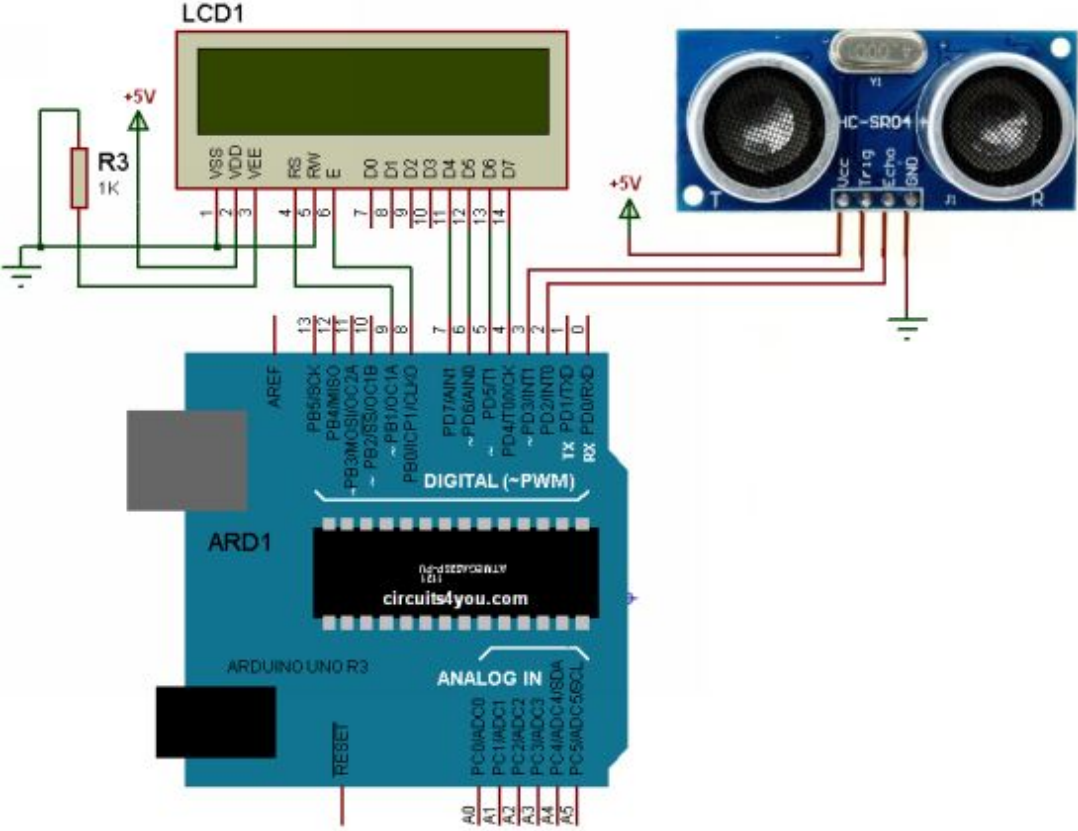
## 9.1 Introduction

This application is based upon the reflection of sound waves. Sound waves are defined as longitudinal pressure waves in the medium in which they are travelling. Subjects whose dimensions are larger than the wavelength of the impinging sound waves reflect them; the reflected waves are called the echo. If the speed of sound in the medium is known and the time taken for the sound waves to travel the distance from the source to the subject and back to the source is measured, the distance from the source to the subject can be computed accurately. This is the measurement principle of this application.

### Components Required

1. Arduino Uno
2. HCSR04 Ultrasonic Distance Sensor
3. 16x2 LCD
4. 1K Resistors

# 9.2 Ultrasonic Distance Meter Circuit



**Figure 9.1: Ultrasonic Distance Meter Circuit**

LCD displays measured distance using ultrasonic sensor in inches and cm's.

## 9.3 Ultrasonic Distance Meter Arduino Code

Here the medium for the sound waves is air, and the sound waves used are ultrasonic, since it is inaudible to humans. Assuming that the speed of sound in air is 1100 feet/second at room temperature and that the measured time taken for the sound waves to travel the distance from the source to the subject and back to the source is  $t$  seconds, the distance  $d$  is computed by the formula  $d=1100 \times t$  inches. Since the sound waves travel twice the distance between the source and the subject, the actual distance between the source and the subject will be  $d/2$ .

```
/*
=====
circuits4you.com
Distance Measurement
Ultrasonic sensor Pins:
  VCC: +5VDC
  Trig : Trigger (INPUT) - Pin 3
  Echo: Echo (OUTPUT) - Pin 2
  GND: GND
=====
*/

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd (9, 8, 7, 6, 5, 4);

int trigPin = 3; //Trig - green Jumper
int echoPin = 2; //Echo - yellow Jumper
long duration , cm , inches ;

void setup () {
  //Serial Port begin
  Serial . begin (9600);
  //Define inputs and outputs
```



```

pinMode ( trigPin , OUTPUT );
pinMode ( echoPin , INPUT );

// set up the LCD's number of columns and rows:
lcd . begin (16, 2);
}

void loop ()
{

// The sensor is triggered by a HIGH pulse of 10 or more microseconds.
// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
digitalWrite ( trigPin , LOW );
delayMicroseconds (5);
digitalWrite ( trigPin , HIGH );
delayMicroseconds (10);
digitalWrite ( trigPin , LOW );

// Read the signal from the sensor: a HIGH pulse whose
// duration is the time (in microseconds) from the sending
// of the ping to the reception of its echo off of an object.
pinMode ( echoPin , INPUT );
duration = pulseIn ( echoPin , HIGH );

// convert the time into a distance
cm = ( duration /2) / 29.1;
inches = ( duration /2) / 74;

lcd . setCursor (0,0);
lcd . print ("Inches:");
lcd . print ( inches );
lcd . print ("in");

lcd . setCursor (0,1);
lcd . print ("Cm:");
lcd . print ( cm );
lcd . print ("cm");

```

```
Serial . print ( inches );  
Serial . print ( "in, " );  
Serial . print ( cm );  
Serial . print ( "cm" );  
Serial . println ();  
  
delay (250);  
}
```

## **Conclusion**

Ultrasonic Distance meter will show the distance on LCD as well as on serial terminal. You can try this project with 7-Segment display also.

