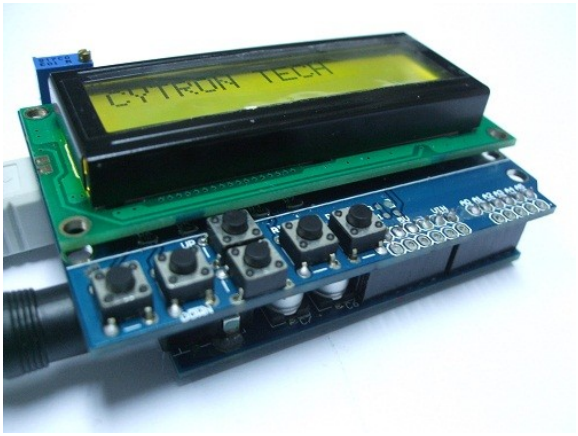
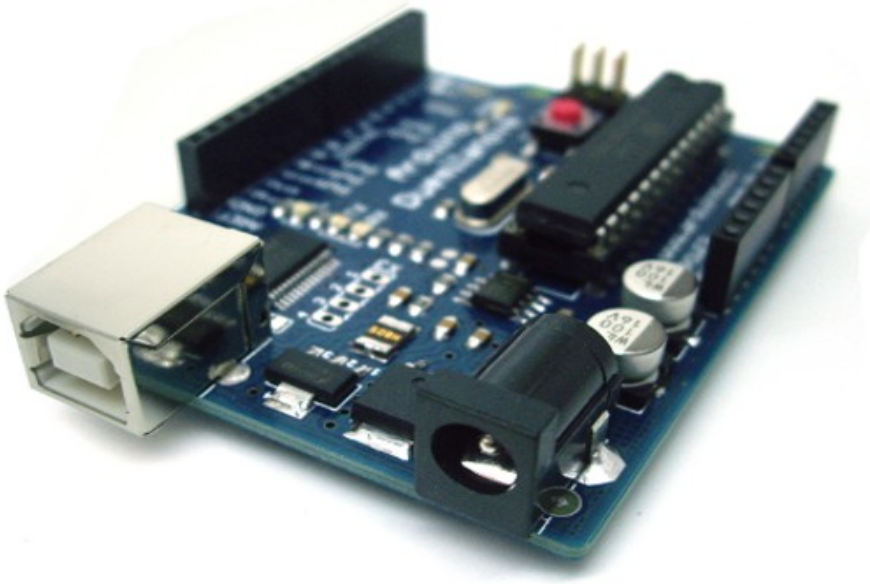
Cytron Technologies



FUN & LEARNING WITH ARDUINO PROJECTS



FUN & LEARNING WITH ARDUINO PROJECTS

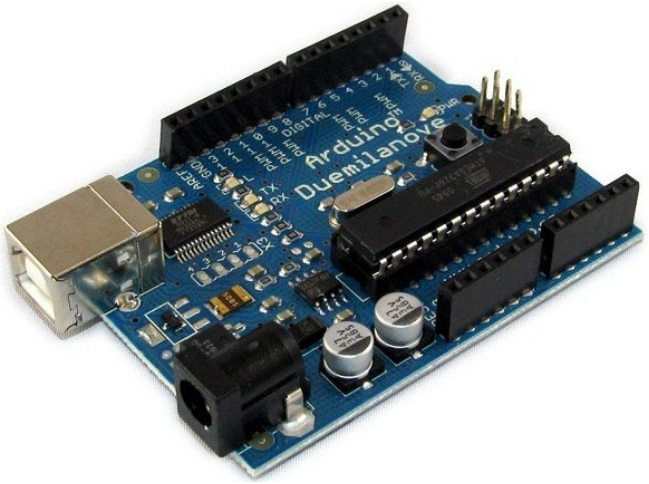
BEGINNER GUIDE

BY

Cytron Technologies

CONTENTS

INTRODUCTION



Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

WHY ARDUINO?

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.

- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino.

- Open source and extensible software- The Arduino software and is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. SImilarly, you can add AVR-C code directly into your Arduino programs if you

want to.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

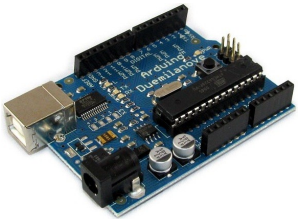
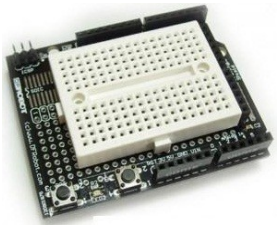
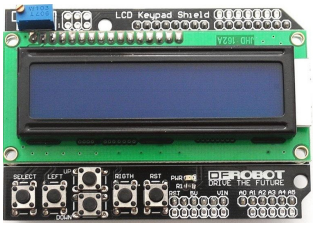
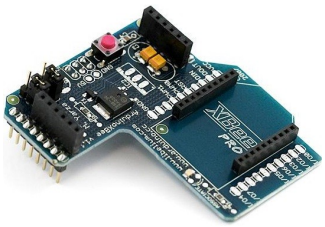
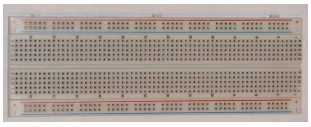
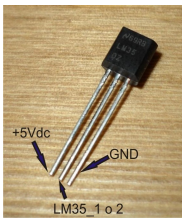
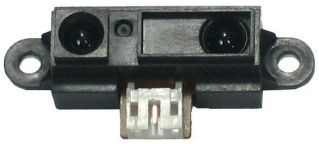
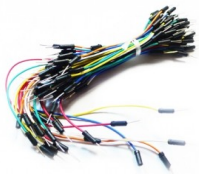
WHAT YOU WILL NEED

Before start any project, make sure you get yourself an Arduino Duemilanove and the necessary Arduino shield that need to use in the coming project. To get the Arduino Duemilanove and other extend shield please click here.

2nd, please download the Arduino software which are required to use to write code and upload to the I/O board here.

Lastly, to get improve of yourself, willingness of learning is always a matter. Coming project are design in simple and easy understanding to help you to get involve in microcontroller easily. It's never too hard to learn, just spend some time with it and you will like it.

COMPONENT REQUIRED

 ARDUINO DUEMILANOVE	 ARDUINO PROTOTYPING SHIELD	 ARDUINO LCD-KEYPAD SHIELD	 ARDUINO GRAPGIC LCD SHIELD
 ARDUINO INPUT SHIELD	 ARDUINO 2-AMP MOTOR DRIVER SHIELD	 ARDUINO-XBEE SHIELD	 SKXBEE
 RESISTOR	 BREADBOARD	 LEDS	 POTENTIALMETER
 LM35 TEMPRETURE SENSOR	 ULTRASONIS RANGE FINDER EZ1	 PIEZO BUZZER	 IR DISTANCE SENSOR
 SPG10 MICRO METAL GEARMOTOR	 RC SREVO MOTOR	 JUMPER WIRES	 DC POWER SUPPLY

OVERVIEW OF ARDUINO DUEMILANOVE

The Arduino Duemilanove ("2009") is a microcontroller board based on the Atmega168 (datasheet) or ATmega328(datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

"Duemilanove" means 2009 in Italian and is named after the year of its release. The Duemilanove is the latest in a series of USB Arduino boards; for a comparison with previous versions, see the index of Arduino boards.

SUMMARY

Microcontroller	ATmega168
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

POWER

The Arduino Duemilanove can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can

supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

- 3V3. A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.

- GND. Ground pins.

INPUT OUTPUT

Each of the 14 digital pins on the Duemilanove can be used as an input or output, using pinMode(), digitalWrite(), anddigitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.

- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Duemilanove has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- I^2C: analog input pins A4 (SDA) and A5 (SCL). Support I^2C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().

- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

GETTING STARTED

For coming project, we are going to use Arduino IDE to develop and upload the code to the Arduino Duemilanove. The Arduino IDE are supported to Window, Mac OS X and also Linux. Be sure to get the one who is suitable to your current using OS. To get this software, please click here.

ARDUINO IDE INSTALLATION

For Arduino software, installation are no needed. All we have to do is download and extract it out and save it anywhere you like.

Next, get yourself an USB-B type cable then plug in to the Arduino while the other end connect to the PC.

INSTALL THE USB DRIVERS

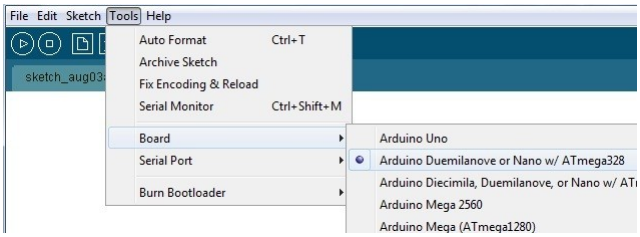
To make your Arduino Duemilanove start talking, you need to install the USB driver which is located inside the Arduino IDE you download (*drivers/FTDI USB Drivers*)

Double click this icon and the installation will executing. After that, your PC/Laptop will be automatically detected the driver install and your Arduino Duemilanove are free to use.

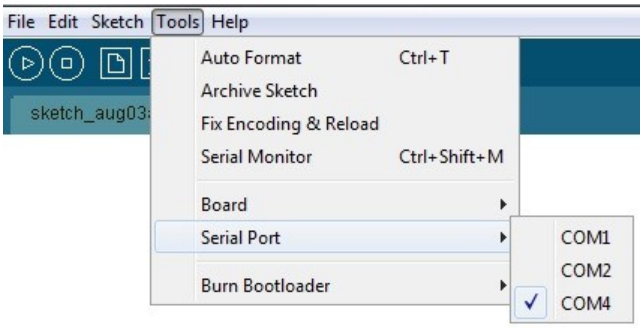
THE ARDUINO IDE SETUP

To start create a project, you may required to double click the arduino.exe which the icon is like the figure beside. This will allow you to open the Arduino IDE.

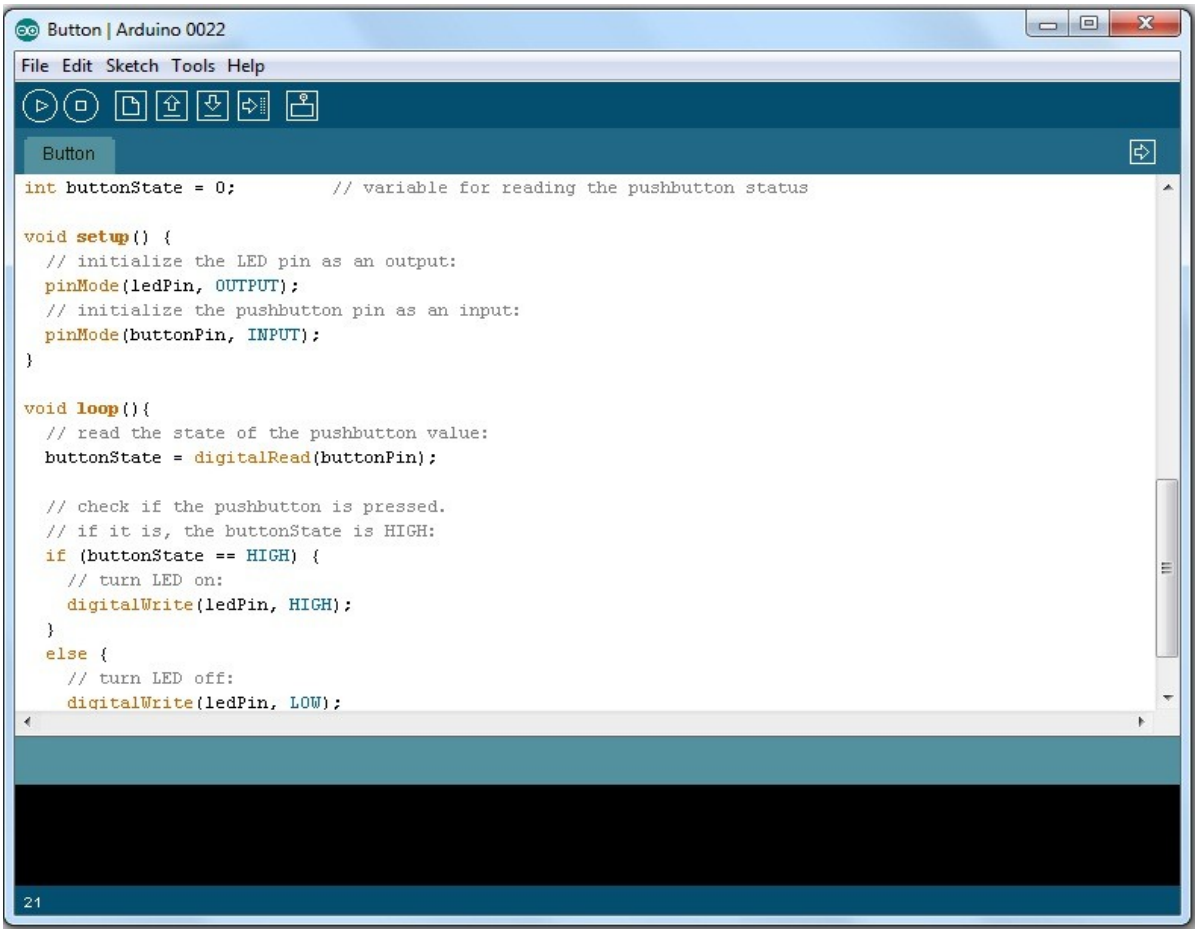
Next step, you may need to choose the Arduino board using for the 1st time such as figure below. Go to **Tools > Board > Arduinio Duemilanove or Nano w/ATmega 328**



Lastly, you may also choose the COM port for Arduino Duemilanove. Same step as above, go to **Tools > Serial Port > COM4** (*Depend on the PC*).



