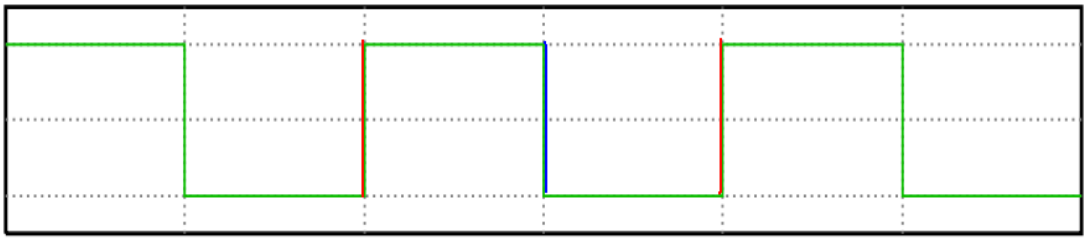
First of all, pay your attention to that fact that sensor's output signal is frequency – frequency of the square wave, in range between 5 and 10 kHz, like shown on image:



In fact, the capacitive sensor is connected to ICM7555 timer IC, which outputs the wave signal, depending on sensor's value.

To get actual RH (Relative Humidity) reading, in percents, we must calculate using the formula described in datasheet:

$$RH = (Offset - F)*Sensitivity/2^{12},$$

where *F* is frequency in Hz measured on FOUT pin.

This formula includes two calibration values - *Offset* and *Sensitivity*. They are individual to each unit, and are stored in tiny on-board I^2C EEPROM, so you need to read them before you can use the sensor. However, if your are not going to replace the sensor or build several devices based on your project, reading calibration values may be done just once. After that you can throw out I^2C code and wiring and include calibration values to your sketch as constants, saving program memory for other things.

MEASURING FREQUENCY

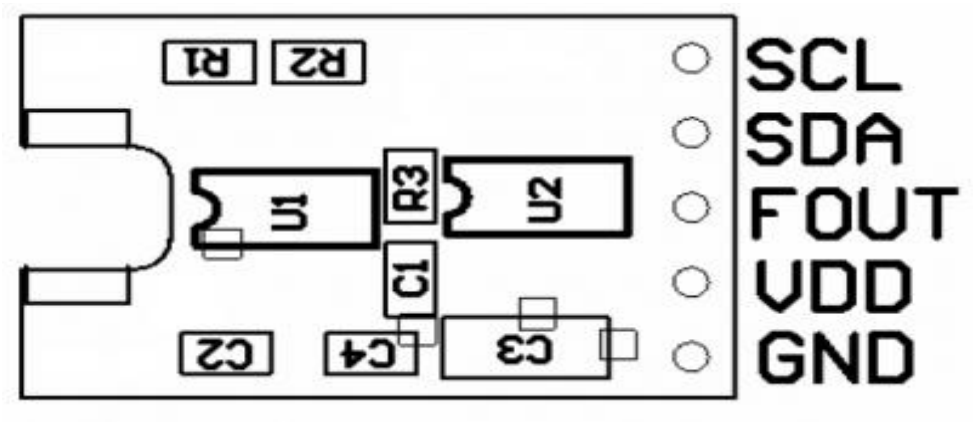
Measuring frequency with Arduino may be relatively simple, thanks to FreqCounter library. **However, if you need more than 2 PWM outputs in your project, things come a bit more complicated. If you need not more that 2 PWM outputs, then just ignore this section,** but if you need, then either consider choosing another sensor – or read the following article: . It's an alternative way to measure frequency, however, it is not as accurate as using hardware timers.

FreqCounter affects 2 of 3 ATMega's timers, and PWM duty cycles on Arduino pins 10, 9, 6 will significantly change. Pin 5 will be taken by FreqCounter – this cannot be changed, due to hardware restrictions. The only outputs available for PWM are pins 11 and 3.