# 10. Digital Timer

---

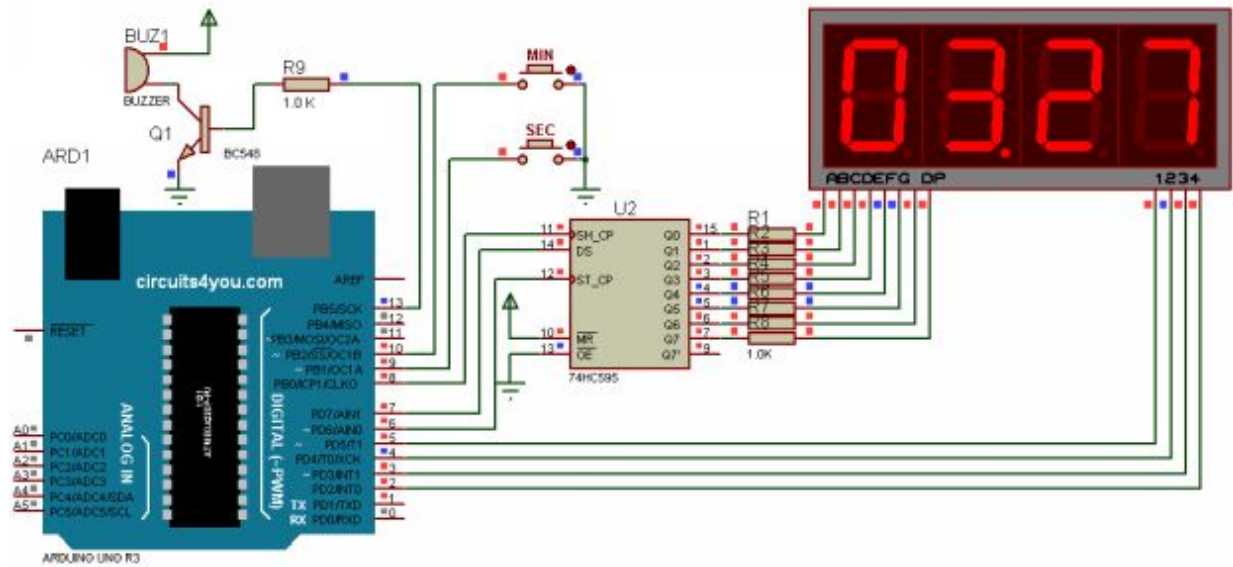
## 10.1 Introduction

Need a few minutes? Pick up the Mini Digital Timer. It's the perfect size for use in the kitchen or around the house, with an easy-to-read display. Use for countdown timer and enjoy an elapsed time graphic plus an alarm.

### Components Required

1. Arduino Uno
2. Buzzer
3. 4-Digit 7-Segment Display Common Cathode
4. 1K Resistors
5. BC548 Transistor
6. 74HC595
7. Switches

## 10.2 Digital Timer Circuit



**Figure 10.1: Digital Timer Circuit**

Two switches are used to set minutes and seconds. As soon as you set the seconds or minutes timer will start to run. It gives beeping sound when time countdown reaches to zero. If both Min and Sec switch is pressed it clears the time.

## 10.3 Digital Timer Arduino Code

```
/*
  Digital Timer using 4-Digit 7-segment Display
  www.circuits4you.com
*/

#include <TimerOne.h>

//Define 74HC595 Connections with arduino
const int Data =7;
const int Clock =8;
const int Latch =6;

const int SEG0 =5;
const int SEG1 =4;
const int SEG2 =3;
const int SEG3 =2;

const int Buzzer =13; //Buzzer

//Up down keys connection
const int Min_key =10;
const int Sec_key =9;

int cc =0;
char Value [4];
const char SegData []=
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

int MIN =0, SEC =0, count =10;
//=====
=====
```

```

//          Setup
//=====
=====
void setup () {
  // initialize the digital pin as an output.
  Serial . begin (9600);
  pinMode ( Data , OUTPUT );
  pinMode ( Clock , OUTPUT );
  pinMode ( Latch , OUTPUT );
  pinMode ( SEG0 , OUTPUT );
  pinMode ( SEG1 , OUTPUT );
  pinMode ( SEG2 , OUTPUT );
  pinMode ( SEG3 , OUTPUT );
  pinMode ( Buzzer , OUTPUT );
  digitalWrite ( Buzzer , LOW ); //Turn off buzzer

  pinMode ( Min_key , INPUT ); //Keys or proximity sensor
  pinMode ( Sec_key , INPUT );

  digitalWrite ( Min_key , HIGH ); //Pull up for keys
  digitalWrite ( Sec_key , HIGH );

  //Initialize Display Scanner
  cc =0;
  Timer1 . initialize (10000); // set a timer of length 100000
  microseconds (or 0.1 sec - or 10Hz => the led will blink 5 times, 5 cycles
  of on-and-off, per second)
  Timer1 . attachInterrupt ( timerIsr ); // attach the service routine here
}
//=====
=====
//          Loop
//=====
=====
void loop () {
  char cMIN [4], cSEC [4];

```



```

if( count ==0)
{
  count =10; //1 Second as we have loop delay of 100mSec
  if( MIN >0) //if Minutes are greater than zero
  {
    if( SEC ==0)
    {
      MIN --;
      SEC =60;
    }
  }
  SEC --;
  //Check that timer is zero
  if( MIN ==0 && SEC ==1) //Second is kept one to avoid beeping at
normal zero
  {
    digitalWrite ( Buzzer , HIGH ); //Turn on Buzzer
  }
}
if( MIN > 0 || SEC > 0)
{
  count --;
}

if( digitalRead ( Min_key )== LOW )
{
  delay (10);
  MIN ++;
  digitalWrite ( Buzzer , LOW ); //Turn off Buzzer
}

if( digitalRead ( Sec_key )== LOW )
{
  delay (10); //debounce
  SEC ++;
  if( SEC >59) //60 seconds = 1 minute