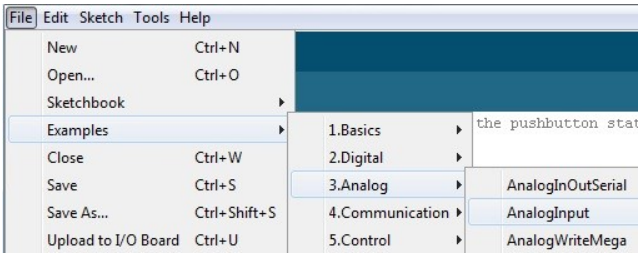
THE ARDUINO IDE



When you open open up the Arduino IDE, it will look similar to the figure above and it may be slightly different to those who are using Mac OS or Linux but the main function of this IDE are still the same.

Above there was a Toolbar which consist of 7 item which is which is start from the left (compile , stop, new, open, save, upload and serial monitor).

On the other hand, Arduino IDE also have prepare lots of example for user to refer the basic of the operating code. By clicking File > Example, there was lots of sample program to be explore.



Verify/ Compile **Stop** **New** **Open** **Save** **Upload** **Serial monitor**

Verify/Compile	Check the code for error
Stop	Stop the serial monitor, or un-highlights other button
New	Creates a new blank Sketch
Open	Show a list of Sketches in your Sketches book
Save	Save the current Sketch
Upload	Upload the current Sketch to Arduino
Serial Monitor	Display serial data being sent from Arduino

Verify/ Compile is use to check the code writing whether is correct or not before upload to Arduino.

Stop button are use to stop the operation of the serial monitor and also un-highlighted other selected button. While press stop in serial monitor, it wil stop the operation of it and user may take a 'snapshot' of the serial data they examine.

New button are use to create a complete new sketch for user to develop their new program.

Open button will present you a list of the sketches store in the sketch book as well as a list of example where allow user to try out.

Save button will save the sketch you developed code to the folder you required. The Arduino IDE will automatically create an folder and store the code.

Upload to I/O board will allow user to load their code onto the Arduino with the current sketch. You may need to make sure the correct board and port using(Tools menu) before uploading. You are encourge to save before uploadto avoid error which make your system hang or IDE to crash. It is advisable to verify/ compile the code before uploading to ensure the code have no errors.

Serial monitor is an very useful too to communicate with computer while it still can use to debugging your code. The monitor will display the data sending in and out from the computer to the Arduino.

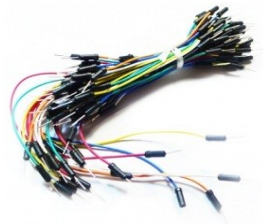
Well, don't get too worry about the IDE using as now you have already pick up some important step to deal with it. So, let's go to explore some projects code now!

PROJECT 0

LED DICE

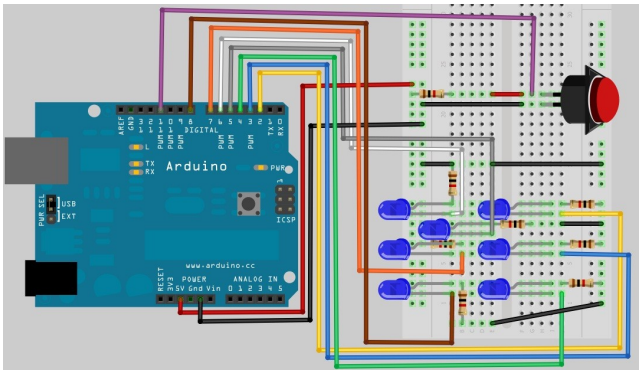
Dice, which all of us are normally will use it when playing a games especially monopoly. But have we ever head of LED dice? LED dice, one of the interesting project using LED except from LED blinking. In here, we going to experience on how to make an LED dice with a few LEDs interface with Arduino Duemilanove.

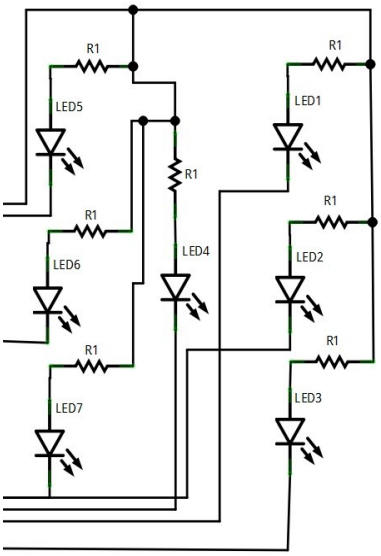
COMPONENT NEEDED

	ARDUINO-PROTOTYPING SHIELD
	JUMPER WIRE
	LEDs
	1K RESISTOR

CONNECTION

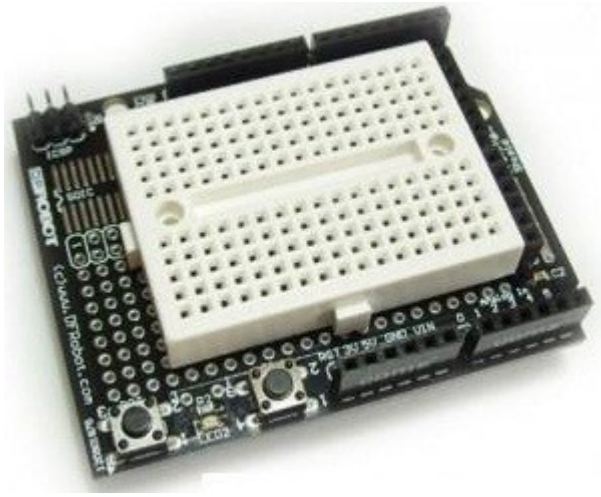
Firstly, we arrange the LEDs like the shape of the dice. Then connect the *LED1 to LED7 to digital port-2 until port-8*. Connect a pull-up switch to control the operation of the LED dice which is connected to digital port 11. Every LEDs are connected with an 1K resistor to prevent over current LED which will make the LED spoil.

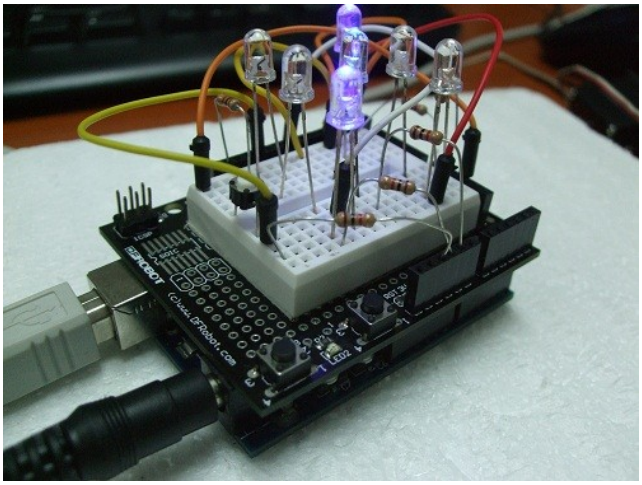




ADDITIONAL INFORMATION

ARDUINO PROTOTYPING SHIELD



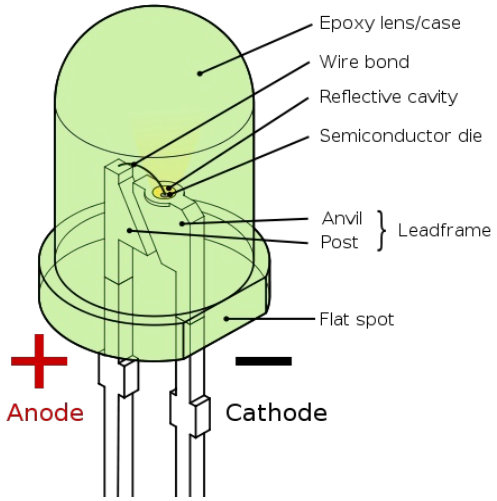


This is a design for an open-source prototyping shield for Arduino NG/Diecimila. It has tons of cool features, to make prototyping on your Arduino easy. Please refer to Arduino Prototyping Shield schematic for detail connection.

Features:

- Reset button up top
- ICSP header
- Lots of GND and +5V rails
- DIP prototyping area makes it easy to add more chips.
- SOIC prototyping area above USB jack for up to 14-pin SOIC chip, narrow medium or wide package.
- A 'mini' breadboard included
- Extra 6mm button
- Compatible with: Arduino Duemilanove, Arduino NG, Arduino Diecimila.

LIGHT EMITTING DIODE



There are few method to determine the anode and cathode side of the LED such as :

1. The flat spot on the lens/case of the LED is cathode.
2. The short lead (or leg) is cathode.
3. The flag symbol inside the lens is cathode.

For further information about LED, please refer to this webpage.

BREADBOARD

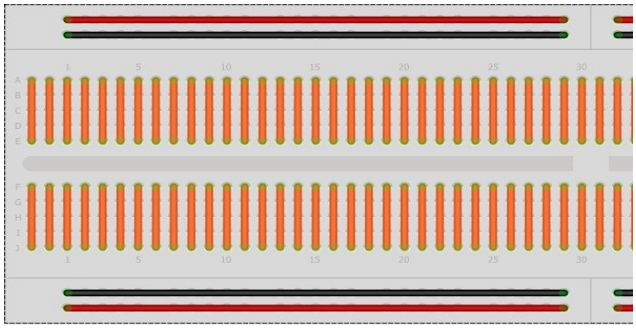
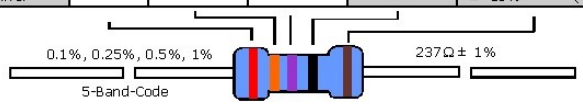


Figure above have shows that the connection of half of the breadboard. The RED and BLACK colour line are usually been connect to power supply (VDD) and ground (GND). They are connected all the way from the beginning to the end but they did not connect to each other.

Besides that, the orange colour line are the part that we usually use to place the electronic component. They are connected in a straight line.
There is a gap in the middle of the breadboard which are not connected to anything. This allow you to put integrated circuit across the gap and have each pin of the chip go to the different set of holes and therefore a different rial.

RESISTOR COLOUR TABLE

COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE	
Black	0	0	0	1Ω		
Brown	1	1	1	10Ω	± 1%	(F)
Red	2	2	2	100Ω	± 2%	(G)
Orange	3	3	3	1KΩ		
Yellow	4	4	4	10KΩ		
Green	5	5	5	100KΩ	±0.5%	(D)
Blue	6	6	6	1MΩ	±0.25%	(C)
Violet	7	7	7	10MΩ	±0.10%	(B)
Grey	8	8	8		±0.05%	
White	9	9	9			
Gold				0.1	± 5%	(J)
Silver				0.01	± 10%	(K)



Above are the example of resistor with the colour

code example. For more detail, refer to this webpage.

CODE OVERVIEW

```
//LEDs
const int pinLeds1 = 2;
const int pinLeds2 = 3;
const int pinLeds3 = 4;
const int pinLeds4 = 5;
const int pinLeds5 = 6;
const int pinLeds6 = 7;
const int pinLeds7 = 8;
const int buttonPin = 11;
```

const int pinLeds1 = 2;
Define Arduino pin 2 as pinLeds1 and so on for total of 7 LEDs.

```
// ramdom number will store in ran
unsigned char ran = 0;

// Push Button
int buttonState;
```

unsigned char ran = 0;
Define ran as 0 while defining the maximum storage as a byte.

Int buttonState;
Define the buttonState as an integer.

```
void setup ()
{
  pinMode (pinLeds1, OUTPUT);
  pinMode (pinLeds2, OUTPUT);
  pinMode (pinLeds3, OUTPUT);
  pinMode (pinLeds4, OUTPUT);
  pinMode (buttonPin, INPUT);
  pinMode (pinLeds5, OUTPUT);
  pinMode (pinLeds6, OUTPUT);
  pinMode (pinLeds7, OUTPUT);
  randomSeed(analogRead(0));
}
```

void **setup** ()
In here, we use to declare the pin using whether is Input or output pin.

pinMode (pinLeds1, OUTPUT);
Declare the pin as an OUTPUT pin.

pinMode (buttonPin, INPUT);
Declare the pin as an INPUT pin.

randomSeed (analogRead(0));
Initializes the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence.

```
void loop()
{
  buttonState = digitalRead(buttonPin);
  while (buttonState == LOW){
    ran = random(1, 7);

  // now arduino will toss the die
   if (ran == 1){
      digitalWrite (pinLeds4, HIGH);
      delay(50);
      while (digitalRead(buttonPin) == HIGH);
      led_clear();
   }
```

void **loop**()
The program under void loop will be loop forever.

