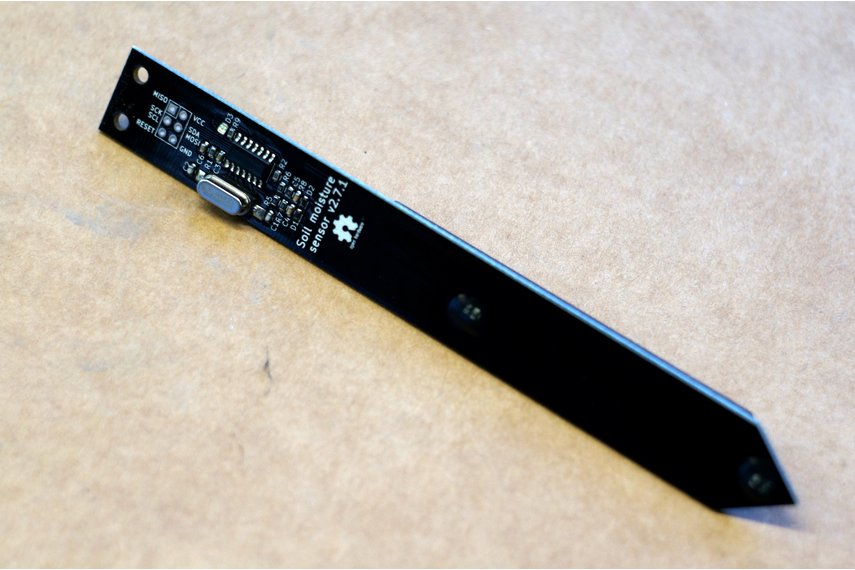
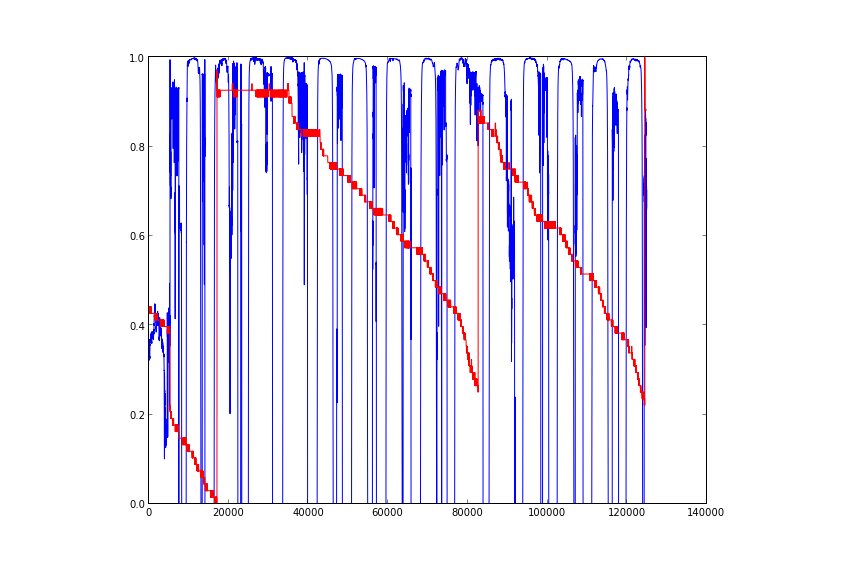
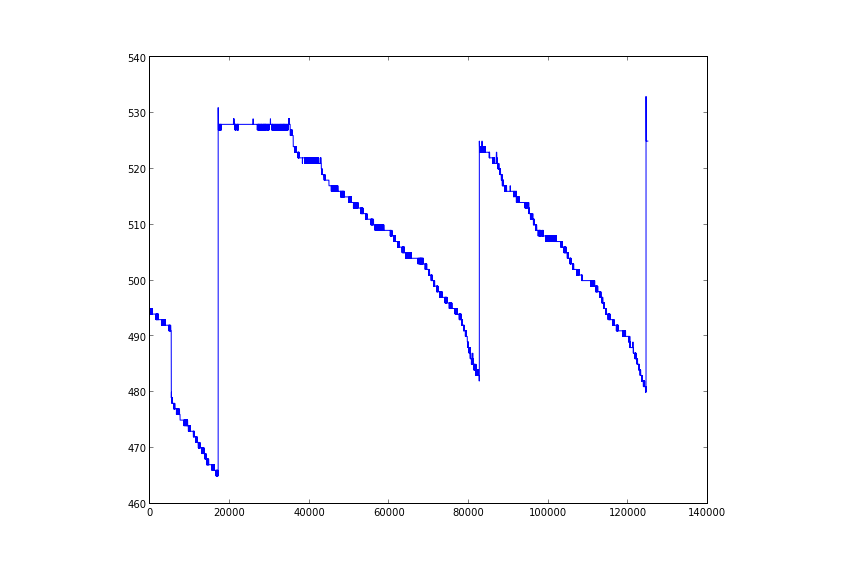
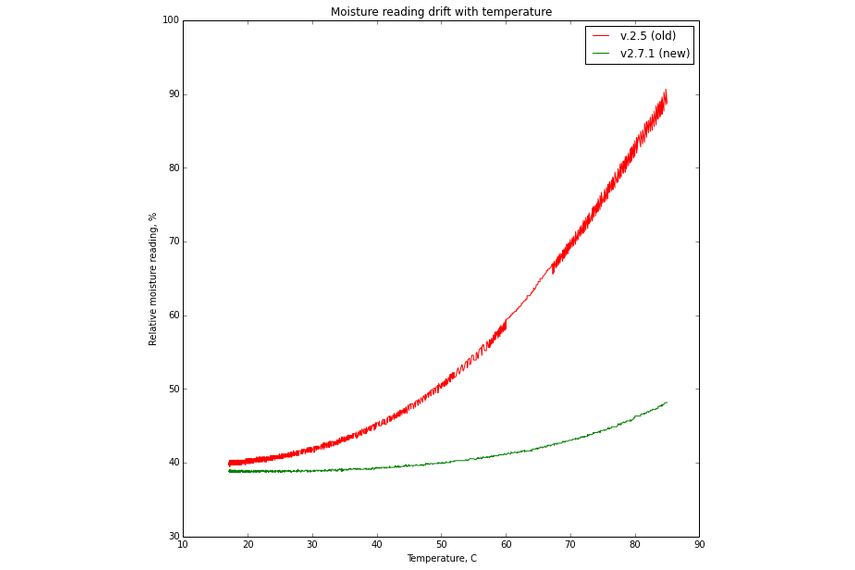
I2C Soil moisture sensor