```

```

    {
        MIN ++;
        SEC =0;
    }
    digitalWrite ( Buzzer , LOW ); //Turn off Buzzer
}

if( digitalRead ( Sec_key )== LOW &&
digitalRead ( Min_key )== LOW ) //If both Switches pressed clear timer
{
    MIN =0;
    SEC =0;
    digitalWrite ( Buzzer , HIGH ); //Give Beep
    delay (500);
    digitalWrite ( Buzzer , LOW );
}

//Display Count on Segments
sprintf ( cMIN ,"%02d", MIN ); //We get ASCII array
sprintf ( cSEC ,"%02d", SEC ); //We get ASCII array
Serial . println ( cMIN ); //Print Count on Serial for debug

Value [0]= cMIN [0] & 0x0F; //Anding with 0x0F to remove upper
nibble
Value [1]= cMIN [1] & 0x0F; //Ex. number 2 in ASCII is 0x32 we want
only 2

Value [2]= cSEC [0] & 0x0F;
Value [3]= cSEC [1] & 0x0F;
delay (100);
}
//=====
=====
//
//
//
//
//=====
=====

```

```

void timerIsr ()
{
    cc ++;
    if( cc ==5) //We have only 4 digits
    { cc =1;}
    Scanner ();
    TCNT0 =0xCC;
}
//=====
//
//      Generates Digit
//=====
void DisplayDigit (char d )
{
    int i ;

for( i =0; i <8; i ++ ) //Shift bit by bit data in shift register
{
    if(( d & 0x80)==0x80)
    {
        digitalWrite ( Data , HIGH );
    }
    else
    {
        digitalWrite ( Data , LOW );
    }
    d = d <<1;

    //Give Clock pulse
    digitalWrite ( Clock , LOW );
    digitalWrite ( Clock , HIGH );
}
//Latch the data
digitalWrite ( Latch , LOW );
digitalWrite ( Latch , HIGH );
}

```

```

//=====
=====
//          SCAN DISPLAY FUNCTION
//=====
=====
void Scanner ()
{
switch ( cc ) //Depending on which digit is selcted give output
{
case 1:
    digitalWrite ( SEG3 , HIGH );
    DisplayDigit ( SegData [ Value [0]]);
    digitalWrite ( SEG0 , LOW );
break;
case 2:
    digitalWrite ( SEG0 , HIGH );
    DisplayDigit ( SegData [ Value [1]] | 0x80); //Decimal Point
    digitalWrite ( SEG1 , LOW );
break;
case 3:
    digitalWrite ( SEG1 , HIGH );
    DisplayDigit ( SegData [ Value [2]]);
    digitalWrite ( SEG2 , LOW );
break;
case 4:
    digitalWrite ( SEG2 , HIGH );
    DisplayDigit ( SegData [ Value [3]]);
    digitalWrite ( SEG3 , LOW );
break;
}
}
//=====
=====

```

## Conclusion

This timer can be used in many applications, you can add start and stop button similar to commercial timers.

# 11. Automatic Irrigation System

---

## 11. 1 Introduction

Automatic Irrigation system monitors the soil moisture and depending on set points turns on/off the pump which is connected to the relay. This way you can keep soil moisture to a set point.

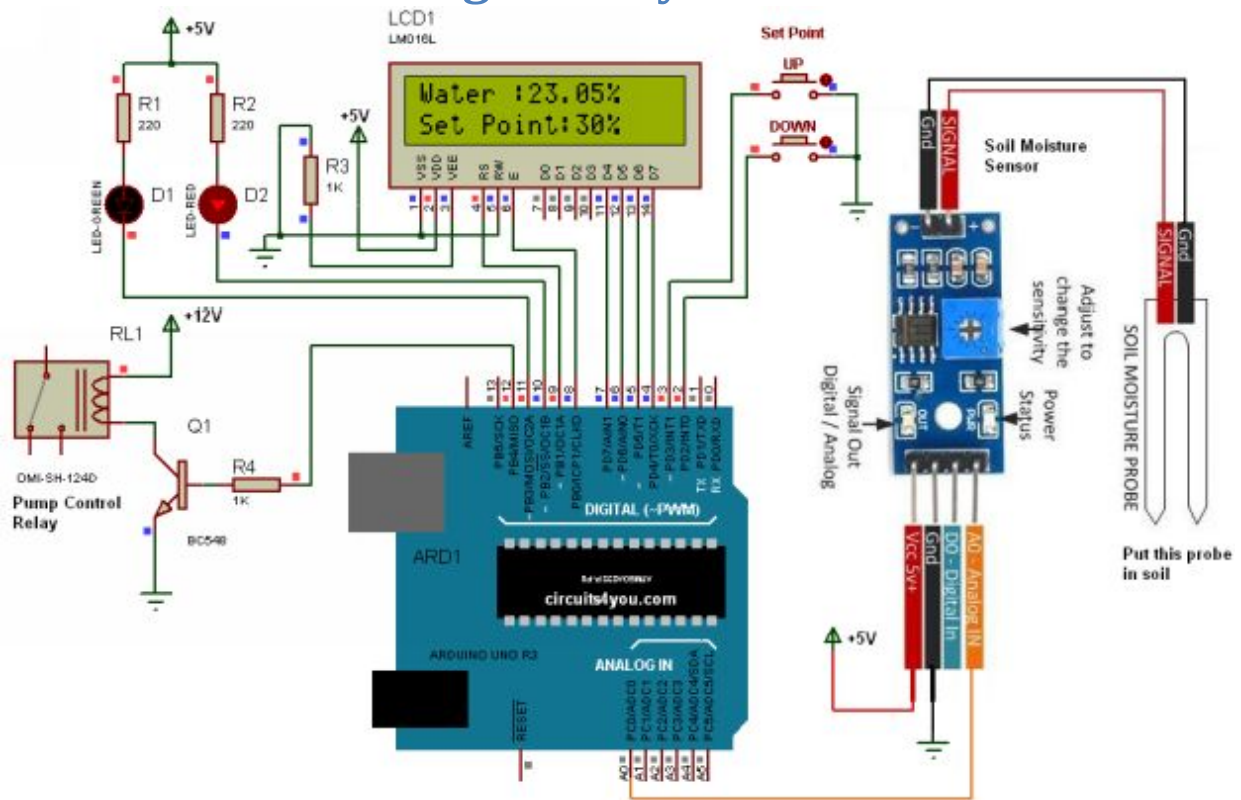
This system automatically waters the plants when we are on vacation. As we are setting the soil moisture level, we need not to worry about too much of watering and the plants end up dying anyway.