ButtonState = digitalRead(buttonPin);
Read the input value from buttonPin = 11, then store the result in ButtonState.

while (buttonState == LOW){
ran = random(1,7)};
Loop forever until the buttonState,LOW(0), is detected. After that execute the random number choosing between 1 to 7.

if (ran == 1)
If ran is 1 then execute the following program.

digitalWrite (pinLeds4, HIGH)
Digitally write the pinLeds4 to High(1) the digital pin 4.

delay(50);
Delay for 50ms.

while(digitalRead(buttonPin) == HIGH);
Check if the button pin is release. If yes, then stay there until is press again.

led_clear();
Call and execute the led_clear program.

```
void led_clear (void)
{
  digitalWrite (pinLeds1, LOW);
  digitalWrite (pinLeds2, LOW);
  digitalWrite (pinLeds3, LOW);
  digitalWrite (pinLeds4, LOW);
  digitalWrite (pinLeds5, LOW);
  digitalWrite (pinLeds6, LOW);
  digitalWrite (pinLeds7, LOW);
}
```

This subroutine are operate to LOW(0) or off all the LED pins by digitally write the pinLeds from 1 to 7 to LOW(0).

References:

1. http://en.wikipedia.org/wiki/Light-emitting_diode

2. http://www.elexp.com/t_resist.htm

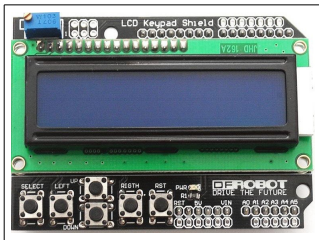
3. http://www.cytron.com.my/usr_attachment/Arduino-proto.pdf

PROJECT 1

"Hello World" ON LCD

Arduino LCD Keypad Shield or navigation shield which come with 6 momentary push button for menu navigation and also a 2x16 LCD. It only required to plug and use with Arduino main board and there no soldering or fly-wiring are required.

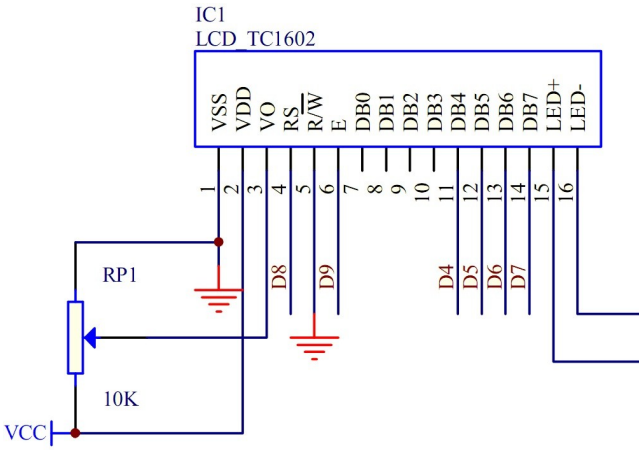
COMPONENT NEEDED

	ARDUINO LCD-KEYPAD SHIELD

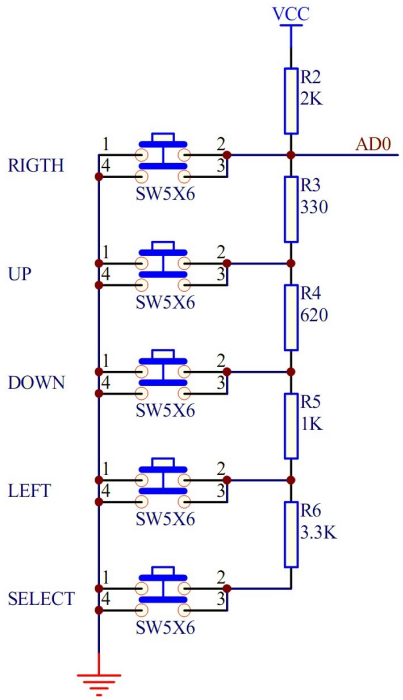
CONNECTION

For the LCD Keypad Shield, it use total of 6-pins to control the LCD display which is pin-4,5,6,7,8,9. For LCD data, it use pin-4,5,6,7, while for the RS and Enable pin, it use pin-8 and 9. The Arduino-LCD Keypad Shield are only required to plug into the Arduino main board and there was no soldering are required such as shown in figure below.

Arduino LCD 2x16 Schematic



Arduino Keypad Schematic



ADDITIONAL INFORMATION

The Arduino-LCD Keypad Shield are operate in 4-bits mode. To ease user, Arduino Team have prepared the <LiquidCrystal.h> file which we only require to modified and change the pin using only. For more information please check the Arduino LCD keypad Shield schematic.

For more example of the Arduino LCD Keypad coding, please refer to this webpage.

CODE OVERVIEW

```
#include <LiquidCrystal.h>

/*
  The circuit:
 * LCD RS pin to digital pin 8
 * LCD Enable pin to digital pin 9
 * LCD D4 pin to digital pin 4
 * LCD D5 pin to digital pin 5
 * LCD D6 pin to digital pin 6
 * LCD D7 pin to digital pin 7
 * LCD R/W pin to ground
*/

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

#include <LiquidCrystal.h>
Include the LiquidCrystal header file which is already prepared inside the Arduino libraries.

LiquidCrystal.lcd(8, 9, 4, 5, 6, 7);
In here, we set the pin using for LCD 4-bit mode by refer to Arduino Liquid Crystal webpage which is LiquidCrystal(rs, enable, d4, d5, d6, d7).

```
int analogPin = A0;
int adc_key_old;
int adc_key_in;
int NUM_KEYS = 5;
int key=-1;
int adc_key_val[5] ={30, 150, 360, 535, 760 };

char msgs[5][15] = {"Right Key OK ",
                    "Up Key OK    ",
                    "Down Key OK  ",
                    "Left Key OK  ",
                    "Select Key OK"};
```

Int analogPin = A0;
Define the A0 as analogPin while assign a space to it as an integer.

Int adc_key_val[5] = {30 , 150 , 360, 535, 760};
Define the adc_key_val from 0 to 4 which contain the number inside the bracket. E.g: adc_key_val[1] = 150;

char msgs [5][15] = {“Right Key OK”,

```
"Up Key OK    ",
"Down Key OK",
"Left Key OK    ",
"Select Key OK"};
```

Define the each ASCII value in char which content of total 5 string and there was 15 ASCII value for each string.

```
void setup ()
{
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("  CYTRON TECH.");
  lcd.setCursor(0,1);
  lcd.print(" Eg. LCD Shield");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Hello World");
  lcd.setCursor(0,1);
  lcd.print("Pls press any");
  adc_key_old = analogRead(analogPin);
}
```

lcd.begin(16,2);
Specified the dimension (width and height)of the LCD display.

lcd.clear();
Clear the screen of the LCD display.

lcd.print(" CYTRON TECH.");
Send the ASCII code to the LCD display according to the text enter.

lcd.setCursor(0,1);
Position the LCD cursor,that is set the location at which subsequent text written to the LCD will be displayed.

```
void loop()
{
  adc_key_in = analogRead(analogPin);
  adc_key_in = get_key(adc_key_in);
  lcd.setCursor(0, 1);
  lcd.print(msgs[adc_key_in]);
}
```

adc_key_in = analogRead(analogPin);
Read the value from the specified pin (analogPin) and store the value in the adc_key_in.

adc_key_in = get_key(adc_key_in);
Send the adc_key_in value to the get_key subroutine.

lcd.print(msgs[adc_key_in]);
Print the text on the msgs[] to the LCD display according to the adc_key_in.

```
int get_key(unsigned int input)
{
        int k;

        for (k = 0; k < NUM_KEYS; k++)
        {
                if (input < adc_key_val[k])
                {

    return k;
        }
        }

    if (k >= NUM_KEYS)
        k = -1;     // No valid key pressed

    return k;
}
```

for(k = 0; k < NUM_KEYS; k++)
Loop from k=0 to k=4.

If (input < adc_key_val[k])
Check of the value from the analogPin are less than adc_key_val[k].

return k;
Replace the original k value with the number of k

from 0 to 4.

```
If (k >= NUM_KEYS)
   k = -1;
```

If detected the k value are large, assign k = -1.

References:

1. http://arduino.cc/en/Tutorial/HomePage

2. http://www.cytron.com.my/usr_attachment/LCDKeypad%20Shield%20SCH.pdf

PROJECT 2

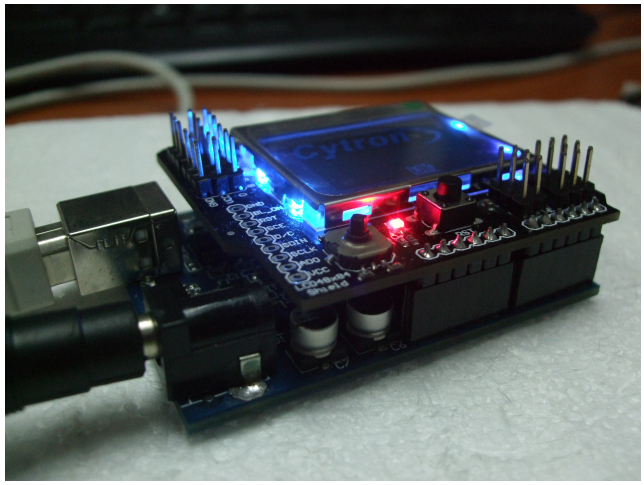
GRAPHIC LCD DISPLAY

Arduino Duemilanove have come with a Graphic LCD shield which allow allow user to play with more pixel besides LCD. This shield have 48x84 pixel resolution. If your project need more information to display, this LCD shield can apparently meet your needs. It is able to display English, Chinese, even images.

COMPONENT NEEDED

	ARDUINO -GRAPHIC LCD SHIELD

CONNECTION



Referring to the Figure above, the GLCD are only required to plug into Arduino Duemilanove then put in the coding, the shield are ready to display.

ADDITIONAL INFORMATION

For Arduino- GLD shield, It also integrates a 5 DOF joystick. The shield has 6 Digital IO and 5 Analog IO. Besides that, we may control the backlight by just use digitalWrite(7, HIGH/LOW) to turn it on and off. Don't forget to put pinMode(7, OUTPUT) in void setup();.

Well, reading the joystick position is accomplished via analogRead(0);. It returns the following values as such:

- Up – 505
- Down – 0
- Left – 740
- Right – 330
- pressed in – 144
- Idle (no action) – 1023

CODE OVERVIEW

```
#include <LCD4884.h>
#include "cytron.h"

int adc_key_in;
int joy_stick = 0;
int  adc_key_val[5] ={50, 200, 400, 600, 800 };

#define LCD_Backlight 7
#define NUM_KEYS 5
```

int adc_key_in;

Define the adc_key_in as a storage space of integer.

int joy_stick = 0;
Define the joy_stick I/O as analog port 0.

int adc_key_val[5] = {50,200,400,600,800};
Initial the joy stick ADC value.

#define LCD_Backlight 7
Define the LCD_Backlight pin as ditial pin 7.

#define NUM_KEYS 5
Define the total number of key is 5.

```
void setup()
{
lcd.LCD_init();
lcd.LCD_clear();

pinMode(LCD_Backlight, OUTPUT);
}
```

lcd.LCD_init();
Graphic LCD Initialize.

Lcd.LCD_clear();
Clear the GLCD screen.

pinMode(LCD_Backlight, OUTPUT);
Make the LCD_Backlight pin as output.

```
lcd.LCD_draw_bmp_pixel(2,1, cytron,80,24);
delay(2000);
lcd.LCD_clear();
```

lcd.LCD_draw_bmp_pixel(2,1,cytron,80,24);
Display the picture in Bitmap format in (2nd row, 1st column, cytron logo, picture's width, picture's height).

```
for (int a=0; a<6; a++)
  {
  lcd.LCD_write_string(0,a,"01234567980123", MENU_NORMAL);
  delay(200);
  }
```

lcd.LCD_write_string(0,a,”01234567890123”,MENU_NORMAL);
Write a string of number in every row.

