Water Level Sensor report

Method 1: voltage measurement between two immersed rods (resistive based)

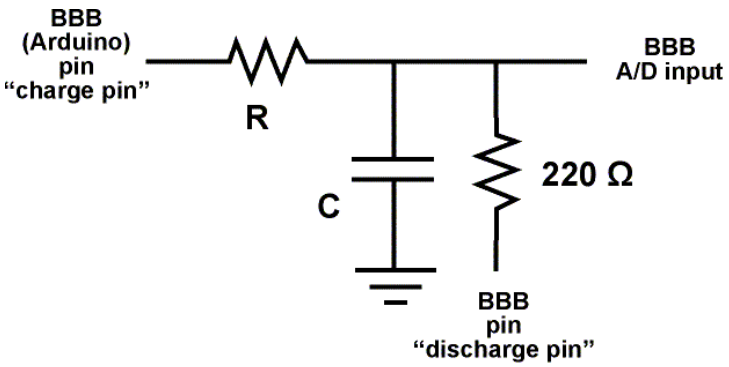
The method proved to be of high tolerance and very low accuracy, stability and repeatability.

Method 2: Capacitance measurement using Arduino

Charge and discharge the capacitor and measure the capacitance using the charging duration and the value of the resistance used.

Demonstration and code available on: https://www.arduino.cc/en/Tutorial/CapacitanceMeter

Arduino connections:



Schematic Diagram for Capacitance Meter

The method is accurate and stable for capacitance the microfarad order, not less. The two aluminum plates have a capacitance of the picofarads order, for which the values returned by the Arduino suffer from accumulation and overflow. I couldn't find a concrete explanation for that. However, the increase in capacitance can be detected by the user, but difficult to handle by a script. The number of measurement per second for a resistance of 70MOhm in 500. Such a high resistance is required since the Arduino clock can only report values in microseconds at most. For this reasons I have disregarded this method currently.

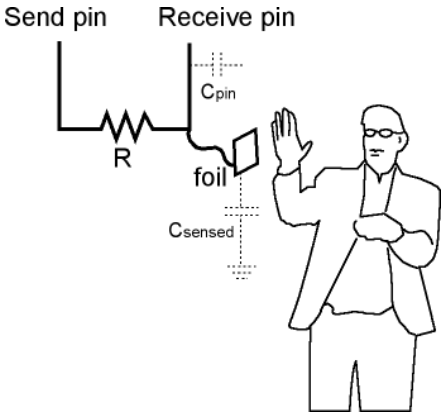
Method 3: Capacitive Sensing Library by Paul Badger, Touch sensor-based project using Arduino

The method description is available on:
http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense

I didn't read through the script and the method yet, but it seems good and fast enough (1 reading/ms).

A demonstration video for using this method is available:
https://www.youtube.com/watch?v=_GPvCxkWgbE

A capacitance of about 200 kOhm is required.



Connections:

Resistor: between send pin and receive pin

Capacitor: between receive pin and ground

Choose the resistance in a way the returned value does not saturate for the water level measurement region, and you still have enough measurement/sec.

100 kOhm: 900 meas/sec, output range: 0-800

200 kOhm: 750 meas/sec, output range: 0-2000

300kOhm: 600 meas/sec, output range: 0-3000

If the system is reset, the starting value will be considered as error and will be subtracted from the actual reading. To solve this problem the send and receive pins should be short circuited. To apply that, using a relay is possible. A digital pin can trigger the relay each time the setup is restarted.

PWM output

References:

https://arduino-info.wikispaces.com/Analog-Output

https://arduino-info.wikispaces.com/PWM

https://arduino-info.wikispaces.com/Arduino-PWM-Frequency

A digital pin of the Arduino is used to output the sensor value as a PWM signal. The duty cycle of the PWM signal is calculated as follow:

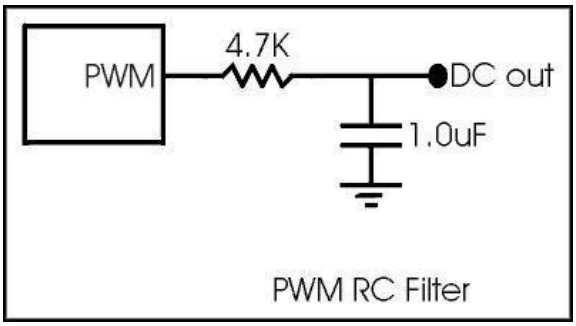
dutyCycle = 255*(total1-CapMin)/(CapMax-CapMin)

where total1 is the current sensor value returned by the Arduino, and CapMin and CapMax are the lower and the upper limits of that value.

The frequency of the PWM signal can be modified: from 30Hz and up to 30kHz.

Low Pass filter

The Bechhoff module will require a true analogue signal. So a low pass filter is required.



A resistance of 1kOhm and a capacitor of 2uF are used.

The low pass filter performance can be automatically tested via this link:

http://sim.okawa-denshi.jp/en/CRtool.php

The stability of the output signal is now in the rage of $\pm 0.1V$, which (theoretically) gives a total of about 50 readings (for a 5 volt interval), that is a resolution of 1mm for a 5 cm water level range (not approved yet, reading tolerance should be tested with a better sensor design).

Fine tuning of R and C with the correct upper and lower limits is required to achieve the best possible results.

Calibration

Calibration example: https://www.youtube.com/watch?v=3Qvdl0q9soE

Having the readings from the Arduino through the Beckhoff module, the calibration process can be handled through ATLAB. A further filtering is required to have a stable signal for a specific water level. Than each stable reading is assigned to its relative water level. Nonlinearity is likely to occure.

The reading speed and accuracy can then be challenged.