## Components Required

1. Arduino Uno
2. Soil moisture sensor
3. 16x2 LCD
4. 1K, 220E Resistors
5. 12V Relay
6. BC548 Transistor
7. Switches



## 11.2 Automatic Irrigation System Circuit



**Figure 11.1: Automatic Irrigation System Circuit**

LCD displays the current moisture level and set point. Set point can be adjusted using push buttons. Connect relay output to water pump. Put soil moisture tip in soil where you want to maintain soil moisture.

## 11.3 Automatic Irrigation System Arduino Code

```
/*
  circuits4you.com
  Automatic Irrigation System
*/
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd (9, 8, 7, 6, 5, 4);

const int LED_RED =10; //Red LED
const int LED_GREEN =11; //Green LED
const int RELAY =12; //Lock Relay or motor

//Key connections with arduino
const int up_key =3;
const int down_key =2;

int SetPoint =30;
//=====
=====
//          SETUP
//=====
=====
void setup (){
  pinMode ( LED_RED , OUTPUT );
  pinMode ( LED_GREEN , OUTPUT );
  pinMode ( RELAY , OUTPUT );
  pinMode ( up_key , INPUT );
  pinMode ( down_key , INPUT );

  //Pull up for setpoint keys
  digitalWrite ( up_key , HIGH );
  digitalWrite ( down_key , HIGH );

  // set up the LCD's number of columns and rows:
  lcd . begin (16, 2);
```

```

// Print a message to the LCD.
lcd . print ("circuits4you.com");
lcd . setCursor (0,1); //Move cursor to second Line
lcd . print (" Irrigation ");
delay (1000);
lcd . setCursor (0,1);
lcd . print (" System ");
digitalWrite ( LED_GREEN , HIGH ); //Green LED Off
digitalWrite ( LED_RED , LOW ); //Red LED On
digitalWrite ( RELAY , LOW ); //Turn off Relay
delay (2000);
}
//=====
=====
// LOOP
//=====
=====
void loop (){
  double WaterLevel = ((100.0/1024.0) * analogRead ( A0 )); //Map it in
  0 to 100%

  lcd . setCursor (0,0);
  lcd . print ("Water:"); //Do not display entered keys
  lcd . print ( WaterLevel );
  lcd . print ("% ");

  //Get user input for setpoints
  if( digitalRead ( down_key )== LOW )
  {
    if( SetPoint >0) //Not less than zero
    {
      SetPoint --;
    }
  }
  if( digitalRead ( up_key )== LOW )
  {
    if( SetPoint <99) //Not more than 100%

```

```

    {
        SetPoint ++;
    }
}

//Display Set point on LCD
lcd . setCursor (0,1);
lcd . print ("Set Point:");
lcd . print ( SetPoint );
lcd . print ("%  ");

//Check Temperature is in limit
if( WaterLevel > SetPoint )
{
    digitalWrite ( RELAY , LOW ); //Turn off water pump
    digitalWrite ( LED_RED , HIGH );
    digitalWrite ( LED_GREEN , LOW ); //Turn on Green LED
}
else
{
    digitalWrite ( RELAY , HIGH ); //Turn on water pump
    digitalWrite ( LED_GREEN , HIGH );
    digitalWrite ( LED_RED , LOW ); //Turn on RED LED
}

delay (100); //Update at every 100mSeconds
}
//=====
=====

```

## **Conclusion**

This system can be applied to garden using multiple soil moisture sensors. This code is similar to temperature controller circuit.

## 12. Mood Lamp

---

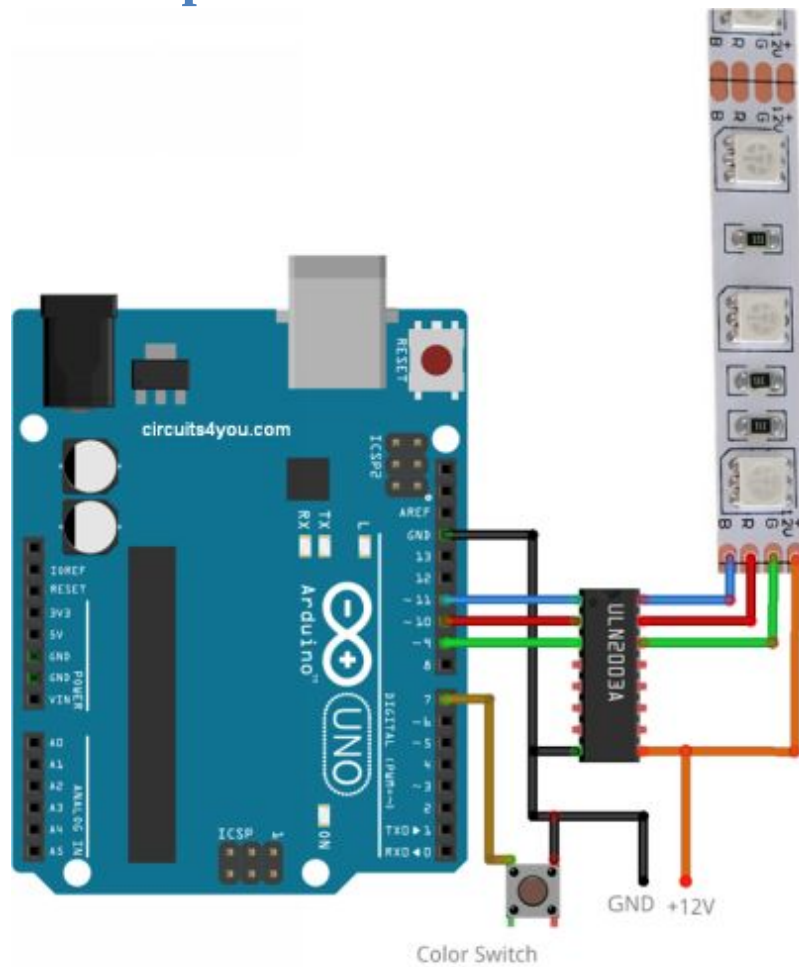
## 12.1 Introduction

Philips offers LivingColor full color lamps, Mood lamp is similar to the Living Color lamp uses buttons to set the color, from this project we get clear idea to control RGB LED Strip to set desired color.