```
lcd.LCD_write_string_big(0, 0, "012345", MENU_NORMAL);
lcd.LCD_write_string_big(0, 3, "-+-+-+", MENU_NORMAL);
delay(1000);
lcd.LCD_clear();
```

lcd.LCD_write_string_big(0,0,”012345”,MENU_NORMAL);
Write a string of number in big font size.

References:

1. http://tronixstuff.wordpress.com/2011/03/12/the-dfrobot-lcd4884-lcd-shield/

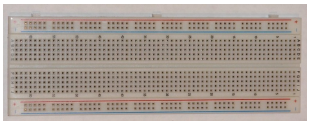
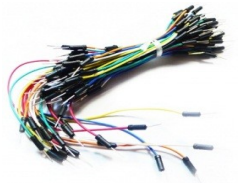
2. http://www.dfrobot.com/wiki/index.php?title=LCD4884_Shield_Fro_Arduino_(SKU:DFR0092)

PROJECT 3

LED CHASER

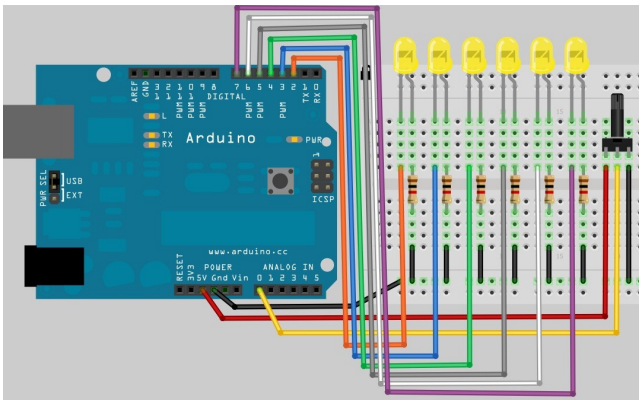
Dice, which all of us are normally will use it when playing a games especially monopoly. But have we ever head of LED dice? LED dice, one of the interesting project using LED except from LED blinking. In here, we going to experience on how to make an LED dice with a few LEDs interface with Arduino Duemilanove.

COMPONENT NEEDED

	BREADBOARD
	JUMPER WIRE
	LEDs
	1K RESISTOR
	5K POTENTIALMETER

CONNECTION

Referring to the figure below, there are total 6 LEDs which is anode are connected to *digital port (2,3,4,5,6,7)*. While other end are connected to 1K resistor and to ground *GND*. Next, connect the potentialmeter which one end RED wire to *VDD(5V)*, the other end **BLACK** wire connected to ground *GND* while the middle pin YELLOW wire is connect to *analog input A0*.



ADDITIONAL INFORMATION

Potentiometer Basics

A potentiometer is a type of variable resistor, made so that it can be easily adjusted. A resistor is an electronic component whose function is to resist the flow of electricity. The more resistance, the slower the electric current flows. Because a potentiometer allows a user to adjust the resistance, it can be used to adjust a circuit while it is running.

Potentiometer Structure

A potentiometer has three contacts - two fixed contacts on either end and a movable one called a wiper. As the wiper moves closer to one fixed contact, it moves further away from the other, decreasing the resistance with the closer one and increasing it with the further one. Some potentiometers, called rheostats only have two contacts - a fixed one and a wiper. In both cases, the position of the wiper is usually adjusted by turning a knob or moving a slider.

CODE OVERVIEW

```
//LEDs
byte pinLeds[] = {2,3,4,5,6,7};
```

byte pinLeds[] = {2,3,4,5,6,7};

Define the pin using in array . Which in this project, the pin using are pin 2,3,4,5,6 and pin 7.

```
void setup ()
{
unsigned char x;
  for (x=0 ; x<6 ; x++)
    {
      pinMode(pinLeds[x],OUTPUT);
    }
}
```

for (x=0 ; x<6 ; x++)
{
pinMode (pinLeds[x],OUTPUT)
}

Loop from x=0 until x=5 and assign the pin to output.
E.g: When x = 3. pinMode(pinLeds[3],OUTPUT);

```
void loop()
{
     delay_period = analogRead(analogPin);

      // turn on the current LED
      digitalWrite(pinLeds[currentLED], HIGH);
      delay(delay_period);
      digitalWrite (pinLeds[currentLED], LOW);

      // increment by the direction value
      currentLED += direction;
      // change direction if we reach the end
      if (currentLED == 5) {direction = -1;}
      if (currentLED == 0) {direction = 1;}
}
```

delay_period = analogRead(analogPin);

Read the ADC value from analogPin,A0, by assigning the code analogRead. After that store the value in delay_period. The ADC for Arduino microcontroller (ATMEGA398P) containing 10 bits of ADC so the output value may between 0 to 1022 which will be taken as an delay value.

CurrentLED += direction;
Decide the LED direction by assigning the value by +1 or -1.

if (currentLED == 5) {direction = -1;}
if (currentLED == 0) {direction = 1;}
Check the LED location and decide whether is 5 or 0. If is 5, then assign -1. If is 0, then add 1.

References:

1. http://en.wikipedia.org/wiki/Potentiometer

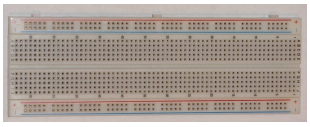
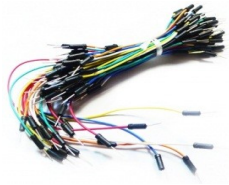
2. http://www.wisegeek.com/what-is-a-potentiometer.htm

PROJECT 4

KNIGHT RIDER LIGHT BAR

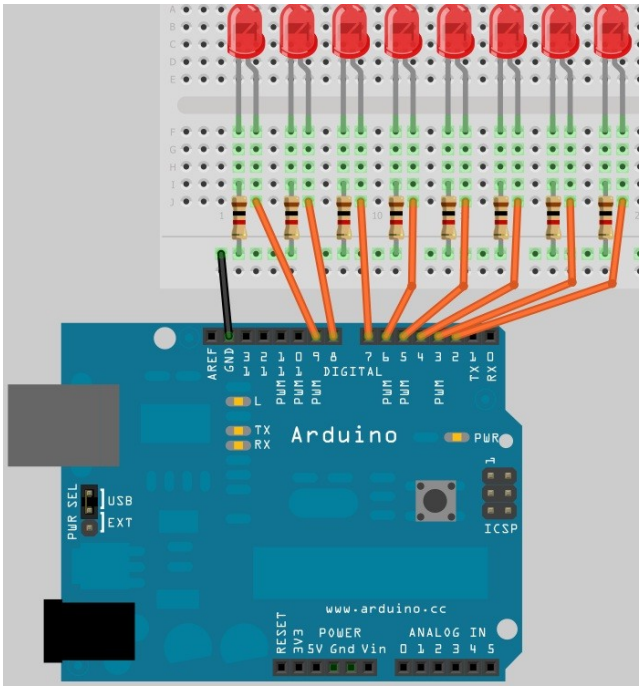
Ever watch knight rider movie? The car (KITT) have a very cool light bar in front and where most people like to have one of it. Here, we are going to make one of it using Arduino Duemilanove interfacing with 8 LEDs. Enjoy..

COMPONENT NEEDED

	BREADBOARD
	JUMPER WIRE
	LEDs
	1K RESISTOR

CONNECTION

There are total of 8 LEDs using which he anode are all connected to *digital port (2,3,4,5,6,7,8,9)* which ORANGE wires. On other hand, the cathode of LEDs are connected to 1K resistor then go to ground *(GND)*.



ADDITIONAL INFORMATION

For this project, user are require to download a new library file for Arduino here. This is a SoftPWM library which use to generate simple PWM using timer 2.

To insert the file into Arduino library, please follow the following steps.

1. After download, extract the file and save it at your arduino-0022\libraries.

2. Restart your arduino.exe (if you are currently using it).

CODE OVERVIEW

```
#include <SoftPWM.h>

#define DELAY 40

int leds[8] = {2, 3, 4, 5, 6, 7, 8, 9};
```

#include <SoftPWM.h>
Include the header file always at the beginning of the each program.

Int leds[8] = {2,3,4,5,6,7,8,9};
Define the LEDs pin using on each digital port.

```
void setup()
{
  SoftPWMBegin();

  for (int i = 0; i < 8; i++)
    SoftPWMSet(leds[i], 0);

  SoftPWMSetFadeTime(ALL, 30, 200);
}
```

SoftPWMBegin();
Initializes the library - sets up the timer and other tasks.

SoftPWMSet (leds[i], 0);
SoftPWMSet(***pin***, ***value***)
~ ***pin*** is the output pin.
~ ***Value*** is a value between 0 and 255 (inclusive).

SoftPWMSetFadeTime (ALL, 30, 200);
SoftPWMSetFadeTime(*pin*,*fadeUpTime*,*fadeDownTime*)
~ ***pin*** *i*s the output pin.
~ ***Fadeuptime*** *i*s the time in milliseconds that it will take the channel to fade from 0 to 255.
~ Range: 0 to 4000
~ ***fadedowntime*** is the time in milliseconds that it will take the channel to fade from 255 to 0.
~ Range: 0 to 4000

```
for (i = 0; i < 3; i++)
{
  SoftPWMSet(leds[i+1], 255);
  SoftPWMSet(leds[6-i], 255);
  SoftPWMSet(leds[i], 0);
  SoftPWMSet(leds[7-i], 0);
  delay(DELAY);
}

delay(250);
```

SoftPWMSet (leds[i+1], 255);
Set the desire pin to HIGH(1) and LOW(0)
E.g: if i = 0, (leds[i + 1] = 1, 255) (pin-3, ON)
(leds [6 – i] = 6, 255) (pin-6, ON)
(leds [i] , 0) (pin-0 , OFF)
(leds [7-i], 0) (pin-7, OFF)

References:

1. http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1276737123

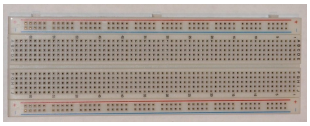
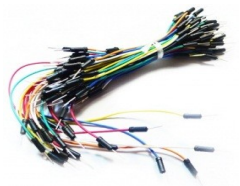
2. http://code.google.com/p/rogue-code/wiki/SoftPWMLibraryDocumentation#Description

PROJECT 5

RC SERVO MOTOR POSITION

Nowadays, most hobbyist like to use RC servo motor to control their robot or others than that which need a precise position while a cheapest price. For this project, we are going to discover how to control 2 RC servo motor together using Arduino Duemilanove with Arduino Input Shield.

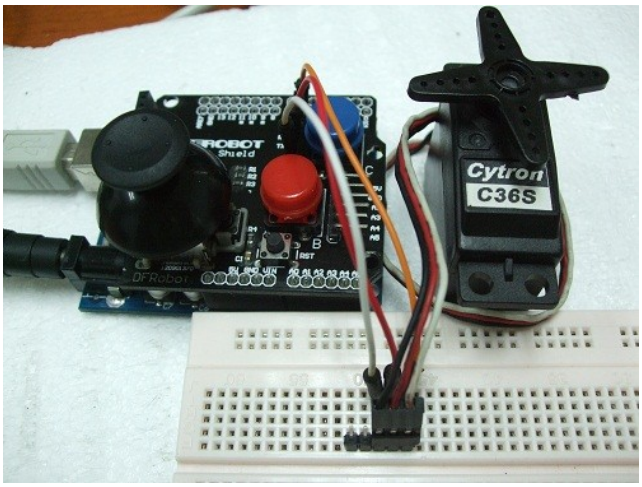
COMPONENT NEEDED

	BREADBOARD
	RC SERVO MOTOR
	ARDUINO INPUT SHIELD
	JUMPER WIRE

CONNECTION

Servo motors have three wires: power, ground, and signal. The power wire is typically RED, and should be connected to the *5V* pin on the Arduino board. The ground wire is typically **BLACK** or BROWN and should be connected to a *GND* pin on the Arduino board. The signal pin is typically YELLOW, ORANGE or **WHITE** and should be connected to a *digital pin 6* on the Arduino board.

Note servos draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the +5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together.

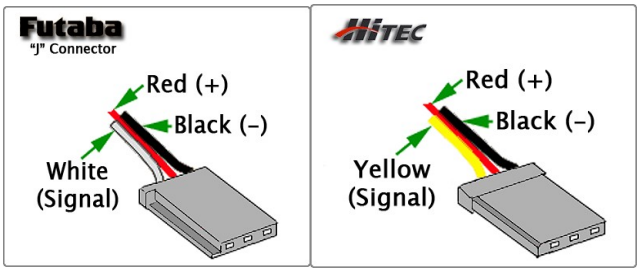


ADDITIONAL INFORMATION

ARDUINO RC SERVO LIBRARY

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds. The Servo library supports up to 12 motors on most Arduino boards.

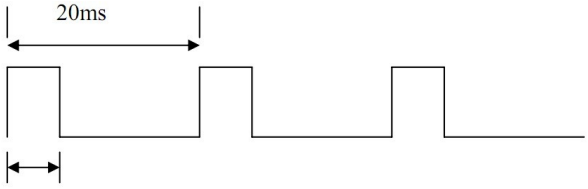
WIRE CONNECTION



Above are the picture showing that the different colour of wire using in different brand of servo motor. For more information, please refer to this servo wiring information.

HOW RC SERVO WORK

To control the RC servo position, we have to send a 20ms plus to the RC servo with a 0.5ms to 2.5ms pulse width to determine the servo position such as the figure showing below. For detail, please refer to this RC servo datasheet.



Pulse Width (0.5ms-2.5ms)

CODE OVERVIEW

```
#include <Servo.h>

Servo servo1;

int potpin_x = A0;
int potpin_y = A1;
const int joy_pushbutton = 5;
const int pushbuttonB = 3;
const int pushbuttonC = 4;

int val;
int mode = 1;
int current_joy_mode;
```

Servo servo1;
Define the servo name as servo1.

Const int joy_pushbutton = 5;
Set the joy_pushbutton as pin 5 and also as the constant value.

Int val;
Define and store the value of RC servo motor.

Int mode = 1;
Define and store number 1 inside at the beginning.

Int current_joy_mode;
Define and store the mode of the joystick.

```
void setup ()
{
  servo1.attach(6);
  pinMode(joy_pushbutton,INPUT);
  pinMode(pushbuttonB,INPUT);
  pinMode(pushbuttonC,INPUT);
}
```

servo1.attach(6);
Define that the servo1 are attach to the digital pin-6.

```
switch(mode)
  {
  case 1: val = analogRead(potpin_x);
          val = map(val, 0, 1023, 0, 179);
          servo1.write(val);
          delay(15);
          break;
```

val = map(val, 0 , 1023, 0, 179);
Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

servo1.write(val);
Apply the value from 'val' to the servo1 which contain of any number within 0 to 179.

break;
Function to break out from the current loop.

```
case 3: for(val = 0; val < 180; val += 1)
          {
          servo1.write(val);
          delay(15);
          }
        for(val = 180; val > 1; val -= 1)
          {
          servo1.write(val);
          delay(15);
          }
        mode =  current_joy_mode;
        break;
```

for (val = 0; val < 180; val +=1)
Make the val increase 1 each step from 0 to 179.

mode = current_joy_mode;
Restore back to mode of he joystick whether in mode 1 or 2.

