# Aurora Monitor – Quick Start Guide

I feel it important for students to play with kit such as this to quickly develop an understanding of the system.

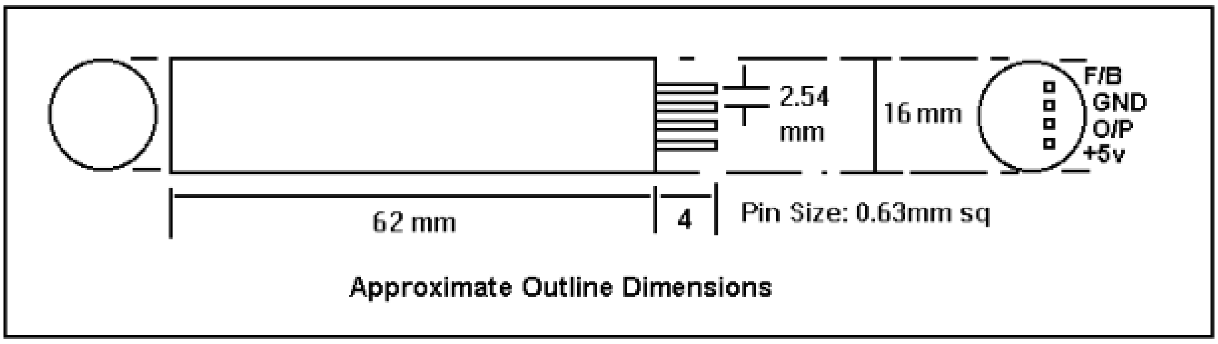
For a full system you will need..

- A Raspberry PI
- An Arduino Uno
- A powersupply – one can start with a lab psu

## The FGM Sensor

I would start with becoming familiar with the sensor

- fg-3 sensor (fgsensors.com)
- Oscilloscope
- Lab P.S.U



The original Speake sensor sensor (above) has 4 pins – we only need ground (GND), output (O/P) and +5V. The newer fg-sensor has only GND, O/P and +5V

If you have a breadboard you should be able to plug the sensor into this for temporary connections. Failing that I would solder 3 leads of about 30cm to an 8 pin dip socket – having first checked that the sensor will fit, otherwise solder directly to the FGM pins taking care not to overheat it – one can use a metal clip or similar as a heat sink.

Then connect the gnd and +5V to a lab power supply. The sensor is not too fussy about input voltages 4.5-5.5V work. However the output does vary if the input voltage changes. For an aurora monitor you will need to construct a simple stabilised supply.

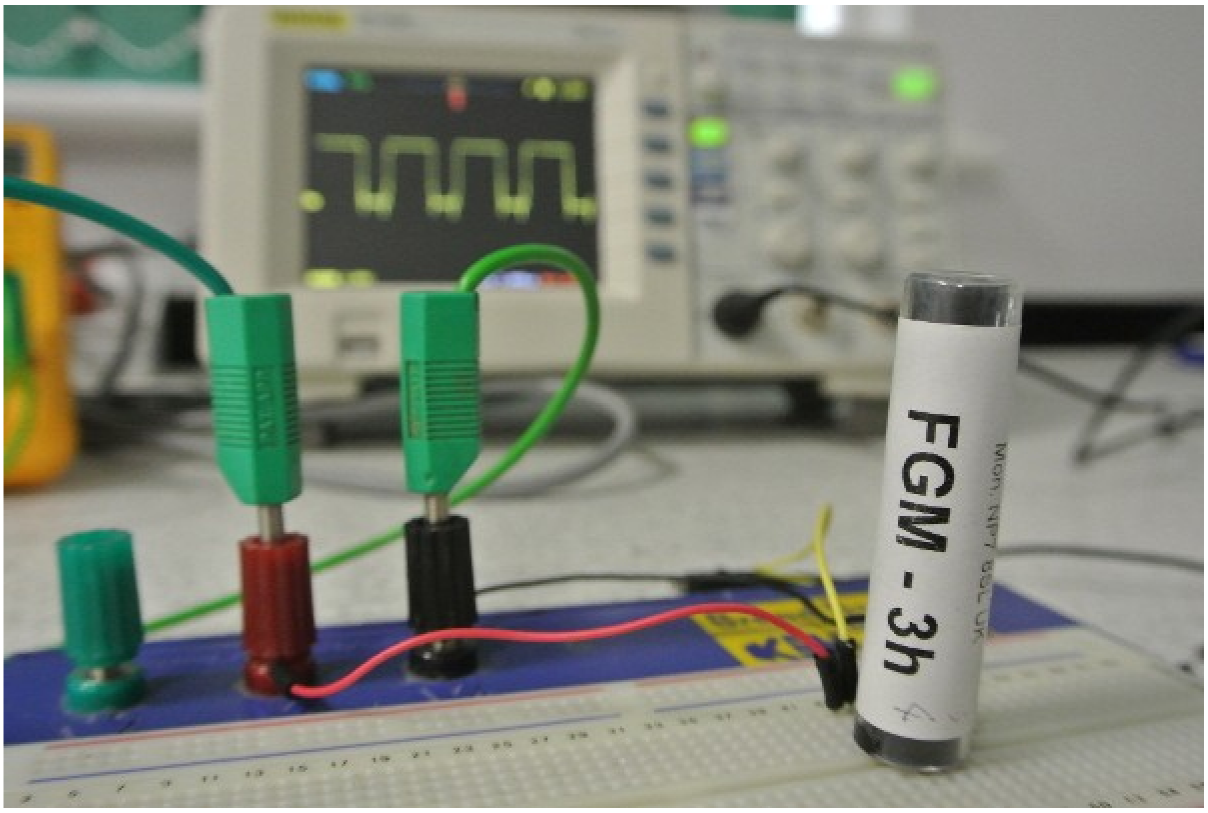
Now connect the input lead of an oscilloscope to the output – O/P pin.