### Components Required

1. Arduino Uno
2. RGB LED Strip
4. ULN2003
5. Switch

## 12.2 Mood Lamp Circuit



**Figure 12.1: Mood Lamp Circuit**

Mood lamp circuits uses ULN2003 to drive RGB LED Strip we are using PWM Channels of Arduino Uno. Single switch to control the color.



## 12.3 Mood Lamp Arduino Code

```
//=====
//
//      RGB LED Color Control
//      circuits4you.com
//=====
//
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 7;  // the number of the pushbutton pin
const int BLUEledPin = 11;  // LED pin
const int REDledPin = 10;  // LED pin
const int GREENledPin = 9;  // LED pin

// variables will change:
int buttonState = 0;      // variable for reading the pushbutton status
int color =0;
//=====
//
//      SETUP
//=====
//
void setup () {
  // initialize the LED pin as an output:
  pinMode ( BLUEledPin , OUTPUT );
  pinMode ( REDledPin , OUTPUT );
  pinMode ( GREENledPin , OUTPUT );

  // initialize the pushbutton pin as an input:
  pinMode ( buttonPin , INPUT );
  digitalWrite ( buttonPin , HIGH ); //Activate internal pull up for switch
}
//=====
//
//      LOOP
//=====
//=====
```

```
void loop (){
  // read the state of the pushbutton value:
  buttonState = digitalRead ( buttonPin );

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if ( buttonState == LOW ) {
    delay (300);
    color ++;
    if( color >10)
    { color =0;}
  }

  //Set different color values refer HTML Color codes
  switch ( color )
  {
    case 0:
      SetColor (255,0,0);
      break;
    case 1:
      SetColor (255,255,0);
      break;
    case 2:
      SetColor (0,0,255);
      break;
    case 3:
      SetColor (128,128,20);
      break;
    case 4:
      SetColor (0,255,255);
      break;
    case 5:
      SetColor (55,100,100);
      break;
    case 6:
      SetColor (0x00,0xA8,0xA9);
      break;
  }
}
```

```

case 7:
  SetColor (0xCC,0x66,0x66);
break;
case 8:
  SetColor (0x12,0xA2,0x7E);
break;
case 9:
  SetColor (0xF0,0x80,0x32);
break;
case 10:
  SetColor (0x30,0xFF,0xFF);
break;
}
}
//=====
=====
//          Write Color Value
//=====
=====
void SetColor (char R ,char G ,char B )
{
  analogWrite ( REDledPin , R );
  analogWrite ( GREENledPin , G );
  analogWrite ( BLUEledPin , B );
}

```

## **Conclusion**

Simple PWM generation of Arduino can be used for controlling colors of RGB LED strip. You can try this project with combination of Bluetooth module.

# 13. Bluetooth based home automation

---

## 13. 1 Introduction

Now days everyone talks about smart home and smart phones. Let's see how we can convert simple electrical switch board into smart android app controlled. Here we are using arduino and Bluetooth module to control relays. Android app acts as wireless serial link between arduino an android app.

You can download android app “BlueHome.apk” from google drive link

<https://drive.google.com/file/d/0ByhR7XNrC0R0MXEtYmRld1lySkU/view?usp=sharing>

## Components Required

1. Arduino Uno
2. Bluetooth Module HC-06 (*Note: Android App works only with HC06*)
3. 4-Channel Relay Board
4. Connecting Wires





## 13.3 Bluetooth based home automation arduino code

```
/*  
circuits4you.com  
Bluetooth based home automation  
*/  
  
//Define Relay Connections  
#define Relay1 8  
#define Relay2 9  

```

```

break;
case 0x05:
    digitalWrite ( Relay2 , HIGH ); //Relay 1 on when inByte equals
A
    break;
case 0x06:
    digitalWrite ( Relay2 , LOW ); //Relay 1 off when inByte equals
B
    break;
case 0x07:
    digitalWrite ( Relay3 , HIGH ); //Relay 1 on when inByte equals
A
    break;
case 0x08:
    digitalWrite ( Relay3 , LOW ); //Relay 1 off when inByte equals
B
    break;
case 0x09:
    digitalWrite ( Relay4 , HIGH ); //Relay 1 on when inByte equals
A
    break;
case 0x0A:
    digitalWrite ( Relay4 , LOW ); //Relay 1 off when inByte equals
B
    break;
case 0x0B:
    digitalWrite ( Relay1 , HIGH ); //All Relays ON when inByte
equals 1
    digitalWrite ( Relay2 , HIGH );
    digitalWrite ( Relay3 , HIGH );
    digitalWrite ( Relay4 , HIGH );
    break;
case 0x0C:
    digitalWrite ( Relay1 , LOW ); //All Relays OFF when inByte
equals 2
    digitalWrite ( Relay2 , LOW );
    digitalWrite ( Relay3 , LOW );

```

```
digitalWrite ( Relay4 , LOW );  
break;  
}  
}  
}
```

## **Testing Procedure**

- 1. Install Android app on your phone.**
- 2. Allow third party download**
- 3. Pair HC-06**
- 4. Then start the software**
- 5. Press Connect button on android app**
- 6. Press on off buttons on android app and see the results.**

