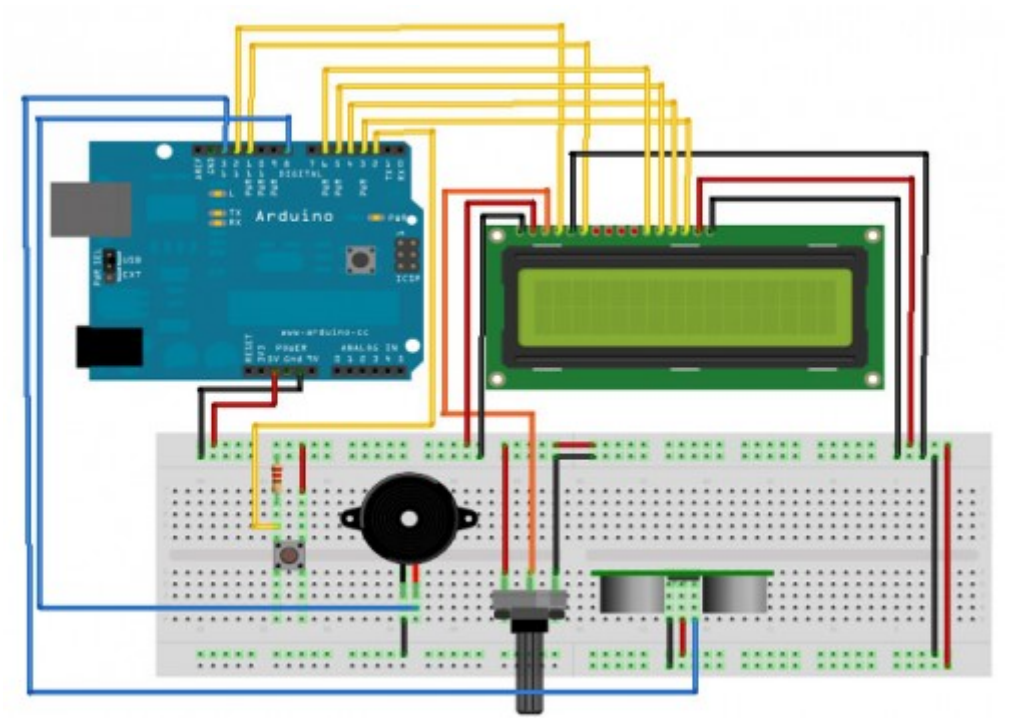
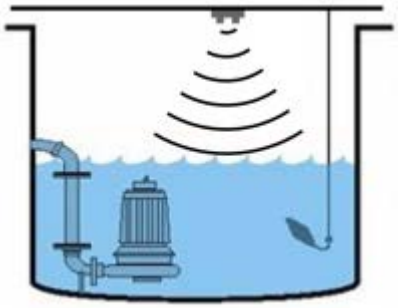
(မှီငြမ်း/Reference: https://www.open-electronics.org/water-tank-level-display-with-arduino/ မှ ဘာသာပြန်ဆို ပါသည်၊)

Water Tank level display with Arduino

ရေလှောင်ကန် အတွင်းရေ၏ အမြင့်ကို အာဒွီနို ဖြင့်ဖော်ပြခြင်း *By Boris Landoni on August 28, 2011*