```
if ((digitalRead(joy_pushbutton)) == LOW) {
  while((digitalRead(joy_pushbutton)) == LOW);
  if (++mode > 2)
      mode = 1;
       current_joy_mode = mode;
}

if ((digitalRead(pushbuttonB)) == LOW) {
  while((digitalRead(pushbuttonB)) == LOW);
    mode = 3;
    }

if ((digitalRead(pushbuttonC)) == LOW) {
  while((digitalRead(pushbuttonC)) == LOW);
    mode = 4;
    }
```

if ((digitalRead(joy_pushbutton)) == LOW)
Check if the joystick pushbutton is press.

while((digitalRead(joy_pushbutton)) == LOW);
Wait for debounce if the button is still pressing.

current_joy_mode = mode;
Store the current joystick mode.

References:

1. http://www.cytron.com.my/usr_attachment/RC_Servo_User_Manual.pdf

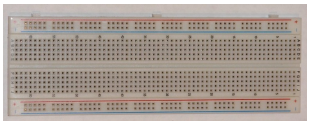
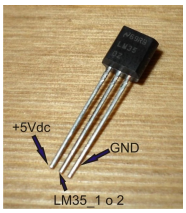
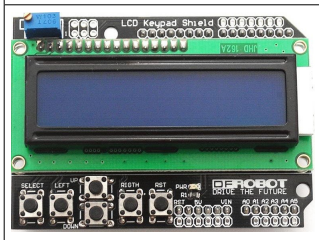
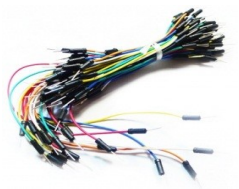
2. http://www.fatlion.com/sailplanes/servos.html

PROJECT 6

TEMPERATURE SENSOR TO LCD DISPLAY

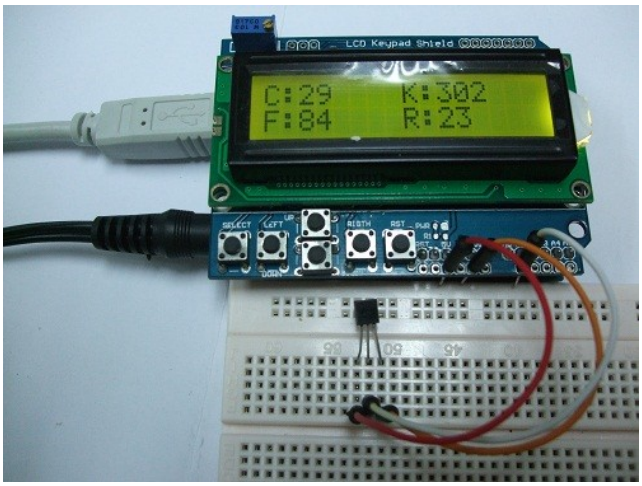
Arduino Duemilanova have come with 6 Analog to Digital Converter I/O with 10-bits resolution. In here, we going to explore on how to interface LM35 temperature sensor to Arduino and Display the output on Arduino LCD-Keypad shield.

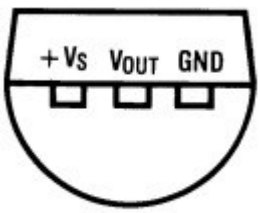
COMPONENT NEEDED

	BREADBOARD
	LM35 TEMPRETURE SENSOR
	ARDUINO LCD-KEYPAD SHIELD
	JUMPER WIRE

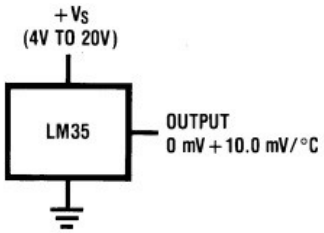
CONNECTION

Connect LM35 RED wire(VDD) to +5V, ORANGE wire (GND) to ground and lastly the middle pin **WHITE** wire of LM35 to Arduino analog port A1.





BOTTOM VIEW



Basic Centigrade Temperature Sensor (+2˚C to +150˚C)

ADDITIONAL INFORMATION

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4˚C at room temperature and ±3⁄4˚C over a full −55 to +150˚C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range.

Features

~ Calibrated directly in ° Celsius (Centigrade)
~ Linear + 10.0 mV/°C scale factor
~ 0.5°C accuracy guaranteeable (at +25°C)
~ Rated for full −55° to +150°C range
~ Suitable for remote applications
~ Low cost due to wafer-level trimming
~ Operates from 4 to 30 volts
~ Less than 60 µA current drain
~ Low self-heating, 0.08°C in still air
~ Nonlinearity only ±1⁄4°C typical
~ Low impedance output, 0.1 W for 1 mA load.

CODE OVERVIEW

```
temp = analogRead(temp_analogPin);

celsius = temp/2;
lcd.setCursor(2, 0);
lcd.print(celsius,DEC);

kelvin = celsius + 273;
lcd.setCursor(10, 0);
lcd.print(kelvin,DEC);

fehrenheit = (celsius*18)+320;
fehrenheit = fehrenheit/10;
lcd.setCursor(2, 1);
lcd.print(fehrenheit,DEC);

reaumur = celsius*8;
reaumur = reaumur/10;
lcd.setCursor(10, 1);;
lcd.print(reaumur,DEC);

delay(1000);
temp_value_clear();
```

temp = analogRead(temp_analogPin);
Read analog pin from LM35 temperature sensor input pin and store at temp.

celsius = temp/2;
LM35 are design to give the exact value of temperature in Celsius. To get the celsius value, the ADC value are required to divided by 2 .

kelvin = celsius + 273;
Kelvin formula are (Celsius + 273.15). But for microcontroller, it will not read the result after the DOT.

fehrenheit = (celsius*18)+320;
fehrenheit = fehrenheit/10;
Fahrenheit formula are ([celsuis*1.8]+32). In the coding, the formula have been times will 10 times to get the exact result.

reaumur = celsius*8;
reaumur = reaumur/10;
Reaumur formula are (celsius*0.8). To get the exact result, we times the value 10 time larger.

References:
http://www.national.com/ds/LM/LM35.pdf

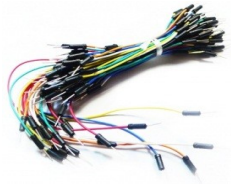
PROJECT 7

ULTRASONIC RANGE FINDER TO LCD DISPLAY

Ultrasonic range finder MAXSONAR-EZ1 which can measure the range from 6-inch to 254-inch with also an incredibly small package with ultra low power consumption. It's provided sonar range information with 1-inch each step. The interface format is include pulse-width, analog and digital output. For this project, we are going to interface this sensor with Arduino and display the output range in inch in LCD display.

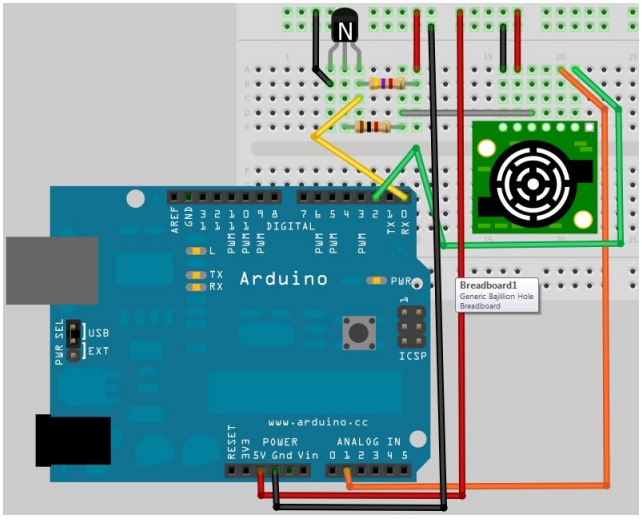
COMPONENT NEEDED

	MAXBOTIC MAXSONAR-EZ1
	NPN TRANSISTOR 2N2222
	1K RESISTOR x1 4K7 RESISTOR x1

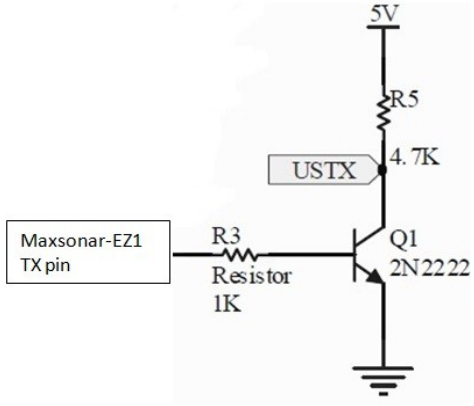
	JUMPER WIRE

CONNECTION

Referring to the figure below are the connection of Maxsonar-EZ1 to Arduino. For analog The RED wire are connected to *5V* while the **Black** wire are ground GND. As for the Analog output, connect the ORANGE wire to *Analog port 2(A2)* and for the PW output are connected to *Digital port 2* which show in GREEN wire.



For UART connection, please refer schematic below.



ADDITIONAL INFORMATION

MAXSONAR-EZ1 Features

· Continuously variable gain for beam control and side lobe suppression.
· Object detection includes zero range objects.
· Single 5V supply with 2mA typical current draw.
· Readings can occur up to every 50mS, (20-Hz rate).
· Free run operation can continually measure and output range information.
· Triggered operation provides the range reading as desired.
· All interfaces are active simultaneously
· Serial, 0 to 5V
  · 9600Baud, 81N
  · Analog (10mV/inch)
  · Pulse width (147uS/inch)
· Learns ringdown pattern when commanded to start ranging.
· Designed for protected indoor environments.
· Sensor operates at 42KHz.
· High output 10V PP square wave sensor drive.

Please check the Maxsonar-EZ1 datasheet for more information.

CODE OVERVIEW

```
//variables needed to store values
long pulse, anVolt, inches;
int sum=0;
int avgrange=10;
```

Variables that declare to store the value in long and integer.

```
  pinMode(range_analog_analogPin, INPUT);
  pinMode(range_pwm_Pin,INPUT);

  adc_key_in = analogRead(analogPin);
  adc_key_in = get_key(adc_key_in);

Serial.begin(9600);
digitalWrite(range_uart_TxPin,HIGH) ;
```

adc_key_in = analogRead(analogPin);
Read the ADC input from the keypad.
adc_key_in = get_key(adc_key_in);
Range the ADC keypad value from 1 to 5.

Serial.begin(9600);
Set the Baud rate of serial communication to 9600.

digitalWrite(range_UART_TxPin,HIGH);
HIGH(1) the UART transmit pin.

```
while(adc_key_in == -1)
  {
  for(int i = 0; i < avgrange ; i++)
     {
       anVolt = analogRead(range_analog_analogPin);
       sum += anVolt;
       delay(10);
     }

  inches = sum/avgrange;
  inches = inches/2;
  lcd.setCursor(6,0);
  lcd.print(inches,DEC);

  sum = 0;

  adc_key_in = analogRead(analogPin);
  adc_key_in = get_key(adc_key_in);
  delay(100);
  range_value_clear();
  }
break;
```

while (adc_key_in == -1)
Loop forever if the adc_key_in is equal to -1.

anVolt = analogRead(range_analog_analogPin);
Read the ADC output value from Maxsonar EZ1 output pin.

sum += anVolt;
Add all the ADC value together.

inches = sum/avrange;
Divided the sum value by 10 to get the average result.

Inches = inches/2;
To get the result in Inches, the average value must be divide by 2 because 1024/522 = ~2.
sum = 0;
Reset the sum value.

range_value_clear();
Clear the LCD value display screen.

```
lcd.setCursor(0,1);
lcd.print("PWM MODE   ");
unsigned int range_pwm;
while(adc_key_in == -1)
 {
 pulse = pulseIn(range_pwm_Pin , HIGH);
 inches = pulse/147;
 lcd.setCursor(6,0);
 lcd.print(inches,DEC);

 adc_key_in = analogRead(analogPin);
 adc_key_in = get_key(adc_key_in);
 delay(100);
 range_value_clear();
 }
break;
```

pulse = pulseIn(range_pwm_Pin , HIGH);
Detect the input HIGH pulse on the desire pin and range it in milisecond.

inches = pulse/147;
Divide the input pulse by 147 due to each inch of range are range to 147ms.

```
while(adc_key_in == -1)
  {
  for (l=0 ; l<6 ; l++)
    {
   while (!Serial.available());
      inches = Serial.read();
        if(inches == 'R') data[k=0] = inches;
         if(data[0] == 'R') data[k++] = inches;
          if (k>4) k = 4;
    }

 lcd.setCursor(6,0);
 lcd.write(data[1]);
 lcd.write(data[2]);
 lcd.write(data[3]);

 adc_key_in = analogRead(analogPin);
 adc_key_in = get_key(adc_key_in);
 delay(100);
 range_value_clear();
 }
break;
```

while(!**Serial**.available());
Check whether there is data available in serial communication.

Inches = **Serial**.read();
Read and store the data from serial into inches.

If(inches == 'R') data[k=0] = inches;
Check whether the data receive is 'R'. If yes then store in data[0].

If(data[0] == 'R') data[k++] = inches;
If the data[0] is equal to 'R'. Then the next data are store from data[1] to data[4].

If (k>4) k=4;
Check if K is greater then 4. If yes then reset back to 4.

