```
// Project: Cistern volume measurement using Arduino Uno R3, Arduino Ethernet Shield
// and using a  JSN SR04T V2.0  as an alternative to an ultrasonic sensors HR-SR04


// calling libraries
#include <stdint.h> // Limiting of value lenght
#include <Ethernet.h> // calling Ethernetshield Arduino at "uno" board-type Arduino/Genuino

// Timing e.g., for test run
#define SENDEINTERVALL 120              //120 = cycle time all 120 sec to Thingspeak, in tests 20 sec

//Variables
const char* server = "api.thingspeak.com";
String apiKey = "XXXXXXXXXXXXXXXX"; // write API: please insert from your own channel

uint8_t senden_erlaubt = 0;
uint8_t zaehler = SENDEINTERVALL;
uint8_t trigger = 7; // Arduino Pin please verify
uint8_t echo = 6; // Arduino Pin please verify
float abstand = 0.0;
float sendewert = 3000.0;

uint32_t sent = 0;
uint32_t failed = 0;

// Variables for volume calculation and average value
float Volumen = 0.0;
float Vol_m3;
float Vol_sum = 0;     // Total value all measurements
float Vol_temp = 0;    // for average calculation
int Szaehler = 1;      // counter for cycle
float fuellhoehe = 0.0;


// **** ETHERNET settings ****

byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0xFB, 0x80};  //Example only
// enter above the address, which is labeled on the backside of the board
EthernetClient client;

// Creates a client which can connect to a specified internet IP address and port
// (defined in the client.connect() function)

void setup()
{
  Serial.begin(9600);   // Calling serial connection for monitoring in Arduino IDE
  Ethernet.begin(mac);

// Definition of pin mode use
  pinMode(trigger, OUTPUT);   // Output for Triggerpulse
```

```
 pinMode(echo, INPUT);      // Input for Echo
 digitalWrite(trigger,LOW);
 Serial.println("Measurement of distance and calculation to m3:");
 Serial.print("Cycle is running – please wait ...");
 Serial.print(SENDEINTERVALL);
 Serial.println(" Sekunden ");
}

// **** Programme execution  ****

void loop()
{

 delay(1000);
 zaehler--;
 if(zaehler <= 6)            // 7 measurements
 {
  ultrasonic();             // Calling subroutine ultrasonic
  Szaehler++;               // Szaehler counting up for average calculation
 }
 if(zaehler <= 0)
 {
  sendtoThingSpeak();       // only 7th measurement will be transmitted
  zaehler = SENDEINTERVALL;   // calling subroutine SENDEINTERVALL
  Szaehler = 1;             // average calculation – reset of loop counter to 1
  Vol_temp = 0;             // temporary volume
 }
}

void ultrasonic()  // subroutine
{

// Variante with JSN sensor
 digitalWrite(trigger, HIGH);
 delayMicroseconds(20);    // with this sensor type 20 microsec waiting. 10 at HC-SR04
 digitalWrite(trigger, LOW);

 /* Variant using the HC-SR04
 digitalWrite(trigger, LOW);   // switching of  trigger signal
 delay(5);
 digitalWrite(trigger, HIGH); // switching on trigger signal
 delay(10);
 digitalWrite(trigger, LOW);
*/
 float dauer = 0.0;
 dauer = pulseIn(echo, HIGH);   // Calculation of total travelling time

 abstand = (dauer/2) * 0.03432; // calculation of distance

//multiplication with velocity of sound in cm/micorseconds to get cm.
// http://www.sengpielaudio.com/TemperaturSchall.htm at 15 C the value  to be set to 340,51 m/s.
// At 20 Celsius = 343,42 m/s
```

```
  abstand = abstand + 3; // The JSN sensor had an offset of 3cm
  abstand = round(abstand);  // Rounding since JSN sensor returns alternating values


// **** Distance to volume ****
// **** Need to measure the final distance from bottom to sensor in cm ****

fuellhoehe = 277 - abstand;   // adapt your distance
Volumen =  125 * 125 * 3.14159 * fuellhoehe;  // cistern diameter Mall 250 cm (r = 125 cm)

// average value calculation
Vol_m3 = Volumen / 1000000; // change from cm3 to m3
Serial.print(F("Loop value of volume, Result current loop in Vol_m3:   "));
