Capacitive Fluid Level Sensor

by **dragonator** on September 27, 2014

Table of Contents

Capacitive Fluid Level Sensor . . . 1
Intro: Capacitive Fluid Level Sensor . . . 2
Step 1: How it works . . . 2
Step 2: What do you need . . . 3
Step 3: Cutting the tubes to length . . . 5
Step 4: Wrapping the outer tube . . . 6
Step 5: Assembling the inner tube . . . 7
Step 6: Assembling and sealing the tube . . . 8
Step 7: Drill holes . . . 9
Step 8: Output wire . . . 11
Step 9: Testing it . . . 11
Step 10: Adding the brain . . . 12
Barebone arduino . . . 12
Measuring capacitance . . . 12
Step 11: Soldering electronics . . . 13
Step 12: Firmware . . . 17
Uploading the firmware . . . 17
How the code works . . . 17
An important bug in Wire.h . . . 17
File Downloads . . . 18
Step 13: How to use it . . . 18
Step 14: Future improvements . . . 20
Related Instructables . . . 20
Advertisements . . . 20
Comments . . . 20

Intro: Capacitive Fluid Level Sensor

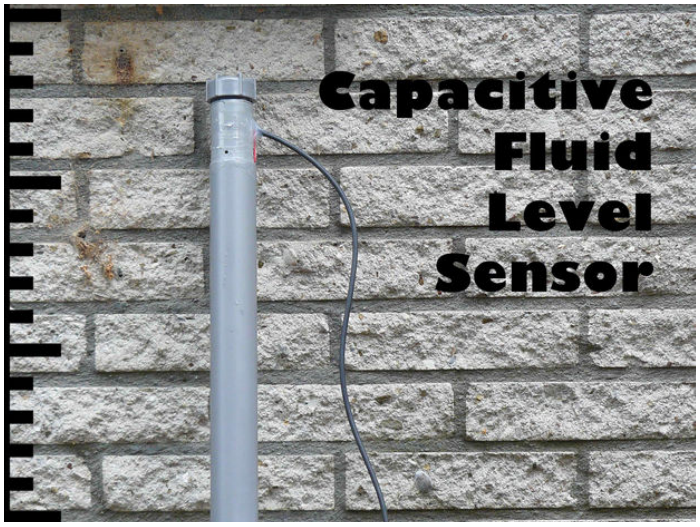
For the next spring I plan to make a fully automatic hydroponic setup. While spring is over 6 months away from now (it is pretty much autumn right now), there is still plenty to do. One of the things I need for automation is a water level sensor, so the system knows how much water is in the system (used for dosing the additives and warning me about low water level). This sensor can be used for anything that has a large height of liquid to be measured, most notably the water level in hydro- and aquaponics setups and ponds. It is not accurate to a mm, but it can tell the height within 1-2cm, more than enough for most applications.

There are many ways of measuring a level, including but not limited to: mechanical (with floats), optical, electrical (resistive), acoustically (ultrasonic) and electromechanical (pressure). The way I picked for my sensor is a Capacitive. It uses the different dielectric properties of various liquids and air to measure the level.

The advantage of capacitive level sensors is that they can be used for basically any solid and liquid. They have no moving parts and scales incredibly easy. They make no contact to the liquid or solid being measured, and so can also be used for more corrosive liquids. The disadvantage is that they need to be calibrated for the liquid being measured and that they can become weak when very large (though even at several meters this is not a problem).

The whole sensor can easily be built in a weekend if you have the materials laying around. It is made with stuff that technically isn't food safe, but it is aquarium and pond safe (silicone and PVC, both used for ponds). Initially I was going to seal the tubes with epoxy, but that can be rather toxic if not used correctly. The sensor is technically waterproof and completely weatherproof.

The sensor uses a barebone Arduino uno, but can also be controlled by any other uno like controller (when you make the electronics compartment larger). The sensor is controlled with I2C through another microcontroller that is I2C compatible.



Step 1: How it works

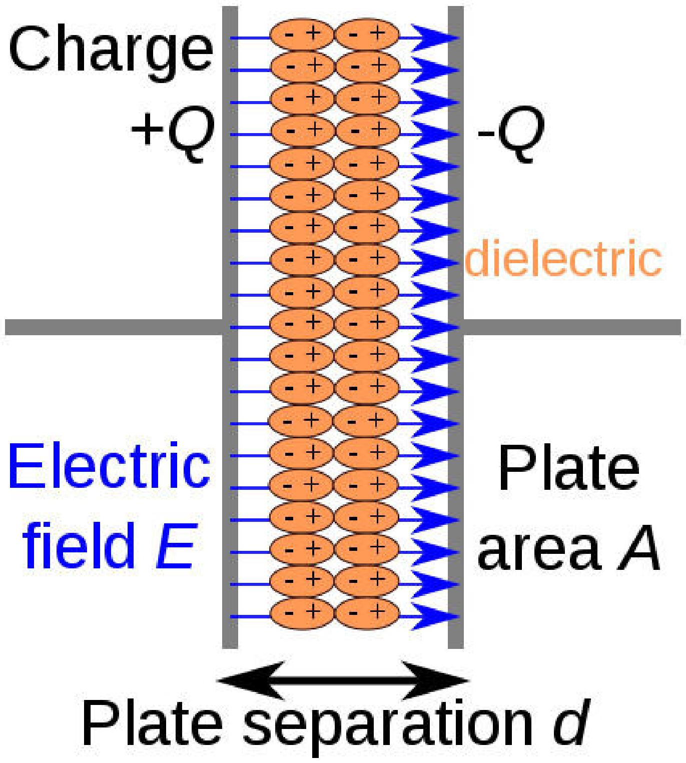
To understand this sensor you will need to understand capacitors. As always wikipedia is the best source for this information: http://en.wikipedia.org/wiki/Capacitor.

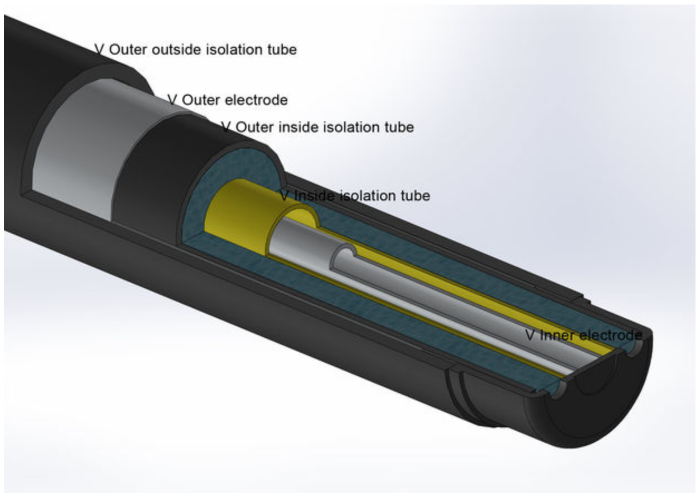
The TL;DR version is this. A capacitor has 2 conducting plates (Electrodes). When a charge is applied to these plates, the space in between the plates will also get a charge. The charge that can be between these plates depends on the material in between the plates (the Dielectric). The ability of a material to be charged is called relative permittivity. Vacuum has by definition 1, air is for all intents and purposes 1 and water at room temperature has 80. It is this difference that will be measured with the capacitive level sensor.

The capacitive level sensor has the 2 conducting plates in the form of 2 electrically isolated aluminium tubes, a smaller tube in a larger tube. The space between the tubes is the dielectric. When the tube is empty, the space is occupied by air. when the tube starts to fill, more and more of the space will be occupied by water. Water holds more charge than air and thus the capacitance will rise (mostly) linearly with the water level.

Determining capacitance is done by charging the capacitor and looking at how long it takes. The sensor doesn't bother with exact values, it just looks at the time it took and compares that to minimum and maximum values.

Sounds simple enough, now lets build one.





Step 2: What do you need

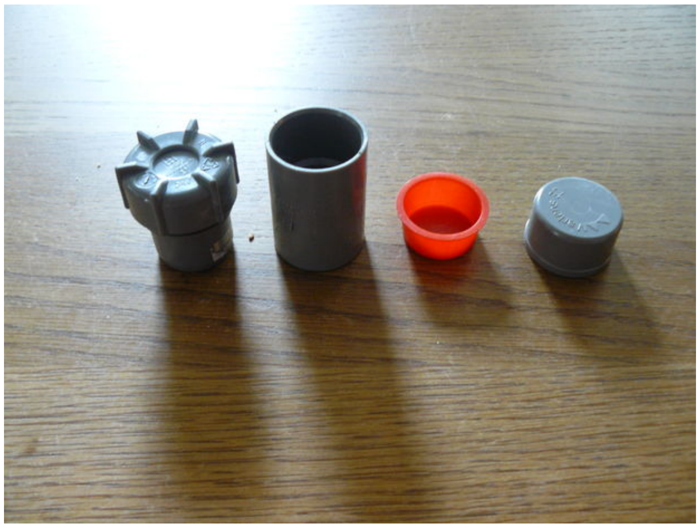
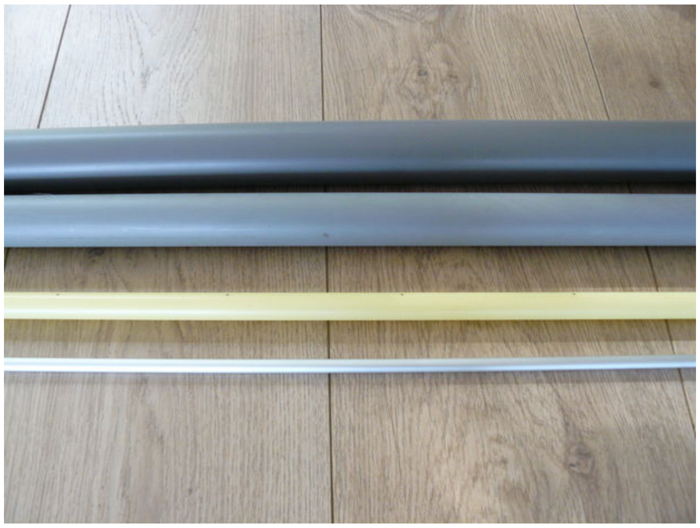
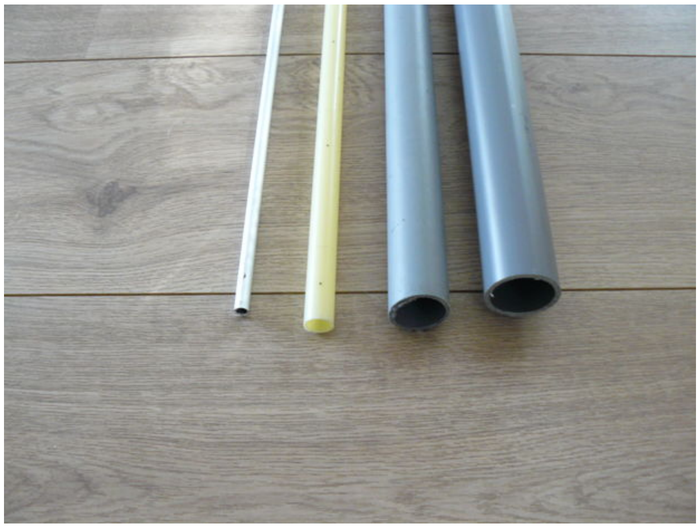
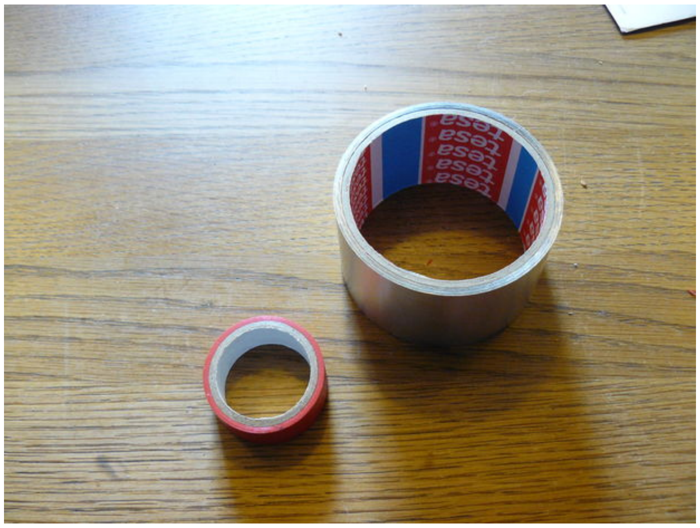
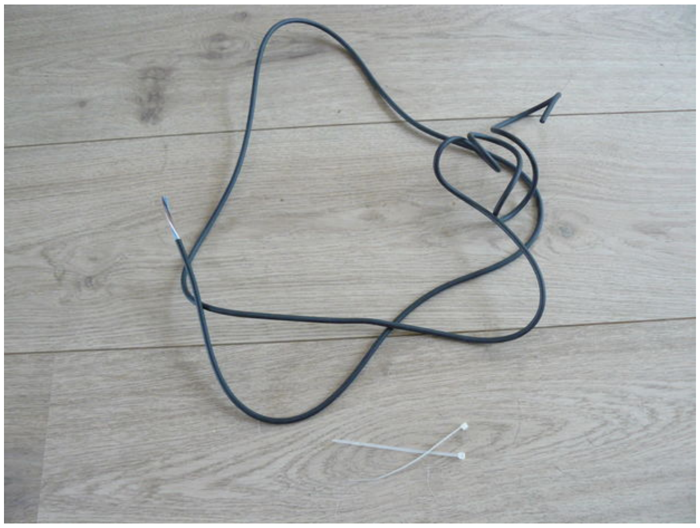
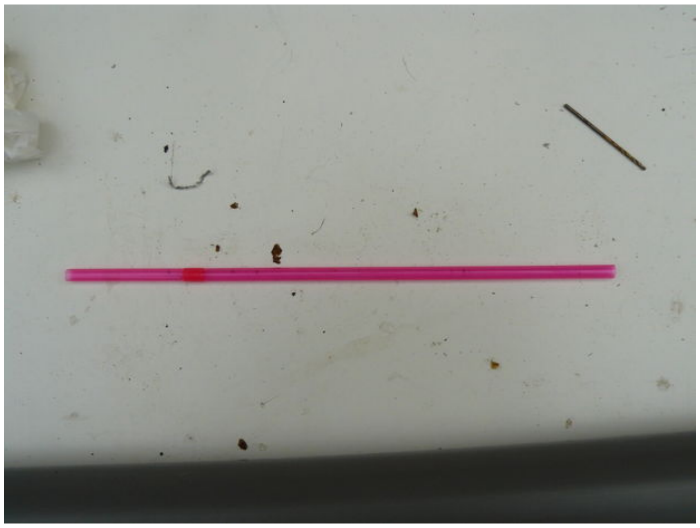
To make the level sensor, you will need several materials and tools. I will tell what I used, but the exact material really don't matter for the design. Also the exact lengths change with the length of the tube. The longest tube is the aluminium tube, all others are as long or slightly shorter. The caps I used are Dutch PVC fittings. I do not know how this should be built in the USA, so imagination is handy here.