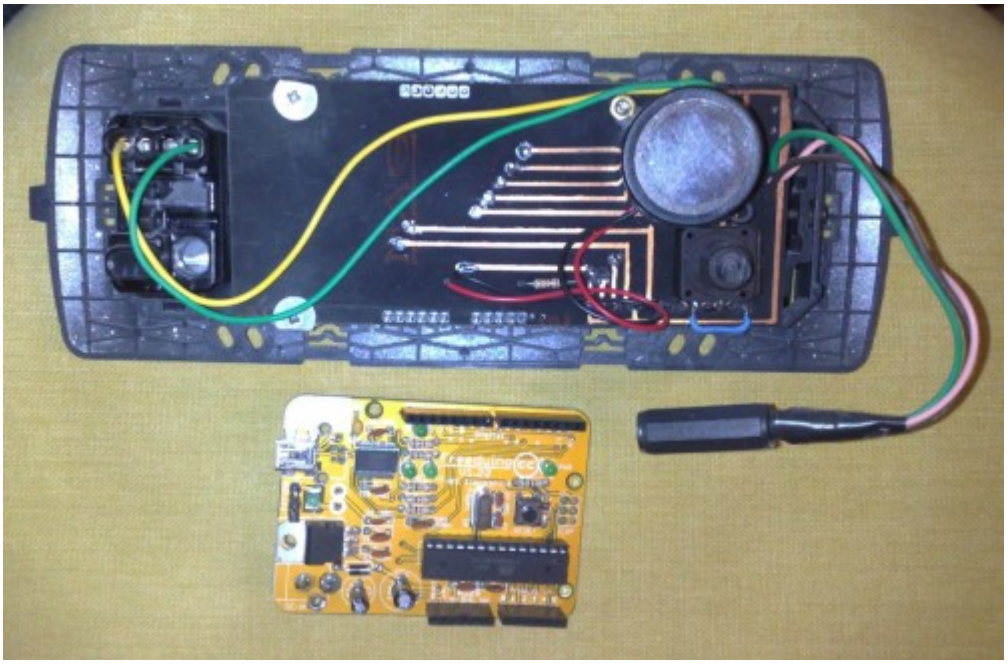
ဤ ပရောဂျက် သည် အာဒွီနို အသုံးပြုပြီး ရေတွင်း (သို့) ရေလှောင်ကန် အတွင်းရှိ ရေ အနိမ့် အမြင့်ကို ဖော်ပြပေးခြင်းဖြစ်သည်။



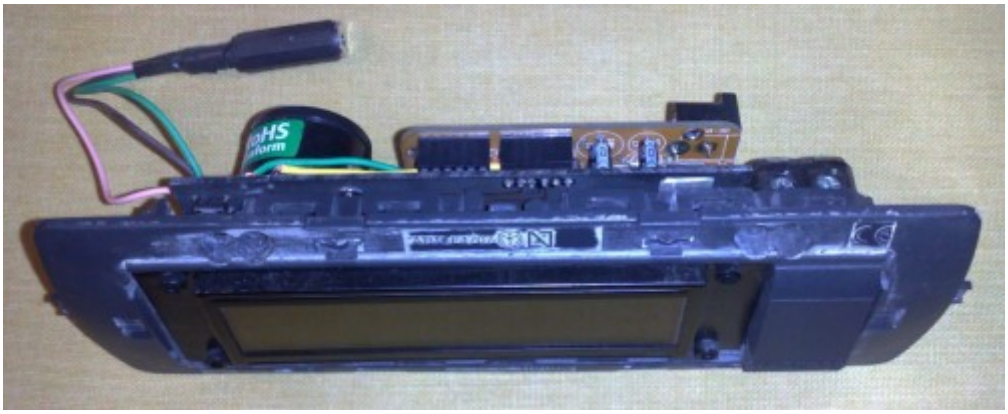
ဤပရောဂျက်တွင် အစိပ်အပိုင်း တချို့ ဖြင့်ဆက်စပ်တည်ဆောက်ထားသည်။ ဆိုနာ ဆင်ဆာ (Sonar Sensor) ကို ရေတွင်း (သို့) ရေလှောင်ကန် ၏ အပေါ်ရေနှင့် အမြဲလွတ်ကင်းနိုင်သည့် နေရာတွင်တပ်ဆင် ထားပြီး အောက်သို့ မျက်နှာမူထားသည်။ သို့ မှသာလျှင် ဆိုနာဆင်ဆာ တည်ရှိရာ နေရာမှ ရေတွင်း (သို့) ရေလှောင်ကန် အတွင်းရှိ ရေ၏ မျက်နှာပြင်ကို အကွာအဝေးကို တိုင်းတာနိုင်မည်။