# 14. Traffic Light Controller

---

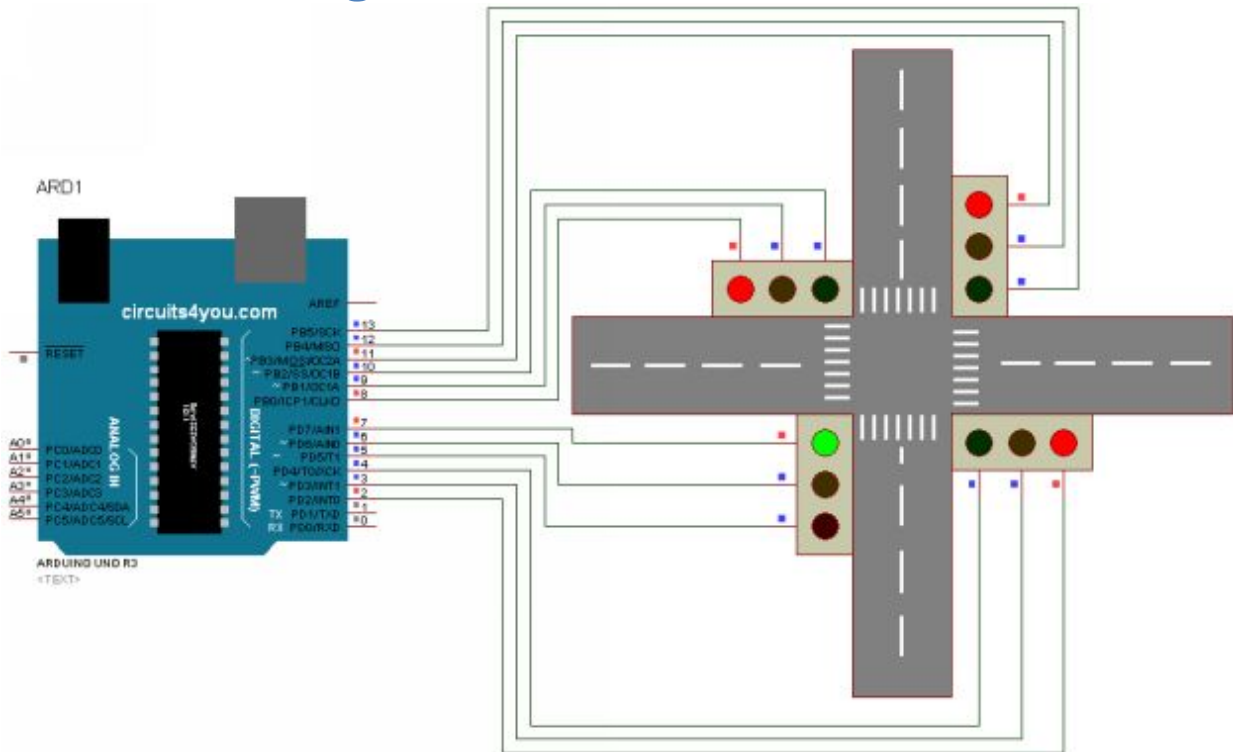
## 14.1 Introduction

The normal function of traffic lights requires more than slight control and coordination to ensure that traffic moves as smoothly and safely as possible and that pedestrians are protected when they cross the roads. A variety of different control systems are used to accomplish this, ranging from simple clockwork mechanisms to sophisticated computerized control and coordination systems that self-adjust to minimize delay to people using the road. Here we are using arduino to do the work.

### Components Required

1. Arduino Uno
2. Red LEDs
3. Green LEDs
4. Yellow LEDs
5. 1K Resistors

## 14.2 Traffic Light Controller Circuit



**Figure 14.1: Traffic Light Controller Circuit**

Connect LEDs in series with 1K current limiting resistor and make placement as shown in figure 14.1.

## 14.3 Traffic Light Controller Code

Controlling of traffic light is all about signal timing and sequencing.

```
//=====
=====
// Traffic Light Controller
// circuits4you.com
//=====
=====
//Arduino Connections with Traffic Lights (LEDs)
const int R_1 =11;
const int Y_1 =12;
const int G_1 =13;

const int R_2 =8;
const int Y_2 =9;
const int G_2 =10;

const int R_3 =5;
const int Y_3 =6;
const int G_3 =7;

const int R_4 =2;
const int Y_4 =3;
const int G_4 =4;

//=====
=====
//      CONNECTION SETUP
//=====
=====
void setup ()
{
//Make all LEDs digital Outputs
pinMode ( R_1 , OUTPUT );
```



```
pinMode ( Y_1 , OUTPUT );  
pinMode ( G_1 , OUTPUT );
```

```
pinMode ( R_2 , OUTPUT );  
pinMode ( Y_2 , OUTPUT );  
pinMode ( G_2 , OUTPUT );
```

```
pinMode ( R_3 , OUTPUT );  
pinMode ( Y_3 , OUTPUT );  
pinMode ( G_3 , OUTPUT );
```

```
pinMode ( R_4 , OUTPUT );  
pinMode ( Y_4 , OUTPUT );  
pinMode ( G_4 , OUTPUT );
```

```
//Circuit Power on State all RED
```

```
digitalWrite ( R_1 , HIGH );  
digitalWrite ( Y_1 , LOW );  
digitalWrite ( G_1 , LOW );
```

```
digitalWrite ( R_2 , HIGH );  
digitalWrite ( Y_2 , LOW );  
digitalWrite ( G_2 , LOW );
```

```
digitalWrite ( R_3 , HIGH );  
digitalWrite ( Y_3 , LOW );  
digitalWrite ( G_3 , LOW );
```

```
digitalWrite ( R_4 , HIGH );  
digitalWrite ( Y_4 , LOW );  
digitalWrite ( G_4 , LOW );  
}
```

```
//=====
```

```
//          Programming
```

```

//=====
=====
void loop ()
{
  int YellowTime =2000; //2 Seconds
  int GreenTime =30000; //30 Seconds

//1. =====
  digitalWrite ( R_1 , LOW );
  digitalWrite ( Y_1 , LOW );
  digitalWrite ( G_1 , HIGH );
  delay ( GreenTime );
//1. =====
  digitalWrite ( R_1 , LOW );
  digitalWrite ( Y_1 , HIGH );
  digitalWrite ( G_1 , LOW );
  delay ( YellowTime );
  digitalWrite ( R_1 , HIGH );
  digitalWrite ( Y_1 , LOW );

//2. =====
  digitalWrite ( R_2 , LOW );
  digitalWrite ( Y_2 , LOW );
  digitalWrite ( G_2 , HIGH );
  delay ( GreenTime );
//2. =====
  digitalWrite ( R_2 , LOW );
  digitalWrite ( Y_2 , HIGH );
  digitalWrite ( G_2 , LOW );
  delay ( YellowTime );
  digitalWrite ( R_2 , HIGH );
  digitalWrite ( Y_2 , LOW );

//3. =====
  digitalWrite ( R_3 , LOW );
  digitalWrite ( Y_3 , LOW );
  digitalWrite ( G_3 , HIGH );