```
if (adc_key_in == 1)
    {
      adc_key_in = -1;
      delay(400);
      if (--mode < 1)
        {
          mode = 3;
        }
    }

if (adc_key_in == 2)
    {
      adc_key_in = -1;
      delay(400);
      if (++mode > 3)
        {
          mode = 1;
        }
    }
```

if (adc_key_in ==1)
Check if the keypad is press UP. If yes then do the following coding.

if (adc_key_in ==2)
Check if the keypad is press DOWN. If yes then do the following coding.

References:

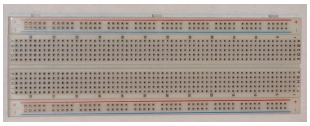
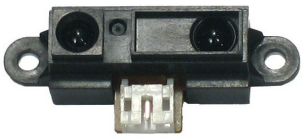
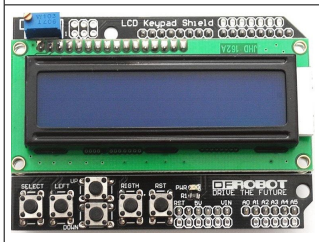
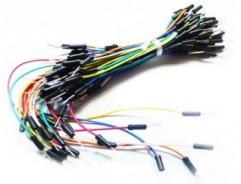
1. http://www.arduino.cc/playground/Main/MaxSonar

PROJECT 8

ANALOG DISTANCE SENSOR TO LCD DISPLAY

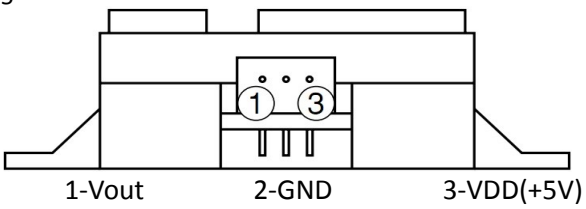
Analog distance sensor, which are mainly use to measure the distance of an object. The advantage of this sensor is the accuracy of sensor can measure the object distance up to centimeter(cm). In this project, we going to interface this sensor to Arduino Deuemilanove and display the value it Arduino LCD Keypad shield.

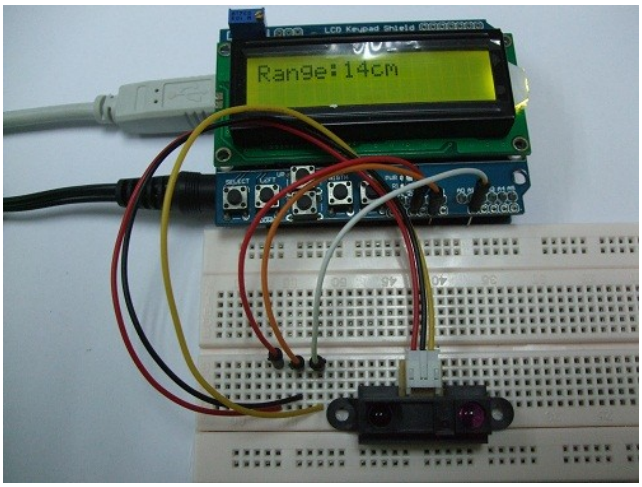
COMPONENT NEEDED

	BREADBOARD
	ANALOG DISTANCE SENSOR
	ARDUINO LCD-KEYPAD SHIELD
	JUMPER WIRE

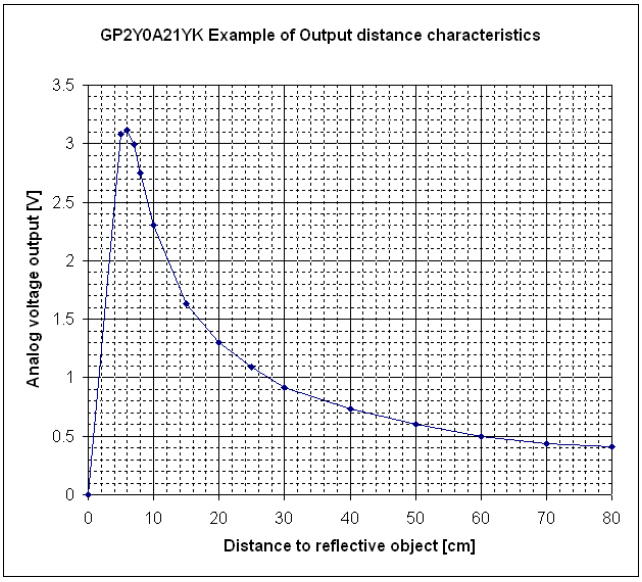
CONNECTION

Analog distance sensor have 3 pins which is VDD, GND, Vout. Connect the *Vout to analog pin-1*, *GND to ground* and *VDD to 5V.*

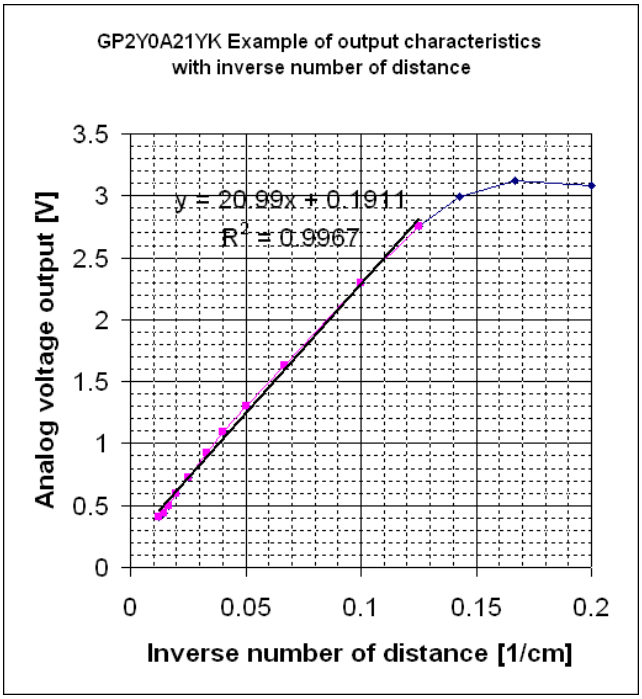




ADDITIONAL INFORMATION



Referring to the chart above are the output voltage(V) versus distances(cm) of the analog distance sensor. By plotting the output voltage versus inverse number distance (1/cm), we can get the linear relation such as in figure below.



The linear equation is **y = 20.99x + 0.19**, where y is voltage output and x invert distance. Notice that the trusted output voltage are about 0.4 to 2.8 V.

CODE OVERVIEW

```
void loop()
{
  range = analogRead(range_analogPin);
  Vout = (range*500000)/1024;

  if ((Vout > 43945)&&(Vout < 279785))
    {
    range = (Vout - 19000)/2099;
    lcd.setCursor(6,0);
    lcd.print(1000/range,DEC);
    }
  else
    {
    lcd.setCursor(0,1);
    lcd.print("OUT OF RANGE");
    }

  delay(100);
  range_value_clear();
}
```

Vout = (range*500000)/1024;
Get the ADC value and apply formula where to make the value in voltage with larger value.
E.g: adc_value = 1022,
~ if using adc_value*5/1024,
Vout = 4

~ if using adc_value* 500000/1024
Vout = 499023

if ((Vout > 43945) && (Vout < 279785))
Check the Vout value whether is between 43945 and 279785.

range = (Vout – 19000)/2099;
Determine the range by apply the formula given.
E.g-1 : if Vout = 279785
279785 – 19000 = 260785
260785 / 2099 = 124
1000 / 124 = 8

```
void range_value_clear(void)
{
lcd.setCursor(6,0);
lcd.print("  ");
lcd.setCursor(0,1);
lcd.print("                ");
}
```

Set the LCD cursor and clear the value.

References:

http://www.cytron.com.my/usr_attachment/GP2Y0A21YK.pdf

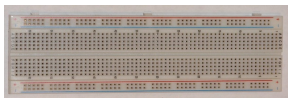
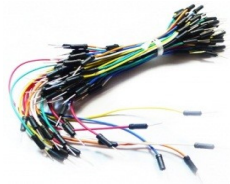
http://onlinescience38.blogspot.com/2011/03/distance-measurement-sensor.html

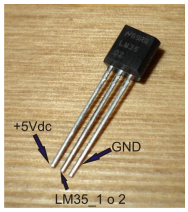
PROJECT 9

LM35 CONTROL DC MOTOR SPEED

In this project, we are going to experience on how to control the DC motor speed in PWM mode according to the changes of the LM35 temperature sensor like air-conditional. The DC motor speed will increase according to the temperature change to higher and decrease on opposite.

COMPONENT NEEDED

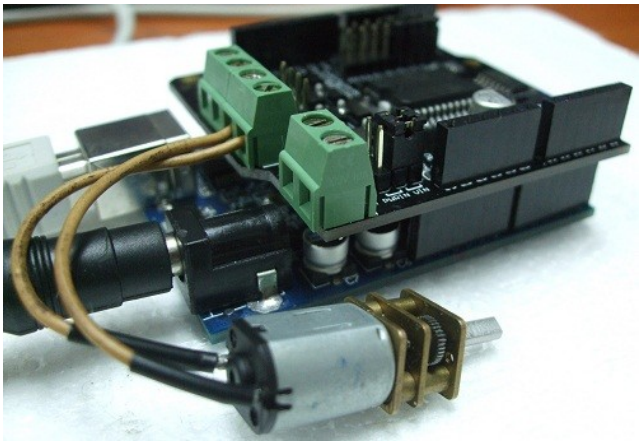
	BREADBOARD
	MICRO METAL GEAR MOTOR
	ARDUINO 2-AMP MOTOR DRIVER SHIELD
	JUMPER WIRE

	LM35 TEMPERATURE SENSOR

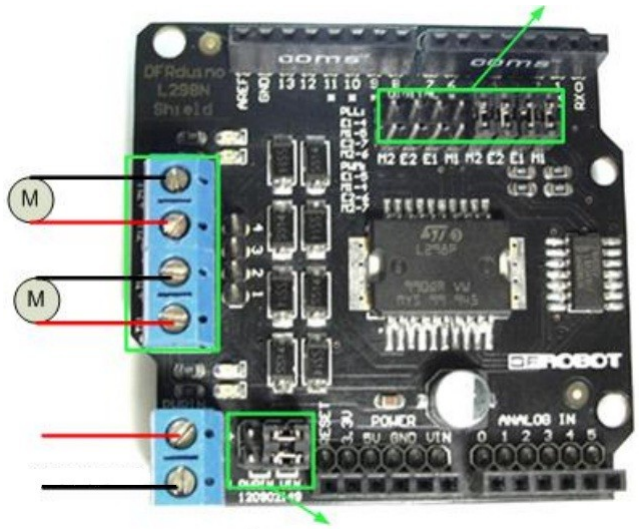
CONNECTION

MOTOR SHIELD CONNECT TO DC MOTOR

Connect the 2 terminal of motor to the M+ and M- terminal of motor. Please refer this website for detail connection.



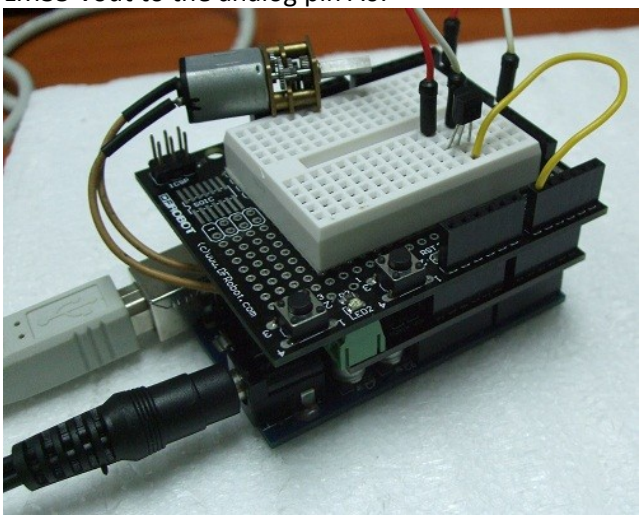
Speed Control Mode



Power Source Selection jumper

LM35 TO ARDUINO PROTOTYPING SHIELD (OPTIONAL)

Connect the LM35 sensor on the prototyping shield while the motor shield are at below. Connect the LM35 Vout to the analog pin A0.



ADDITIONAL INFORMATION

The Shield can switch between PWM speed control mode and PLL phase-locked loop mode through setting the appropriate jumpers. The power supply can be achieved either via Arduino VIN input or PWRIN input on the shield through setting the appropriate jumpers. The speed control is achieved through conventional PWM which can be obtained from Arduino's PWM output Pins 5 and 6. The enable/disable function of the motor control is signaled by Arduino Digital Pins 4 and 7.

The Motor shield can be powered directly from Arduino or from external power source. It is strongly recommended to use external power supply to power the motor shield.

- Logic Control Voltage：5V (From Arduino)
- Motor Driven Voltage ：4.8～35V (From Arduino or External Power Source)
- Logic supply current Iss：≤36mA
- Motor Driven current Io：≤2A
- Maximum power consumption ：25W（T=75℃）
- PWM、PLL Speed control mode
- Control signal level:
  High：2.3V ≤ Vin ≤ 5V
  Low：-0.3V ≤ Vin ≤ 1.5V

CODE OVERVIEW

```
int PWM_E1 = 6;
int PWM_M1 = 7;
int PWM_E2 = 5;
int PWM_M2 = 4;

int analogPin = A0;
```

Define the connection of the motor shield PWM mode pin.