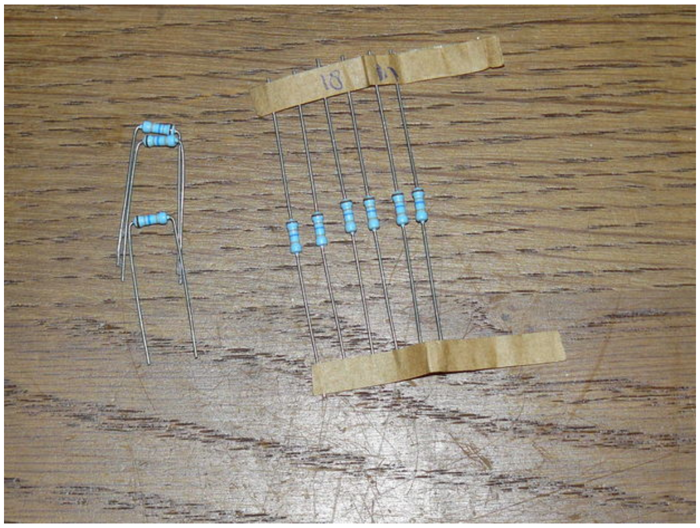
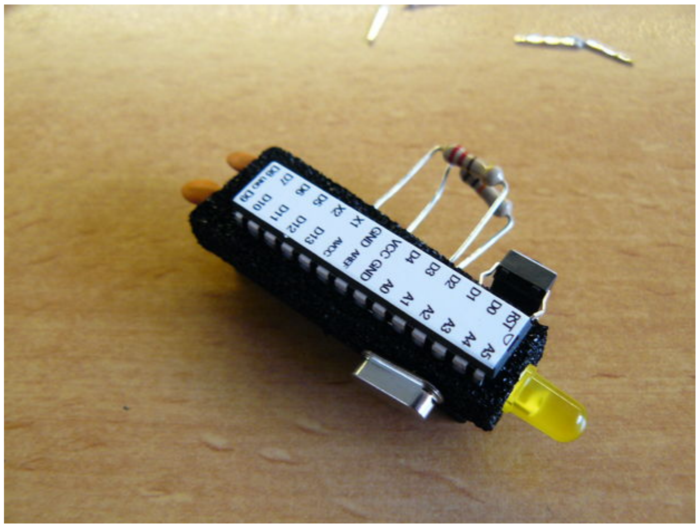
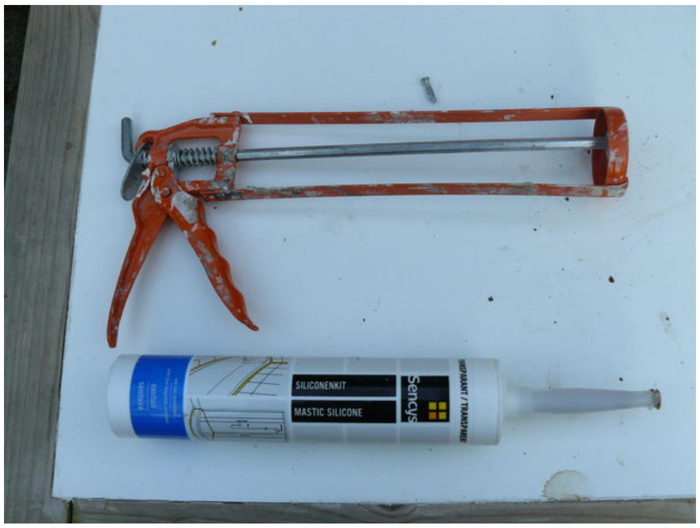
Materials:

- 40mm x 3.2mm PVC tube
- 32mm x 1.6mm PP tube (fits with a 1mm gap into the 40mm pvc)
- 5/8" electrical PVC tube (fits around the aluminium tube)
- 8x1mm aluminium tube
- 32mm PVC endcap
- 32mm PVC coupling
- 32mm PVC screwcap
- 32mm plug
- electrical tape
- Aluminium tape
- Silicone caulk
- Wire
- 4 pole wire
- Tie rips
- A straw
- 1 barebones Arduino uno (or any other mini arduino)
- 3 18Mohm resistors (or comparable high resistance resistors)
- Another Arduino to check and calibrate the sensor (and an Uno to program the barebone)
- 2 4k7ohm resistors for the I2C

Tools:

- A caulk gun
- A cordless drill
- A hacksaw
- Files
- Pliers
- Wire cutters
- Soldering equipment

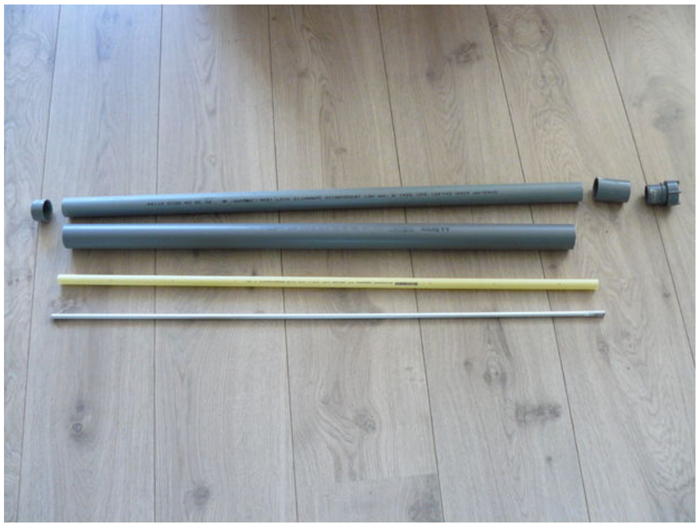
Step 3: Cutting the tubes to length

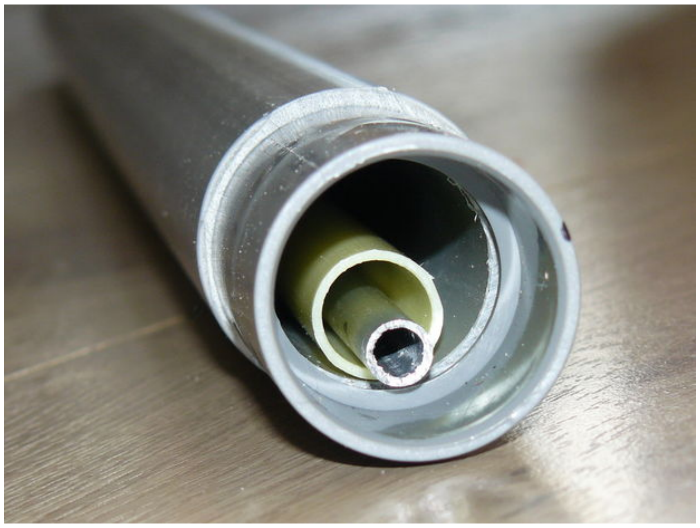
All tubes need to be cut to length. Exact lengths are not relevant here, just relative lengths. The tube around which these measurements are made is the 32mm tube.

The 40mm tube needs to be cut about 5cm shorter than the 32mm tube. This is so that when the coupler and the endcap are placed, none of the 32mm tube is visible.

The 5/8"electrical PVC tube needs to be a tiny bit longer than the 32mm tube. It needs to go through the cap placed in the top.

The aluminium tube needs to be around a cm longer than the electrical PVC tube. This is so that it is easy to seal the tubes and make a connection with the aluminium tube.