Power up.

You should detect a roughly square wave TTL (0-5V) output at 20kHz $\rightarrow$120kHz.

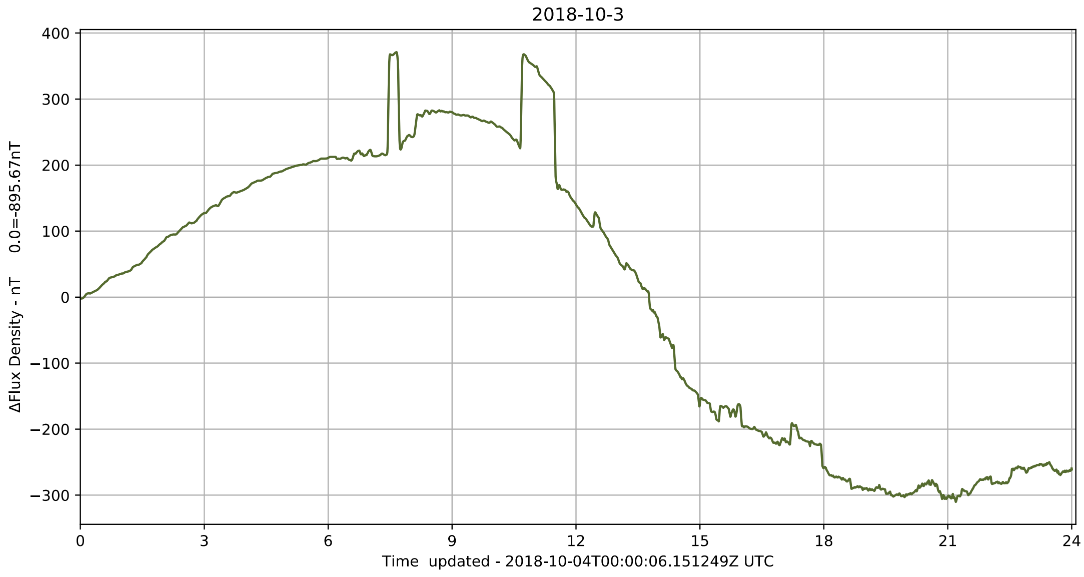
Orientate the sensor roughly East-West and you should be seeing an output of approx 70kHz.

Investigate (play). Moving a small piece of iron or even your own body in the vicinity of the sensor should cause an observable change in the output frequency.



Rotating the sensor we should see a dramatic change in output – probably saturating orientated North-South where the earth's field in strongest. Pointing the sensor north and changing the angle of dip one can also clearly see the declination of the earth's field.

This device really is very sensitive. At the surface the earth's field varies from 25,000-65,000 nT (nano Tesla). This device will reliably detect a change of about 1-2 nT.



The step changes at ~8 and~11 were caused by the moving of a car approx 12m from the sensor.