Serial.print(Vol_m3);
Serial.println("  m3");

//average
Serial.print(F("Vol_Temp before change: "));
Serial.print(Vol_temp);
Vol_sum = Vol_temp + Vol_m3;
Vol_temp = Vol_sum;

Serial.print("   Vol_Temp after change: ");
Serial.print(Vol_temp);
Serial.print("   Vol_sum total after change: ");
Serial.println(Vol_sum);

// **** checking if value of distance is in range ****
     if (abstand >= 300 || abstand <= 0)  // 3m max height measured in the cistern
  {
    Serial.println("Measurement outside range");
    Serial.println(fuellhoehe);
    Serial.println(Vol_m3);
  }
  else
  {
    senden_erlaubt = 1;
    Serial.println("  ");
    Serial.print(F("Szaehler = ")); // checking loop counter
   Serial.println(Szaehler);
    Vol_m3 = Vol_temp;            // using last temporary value of  loop
    Vol_m3 = Vol_m3 / Szaehler;
//average value. This variable needs to be set in Thingspeak to achieve lines or gauge values

/*
   Following rounding of decimals. Reason: the lines in charts will be smoother in Thingspeak
   Vol_m3 = round(Vol_m3*10)/10.0; // rounding to one decimal. If two then using 100
  // Float has two decimals, because only 4 Byte. --> xx.yy
*/

    // **** this section here only for use in Arduino Monitor  ****
    Serial.print(fuellhoehe);
   Serial.print(" cm filling level results in:     ");
```

```
   Serial.print(Vol_m3);
   Serial.println(" m3 volume of water");
  }
}


// Sending volume to Thingspeak using API

void sendtoThingSpeak()
{
  int ConnectionStatus = 0;

  if(senden_erlaubt == 1)
  {
   client.stop();
   do
   {
    ConnectionStatus = client.connect(server, 80);
    if (ConnectionStatus == 1)
    {
     Serial.println("LAN Client connected.");
     Serial.print(F("calculated Vol_m3 =   "));
     Serial.println(Vol_m3);

/*
  Following contains rounding of decimals. Reason: the lines in charts will be smoother in Thingspeak
  Vol_m3 = round(Vol_m3*10)/10.0; // rounding to one decimal. If two then using 100
 // Float has two decimals, because only 4 Byte. --> xx.yy
*/

     sendewert = Vol_m3; // Volume to be send to Thingspeak
     Serial.print(F("average value one decimal of Vol_m3 =   "));
     Serial.println(sendewert);

     String postStr = apiKey;
     postStr += "&field1=";
     postStr += String(sendewert);

     client.print("POST /update HTTP/1.1\n");
     client.print("Host: api.thingspeak.com\n");
     client.print("Connection: close\n");
     client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
     client.print("Content-Type: application/x-www-form-urlencoded\n");
     client.print("Content-Length: ");
     client.print(String(postStr.length()));
     client.print("\n\n");
     client.println(postStr);

     delay(1000);
     sent++;
     Serial.println(F("  "));
     Serial.print(F("Lopp cycle No.  "));
     Serial.print(sent);
```

```
      Serial.print("  ");
      Serial.print(sendewert); //  sendewert = Vol_m3;
      Serial.println(" m3 – transmitted value (sendewert) to Thingspeak");
      Serial.println(" ***** next measurement ***** ");
    }
    else
    {
      Serial.print("connection failed, Status =  ");
      Serial.print(ConnectionStatus);
      Serial.print("   Trying:  ");
      Serial.println(failed);
      delay(500);
      failed++;
      ConnectionStatus = client.connect(server, 80); // to avoid endless loop, e.g., after power drops
     }
    client.flush(); // waiting until all data are send
    client.stop();
   }
   while(ConnectionStatus != 1 && failed < 10);
   senden_erlaubt = 0;
   failed = 0;
  }
  else
  {
   Serial.println("failure, no data transmission");
  }
}
```

//examples, sources:

// https://www.hackster.io/felixg/measurement-of-a-cistern-volume-with-an-ultrasonic-sensor-872456

// https://www.makerblog.at/2015/03/das-arduino-ethernet-shield-erste-installation-und-demo-sketch/