```

```
delay ( GreenTime );  
//3. =====  
digitalWrite ( R_3 , LOW );  
digitalWrite ( Y_3 , HIGH );  
digitalWrite ( G_3 , LOW );  
delay ( YellowTime );  
digitalWrite ( R_3 , HIGH );  
digitalWrite ( Y_3 , LOW );  
  
//4. =====  
digitalWrite ( R_4 , LOW );  
digitalWrite ( Y_4 , LOW );  
digitalWrite ( G_4 , HIGH );  
delay ( GreenTime );  
//4. =====  
digitalWrite ( R_4 , LOW );  
digitalWrite ( Y_4 , HIGH );  
digitalWrite ( G_4 , LOW );  
delay ( YellowTime );  
digitalWrite ( R_4 , HIGH );  
digitalWrite ( Y_4 , LOW );  
}
```

# Conclusion

Test the circuit in simulation and make it.

# 15. RPM Meter

---

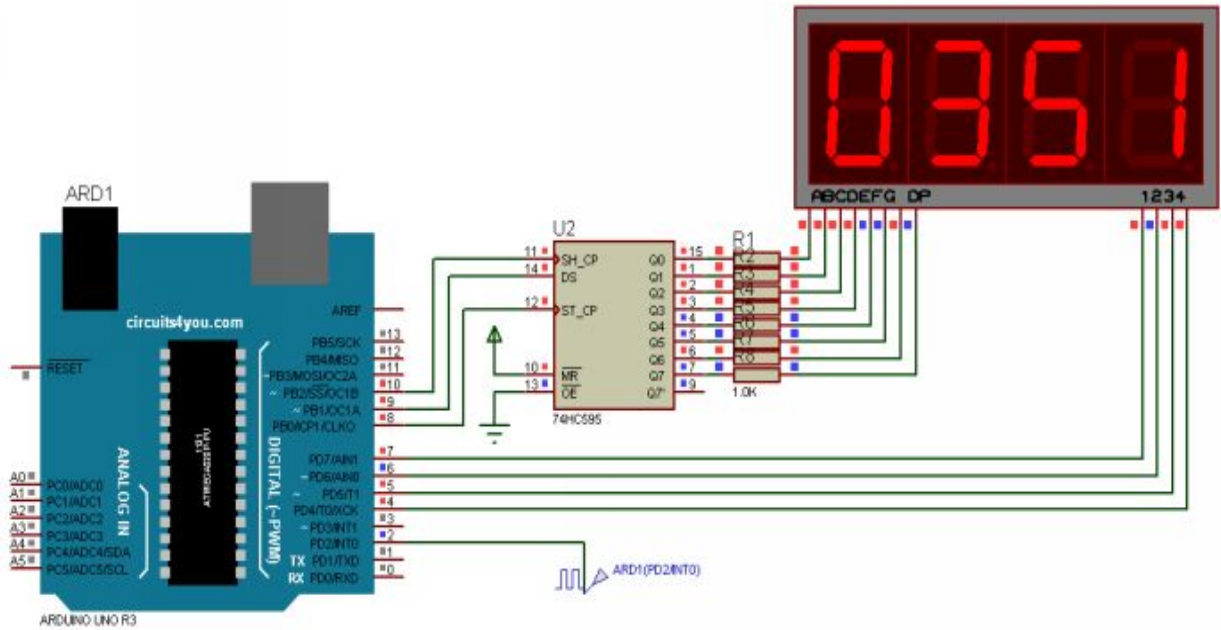
## 15.1 Introduction

A tachometer (revolution-counter, tach, rev-counter, RPM gauge) is an instrument measuring the rotation speed of a shaft or disk, as in a motor or other machine. The device usually displays the revolutions per minute (RPM) on a calibrated analogue dial, but digital displays are increasingly common. Here we make use of arduino to measure RPM.

## Components Required

1. Arduino Uno
2. ACS712-20A
3. 16x2 LCD
4. 1K, 9K Resistors
5. LM35 Temperature sensor

## 15.2 RPM Meter Circuit



**Figure 15.1: RPM Meter Circuit**

Connect pulse input to arduino port 2. You can use any proximity sensor or hall sensor to generate pulses from rotating wheel.



## 15.3 RPM Meter Arduino Code

```
/*
  Digital RPM Meter using 4-Digit 7-segment Display
  www.circuits4you.com
*/

#include <TimerOne.h>

#define MainPeriod 100
long previousMillis = 0; // will store last time of the cycle end
volatile unsigned long duration = 0; // accumulates pulse width
volatile unsigned int pulsecount = 0;
volatile unsigned long previousMicros = 0;

//Define 74HC595 Connections with arduino
const int Clock = 10;
const int Data = 9;
const int Latch = 8;

const int SEG0 = 7;
const int SEG1 = 6;
const int SEG2 = 5;
const int SEG3 = 4;

int cc = 0;
char Value [4];
const char SegData []=
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

//=====
//
//          Setup
//=====

void setup () {
  // initialize the digital pin as an output.
  attachInterrupt (0, myinhandler , RISING );
```

```

pinMode ( Data , OUTPUT );
pinMode ( Clock , OUTPUT );
pinMode ( Latch , OUTPUT );
pinMode ( SEG0 , OUTPUT );
pinMode ( SEG1 , OUTPUT );
pinMode ( SEG2 , OUTPUT );
pinMode ( SEG3 , OUTPUT );

    //Initialize Display Scanner
    cc =0;
    Timer1 . initialize (10000); // set a timer of length 100000
microseconds (or 0.1 sec - or 10Hz => the led will blink 5 times, 5 cycles
of on-and-off, per second)
    Timer1 . attachInterrupt ( timerIsr ); // attach the service routine here
}
//=====
//
//          Loop
//=====
void loop () {
    char rpm [4];
    int RPM ;
    unsigned long currentMillis = millis ();
    if ( currentMillis - previousMillis >= MainPeriod )
    {
        previousMillis = currentMillis ;
        // need to bufferize to avoid glitches
        unsigned long _duration = duration ;
        unsigned long _pulsecount = pulsecount ;
        duration = 0; // clear counters
        pulsecount = 0;
        float Freq = 1e6 / float( _duration ); //Duration is in uSecond so it is
1e6 / T
        Freq *= _pulsecount ; //calculate F
        //Convert Freq to RPM

