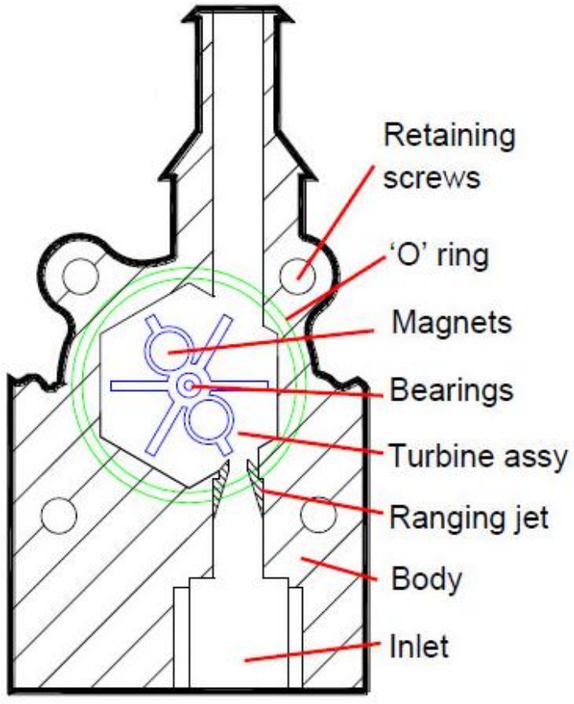
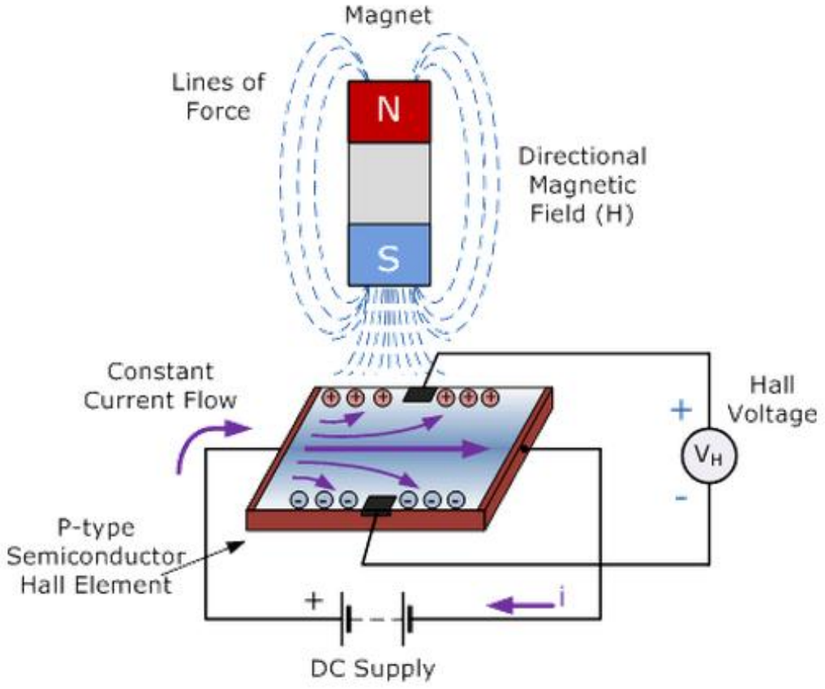
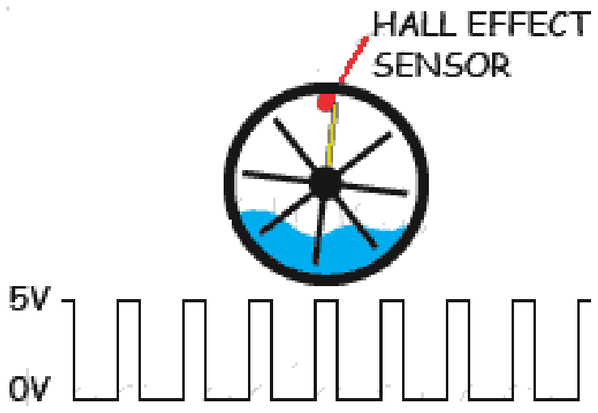
WATER FLOW SENSOR







