Automatic Water Management System

Using IOT

INTRODUCTION:

Using this automatic water management system we can save water, power and manual operation. This water management system will check the water level in the tank and give instruction to the motor to send the water to the tank and also another sensor will check if the water is flowing or not and instruct the water level sensor to sense the and instruct the motor to send the water to tank and both data of water level and when the water has last came to the THING SPEAK

Equipments used:

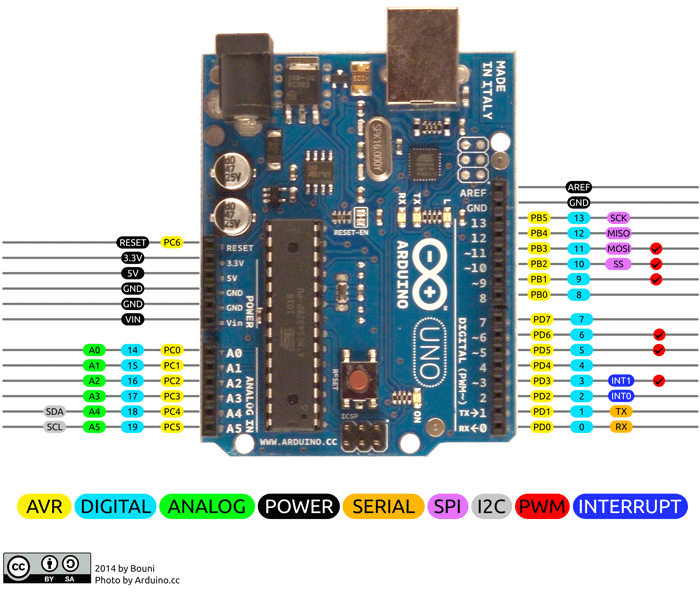
* Arduino UNO
* Ultrasonic sensor
* Water flow sensor
* DC Water pump
* ESP8266 Module
* Motor driver IC 293D

ARDUINO UNO:

In project we used arduino to control all sensors, modules and DC water pump.

[Arduino](http://arduino.cc/) is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a [microcontroller](http://en.wikipedia.org/wiki/Microcontroller)) and a piece of [software](http://arduino.cc/en/Main/Software), or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

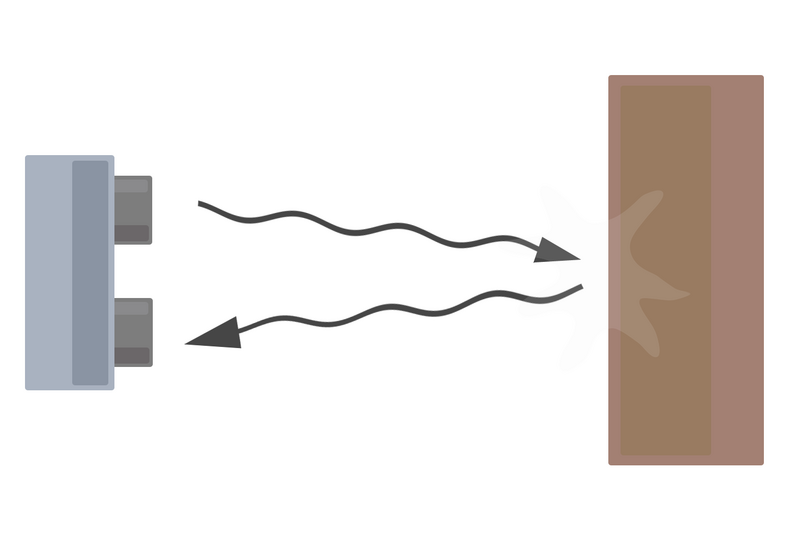
[](https://components101.com/sites/default/files/component_pin/Arduino-Uno-Pin-Diagram.png)