```
void loop()
{
int value;
value = (analogRead(analogPin))*5;
digitalWrite(PWM_M1,HIGH);
digitalWrite(PWM_M2, HIGH);
analogWrite(PWM_E1, value);
analogWrite(PWM_E2, value);
delay(30);
}
```

value = (analogRead(analogPin))*5;

Read the ADC value from LM35 temperature sensor. In this case, the value is time with 5 due to the ADC output value from temperature are too low to move the DC motor. The PWM value are between 0 to 255.

References:

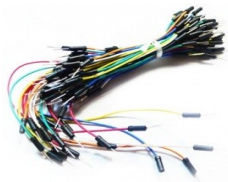
http://droboticsonline.com/ebaydownloads/L298_Motor_Shield_Manual.pdf

PROJECT 10

SERIAL COMMUNICATION CONTROL RC SERVO

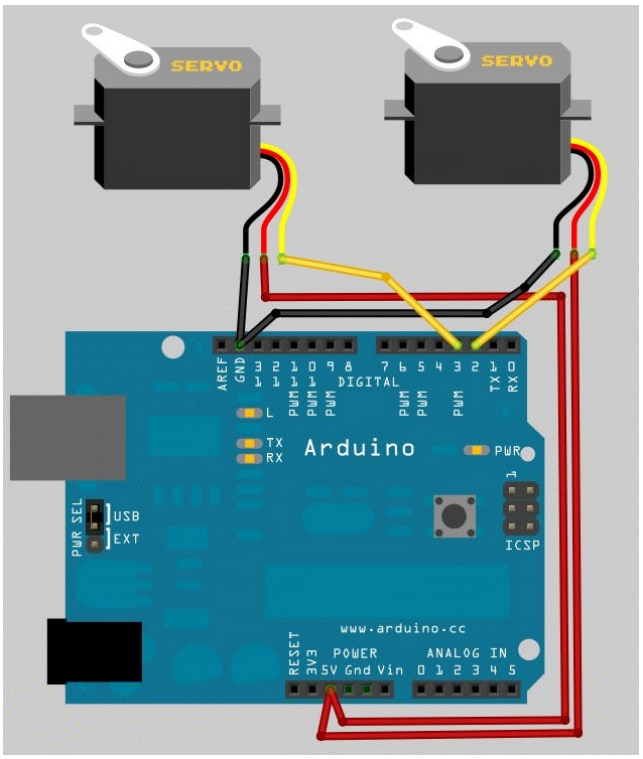
Wondering how to control the RC servo motor through your PC/laptop? In this project, we are going to explore on how to control two RC servo's position through the serial monitor of Arduino programmer. By sending the position from 0 to 180 will determine the servo position.

COMPONENT NEEDED

	RC SERVO MOTOR
	JUMPER WIRE

CONNECTION

in this project, there are 2 RC servo motor signal wire (YELLOW) connected to Arduino Duemilanove *digital pin-2 and pin-3*. RED wire are connected to *+5V* while **BLACK** wire connected to *ground(GND)*.



CODE OVERVIEW

```
int value;
unsigned int old_value;
unsigned char i;
```

int value;
Assign the value as integer to store the value of servo position from 0 to 180.

unsigned int old_value;
Store the last value of the servo position.

```
void setup ()
{
  servo1.attach(2);
  servo2.attach(3);
  Serial.begin(9600);

  servo1.write(90);
  servo2.write(90);
  old_value = 90;
}
```

servo1.write(90);
Initialize and move the servo1 to 90 degree.

old_value = 90;
Set and store the position of servo to old_value.

```
Serial.println();
  value = inByte[0]*100 + inByte[1]*10;
  if (value > 180) value = 180;
  else if (value < 0) value = 0;

   if (value < old_value)
      {
     for(i = old_value ; i > value ; i -= 1)
        {
        servo1.write(i);
        servo2.write(i);
        delay(15);
        }
        old_value = value;
      }
```

If (value > 180) value = 180;
Check if the value is larger than 180. if yes then set it to maximum 180.

else if (value < 0) value = 0;
Check if the value is less than 0. if yes then set it to minimum 0.

if (value < old_value)
Check if the current value of servo position is less then the old_value or last value.

old_value = value;
Assign the current value to old_value as the memory for the microcontroller to determine servo position.

References:

1. http://arduino.cc/en/Tutorial/Sweep

2. http://www.cytron.com.my/usr_attachment/RC_Servo_User_Manual.pdf

PROJECT 11

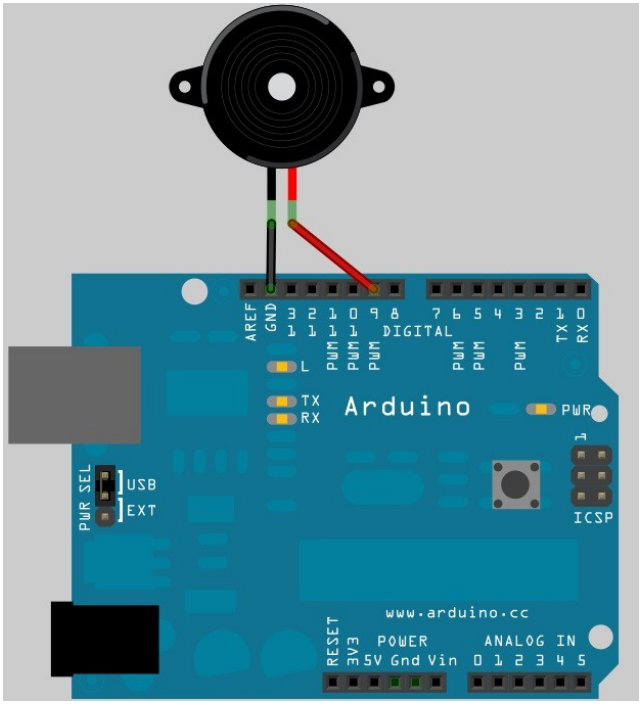
PIEZO BUZZER: MELODY

For this project, we are going to interfacing the piezo buzzer to Arduino Duemilanove and create a simple birthday melody by just applying the Pulse Width Modulation(PWM) signal to it.

COMPONENT NEEDED

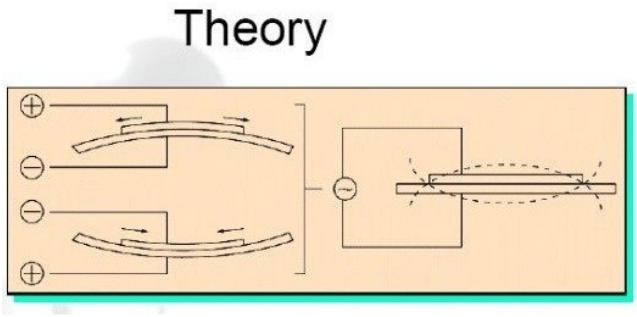
	PIEZO BUZZER

CONNECTION

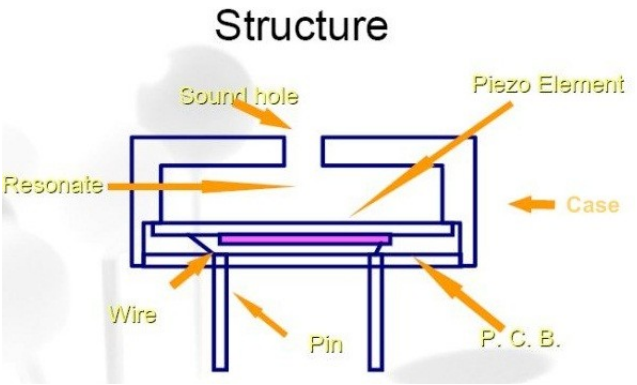


Referring to the figure above. The **BLACK** colour wire are connected to *ground(GND)* while the RED wire are connected to *digital pin-9* as the signal tone.

ADDITIONAL INFORMATION



When we give voltage to the ceramic wafer, it contracts and expands. The tension will drive the attached metal sheet emits the sound.



CODE OVERVIEW

```
/********************
 * Public Constants
 ********************
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
```

Define the pitches value using for typical note.

```
// notes in the melody:
int melody[] = {
   NOTE_C4_1,NOTE_C4, NOTE_D4, NOTE_C4,NOTE_F4,NOTE_E4,
   NOTE_C4_1,NOTE_C4,NOTE_D4,NOTE_C4,NOTE_G4,NOTE_F4,
   NOTE_C4_1,NOTE_C4,NOTE_C5,NOTE_A4,NOTE_F4,NOTE_E4,NOTE_D4,
   NOTE_AS4,NOTE_AS4,NOTE_A4,NOTE_F4,NOTE_G4,NOTE_F4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  6, 6, 3, 3,3,3,
  6, 6, 3, 3,3,3,
  6, 6, 3, 3,3,3,3,
  6, 6, 3, 3,3,3 };
```

int melody[ ]
Preset the melody for birthday song.

int noteDuration[ ]
Preset the delay value for each melody note playing.

```
void loop()
{
  for (int thisNote = 0; thisNote < 26; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(9, melody[thisNote],noteDuration);

    int pauseBetweenNotes = noteDuration + 50;      //delay between pulse
    delay(pauseBetweenNotes);

    noTone(9);                // stop the tone playing
  }
}
```

int noteDuration = 1000/noteDuration[thisnote];
To calculate the note duration, take the 1 second and divided with the note type.
E.g: 1000/3 = 333

tone (9, melody[thisnote], noteDuration);
Tone(pin, frequency, duration).
Generates a square wave of the specified frequency (and 50% duty cycle) on a pin-9.

int pauseBetweenNotes = noteDuration + 50;
Delay between the notes by adding 50ms to the noteDuration.

noTone(9);
Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

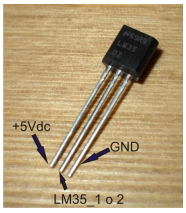
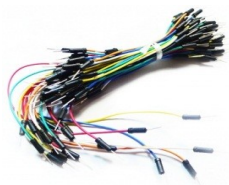
References:
http://www.arduino.cc/en/Tutorial/Tone

PROJECT 12

SERIAL COMMUNICATION CONTROL DC MOTOR

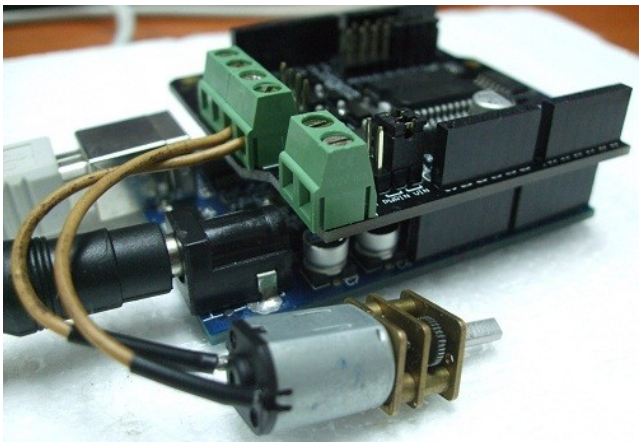
Since we have try how to control DC motor with temperature, how about we try how to control DC motor with our PC/Laptop while sending temperature value to our PC.

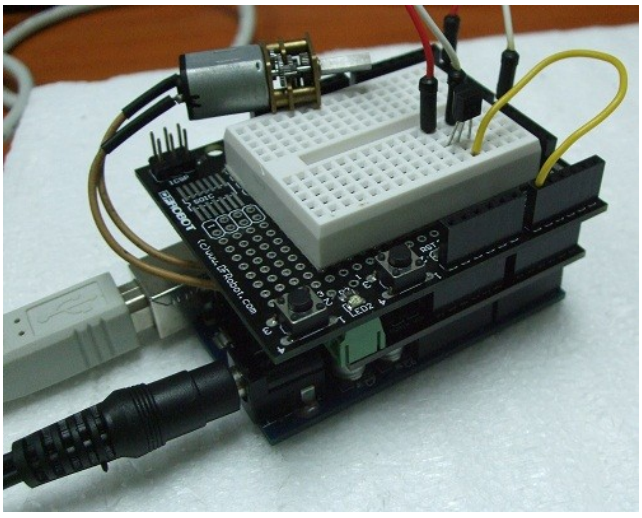
COMPONENT NEEDED

	ARDUINO 2-AMP MOTOR DRIVER SHIELD
	MICRO METAL GEARMOTOR SPG10
	LM35 TEMPERATURE SENSOR
	JUMPER WIRE

CONNECTION

The connection for this project are same with project 9 which the motor are connected to motor shield like figure below. While the LM35 temperature sensor signal pin are connected to *analog pin-0*.





CODE OVERVIEW

```
Temp_Sensor = analogRead(A0);
delay(10);
Serial.print("Current Tempreture Reading(celcius):");
Serial.println(Temp_Sensor/2, DEC);
Serial.print("Motor Speed Require(00 - 25):");

 for (i=0 ; i<2 ; i++) {
   while (!Serial.available());
   inByte[i] = Serial.read();
   inByte[i] = inByte[i]-48;
   }

     value = inByte[0]*100 + inByte[1]*10 + inByte[2];