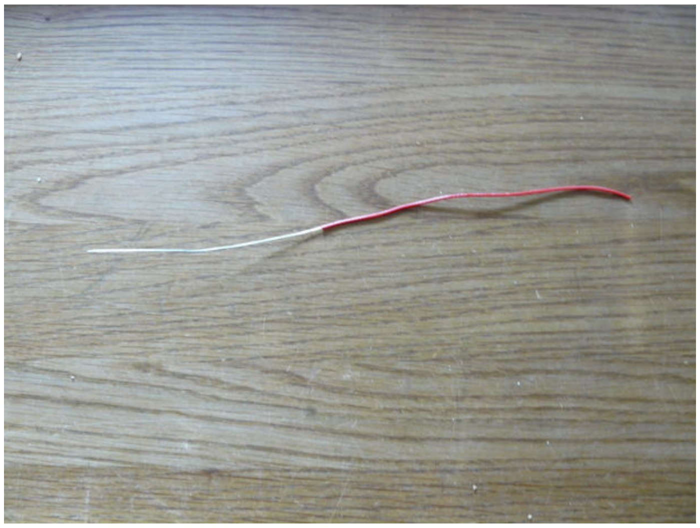
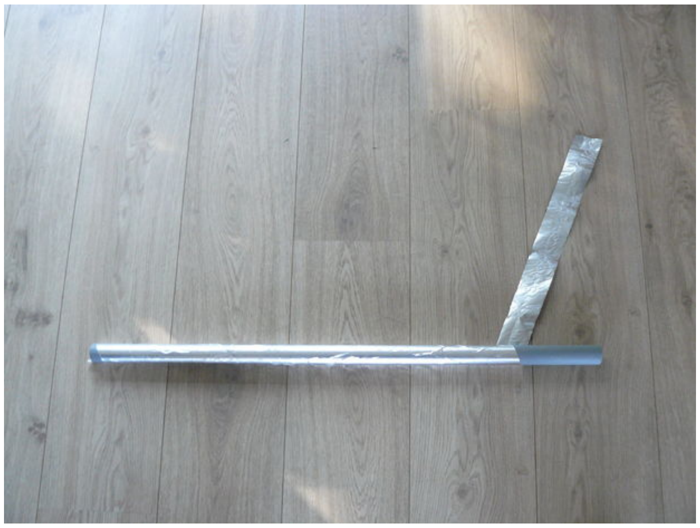


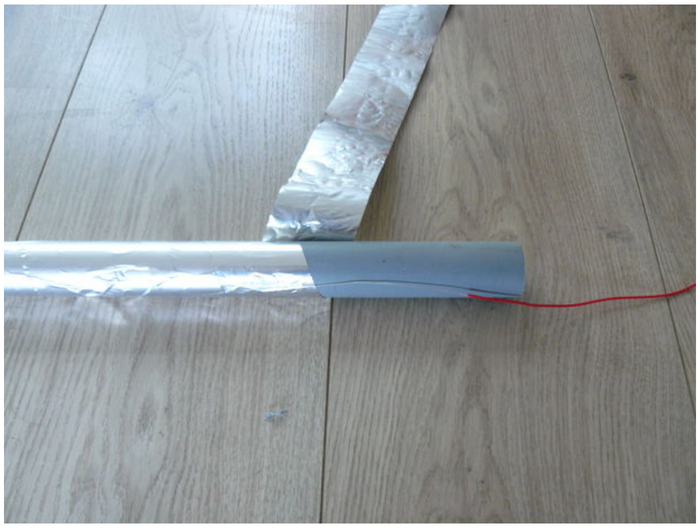


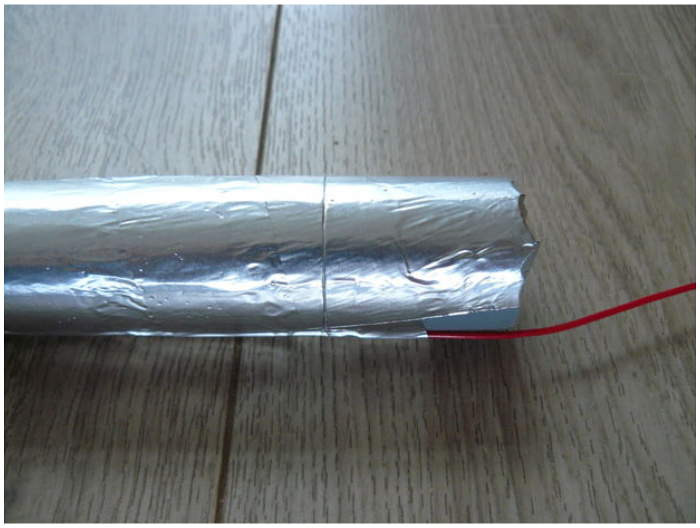
Step 4: Wrapping the outer tube

The outer electrode is made by wrapping aluminium tape around the 32mm tube. Using a solid aluminium tube would have been possible, but it was impossible to find one that could easily fit in PVC tubes.

Spiral the tape around the tube, periodically checking if the the overlap doesn't become too big. At the end, put a 20cm piece of wire with 10cm stripped under the tape. This wire will be connected to the microcontroller later. If the entire tube is wrapped, cut the aluminium tape 1cm above the end cap and the coupling, careful not to cut the wire.





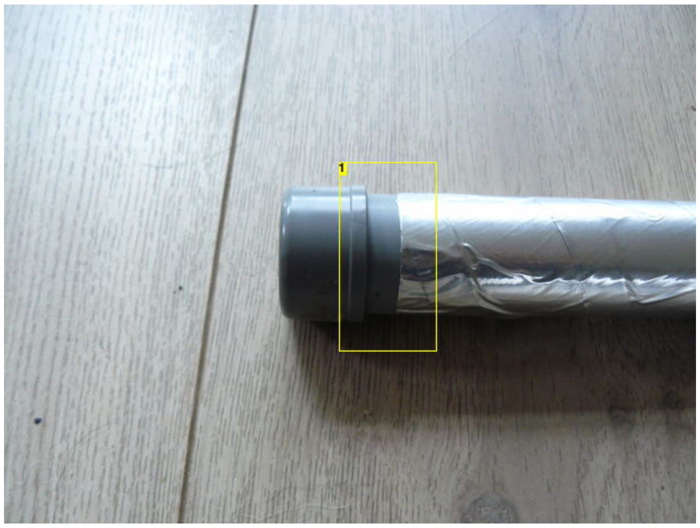
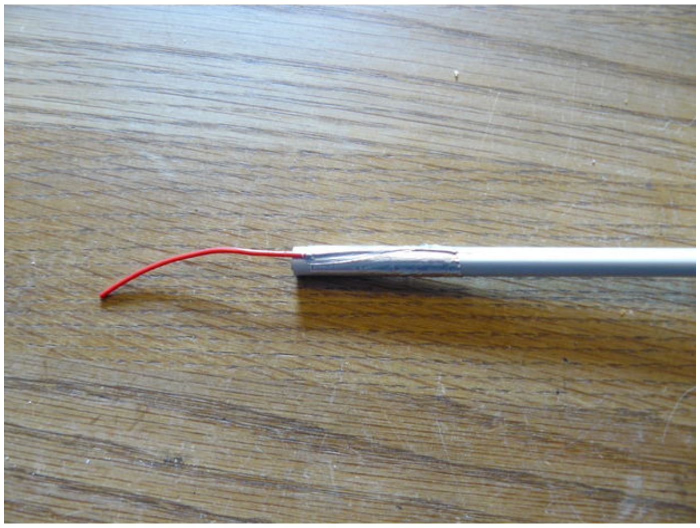
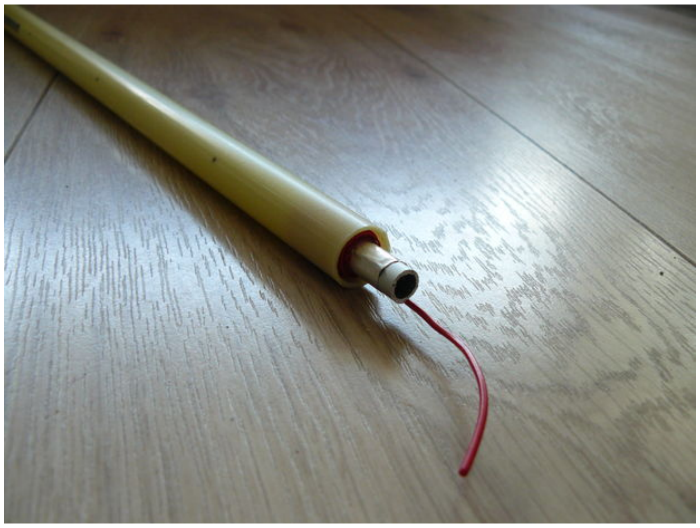


Image Notes
1. This gap here is needed for assembly

Step 5: Assembling the inner tube

For the inner tube the aluminium tube and the smallest PVC tube need to be merged, First use a piece of aluminium tape to tape a 10cm wire with 4cm stripped to the tube. Then use electrical tape to create a spacer for the small aluminium tube so it sits in the centre of the PVC tube. If you cut all tubes right, the aluminium tube should be a few centimetres longer than the PVC tube.





Step 6: Assembling and sealing the tube

With all of the tubes prepared, the whole sensor can be assembled. I apologize for the lack of pictures at this step, but my hands had silicone caulk on them most of the time, so I didn't pick up the camera as much as I should have. To compensate for this, a made a few renders with sections of how the sensor is supposed to be.

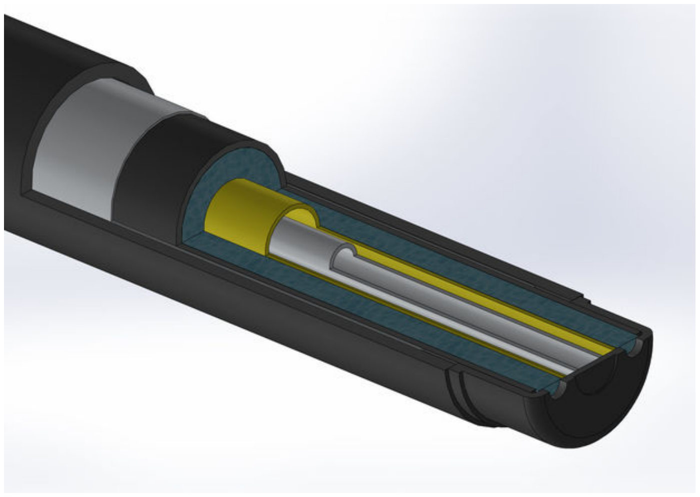
Initially I was going to seal the tubes with Epoxy. Epoxy would have made a stronger and more guaranteed seal, but is also toxic (Bisphenol A, has funny side effects) if not used properly. Rather than risking it, I went with the safer silicone.

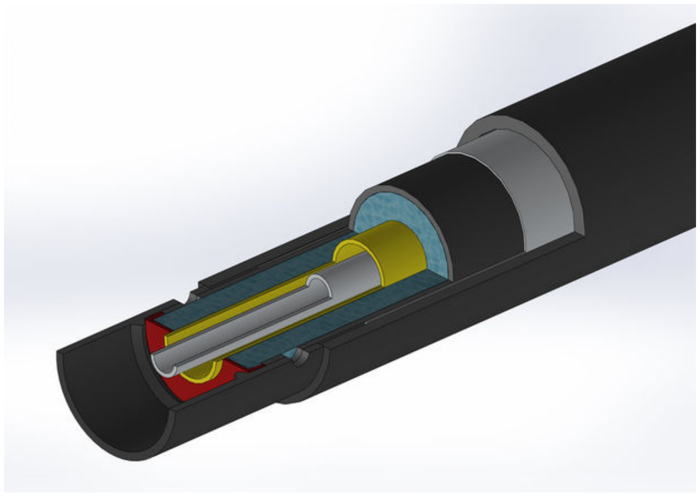
First put a generous dot of silicone in the endcap. Also put a small dab of silicone in the bottom of the inner tube (the side with no wire). Then push the bottom of the inner tube in the centre of the endcap. Push the wrapped 32mm tube in the endcap.

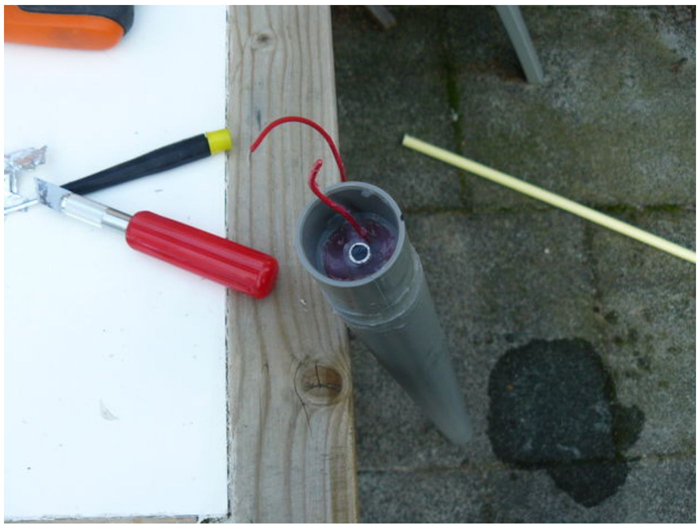
It might be wise to cut a small notch in the 40mm tube, to make space for the wire to pass the coupler. Put silicone caulk all the way around the endcap, then place the 40mm tube with the notch facing up around the 32mm tube.