**Pin Description**

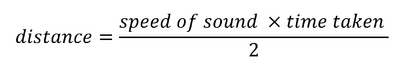
|  |  |  |
| --- | --- | --- |
| **Pin Category** | **Pin Name** | **Details** |
| Power | Vin, 3.3V, 5V, GND | Vin: Input voltage to Arduino when using an external power source.  5V: Regulated power supply used to power microcontroller and other components on the board.  3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.  GND: ground pins. |
| Reset | Reset | Resets the microcontroller. |
| Analog Pins | A0 – A5 | Used to provide analog input in the range of 0-5V |
| Input/Output Pins | Digital Pins 0 - 13 | Can be used as input or output pins. |
| Serial | 0(Rx), 1(Tx) | Used to receive and transmit TTL serial data. |
| External Interrupts | 2, 3 | To trigger an interrupt. |
| PWM | 3, 5, 6, 9, 11 | Provides 8-bit PWM output. |
| SPI | 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK) | Used for SPI communication. |
| Inbuilt LED | 13 | To turn on the inbuilt LED. |
| TWI | A4 (SDA), A5 (SCA) | Used for TWI communication. |
| AREF | AREF | To provide reference voltage for input voltage. |

ULTRASONIC SENSOR: In this project we use this sensor to sense the water level in the tank and send data to arduino.

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.



Since it is known that sound travels through air at about 344 m/s (1129 ft/s), you can take the time for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave. Round-trip means that the sound wave traveled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object). To find the distance to the object, simply divide the round-trip distance in half.





Water flow sensor: In this project we use water flow sensor to sense whether the water is flowing or not and send data to arduino that how much speed the water is flowing.

Water flow sensor consists of a plastic valve body, a water rotor, and a hall-effect sensor. When water flows through the rotor, rotor rolls. Its speed changes with different rate of flow. The hall-effect sensor outputs the corresponding pulse signal. This one is suitable to detect flow in water dispense or coffee machine. We have a comprehensive line of water flow sensors in different diameters. Check them out to find the one that meets your need most.

**Features:**

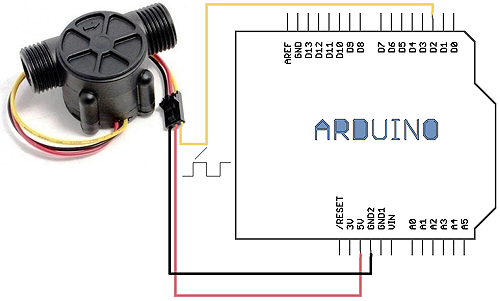
 Compact, Easy to Install

 High Sealing Performance

 High Quality Hall Effect Sensor

**Specifications:**

* Mini. Wokring Voltage: DC 4.5V
* Max. Working Current: 15mA (DC 5V)
* Working Voltage: DC 5V~24V
* Flow Rate Range: 1~30L/min
* Load Capacity: ?10mA (DC 5V)
* Operating Temperature: ?80?
* Liquid Temperature:?120?
* Operating Humidity: 35%~90%RH
* Water Pressure: ?1.75MPa
* Storage Temperature: -25~+ 80?
* Storage Humidity: 25%~95%RH

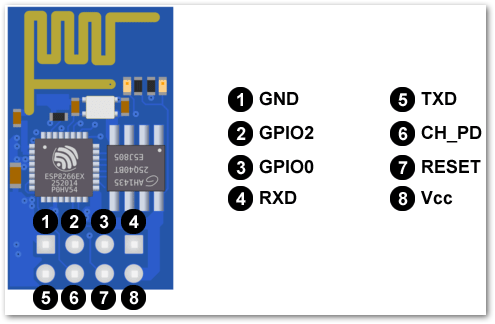


ESP8266 MODULE:

In this project we use this module to send data of ultrasonic sensor and water flow sensor to send into THINGSPEAK.