### Capacitive soil moisture sensor interfaced via I2C. In addition senses light level and ambient temperature.

This is a cheaper "sensor mode" only version of my plant watering alarm [Chirp](https://www.tindie.com/products/miceuz/chirp-plant-watering-alarm/).

I've noticed that quite a lot of people use Chirp as a sensor, so I've decided to release a version of device optimised just for that.

The sensor can be read via I2C protocol and provides these features:

* Soil moisture sensing
* Light sensing
* Temperature sensing
* Reset chip
* I2C address change

This is the second version of my sensor with some improvements:

* 16MHz crystal for better repeatability and better Raspberry Pi support
* Thermistor for accurate temperature measurements in the soil
* Increased moisture reading resolution (almost double!)
* Holes to fasten a cable with a zip-tie

## Technical data

* Version 2.7.1
* Supply voltage 3.3V - 5V
* Current consumption: 1.1mA @ 5V, 0.7mA @ 3.3V when idle, 14mA @ 5V, 7.8mA @ 3.3V when taking a measurement. When constantly polling sensor at full speed, current consumption averages to 4.5mA @ 5V, 2.8mA @ 3.3V
* Operating temperature 0°C - 85°C
* Moisture reading drift with temperature - <10% over full temp range
* Don't forget to provide pullups for SCL and SDA lines
* Default I2C address is 0x20 (hex)
* To read soil moisture, read 2 bytes from register 0
* To read light level, start measurement by writing 3 to the device I2C address, wait for 3 seconds, read 2 bytes from register 4
* To read temperature, read 2 bytes from register 5
* To change the I2C address of the sensor, write a new address (one byte [1..127]) to register 1; the new address will take effect after reset
* To reset the sensor, write 6 to the device I2C address.

## How to interpret the readings

Both light and moisture sensors give relative values. Meaning, more moisture will give you higher reading, more light, lower reading.

Moisture is somewhat linear. I test all sensors before shipping and they give about 290 - 310 in free air at 5V supply.

I didn't measure linearity of the light sensor, it gives 65535 in a dark room away form desk lamp. When it's dark, it takes longer to measure light, reading the light register while measurement is in progress will return the previous reading. Be aware, light sensor is pretty noisy.

Temperature is measured by the thermistor on the tip of the sensor. Calculated absolute measurement accuracy is better than 2%. The returned value is in tenths of degrees Celsius. I.e. value 252 would mean 25.2°C.