Cut the 32mm plug so it fits in the coupler and cut a hole big enough for the 5/8" PVC to fit through. Put the plug in the endcap and test fit to make sure the 5/8" PVC goes through the plug. Then use silicone caulk anywhere where no water should be going (which is basically everywhere) and push the coupler over the 32mm tube. The glue the screwcap in the coupler.

Let the tube dry for a few hours before you continue.







Step 7: Drill holes

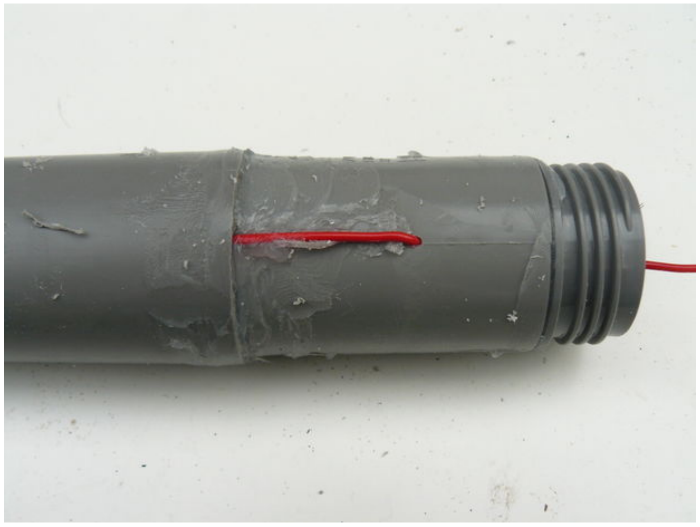
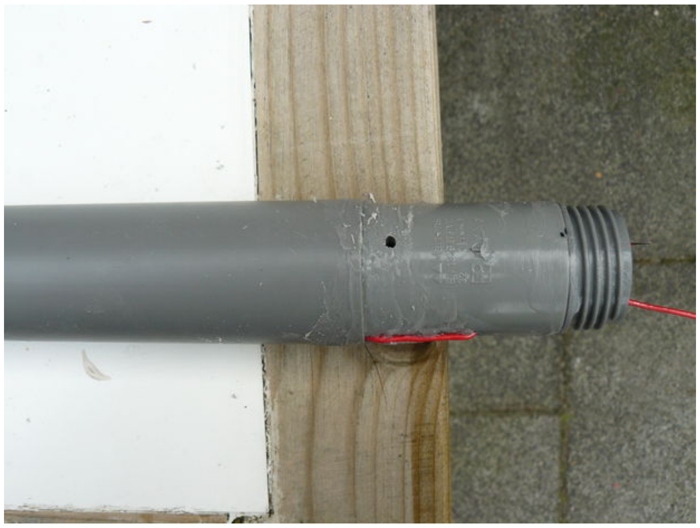
When the tube is dry enough to handle, the holes can be drilled. It is absolutely critical here that all the holes cut do not penetrate the centre tubes. Also do not drill anywhere where the outer tube was wrapped. None of the metal parts should ever make contact with the fluid being measured.It is safest to drill in the coupler and endcap.

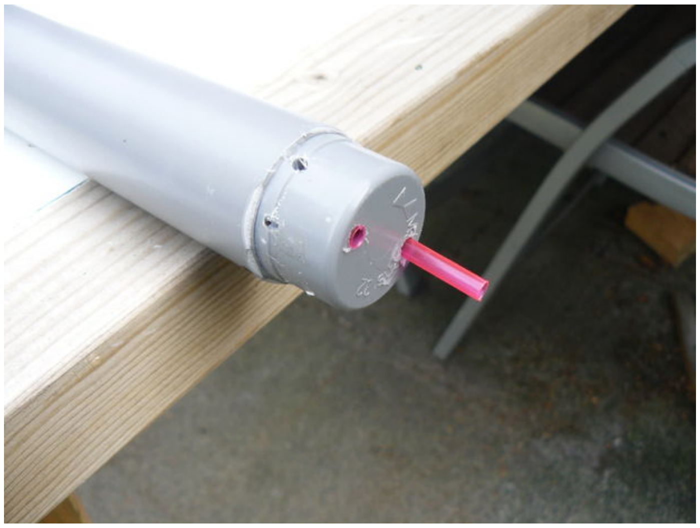
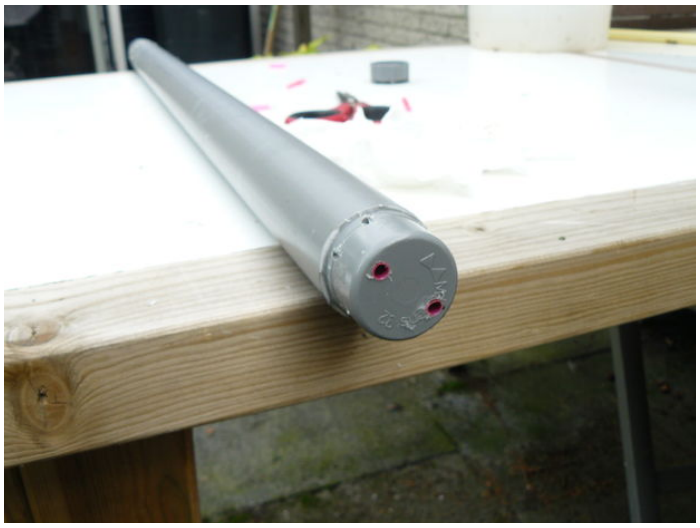
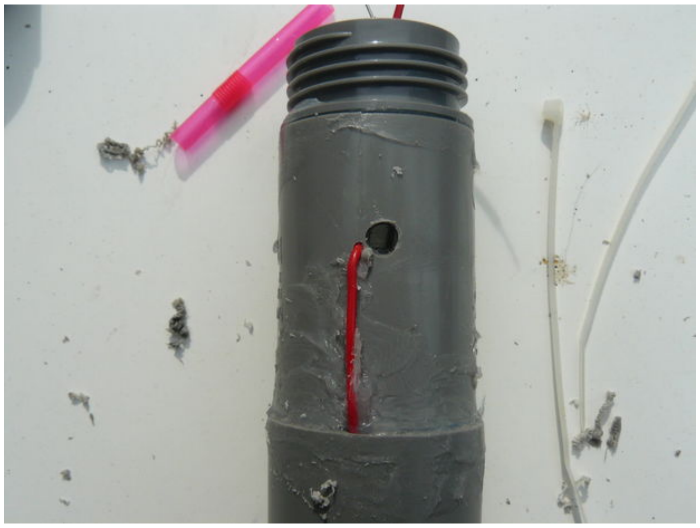
First the hole was drilled for the wire of the outer electrode. This wire was then pushed through the hole to the watertight inside of the sensor, where later the microcontroller will be housed. Simply use a drill that is the size of the wire to drill a hole through the coupler and inside of the screwcap holder. This hole will later be sealed with silicone.

All holes fluid holes were made with a 4.8mm drill, but this was only because this was the size of the straws. At the top, 2 holes were drilled through the coupler under the red cap. These holes allow air to escape when the water level in the tube is rising or falling.

At the bottom 2 holes were drilled in the endcap. These holes are between the 5/8" inner tube and the 32mm outer tube. Short pieces of straw (3-4cm) were pushed through the holes because the silicone was still wet and I feared the holes might collapse.

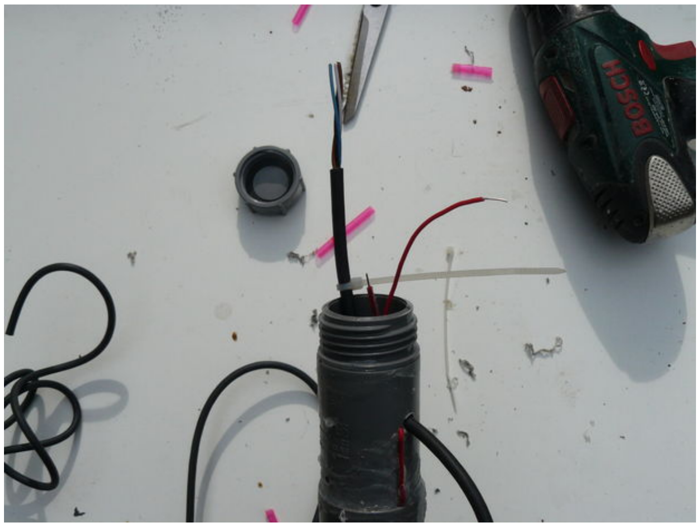
Last the 5mm hole for the output wire was drilled.

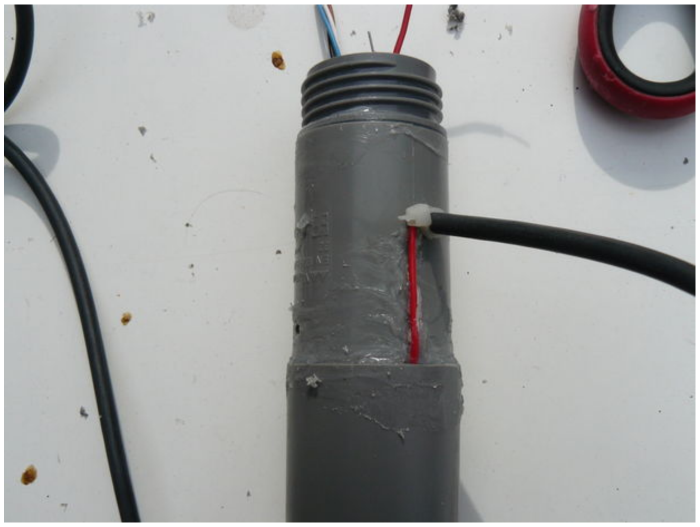



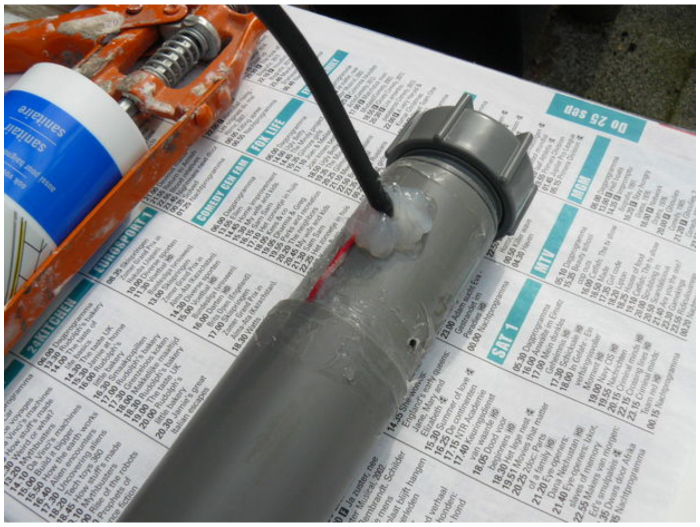




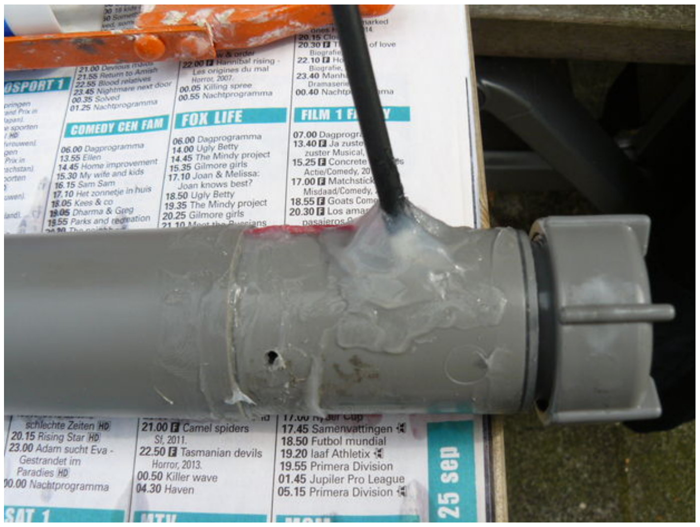
Step 8: Output wire

The output wire is a 4 pin wire that will connect the sensor through I2C to the main controller that will need the value. To get the wire in place, I used 2 tie rips. I ran the wire through the hole and tightened the first tie rip around the wire. Then I pulled the wire back until the tie rip hit the tube and I fastened another tie rip around the wire on the outside. Then I used silicone caulk to seal both the output wire and the outer electrode wire.