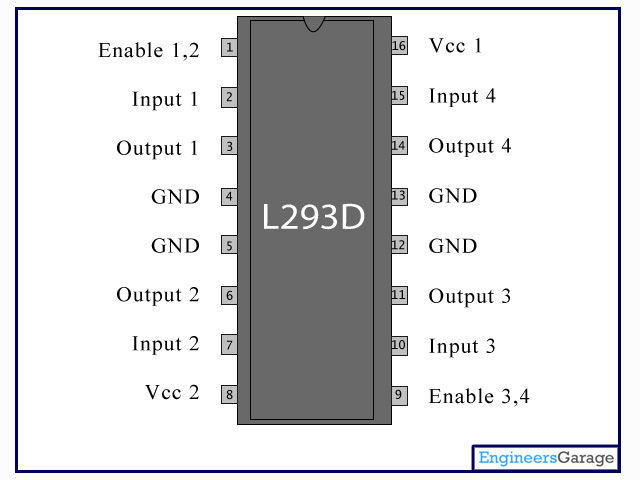
The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much Wi-Fi-ability as a Wi-Fi Shield offers (and that’s just out of the box)! The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts.

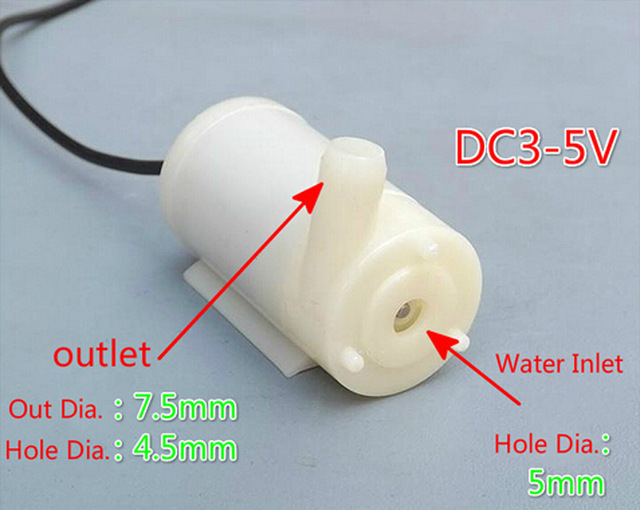


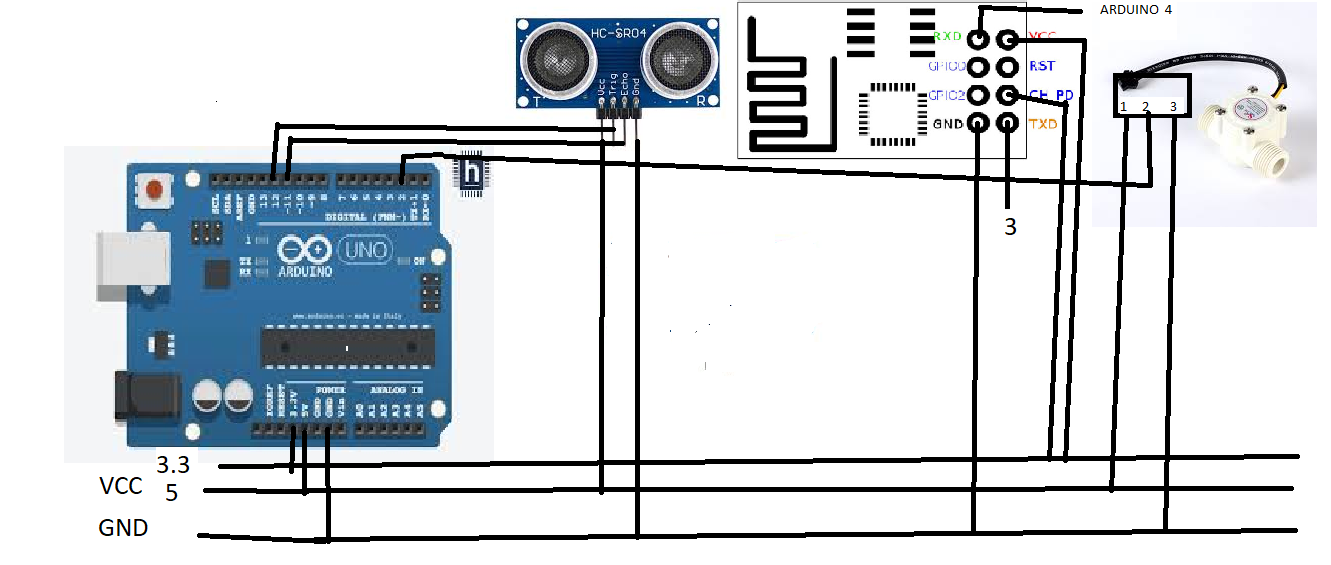
DC WATER PUMP: In this project we use 5v of dc water pump connected to arduino to send water to tank when arduino gives command.