ရေတွင်း(သို့) ရေလှောင်ကန်၏ အောက်ခြေမျက်နှာပြင် အကွာအဝေး နှင့် ဆိုနာဆင်ဆာ ၏ ဖတ်ရှုထားသော ရေမျက်နှာပြင် အကွာအဝေး၏ ခြားနားခြင်းသည် အဆိုပါ ရေတွင်း (သို့) ရေလှောင်ကန် အတွင်းရှိရေ၏ အမြင့် အဖြစ်ယူဆနိုင်သည်။ ရေတွင်း (သို့) ရေလှောင်ကန် ၏ အတိုင်းအတာ နှင့် ရောက်ရှိနေ သော ရေမျက်နှာပြင်အမြင့်ကိုသိရှိလျှင် ရေသိုလှောင်ပမာဏကို တွက်ချက်နိုင်သည်၊ ကြိုတင်တွက်ချက်ထားသော အချက်အလက် များအရ အာဒူရီနို သည် ရေမျက်နှာပြင်အမြင့် နှင့် ရေပမာဏကို ဖေါ်ပြပေးနိုင်သည်။

အထက်ပါ ပုံတွင် ရေညီမျဉ်းအတိုင်းဖေါ်ပြခြင်းသည် ရေ၏အမြင့်ကို အလွယ်တကူ ဖတ်ရှုနိုင်ရန် အတွက် ဖြစ်သည်။



ရေ၏အမြင့်သည် ပထမ သတ်မှတ်အမြင့်ကို ရောက်ရှိသောအခါ အချက်ပေး သံကို စတင်ထုတ်လွှတ်မည်ဖြစ်သည်။ ဒုတိယ သတ်မှတ် အမြင့်ကို ရောက်ရှိသောအခါ အချက်ပေးသံကို ပို၍ မြန်စွာထုတ်လွှတ်ပြီး အဆိုပါ အမြင့်အောက်သို့ မရောက်မချင်း သို့မဟုတ် လူကိုယ်တိုင် မပိတ်မချင်း ဆက်လက် ၍ အချက်ပေးနေမည်။