Serial.print("\n");
Serial.print(value, DEC);
Serial.println("PWM");
digitalWrite(PWM_M1,HIGH);
digitalWrite(PWM_M2, HIGH);
analogWrite(PWM_E1, value);
analogWrite(PWM_E2, value);
delay(30);
```

Serial.print(“Current Tempreture Reading(celsius)”);
Send a string of words to PC/Laptop.

Serial.println(Temp_Sensor/2, DEC);
Send the temperature value to PC/Laptop which the value in Decimal and print it in new line.

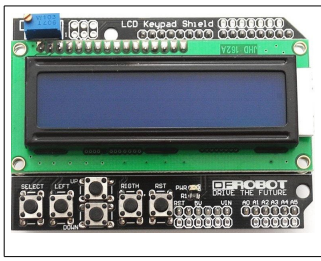
References
http://droboticsonline.com/ebaydownloads/L298_Motor_Shield_Manual.pdf

PROJECT 13

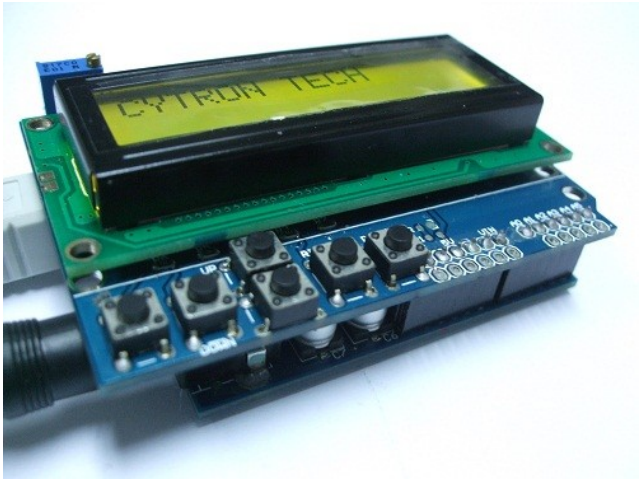
UART TO COMPUTER

To deal with computer keyboard, we have to know what kind of signal did keyboard giving when you press a key. For this project, we are going to interface the Arduino Duemilanove to computer through USB.

COMPONENT NEEDED

	ARDUINO – LCD KEYPAD SHIELD

CONNECTION



For this project, you are only required to plug in the Arduino LCD-Keypad Shield to the Arduino Duemilanove and the also make sure the USB cable are connected to the computer.

ADDITIONAL INFORMATION

As many user may wondering what kind of the output from the keyboard are sending out to our microcontroller. For you information, the output are same as what we trying to send to LCD to display which is call ASCII where all the words and number are in Hex. Please check the ASCII table here.

CODE OVERVIEW

```
void loop()
{
  for (int lcd_cursor=0; lcd_cursor<32; lcd_cursor++)
    {
    if (lcd_cursor == 15) lcd.setCursor(0,1);
    else if (lcd_cursor == 31) lcd.home();
      while (!Serial.available());
      serial_in = Serial.read();
      lcd.write(serial_in);
    }
}
```

for (int lcd_cursor; lcd_cursor<32; lcd_cursor++)
The lcd_cursor will continuously increasing from 0 to 31 1 step each.

if (lcd_cursor == 15) lcd.setCursor(0,1);
Check if the lcd_cursor is 15. If yes then jump the lcd cursor to next line which is 2nd row, 1st column.

else if (lcd_cursor ==31) lcd.home();
Check if the lcd_cursor is 31. If yes then jump cursor to home which is 1st row, 1st column.

lcd.write(serial_in);
Write a character to the LCD display.

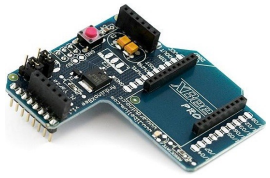
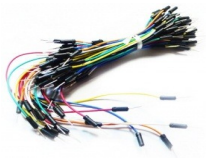
References:
http://arduino.cc/en/Reference/LiquidCrystal

PROJECT 14

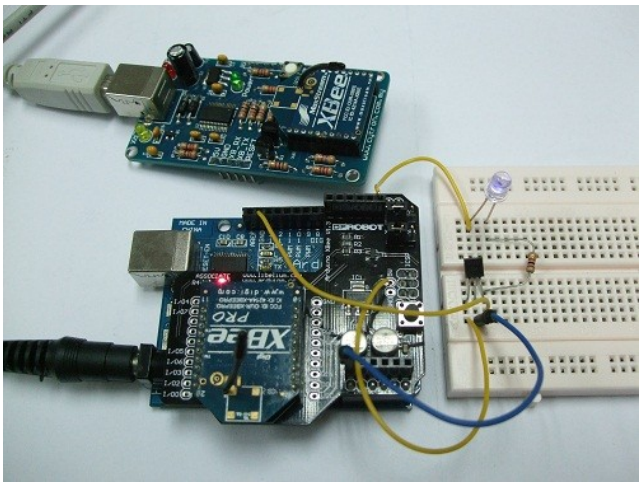
WIRELESS TEMPERATURE LOGGING

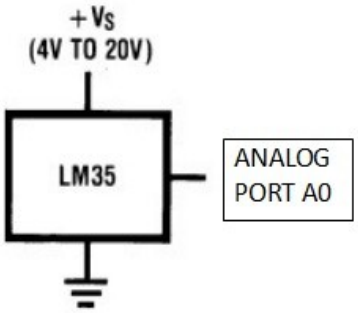
Required to send data in wireless and a long distance? Why not try to use XBee for long distance communication. For this project, we are going to interface LM35 temperature sensor and LED to the PC/Laptop by using XBee with 2 way communication.

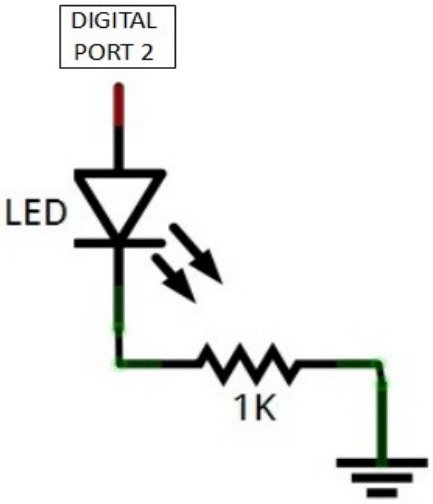
COMPONENT NEEDED

	SKXBEE
	ARDUINO-XBEE SHIELD
	XBEE
	JUMPER WIRE
	1K RESISTOR
	LED

CONNECTION







ADDITIONAL INFORMATION

XBEE ADDRESS SETTING

Before sending data through XBee. Always be remind that it must be set the address of each XBee and the other XBee to let them communicate. To set the address, please check this link.

TEST XBEE BY NETERING SIMPLE CODE

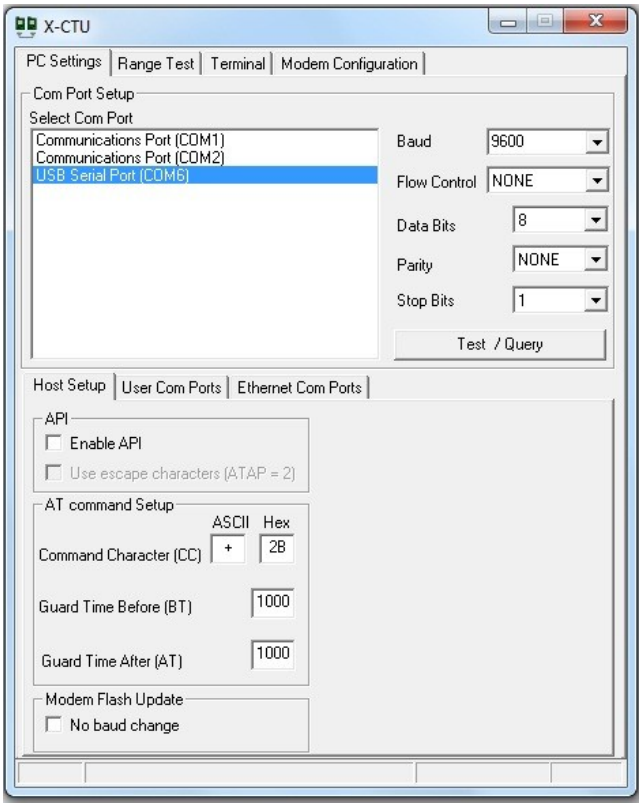
```
void setup()
{
  Serial.begin(9600)
}

void loop()
{
  Serial.print('H');
  delay(1000);
  Serial.print('L');
  delay(1000);
}
```

By entering the code above, we can test the XBee by checking the communication on the X-CTU from the computer between two XBee.

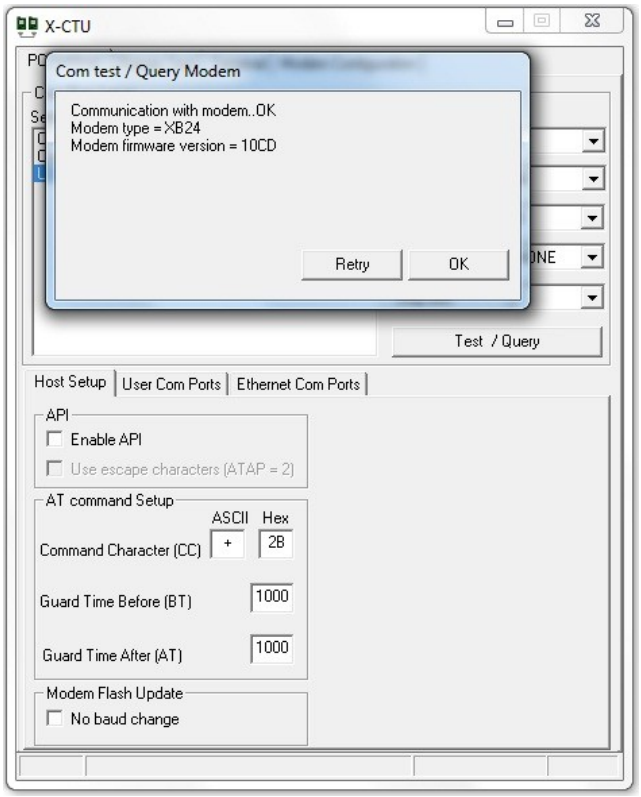
X-CTU

X-CTU are specially design for XBee to check the communication between XBee to computer. To use this software, please download here. There are only a few steps to use this program.

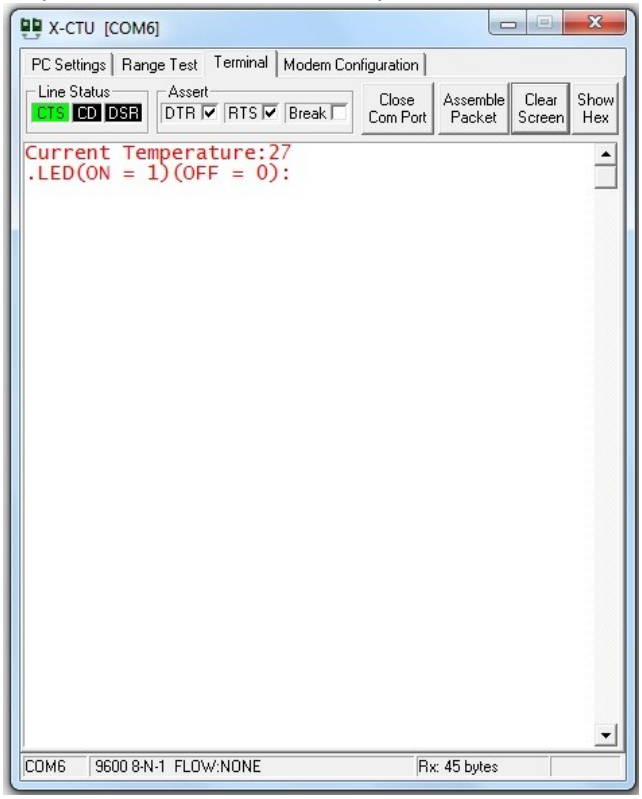
Step 1: Plug in the SKXBee to computer, then open the X-CTU. The window pop-up will be like figure below.



Step 2: Click the USB Serial Port then click the **Test / Queny.** A window will pop-up and tell us that it's OK to use such as figure below. If fail, please check back the connection and repeat the step.



Step 3: Click on the Terminal tap above and Done.



CODE OVERVIEW

```
int temp_analogPin = A0;
int LED = 2;
```

Define the temperature I/O pin by setting it to port A0 and the LED pin is on digital port 2.

```
void loop()
{
  int temp_value,LED_trigger;

  temp_value = analogRead(temp_analogPin);
  temp_value = temp_value/2;

  Serial.print("Current Temperature:");
  Serial.println(temp_value);

  Serial.print("LED(ON = 1)(OFF = 0):");
  while (!Serial.available());
  Serial.println();
  LED_trigger = Serial.read();
  LED_trigger = LED_trigger - 48;
    if (LED_trigger == 1)digitalWrite(LED,HIGH);
    if (LED_trigger == 0) digitalWrite(LED,LOW);
}
```

temp_value = analogRead (temp_analogPin);
Read the temperature value from the analog pin.

temp_value = temp_value/2;
Divide the temp_value by 2.

if (LED_trigger == 1) digitalWrite (LED,HIGH);
Check if the data receive for LED is 1 or 0. If is 1 then ON the LED, if 0 then off the LED.

References:
http://www.arduino.cc/en/Guide/ArduinoXbeeShield