Step 9: Testing it

Before I added a microcontroller, I wanted to make sure the sensor actually worked. I used my trusty LC-meter to measure the capacitance of the sensor when it is empty. The value I got back is 44.6pF. This is incredibly low, but it is a measurable capacitance. Next I put the sensor in the pond to fill it up. When submerged around 50%, it had a capacitance of around 69.4pF. The sensor works.

Step 10: Adding the brain

Barebone arduino

Now it is time to add the brain of the sensor, the microcontroller. The microcontroller needs to be as close as possible to the sensor, so it is mounted right above it. At this point I do have to admit that I slightly misjudged the available space in the top of the sensor. Initially I wanted to use an Arduino nano, but that one didn't fit. Then I tried a Trinket 5V, but that lacked some key internal hardware so it couldn't measure the capacitance. I quickly ran out of options.

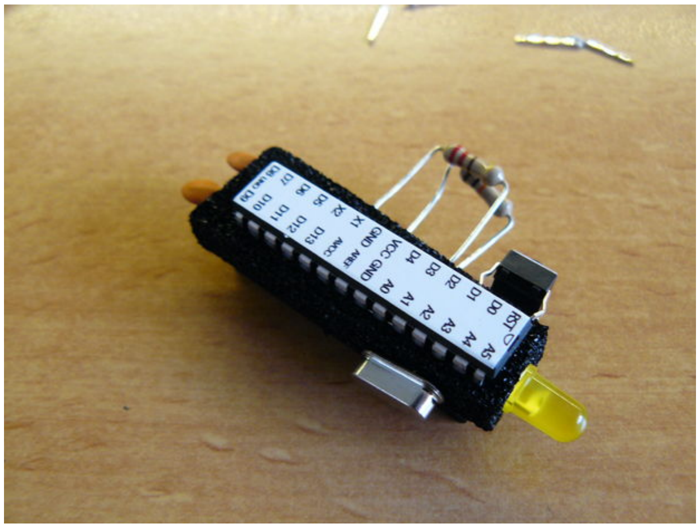
I went to the last thing I had left in stock that I knew would work, a barebone Arduino Uno. It is basically an Arduino uno, but without anything that isn't absolutely critical, like a USB to serial, a PCB and power circuitry. Programming it is a bit more of a hassle, but anyone using this instructable would probably only have to program it once. Click here for more information on the barebone Arduino. ->(Also here)<-

Now for the last issue, please don't yell at me for using I2C, it is the simplest 2 way protocol I could use and for the cable length I have, it will do just fine. In the ideal world I would have rather used a RS485 protocol as well, but it require additional hardware, which I didn't want to add. I2C does work up to 4-5 meters without any additional electronics, and I had no problems with this sensor.

With that information and apology out of the way, lets go onward to success.

Measuring capacitance

To measure capacitance, we will use the CapacitiveSensor library on Arduino (http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense). Using it the way it was intended would be pretty useless and leaves an inaccurate value, but the library does have one property that is really useful for us. It takes a measurable amount of time to measure the capacitance value and by running dozens (or 100) of samples and measuring how long it took, we can get a clear, consistent measurement of the capacitance. My sensor runs 100 samples for each measurement and it takes roughly 400ms to preform all those cycles when the tube is empty. When the tube fills this value rises to 690ms for 800mm.



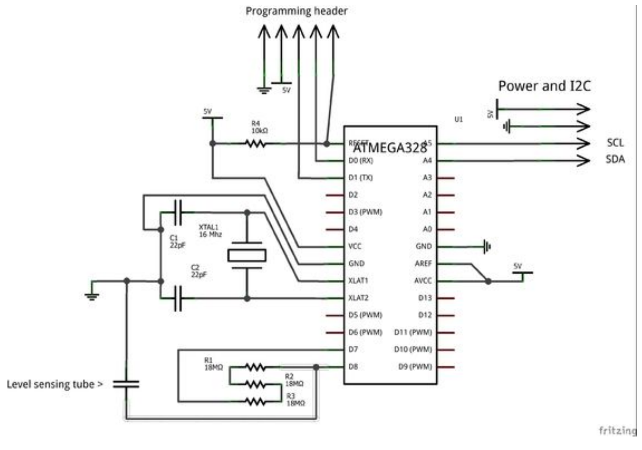
Step 11: Soldering electronics

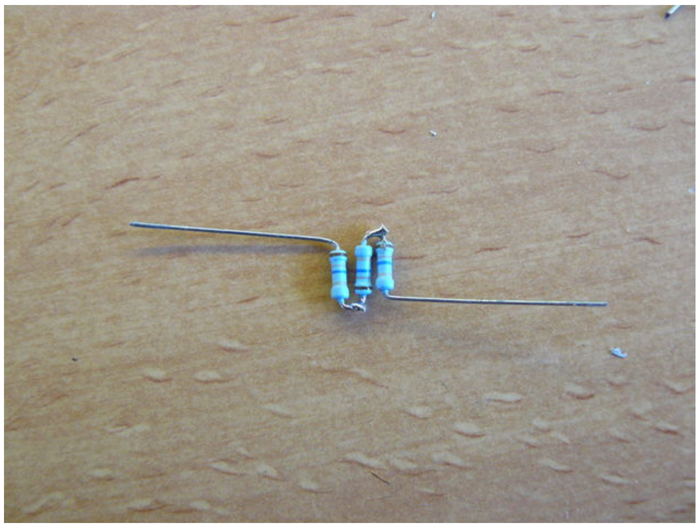
A barebone Arduino needs to be soldered. I decided to dead bug mine, to save space. The complete circuit was made following the schematic, so I won't really bother with explaining pin numbers, but I followed these steps:

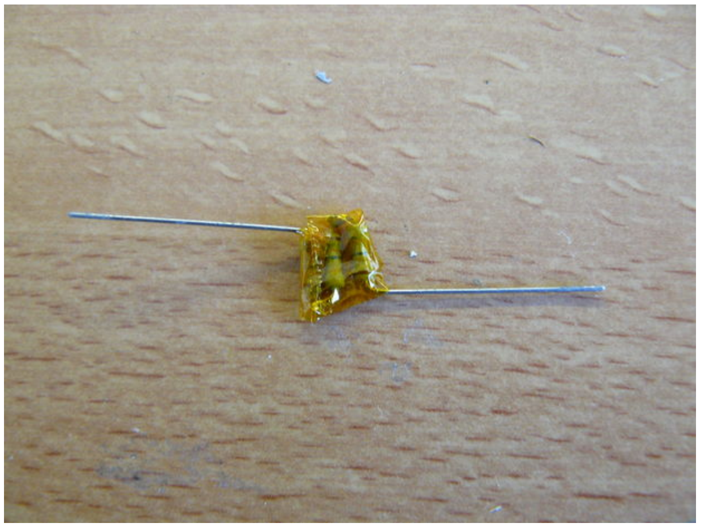
1. Solder 3 18Mohm resistors in series to make 1 54Mohm resistor;
2. Use tape to isolate the 54Mohm resistor;
3. Solder the crystal to the Xtal pins of the arduino;
4. Solder the 2 capacitors to the ground next to the Xtal pins;
5. Solder the 10k resistor from the 5V to the reset;
6. Solder all 5V lines together;
7. Solder the 2 ground pins together;
8. Solder wires to the I2C lines (SDA and SCL, A4 and A5);
9. Solder I2C power wires to the 5V and ground;
10. Solder a programming connector to the 5V, ground, reset, Rx and Tx;
11. Solder the 54Mohm resistor in between the measurement pins;
12. Solder a wire to one of the measurement pins and solder another wire to the ground (capacitor wires);
13. Tin the wires coming from the tube;
14. Solder the I2C and sensor measurement wires to the barebone Arduino;
15. When all is tested, stuff the Arduino in the tube and test again.

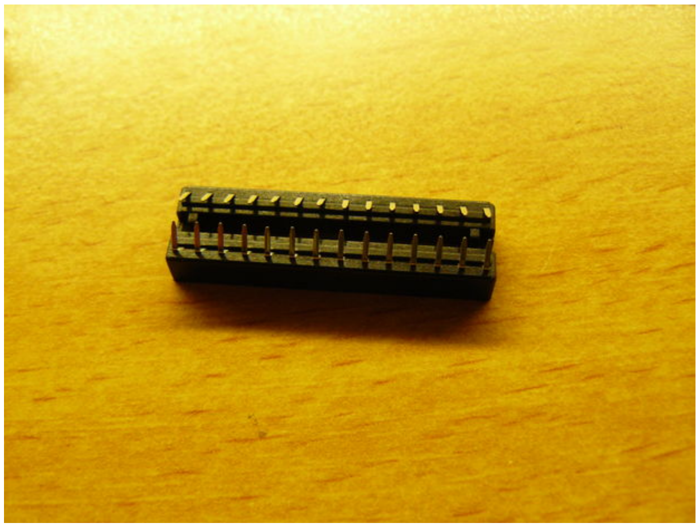
IMPORTANT:

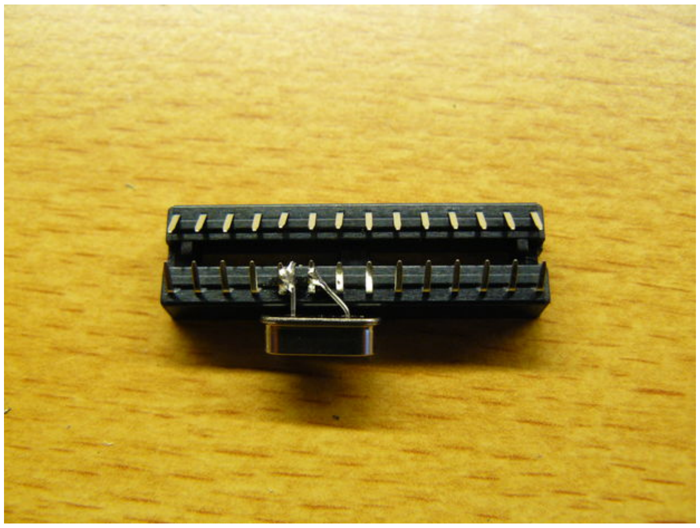
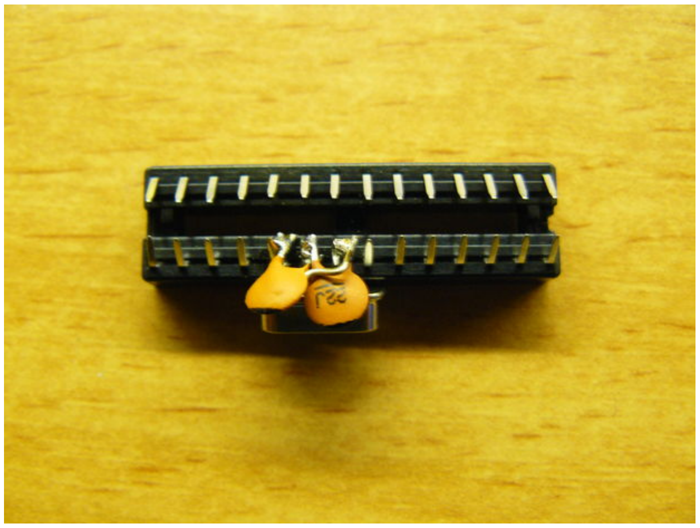
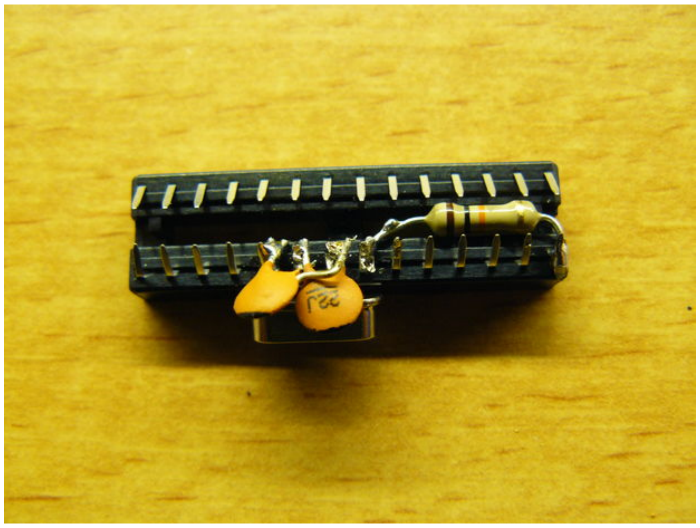
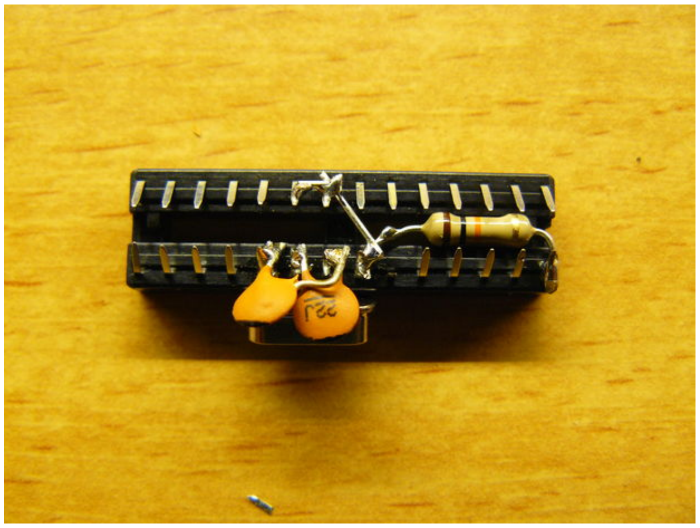
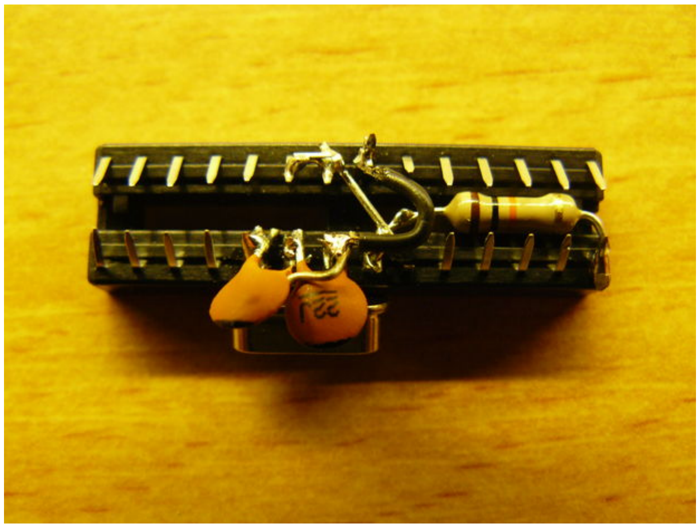
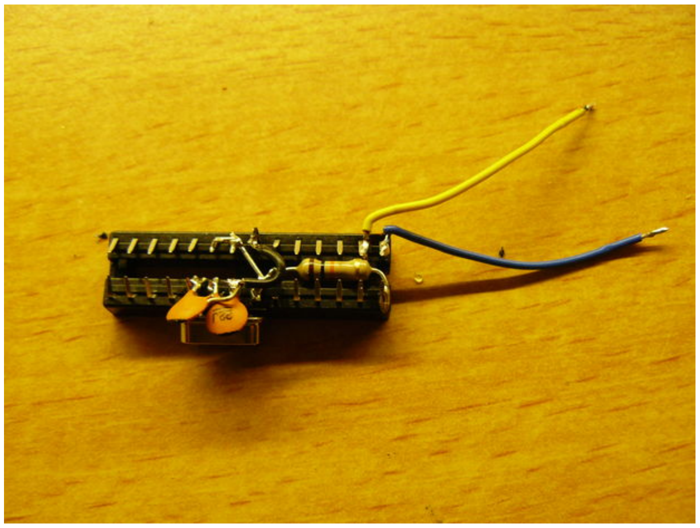
Polarity seems to matter when connecting the tube to the Arduino. I found that when I connect the inner tube to the ground and start measuring on the outer tube, I get NO readable value back. The inner tube needs to be the measured tube and the outer tube needs to be hooked up to ground.

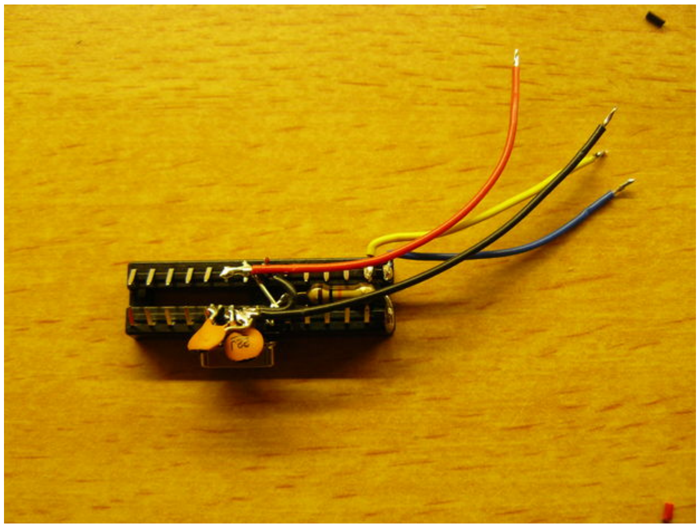
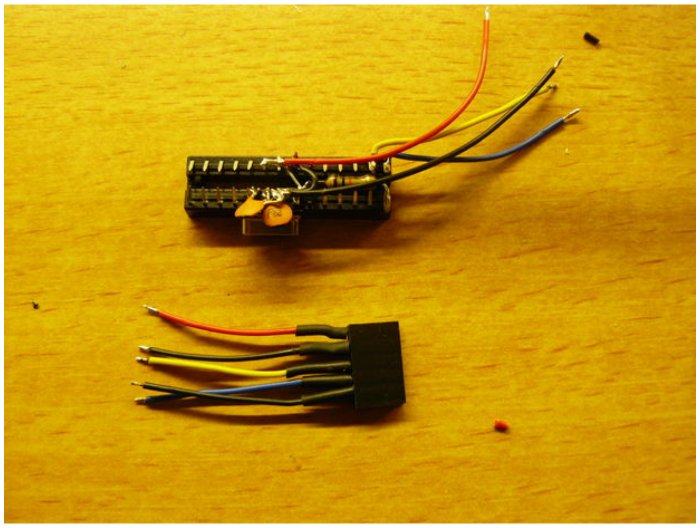
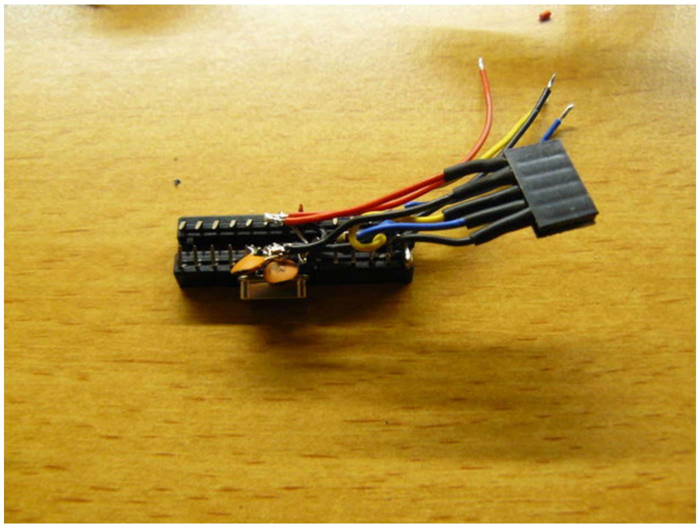
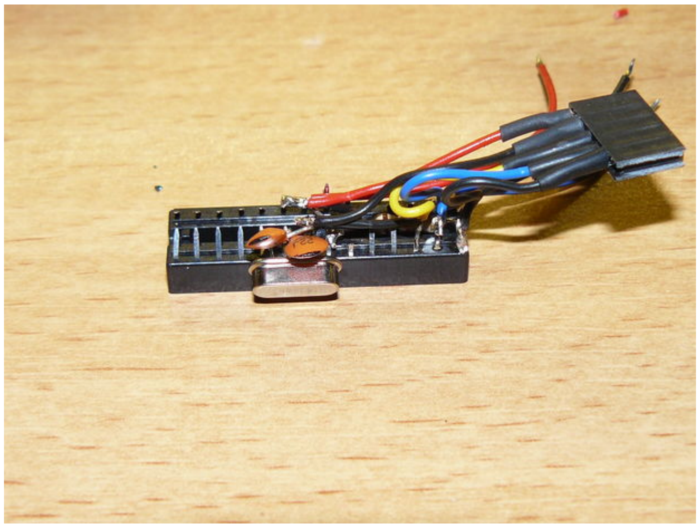
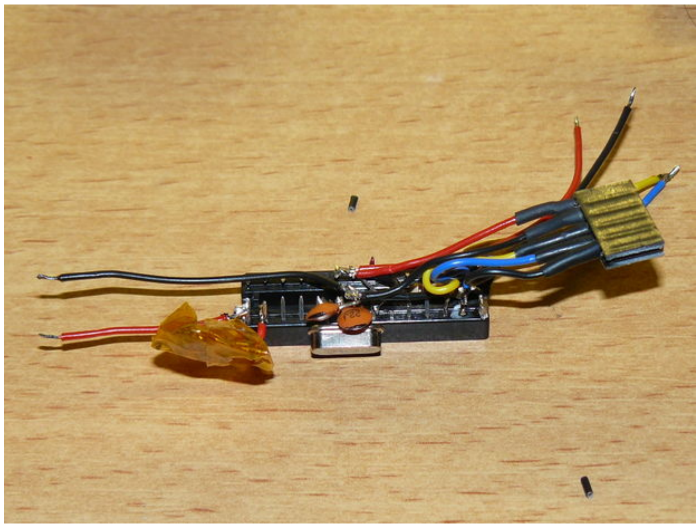
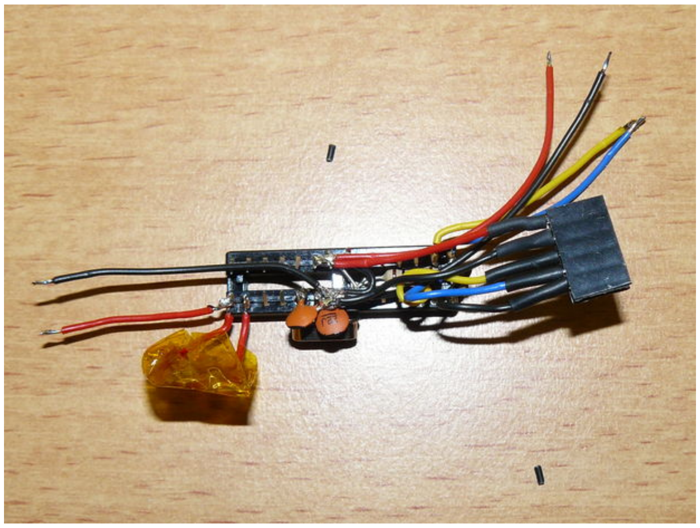