အာဒွီနို သည် အောက်ပါ စကတ်ချ် (Sketch) ဖြင့် လုပ်ဆောင်မှုများကို ထိန်းချုပ်ပေးသည်။

```
/* -*- mode: c -*- */
/**
 * pozzo.pde
 * version: 1.2
 */

#include &lt;LiquidCrystal.h&gt;

#define PING_PIN 13
#define BUZZER_PIN 8
#define SWITCH_INT 0 /* 0 =&gt; pin 2 */
#define PI 3.1415926535898
#define SUPERFICE_BASE (R_POZZO * R_POZZO * PI)
#define SIZE_BAR (16 * 5)
#define ALARM_ICON 0 /* code */
#define SOUND_ICON 6 /* code */
#define SOUND_ICON_ON 7 /* code */

#define R_POZZO 0.5 /* raggio pozzo (m) */
#define H_POZZO 146.0 /* cm */
#define SOGLIA_ALLARME_1 100 /* cm */
#define SOGLIA_ALLARME_2 120 /* cm */
#define DELAY_0 60000 /* ms; 1000 * 60 * 1 = 1 min */
#define DELAY_1 600 /* ms */
#define DELAY_2 200 /* ms */
```

```
/* initialize the library with the numbers of the interface pins */
LiquidCrystal lcd(12, 11, 5, 4, 3, 6);

int mute = 0;

byte *getChar(int n, byte newChar[]) {
  int i;
  byte code[5] = {
    B10000,
    B11000,
    B11100,
    B11110,
    B11111};

  for (i = 0; i &lt; 8; i++)
    newChar[i] = code[n - 1];

  return newChar;
}

void setup() {
  int i;
  float h;
  byte newChar[8];

  /* set up the LCD&#039;s number of rows and columns: */
  lcd.begin(16, 2);

  for (i = 1; i &lt; 6; i++)
    lcd.createChar(i, getChar(i, newChar));

  newChar = {
    B00000,
    B00100,
    B01010,
    B01010,
    B11111,
    B00100,
    B00000,
  };
```

```
  newChar = {
    B00011,
    B00101,
    B11001,
    B11001,
    B11001,
    B00101,
    B00011,
  };

  lcd.createChar(SOUND_ICON, newChar);

  newChar = {
    B00100,
    B10010,
    B01001,
    B01001,
    B01001,
    B10010,
    B00100,
  };

  lcd.createChar(SOUND_ICON_ON, newChar);

  pinMode(BUZZER_PIN, OUTPUT);

  /**
   * LOW to trigger the interrupt whenever the pin is low,
   * CHANGE to trigger the interrupt whenever the pin changes value
   * RISING to trigger when the pin goes from low to high,
   * FALLING for when the pin goes from high to low.
   */
  attachInterrupt(SWITCH_INT, button, RISING);

  /* initialize serial communication */
  Serial.begin(9600);
}

void loop() {
  long hWatherCm;
```

```
  int litres;

  hWatherCm = read_height();
  if (check_alarm(hWatherCm) != 0) /* read again wather height */
    hWatherCm = read_height();

  lcd.clear();

  print_histogram(hWatherCm);

  lcd.setCursor(0, 1);

  lcd.print(hWatherCm);
  lcd.print(&amp;quot; cm - &amp;quot;);

  // litres = SUPERFICE_BASE * (hWather / 100.0) * 1000
  litres = floor(SUPERFICE_BASE * hWatherCm * 10);
  lcd.print(litres);
  lcd.print(&amp;quot; l &amp;quot;);

  lcd.setCursor(14, 1);
  lcd.write(SOUND_ICON);
  lcd.setCursor(15, 1);
  if (!mute)
    lcd.write(SOUND_ICON_ON);
  else
    lcd.write(&amp;#039;X&amp;#039;);

/*
  Serial.print(&amp;quot;cm = &amp;quot;);
  Serial.println(hWatherCm);
*/

  switch (check_alarm(hWatherCm)) {
  case 1:
    lcd.setCursor(0, 0);
    lcd.write(ALARM_ICON);

    buzz(200);
    delay(DELAY_1);
```

```
      break;

  case 2:
    lcd.setCursor(0, 0);
    lcd.write(ALARM_ICON);

    buzz(200);
    delay(200);
    buzz(200);
    delay(DELAY_2);
    break;

  case 0: // no alarm
    delay(DELAY_0);
  }
}

void print_histogram(int hWatherCm) {
  int i;
  int bloks;
  float histogram;

  // hWatherCm : HPOZZO = histogram : SIZE_BAR
  histogram = (SIZE_BAR * hWatherCm) / H_POZZO;
  histogram = histogram + 0.5;

  bloks = (int)histogram / 5;

  for (i = 0; i &lt; bloks; i++)
    lcd.write(5);

  if ((int)(histogram) % 5 &gt; 0)
    lcd.write((int)(histogram) % 5);
}

long read_height() {
  /**
   * establish variables for duration of the ping,
   * and the distance result in centimeters:
   */
```

```
    or more microseconds.
   * Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
   */
  pinMode(PING_PIN, OUTPUT);
  digitalWrite(PING_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(PING_PIN, HIGH);
  delayMicroseconds(5);
  digitalWrite(PING_PIN, LOW);

  /**
   * The same pin is used to read the signal from the PING))): a HIGH
   * pulse whose duration is the time (in microseconds) from the sending
   * of the ping to the reception of its echo off of an object.
   */
  pinMode(PING_PIN, INPUT);
  duration = pulseIn(PING_PIN, HIGH);

  /* convert the time into a distance */
  hWatherCm = H_POZZO - microseconds_to_centimeters(duration);

  if (hWatherCm &lt; 0)
    return 0;

  if (hWatherCm &gt; H_POZZO)
    return H_POZZO;

  return hWatherCm;
}

void buzz(int msec) {
  if (!mute)
    digitalWrite(BUZZER_PIN, HIGH);
  delay(msec);
  digitalWrite(BUZZER_PIN, LOW);
}

int check_alarm(int hWatherCm) {
  if (hWatherCm &gt; SOGLIA_ALLARME_1) {
     if (hWatherCm &lt; SOGLIA_ALLARME_2)
```

```
  return 0;
}

long microseconds_to_centimeters(long microseconds) {
  /**
   * The speed of sound is 340.29 m/s or 29.4 microseconds per centimeter.
   * The ping travels out and back, so to find the distance of the
   * object we take half of the distance travelled.
   */
  return microseconds / 29.387 / 2;
}

void button() {
  //  Serial.println(&amp;quot;Pulsante premuto&amp;quot;);
  mute = !mute;

  lcd.setCursor(15, 1);
  if (!mute)
    lcd.write(SOUND_ICON_ON);
  else
    lcd.write(&amp;#039;X&amp;#039;);
}
```

Translated by Friend Online Store

4.12.2018