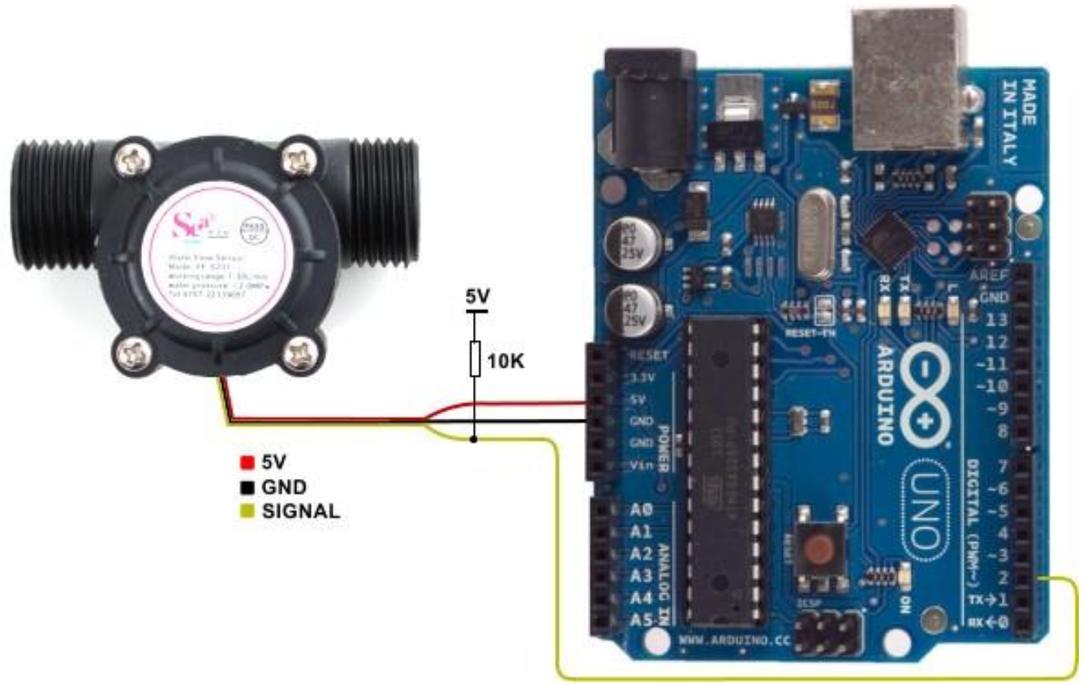
Treno di impulsi con frequenza proporzionale alla portata di fluido

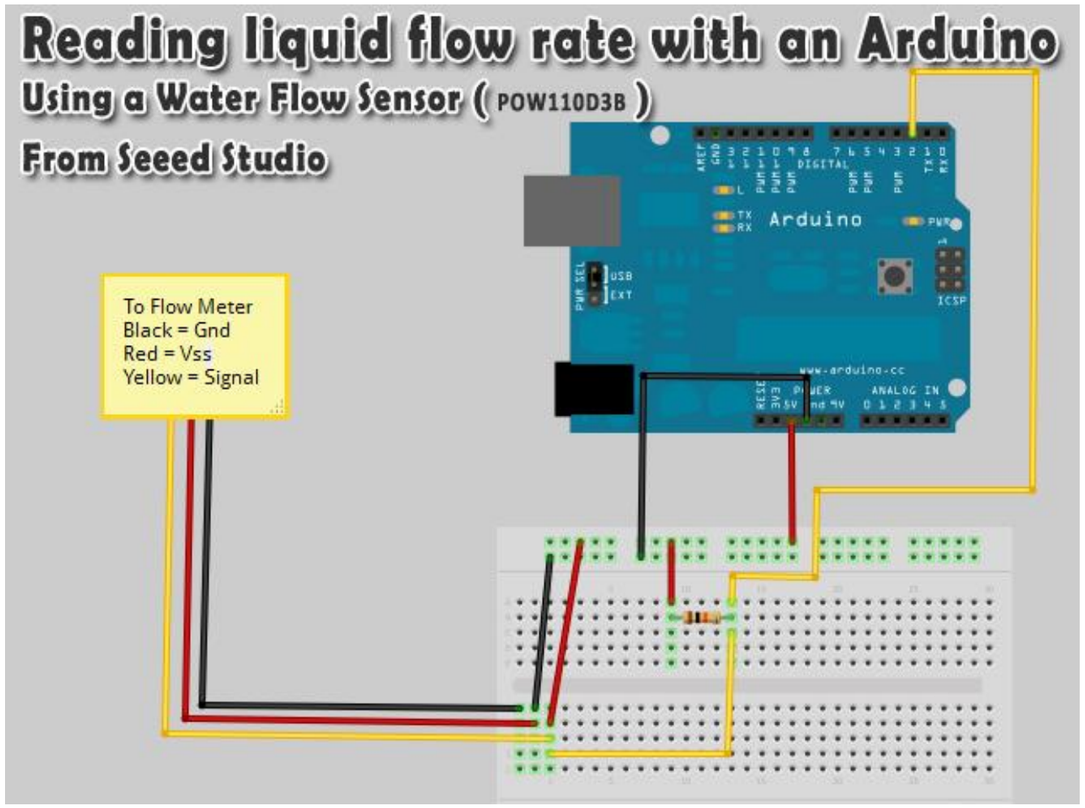
Hall Effect Sensors consist basically of a thin piece of rectangular p-type semiconductor material such as gallium arsenide (GaAs), indium antimonide (InSb) or indium arsenide (InAs) passing a continuous current through itself. When the device is placed within a magnetic field, the magnetic flux lines exert a force on the semiconductor material which deflects the charge carriers, electrons and holes, to either side of the semiconductor slab. This movement of charge carriers is a result of the magnetic force they experience passing through the semiconductor material.