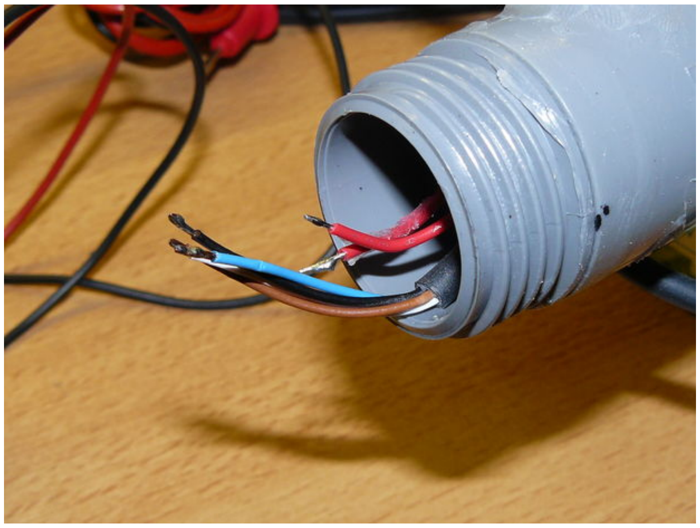


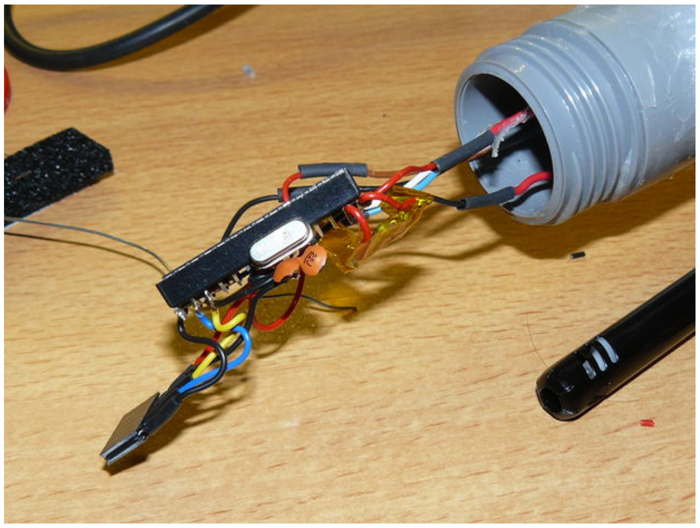


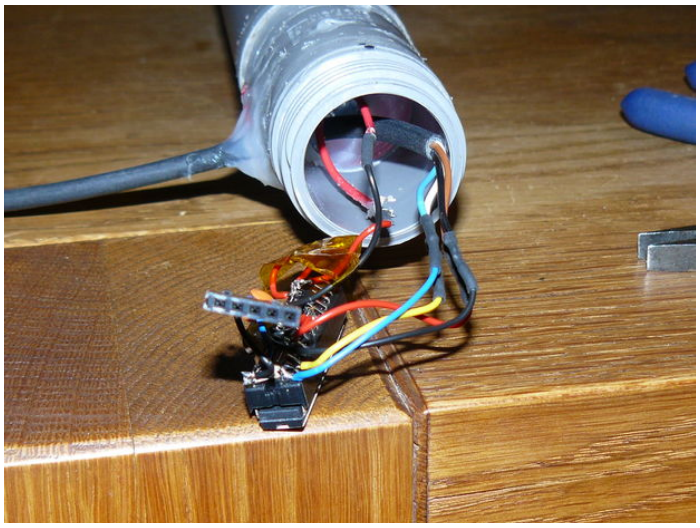


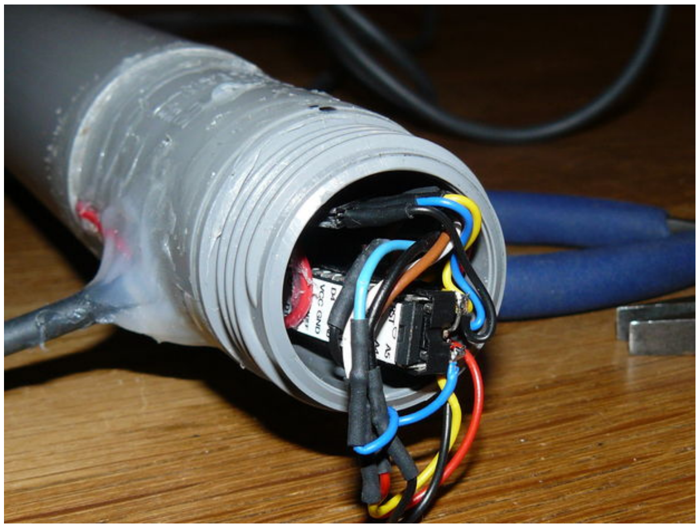







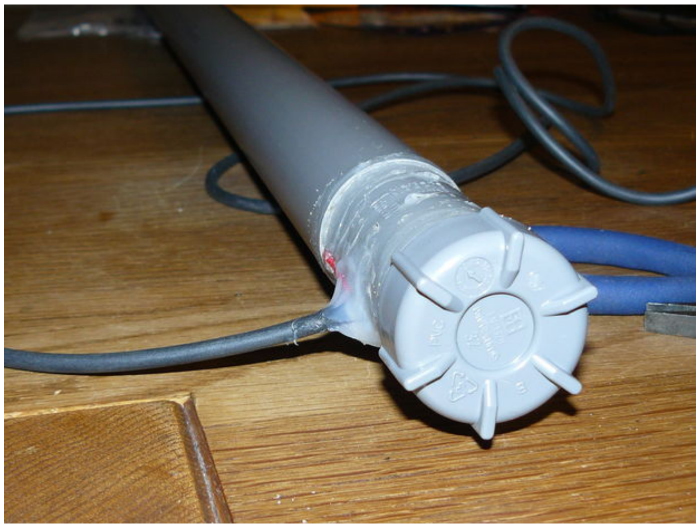

















Step 12: Firmware

Programming the barebone is pretty simple. All you need is an Arduino Uno with the microcontroller removed (it will see the barebone as the controller).

http://arduino.cc/en/Main/Standalone

http://arduino.cc/en/Tutorial/ArduinoToBreadboard

I soldered a header to the 5V, ground, reset, rx and tx, so I can connect my sensor's Arduino to an Arduino uno that has the Atmega chip removed. This way I can program the Arduino the way I would with any other Arduino.

You will also need to install the capacitiveSensor library to the Arduino IDE: http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense

Uploading the firmware

The firmware can be downloaded in the attachments. The CLS_Sensor_firmware is the firmware that needs to be uploaded to the capacitive level sensor. The CLS_sensor_reader firmware can be uploaded to another Arduino and used to communicate with the sensor. The reader Arduino needs to be connected to the sensor through I2C, with 2 4k7ohm pull-up resistors. In the next step all commands that can be used will be explained. You can also use snippets of code from the reader to add to your own code, so you can easily set and read the sensor without having to write the functions yourself.

How the code works

The sensor works fairly straight forward. Every N seconds it notes the time on the millis() timer. Then it starts 100 CapacitiveSensor cycles. Then it measures the time again and subtracts the start time from the end time. This is the raw value by which the sensor compares the fluid level. The sensor also makes a 0-255 value and a distance in mm from the raw value, by comparing it to the set minimum and maximum value.

Values are stored in EEPROM to make them non-volatile. When the sensor powers up, it will read the values from the EEPROM and start measuring.

Also the sensor has an I2C line to communicate with a master microcontroller. The I2C address of the sensor is 42. The master communicates with the sensor the following way.

```
//master request example
Wire.beginTransmission(42); //open communication
Wire.write(30); //tell the sensor what to do (30 = send raw value)
//additional writes can happen in case of 60.
Wire.endTransmission(); //end the transmission
Wire.requestFrom(42,2); //request 2 bytes from the sensor
if(Wire.available())    // if two bytes were received
{
 readValue[0] = Wire.read(); //read the first byte
 readValue[1] = Wire.read(); //read the second byte
}
```

The commands that can be sent to the sensor are:

- 10: send 0-255 level value (returns 1 byte);
- 20: send distance in mm (returns 2 bytes, first the high byte, then the low byte);
- 30: send raw measure time (returns 2 bytes, first the high byte, then the low byte);

- 40: set sensor 0% value (nothing additional happens);
- 50: set sensor 100% value (nothing additional happens);
- 60: set sensor length in the unit you want (2 additional bytes are expected to be sent that specify the length of the sensor your chosen unit)
- 70: read the set lower limit raw value (returns 2 bytes (high byte first) with the set lower raw value);
- 80: read the set upper limit raw value (returns 2 bytes (high byte first) with the set upper raw value);

- 90: read the set sensor length (returns 2 bytes (high byte first) with the set length in the unit you picked)

An important bug in Wire.h

The wire library is has a curious bug in it when 2 Arduino's are communicating. When the master has sent a requestFrom to the slave, the slave can not send back this way:

```
byte sendValue1;
byte sendValue2;
Wire.write(sendValue1);
Wire.write(sendValue2);
```

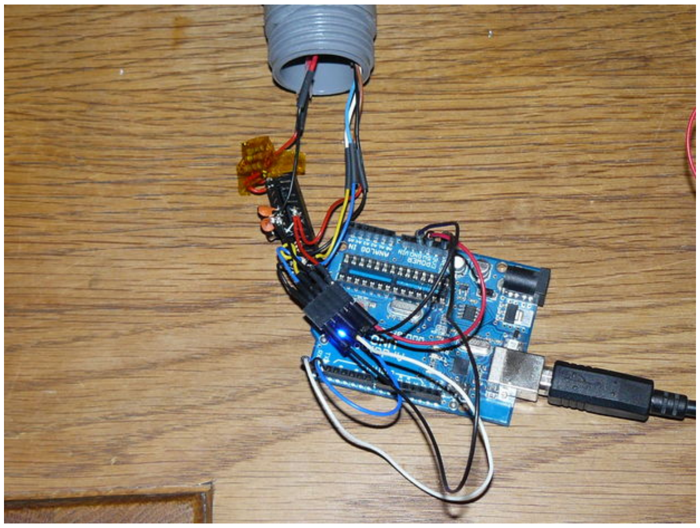
Instead, it needs to send back the value this way:

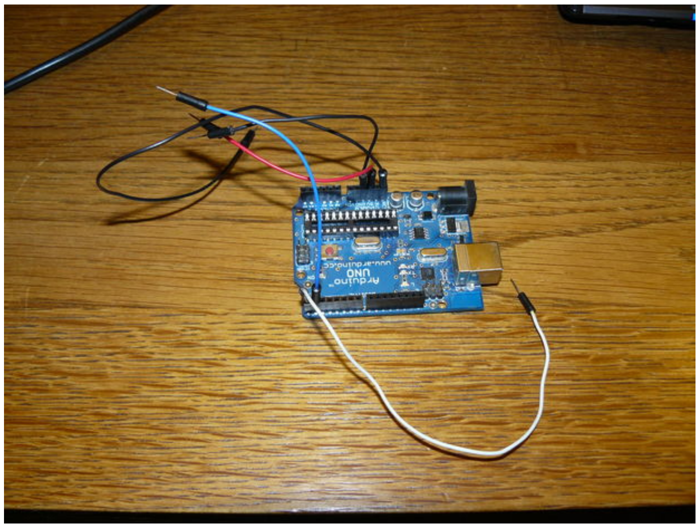
```
byte sendValue[2];
Wire.write(sendValue,2);
```

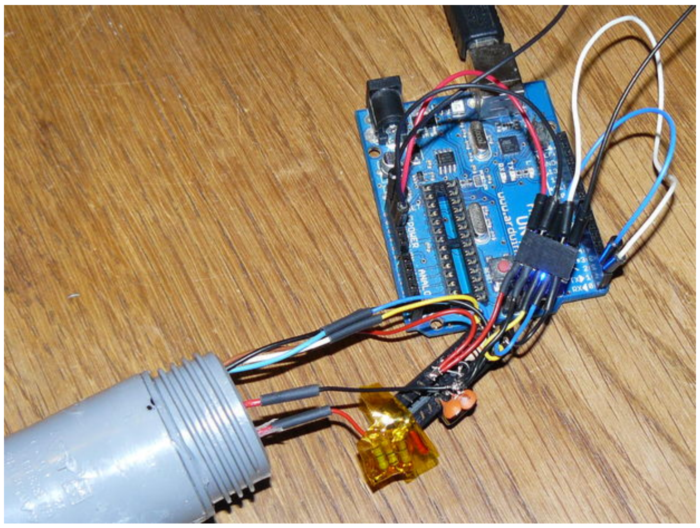
If not done this way, the master will only see the last byte sent, and all others are ignored. It might seem simple, but this issue here cost me 3 days to figure out.