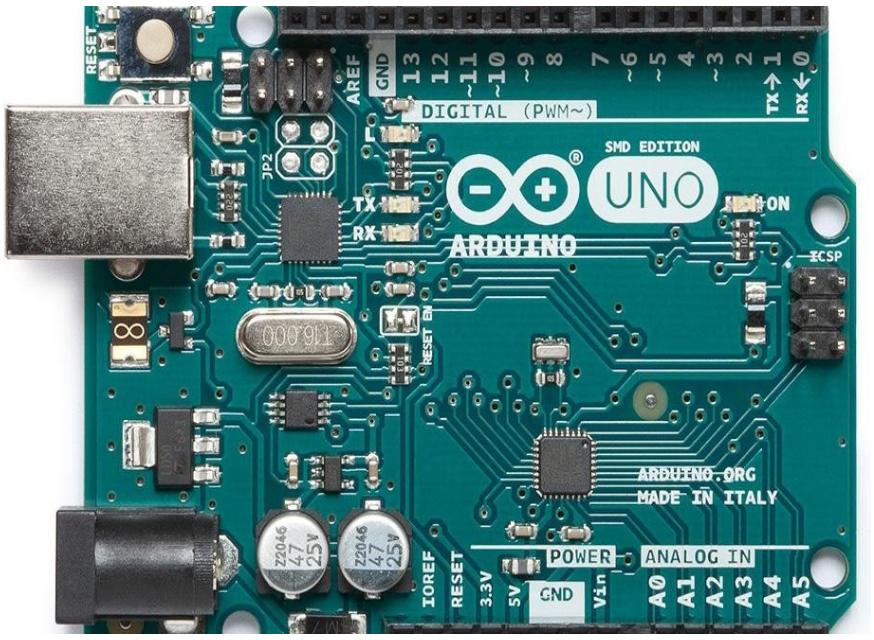
Next…

## Measuring the Output Frequency

The sensor outputs a TTL (5V square wave) whose frequency varies approximately linearly with the external magnetic field. When orientated East-West, where the earth's field is lowest output is approximately 70kHz. This is too fast to be accurately counted by a raspberry Pi so we will use an Arduino (R3) to count the pulses and output the frequency via USB to a Pi.

### Setting Up The Arduino

I would recommend using an Arduino Uno costing about £29. Cheaper options are available though.

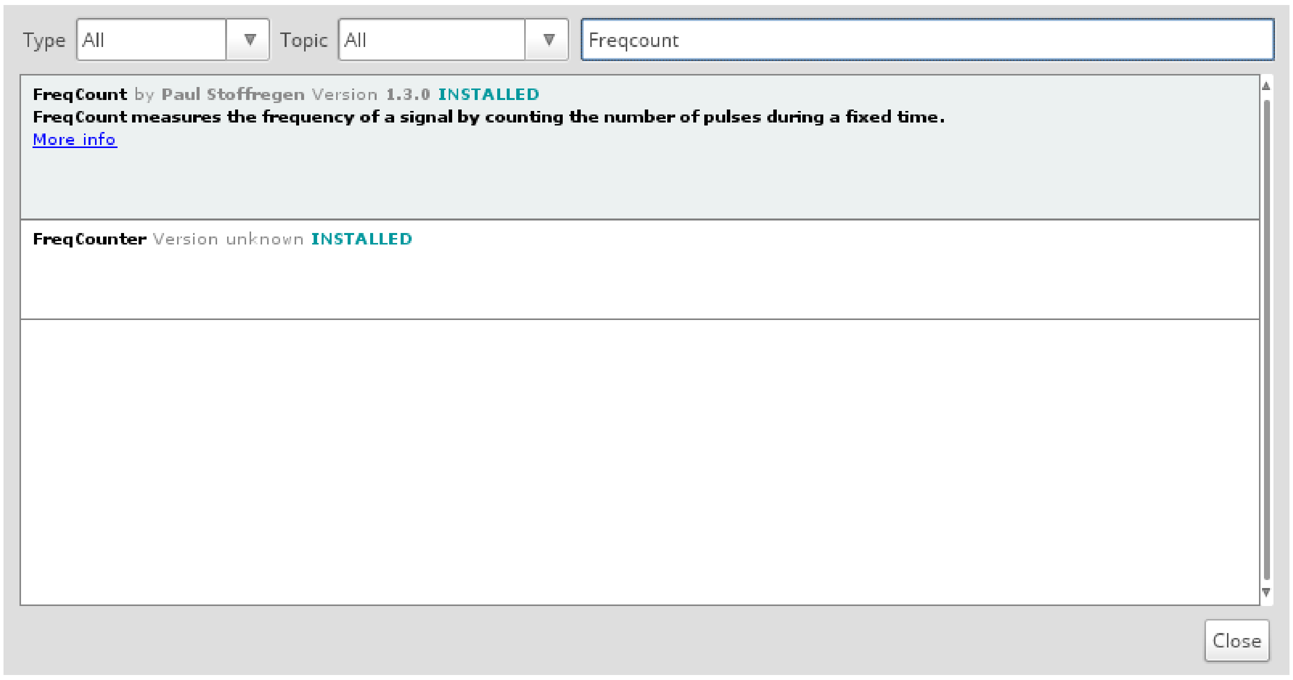


The Arduino is a 'bare metal processor'. Here it will run one simple program automatically when started up.