As these electrons and holes move side wards a potential difference is produced between the two sides of the semiconductor material by the build-up of these charge carriers. Then the movement of electrons through the semiconductor material is affected by the presence of an external magnetic field which is at right angles to it and this effect is greater in a flat rectangular shaped material.

The effect of generating a measurable voltage by using a magnetic field is called the **Hall Effect**after Edwin Hall who discovered it back in the 1870's with the basic physical principle underlying the Hall effect being Lorentz force. To generate a potential difference across the device the magnetic flux lines must be perpendicular, (90^o) to the flow of current and be of the correct polarity, generally a south pole.

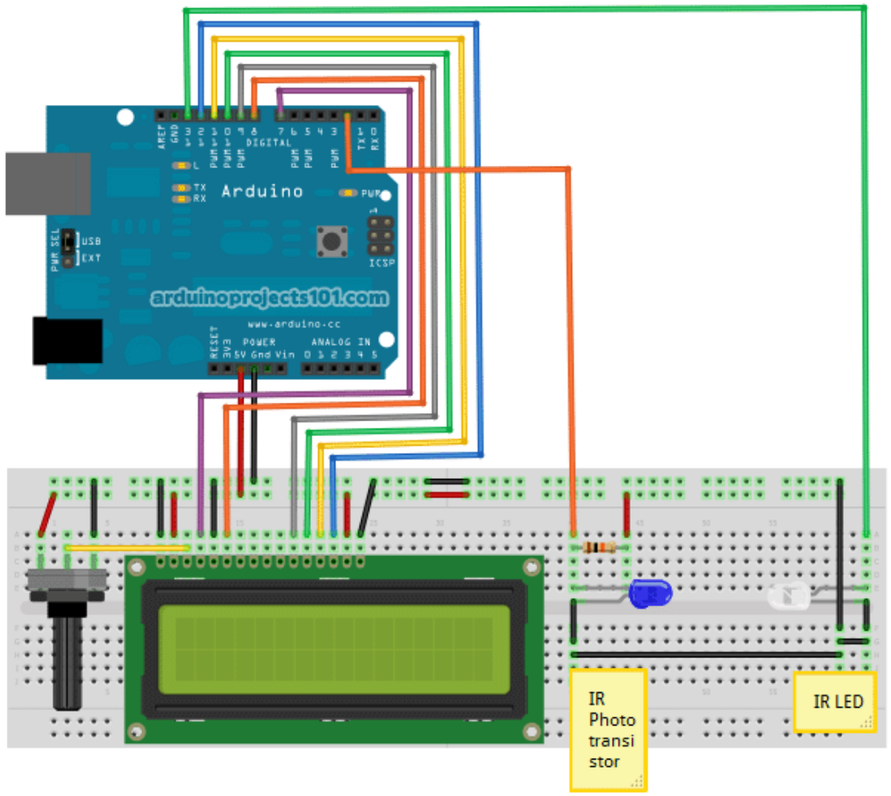
ARDUINO WATER FLOW SENSOR

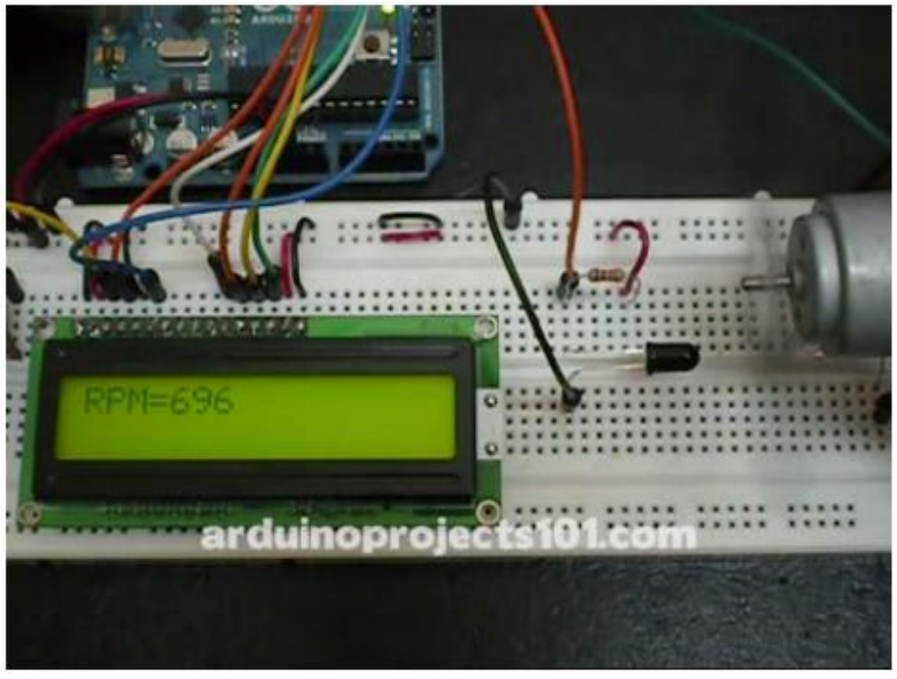





```
// reading liquid flow rate using Seeeduino and Water Flow Sensor from
Seeedstudio.com
// Code adapted by Charles Gantt from PC Fan RPM code written by Crenn
@thebestcasescenario.com
// http:/themakersworkbench.com http://thebestcasescenario.com http://seeedstudio.com

volatile int NbTopsFan; //measuring the rising edges of the signal
int Calc;
int hallsensor = 2;    //The pin location of the sensor

void rpm ()     //This is the function that the interupt calls
{
 NbTopsFan++;  //This function measures the rising and falling edge of the

hall effect sensors signal
}
// The setup() method runs once, when the sketch starts
void setup() //
{
 pinMode(hallsensor, INPUT); //initializes digital pin 2 as an input
 Serial.begin(9600); //This is the setup function where the serial port is

initialised,
 attachInterrupt(0, rpm, RISING); //and the interrupt is attached
}
// the loop() method runs over and over again,
// as long as the Arduino has power
void loop ()
{
 NbTopsFan = 0;      //Set NbTops to 0 ready for calculations
 sei();             //Enables interrupts
 delay (1000);      //Wait 1 second
 cli();             //Disable interrupts
 Calc = (NbTopsFan * 60 / 7.5); //(Pulse frequency x 60) / 7.5Q, = flow rate