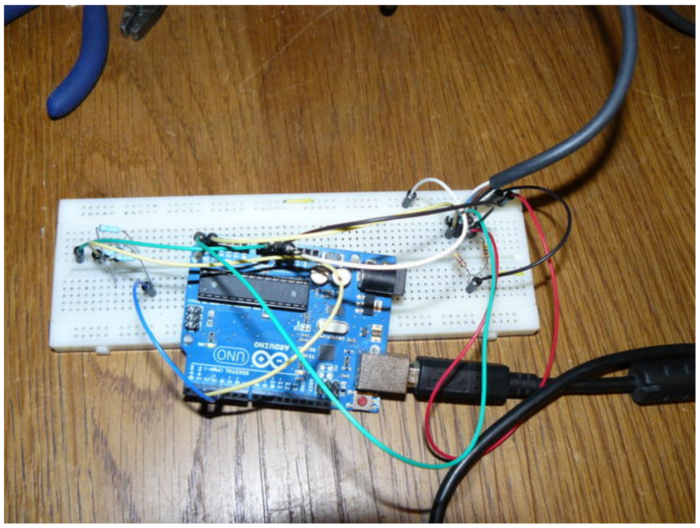
!!!3 DAYS!!!

(If it were possible I would also use capital 3).









File Downloads

CLS_sensor_reader.zip (1 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'CLS_sensor_reader.zip']

CLS_sensor_firmware.zip (2 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'CLS_sensor_firmware.zip']

Step 13: How to use it

The first step is to mount your sensor in a place where you want to use it. I do not have the set up I want to use the sensor in yet, so I will be using a tube filled with water to calibrate and test my sensor.There are 3 things that need to be set up before the sensor can be used. These 3 steps will let the sensor know what values correspond to what values. These values are stored in the EEPROM memory, so even when the sensor is powered off, the values will be retained:

Step 1:
The measuring range of the sensor needs to be set. This is so that the sensor can return the height of the fluid in millimeters (or any other value you want to use really, it is a unit-less value). This can be done through I2C by doing this:

```
word setValueMM = 800;
byte tempSendValue[3] = {60, highByte(setValueMM), lowByte(setValueMM)};
Wire.beginTransmission(42);
Wire.write(tempSendValue, 3);
Wire.endTransmission();
```

Or through the Arduino with the sensor reader firmware and the serial command: 'S'. You will have to modify the value setValueMM in the firmware of the sensor reader to the correct length.

Step 2:

Set the lower limit of the sensor. Fill the tube to the level you want to be registered as 0. This can be completely empty, this can be halfway. When the sensor is filled with the level you want to set as 0, use the following command:

```
Wire.beginTransmission(42);
Wire.write(40);
Wire.endTransmission();
```

Or through the Arduino with the sensor reader firmware and the serial command: 'L'.

Step 3:

Step 3 looks a lot like step 2, just have the level of the sensor to the value you want to have registered as full, 255 in case of a read 0-255, or the length you have specified in step 1. When the tube is at the full level, use this command in Arduino to set the upper limit.

```
Wire.beginTransmission(42);
Wire.write(50);
Wire.endTransmission();
```

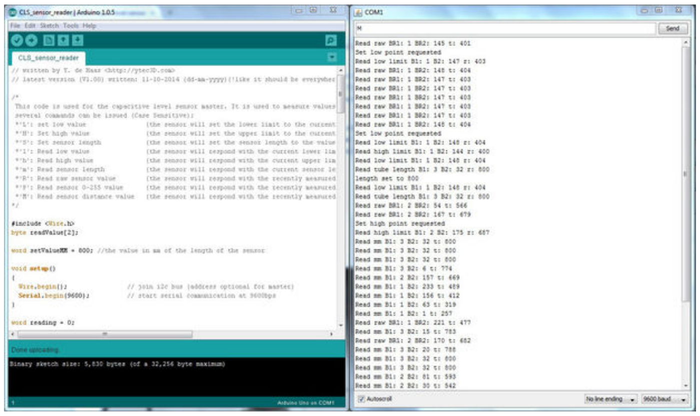
Or through the Arduino with the sensor reader firmware and the serial command: 'H'.

Using it

The sensor is now ready for use. To measure it using the sensor reader Arduino, open the serial window. Here you can talk to the reader Arduino, which will in term talk to the sensor. There are 9 commands you can use, they are case sensitive:

- 'L': set sensor lower value;
- 'H': set sensor upper value;
- 'S': set sensor tube length (value specified in firmware);
- 'l': read set sensor lower value;
- 'h': read set sensor upper value;
- 'm': read set sensor tube length;
- 'R': read current sensor raw value (is in ms);
- 'P': read the sensor 0-255 value;
- 'M': read the sensor level value (in the units you specified);

Look in the firmware of the reader for the snippets responsible for getting and setting the values. You can use these snippets of code in your own Arduino projects (or really any other microcontroller) to control the sensor.



```
// written by Y. de Haas <http://ytec3D.com>
// latest version (V1.00) written: 11-10-2014 (dd-mm-yyyy)(!like it should be everywher

/*
 This code is used for the capacitive level sensor master. It is used to measure values
 several commands can be issued (Case Sensitive):
 *'L': set low value               (the sensor will set the lower limit to the current
 *'H': Set high value              (the sensor will set the upper limit to the current
 *'S': Set sensor length           (the sensor will set the sensor length to the value
 *'l': Read low value              (the sensor will respond with the current lower lim
 *'h': Read high value             (the sensor will respond with the current upper lim
 *'m': Read sensor length          (the sensor will respond with the current sensor le
 *'R': Read raw sensor value       (the sensor will respond with the recently measured
 *'F': Read sensor 0-255 value     (the sensor will respond with the recently measured
 *'M': Read sensor distance value  (the sensor will respond with the recently measured
*/

#include <Wire.h>
byte readValue[2];

word setValueMM = 800; //the value in mm of the length of the sensor

void setup()
{
  Wire.begin();              // join i2c bus (address optional for master)
  Serial.begin(9600);        // start serial communication at 9600bps
}

word reading = 0;
```

```
Done uploading.
Binary sketch size: 5,830 bytes (of a 32,256 byte maximum)
```

```
M
Read raw BR1: 1 BR2: 145 t: 401
Set low point requested
Read low limit B1: 1 B2: 147 r: 403
Read raw BR1: 1 BR2: 148 t: 404
Read raw BR1: 1 BR2: 147 t: 403
Read raw BR1: 1 BR2: 147 t: 403
Read raw BR1: 1 BR2: 147 t: 403
Read raw BR1: 1 BR2: 147 t: 403
Read raw BR1: 1 BR2: 147 t: 403
Read raw BR1: 1 BR2: 148 t: 404
Set low point requested
Read low limit B1: 1 B2: 148 r: 404
Read high limit B1: 1 B2: 144 r: 400
Read low limit B1: 1 B2: 148 r: 404
Read tube length B1: 3 B2: 32 r: 800
length set to 800
Read low limit B1: 1 B2: 148 r: 404
Read tube length B1: 3 B2: 32 r: 800
Read raw BR1: 2 BR2: 54 t: 566
Read raw BR1: 2 BR2: 167 t: 679
Set high point requested
Read high limit B1: 2 B2: 175 r: 687
Read mm B1: 3 B2: 32 t: 800
Read mm B1: 3 B2: 32 t: 800
Read mm B1: 3 B2: 32 t: 800
Read mm B1: 3 B2: 6 t: 774
Read mm B1: 2 B2: 157 t: 669
Read mm B1: 1 B2: 233 t: 489
Read mm B1: 1 B2: 156 t: 412
Read mm B1: 1 B2: 63 t: 319
Read mm B1: 1 B2: 1 t: 257
Read raw BR1: 1 BR2: 221 t: 477
Read mm B1: 3 B2: 15 t: 783
Read raw BR1: 2 BR2: 170 t: 682
Read mm B1: 3 B2: 20 t: 788
Read mm B1: 3 B2: 32 t: 800
Read mm B1: 3 B2: 32 t: 800
Read mm B1: 2 B2: 81 t: 593
Read mm B1: 2 B2: 30 t: 542
```

Step 14: Future improvements

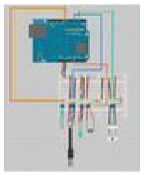
- Move the tubes closer together and have less air in the sensor. Currently the inner tube has a thick layer of air between it and the inner insulator. This air drastically reduces the capacitance of the sensor. By choosing either a bigger inner electrode or a smaller inner insulator, The gap can be drastically reduced.

- Better microcontroller (and more space). Because I misjudges the space available for the controller, I ended up wasting a lot of time getting a working controller. Soldering a barebone arduino cost me a lot of extra time. If I were to design another one, I would make a bigger more accesible microcontroller space that can hold a decent controller.

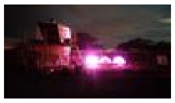
- No more I2C. I2C was the only communication protocol I had on the trinket, but it is designed for use on a circuitboard, not for use in cable separated components. This means that it can only bridge a very limited distance, if no other cables attached, 3-5 meters. RS485 would have been better, but the trinket doesn't support this.

- Get the electrodes closer to each other. Currently, there is quite a big gap between the electrodes. This is very inefficient. A smaller gap makes a capacitor that can hold a larger charge. The gap between the tubes only needs to be big enough to not clog, there are no additional requirements to the gap.

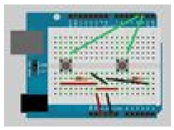
Related Instructables


Aquaponics: EnvDAQ upgrade with Water Temperature Sensor (Grow Bed DAQ) by IAquaponics

Hydroponics with Reclaimed Materials by pumpkinhead24


Hackerspace Earthship (HAESH): Homemade Environmental Control Life Support System (E.C.L.S.S.) by haeshproject

How To Use Sensors for Flood Alerts and Water Detection by edward_Valarm


Aquaponics: Arduino Email & Text Messaging by IAquaponics

How to build a desktop aquaponics system for indoor gardening. (video) by Get Forked

Comments

2 comments **Add Comment**

carlos66ba says: Oct 11, 2014. 10:56 AM **REPLY**
Very good instructable and excellent idea! Thank you for sharing this.

MsSweetSatisfaction says: Oct 11, 2014. 10:49 AM **REPLY**
Very interesting design, I can't wait to see the hydroponic set up! Thanks for sharing!