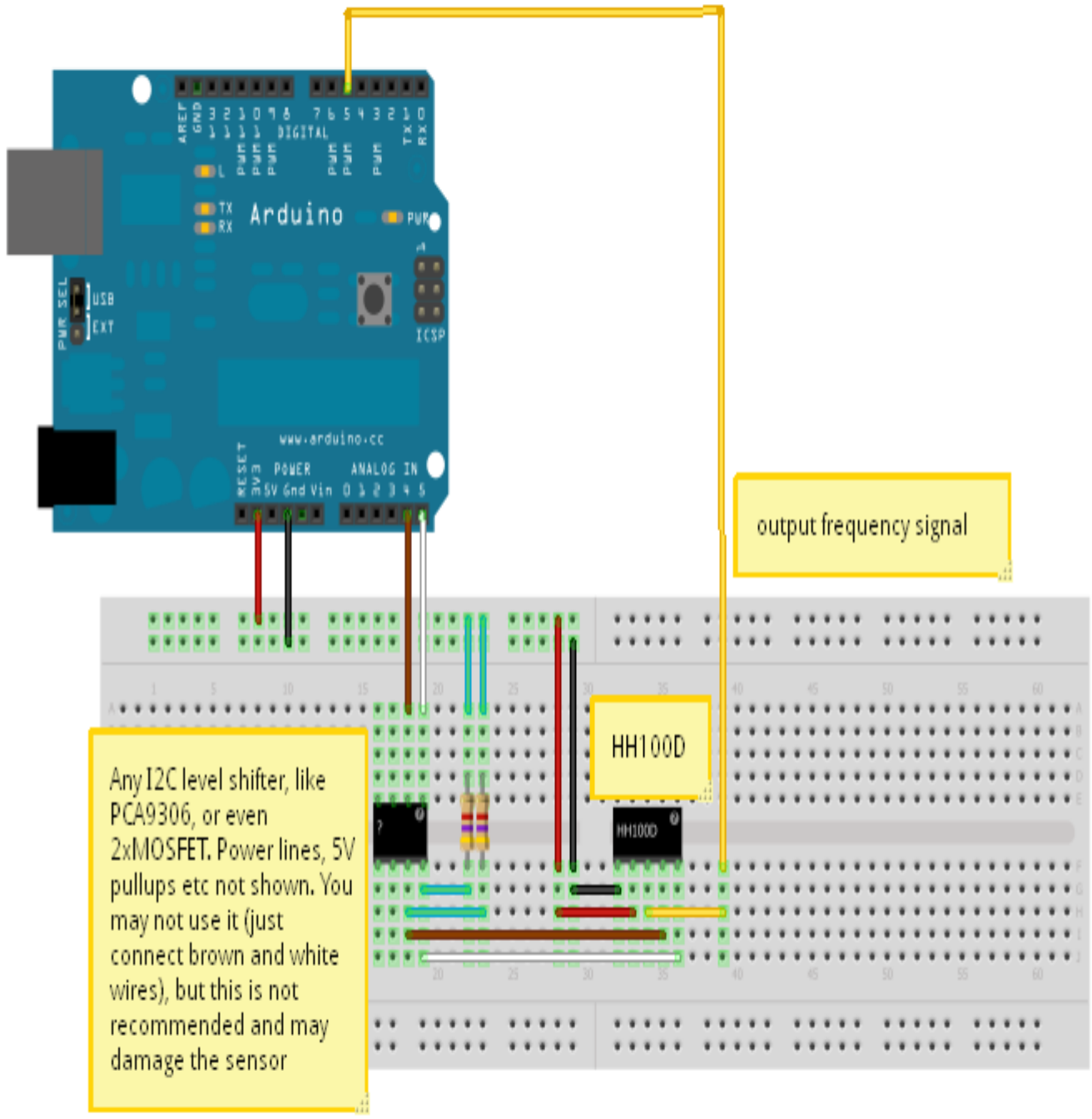
Also I've heard that FreqCounter may conflict with Servo library, but I cannot say that for sure.

CONNECTING THE SENSOR

If you look on the top side of you sensor (the one which has ICs), you will see the following:



Connects them to your Arduino, as shown on the wiring diagram:



- SCL goes to pin A5
- SDA to A4 (these are pins for I^2C, they cannot be changed)
- FOUT to digital pin 5
- VDD to +3V3
- GND to GND
- Also you should attach two 4.7 kOhm pullup resistors between 3V3 line and SCL\SDA lines.

VERY IMPORTANT: This sensor is 3.3v device, and you will probably need a level shifter on SCL and SDA to connect it to 5V-Arduino via I^2C . In my case, everything worked normally without it, but this gives no warranty that your sensor will not be damaged. Read more here: http://www.nxp.com/documents/application_note/AN10441.pdf
I'm not responsible for any damage or loss if you decide to follow my way, without level shifting.

You can use I2C+SMBus Voltage Translator or TI PCA9306 for this.

https://tushev.org/articles/arduino/item/47-interfacing-hh10d-with-arduino