```

```

    RPM = Freq * 60.0 * 1.0; //RPM = Freq * 60 * (Number of
pulses per revolution)
    //Display Freq on Segments
    sprintf ( rpm ,"%04d", RPM ); //We get ASCII array in Volt

    Value [0]= rpm [0] & 0x0F; //Anding with 0x0F to remove upper
nibble
    Value [1]= rpm [1] & 0x0F; //Ex. number 2 in ASCII is 0x32 we
want only 2
    Value [2]= rpm [2] & 0x0F;
    Value [3]= rpm [3] & 0x0F;
}

delay (200);
}

//=====
//
// Generates Bargraph
//=====
//=====

void DisplayDigit (char d )
{
    int i;

for( i =0; i <8; i ++ ) //Shift bit by bit data in shift register
{
    if(( d & 0x80)==0x80)
    {
        digitalWrite ( Data , HIGH );
    }
    else
    {
        digitalWrite ( Data , LOW );
    }
    d = d <<1;
}
}

```

```

    //Give Clock pulse
    digitalWrite ( Clock , LOW );
    digitalWrite ( Clock , HIGH );
}
//Latch the data
digitalWrite ( Latch , LOW );
digitalWrite ( Latch , HIGH );
}
//=====
//
//                                TIMER 1 OVERFLOW INTTERRUPT FOR
DISPALY
//=====
//=====
void timerIsr ()
{
    cc ++;
    if( cc ==5) //We have only 4 digits
    { cc =1;}
    Scanner ();
    TCNT0 =0xCC;
}

//=====
//=====
//                                SCAN DISPLAY FUNCTION
//=====
//=====
void Scanner ()
{
    switch ( cc ) //Depending on which digit is selcted give output
    {
        case 1:
            digitalWrite ( SEG3 , HIGH );
            DisplayDigit ( SegData [ Value [0]]);
            digitalWrite ( SEG0 , LOW );
            break;

```

```

case 2:
    digitalWrite ( SEG0 , HIGH );
    DisplayDigit ( SegData [ Value [1]]);
    digitalWrite ( SEG1 , LOW );
break;
case 3:
    digitalWrite ( SEG1 , HIGH );
    DisplayDigit ( SegData [ Value [2]]);
    digitalWrite ( SEG2 , LOW );
break;
case 4:
    digitalWrite ( SEG2 , HIGH );
    DisplayDigit ( SegData [ Value [3]]);
    digitalWrite ( SEG3 , LOW );
break;
}
}
//=====
=====
//           Interrupt Handler
//=====
=====
void myinhandler () // interrupt handler
{
    unsigned long currentMicros = micros ();
    duration += currentMicros - previousMicros ;
    previousMicros = currentMicros ;
    pulsecount ++;
}

```

## Conclusion

This project can be modified to display frequency and speed.

## 16. References

---

Here you get all simulation file links and hex files.

# Simulation and Hex Files

<http://circuits4you.com/arduino-pro-res1/>

**It is password Protected. Its Password:  
“arduprojvol1”**



**Donot forget to get this book, [Measurement Made simple with Arduino](#) , Available in pdf at [circuits4you.com](http://circuits4you.com) and Kindle format at Amazon**

2016

21



°C

pH

# Measurement Made Simple with Arduino

21 Different Measurements covers all  
physical and electrical parameter  
measurements with code and circuits

Manoj R. Thakur



# Contents of Measurement Made Simple with Arduino

	Title	Page Number
<b>1.</b>	<b><u>Introduction</u></b>	<b>1</b>
	1.1 Arduino Introduction	1
	1.2 Arduino IDE basics	1
	1.3 Arduino Programming	2
	1.4 Arduino Pin-outs	4
<b>2.</b>	<b><u>Voltage Measurement</u></b>	<b>6</b>
	2.1 DC Voltage	7
	2.2 AC Voltage	9
<b>3.</b>	<b>Current Measurement</b>	<b>11</b>
	3.1 DC Current Measurement ASC712	11
	3.2 DC Current Measurement using Shunt Resistor	14
	3.3 AC Current Measurement using ASC712	16
<b>4.</b>	<b>Resistance Measurement</b>	<b>19</b>
	4.1 Normal value resistance measurement	19
	4.2 Low value resistance measurement	21
<b>5.</b>	<b>Capacitance Measurement</b>	<b>23</b>
<b>6.</b>	<b>Frequency Measurement</b>	<b>27</b>
<b>7.</b>	<b>Light Measurement</b>	<b>30</b>
<b>8.</b>	<b>Temperature Measurement</b>	<b>32</b>

<b>9.</b>	Humidity Measurement	<b>34</b>
<b>10.</b>	Pressure Measurement	<b>37</b>
	10.1 Atmospheric pressure, Altitude measurement	38
	10.2 Air Pressure	43
<b>11.</b>	Rain measurement	<b>46</b>
<b>12.</b>	Soil Moisture Measurement	<b>48</b>
<b>13.</b>	pH Measurement	<b>50</b>
<b>14.</b>	Water Flow Measurement	<b>54</b>
<b>15.</b>	Distance Measurement	<b>56</b>
<b>16.</b>	Knock Detection	<b>59</b>
<b>17.</b>	4-20mAmp Industry Standard Measurement	<b>61</b>
<b>18.</b>	Water Level Measurement	<b>63</b>
<b>19.</b>	Rotary Position (Encoder)	<b>66</b>
<b>20.</b>	Color Detection	<b>69</b>
<b>21.</b>	Sound Level Measurement	<b>73</b>

Get it now at [circuits4you.com](http://circuits4you.com)

This Page intentionally left blank