## Controller support

The sensor works fine with Arduino. To talk to the sensor from RaspberryPi due to clock stretching bug, the baudrate of the RasPi I2C bus has to be reduced drastically - around 1000 - 3000.

## Waterproofing

The sensor comes coated with PRF202 - a moisture resistant varnish for electronics. Though sometimes it just does not work, I'm not sure yet why, probably because of different soil composition. Some suggestions on making the sensor more robust after attaching the cable:

* Polyester or epoxy resin - this method is the most bullet proof as the resin is totally resistant to the water. On the par side, note that sensitivity of the sensor will decrease depending on how thick the layer you are going to apply. Also applying the resin in uniform manner presents some challenge.
* PlastiDip - some of my customers have tried this method - easy to apply by spraying and not a lot of loss of sensitivity.
* Rubber balloon - yes, just roll a long balloon over the sensor :)

# i2c-moisture-sensor

I2C based soil moisture sensor. A continuation of the [Chirp - plant watering alarm](https://github.com/Miceuz/PlantWateringAlarm) project. There is also an [RS485](https://github.com/Miceuz/rs485-moist-sensor) and an[analog](https://github.com/Miceuz/soil-moisture-sensor-analog) version available.

## I2C protocol

Available registers for reading and writing.

| **Name** | **Register** | **R/W** | **Data length** |
| --- | --- | --- | --- |
| GET\_CAPACITANCE | 0x00 | (r) | 2 |
| SET\_ADDRESS | 0x01 | (w) | 1 |
| GET\_ADDRESS | 0x02 | (r) | 1 |
| MEASURE\_LIGHT | 0x03 | (w) | 0 |
| GET\_LIGHT | 0x04 | (r) | 2 |
| GET\_TEMPERATURE | 0x05 | (r) | 2 |
| RESET | 0x06 | (w) | 0 |
| GET\_VERSION | 0x07 | (r) | 1 |

### Arduino example

#include <Wire.h>

void writeI2CRegister8bit(int addr, int value) {

Wire.beginTransmission(addr);

Wire.write(value);

Wire.endTransmission();

}

unsigned int readI2CRegister16bit(int addr, int reg) {

Wire.beginTransmission(addr);

Wire.write(reg);

Wire.endTransmission();

delay(20);

Wire.requestFrom(addr, 2);

unsigned int t = Wire.read() << 8;

t = t | Wire.read();

return t;

}

void setup() {

Wire.begin();

Serial.begin(9600);

writeI2CRegister8bit(0x20, 6); //reset

}

void loop() {

Serial.print(readI2CRegister16bit(0x20, 0)); //read capacitance register

Serial.print(", ");

Serial.print(readI2CRegister16bit(0x20, 5)); //temperature register

Serial.print(", ");

writeI2CRegister8bit(0x20, 3); //request light measurement

Serial.println(readI2CRegister16bit(0x20, 4)); //read light register

}

## Address change example

By default the sensor comes with 0x20 set as an address, this is an example on how to change address for indivitual sensor:

#include <Wire.h>

void writeI2CRegister8bit(int addr, int reg, int value) {

Wire.beginTransmission(addr);

Wire.write(reg);

Wire.write(value);

Wire.endTransmission();

}

void writeI2CRegister8bit(int addr, int value) {

Wire.beginTransmission(addr);

Wire.write(value);

Wire.endTransmission();

}

void setup() {

Wire.begin();

Serial.begin(9600);

//talking to the default address 0x20

writeI2CRegister8bit(0x20, 1, 0x21); //change address to 0x21

writeI2CRegister8bit(0x20, 6); //reset

delay(1000); //give it some time to boot

}

/\*loop scans I2C bus and displays foud addresses\*/

void loop()

{

byte error, address;

int nDevices;

Serial.println("Scanning...");

nDevices = 0;

for(address = 1; address < 127; address++ )

{

// The i2c\_scanner uses the return value of

// the Write.endTransmisstion to see if

// a device did acknowledge to the address.

Wire.beginTransmission(address);

error = Wire.endTransmission();

if (error == 0)

{

Serial.print("I2C device found at address 0x");

if (address<16)

Serial.print("0");

Serial.print(address,HEX);

Serial.println(" !");

nDevices++;

}

else if (error==4)

{

Serial.print("Unknow error at address 0x");

if (address<16)

Serial.print("0");

Serial.println(address,HEX);

}

}

if (nDevices == 0)

Serial.println("No I2C devices found\n");

else

Serial.println("done\n");

delay(5000); // wait 5 seconds for next scan

}