in L/hour
 Serial.print (Calc, DEC); //Prints the number calculated above
 Serial.print (" L/hour\r\n"); //Prints "L/hour" and returns a  new line
}
```

Arduino RPM Counter / Tachometer






```
/*
 * Optical Tachometer
 * Uses an IR LED and IR phototransistor.
 * The IR LED is connected to pin 13 and ran continually.
 * Pin 2 (interrupt 0) is connected across the IR detector.
 */

int ledPin = 13;              // IR LED connected to digital pin 13
volatile byte rpmcount;
unsigned int rpm;
unsigned long timeold;

// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void rpm_fun()
 {
  //Each rotation, this interrupt function is run twice, so take
  // that into consideration for calculating RPM
  //Update count
    rpmcount++;
 }

void setup()
 {
  lcd.begin(16, 2);  // intialise the LCD
  //Interrupt 0 is digital pin 2 → where IR detector is connected
  //Triggers on FALLING (change from HIGH to LOW)
  attachInterrupt(0, rpm_fun, FALLING);
  //Turn on IR LED
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  rpmcount = 0;
  rpm = 0;
  timeold = 0;
 }
```

```
void loop()
 {
  //Update RPM every second
  delay(1000);
  //Don't process interrupts during calculations
  detachInterrupt(0);
  //Note that this would be 60*1000/(millis() - timeold)*rpmcount
  //if the interrupt happened once per revolution instead of twice.
  //Other multiples could be used for multi-bladed propellers or fans
  rpm = 30*1000/(millis() - timeold)*rpmcount;
  timeold = millis();
  rpmcount = 0;
  //Print out result to lcd
  lcd.clear();
  lcd.print("RPM=");
  lcd.print(rpm);
  //Restart the interrupt processing
  attachInterrupt(0, rpm_fun, FALLING);
 }
```

Parts List

1) 1x 16×2 parallel LCD display (compatible with Hitachi HD44780 driver)
2) 1x Arduino
3) 1x 10kO potentiometer
4) 1x 10kO resistor
5) 1x IR LED
6) 1x IR Phototransistor
7) Jumper wire

Build

1) Connect all jumper wire as shown in diagram.
2) Connect IR LED to digital pin 13.
3) Connect IR Phototransistor (dark) to digital pin 2.

Make sure shorter lead connected to digital pin 2 and longer lead to Ground.