You will need a PC (Linux or Windows) with the Arduino IDE installed from [https://www.arduino.cc/en/main/software](https://www.arduino.cc/en/main/software)

Firing up the IDE choose

Sketch → Include Library → Manage Libraries ->



FreqCount by Paul Stoffregen Version 1.3.0 INSTALLED
FreqCount measures the frequency of a signal by counting the number of pulses during a fixed time.

FreqCounter Version unknown INSTALLED

Ensure that the Freqcount library is installed - the ide can pull it from the net if not.

Then do a File ..new and paste the following code in.

```
/* FreqCount - Example with serial output
 * http://www.pjrc.com/teensy/td_libs_FreqCount.html
 * This example code is in the public domain.
 */
#include <FreqCount.h>
void setup() {
  Serial.begin(57600);
  FreqCount.begin(1000);/* gatetime in mSeconds*/
}
void loop() {
  if (FreqCount.available()) {
    unsigned long count = FreqCount.read();
    Serial.println(count);
  }
}
```

This repeatedly counts the number of pulses over 1 sec (1000 msecs) and presents this frequency to the seial output (USB line) . This we can read with the PI.

Connect the Arduino to a USB slot on the PC and within the Arduino IDE select Sketch .. Upload to compile the code and upload to the Arduino where it should start running immediately. If all has worked ( probably hasn't) selecting Tools .. Serial Monitor should show the output from the Arduino.

## Setting Up The Pi

Right – this can be involved but we learn a lot about computing here.

We are going to use a Raspberry PI to  save data, plot graphs and upload to your website.

There are two ways we can 'talk' to the PI – either using a screen, keyboard and mouse – like a conventional computer or from a PC over a network – a.k.a 'headless mode'. A monitor and keyboard is handy in a schools lab but for a proper network install we are better with headless.

So the first thing to do is get the PI running. I will assume that you have connected to a hdmi monitor or TV and have a keyboard and mouse attached.

Firstly you will need to install the operating system. This will be Raspian – a version of Linux.

### Follow the instructions at

### https://github.com/starfishprime101/RPI-Seismic-Monitor-Code

## The Monitoring Programs

There are 3 python programs used

- aurora_monitor.py
- geostationModules.py
- globalvariablesModule.py

These should be copied into the 'station' folder on your PI

The 3rd file, globalvariablesModule.py may need editing for your system. Especially Line 5  - home_directory = '/home/GeoPhysics/EFM'.

lines 14 and 26 may want minor editing.

Set Up CronTab to automatically start the Aurora Monitor on reboot using the instructions at https://github.com/starfishprime101/RPI-Seismic-Monitor-Code

### Adding a real Time Clock

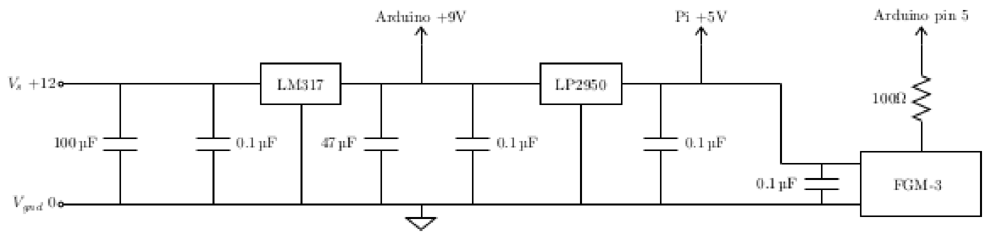
In normal use the PI gets its time signal from Internet. It lacks an internal clock so cannot add correct time to a trace if it is not connected to the internet.

Adding a precise clock module is thus desirable only if you intend to use the sensor away from an Internet connection.

I use the DS3231 Precision R.T.C. from AdaFruit following instructions at https://pimylifeup.com/raspberry-pi-rtc/

## A Precision Power Supply

A simple double-regulated power supply constructed on veroboard may be used to provide a stable voltage to the sensor as well as powering the Arduino and Raspberry Pi thereby ensuring a common ground.


Arduino +9V
Pi +5V
Arduino pin 5
$V_s$ +12
LM317
LP2950
100Ω
100 µF
0.1 µF
47 µF
0.1 µF
0.1 µF
0.1 µF
FGM-3
$V_{gnd}$ 0