For this water pump we use motor driver Ic 293d to control this water motor because arduino cannot drive the motor due to the output volt and current given by arduino is less



DC WATER PUMP:





Program for this project :

#include <NewPing.h>

#include <SoftwareSerial.h>

String apiKey = "BF9HMHD1WEGME4JD"; // 2737PEGIY2F9413T Edit this CHANNEL API key according to your Account

String Host\_Name = "sree"; // Edit Host\_Name

String Password = "sree@123"; // Edit Password

SoftwareSerial ser(3,4); // RX, TX

#define TRIGGER\_PIN 12

#define ECHO\_PIN 11

#define MAX\_DISTANCE 200

int val=0;

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE);

volatile int flow\_frequency; // Measures flow sensor pulses

unsigned int l\_hour; // Calculated litres/hour

unsigned char flowsensor = 2; // Sensor Input

unsigned long currentTime;

unsigned long cloopTime;

void flow () // Interrupt function

{

flow\_frequency++;

}

void setup() {

Serial.begin(115200); // enable software serial

ser.begin(115200); // reset ESP8266

ser.println("AT+RST"); // Resetting ESP8266

char inv ='"';

String cmd = "AT+CWJAP";

cmd+= "=";

cmd+= inv;

cmd+= Host\_Name;

cmd+= inv;

cmd+= ",";

cmd+= inv;

cmd+= Password;

cmd+= inv;

ser.println(cmd); // Connecting ESP8266 to your WiFi Router

pinMode(flowsensor, INPUT);

digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up

attachInterrupt(0, flow, RISING); // Setup Interrupt

sei(); // Enable interrupts

currentTime = millis();

cloopTime = currentTime;

}

void loop() {

currentTime = millis();

// Every second, calculate and print litres/hour

if(currentTime >= (cloopTime + 1000))

{

cloopTime = currentTime; // Updates cloopTime

// Pulse frequency (Hz) = 7.5Q, Q is flow rate in L/min.

l\_hour = (flow\_frequency / 7.5); // (Pulse frequency x 60 min) / 7.5Q = flowrate in L/hour

flow\_frequency = 0; // Reset Counter

Serial.print(l\_hour, DEC); // Print litres/hour

Serial.println(" L/min");

Serial.print("\n \n");

}

int y=l\_hour;

delay(50);

int x=sonar.ping\_cm();

Serial.print("ultra sonic \n");

Serial.print(x);

Serial.print("\n \n");

String cmd = "AT+CIPSTART=\"TCP\",\""; // Establishing TCP connection

cmd += "184.106.153.149"; // api.thingspeak.com

cmd += "\",80"; // port 80

ser.println(cmd);

Serial.println(cmd);

if(ser.find("Error")){

Serial.println("AT+CIPSTART error");

return;

}

String getStr = "GET /update?api\_key="; // prepare GET string

getStr += apiKey;

getStr +="&field1=";

getStr +=String (x);

getStr +="&field2=";

getStr +=String (y);

getStr += "\r\n\r\n";

cmd = "AT+CIPSEND=";

cmd += String(getStr.length()); // Total Length of data

ser.println(cmd);

Serial.println(cmd);

if(ser.find(">")){

ser.print(getStr);

Serial.print(getStr);

}

else{

ser.println("AT+CIPCLOSE"); // closing connection

Serial.println("AT+CIPCLOSE");

}

delay(1000); // Update after every 1 seconds

}