the CANARY

Siddharth D'Souza, Davide Scola & Laura Newton

Applied Measurement and Control,
Rhine-Waal University of Applied Science - 11.07.17

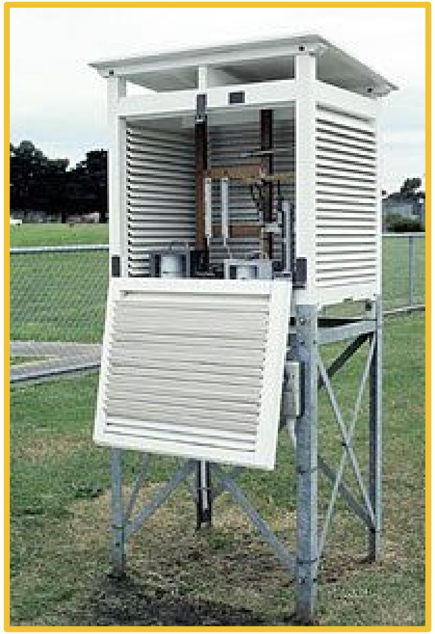
Background

- Environmental Monitoring - LANUV visit
- Measure **air quality**
- Specifically **NO_2**
- Develop a small **portable** device
- Use knowledge from **class**

wallpapersafari.com

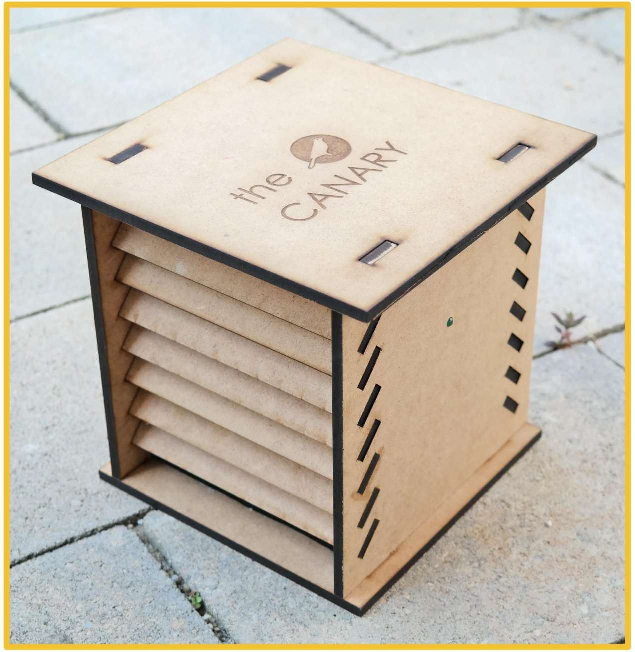
http://www.bmvi.de

The original Stevenson Screen



Source: Government of Australia 2017, Bureau of Meteorology

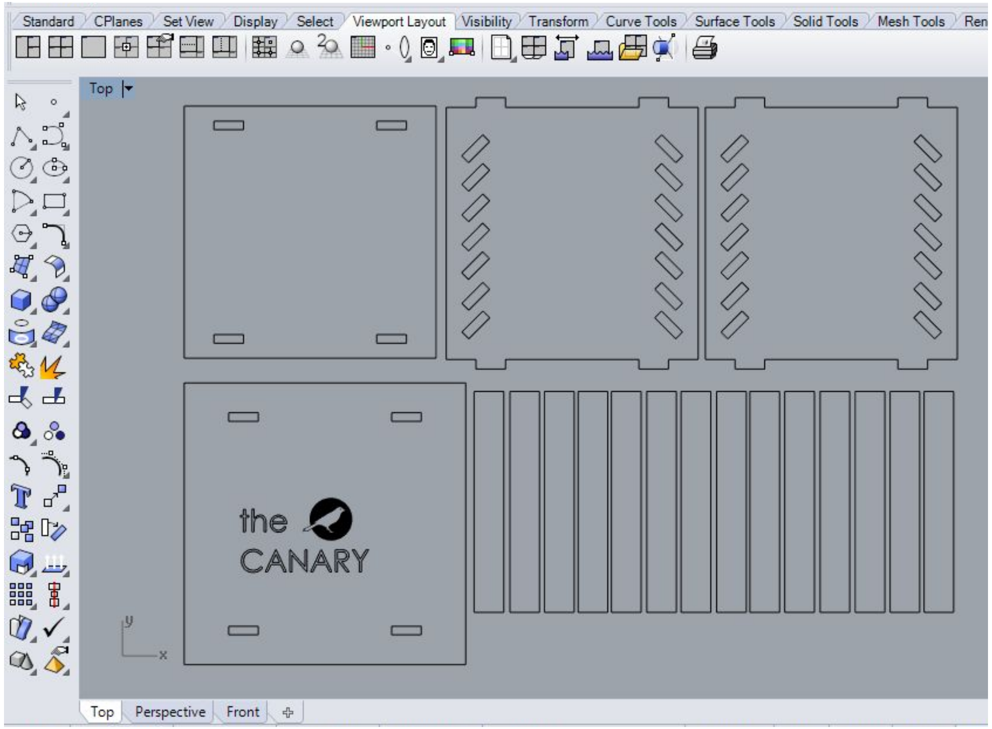
The CANARY

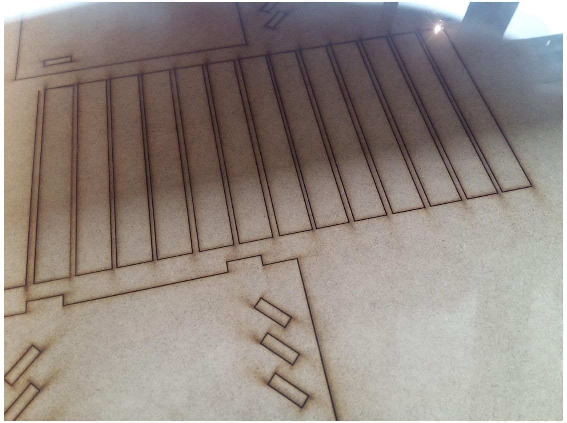


L. Newton

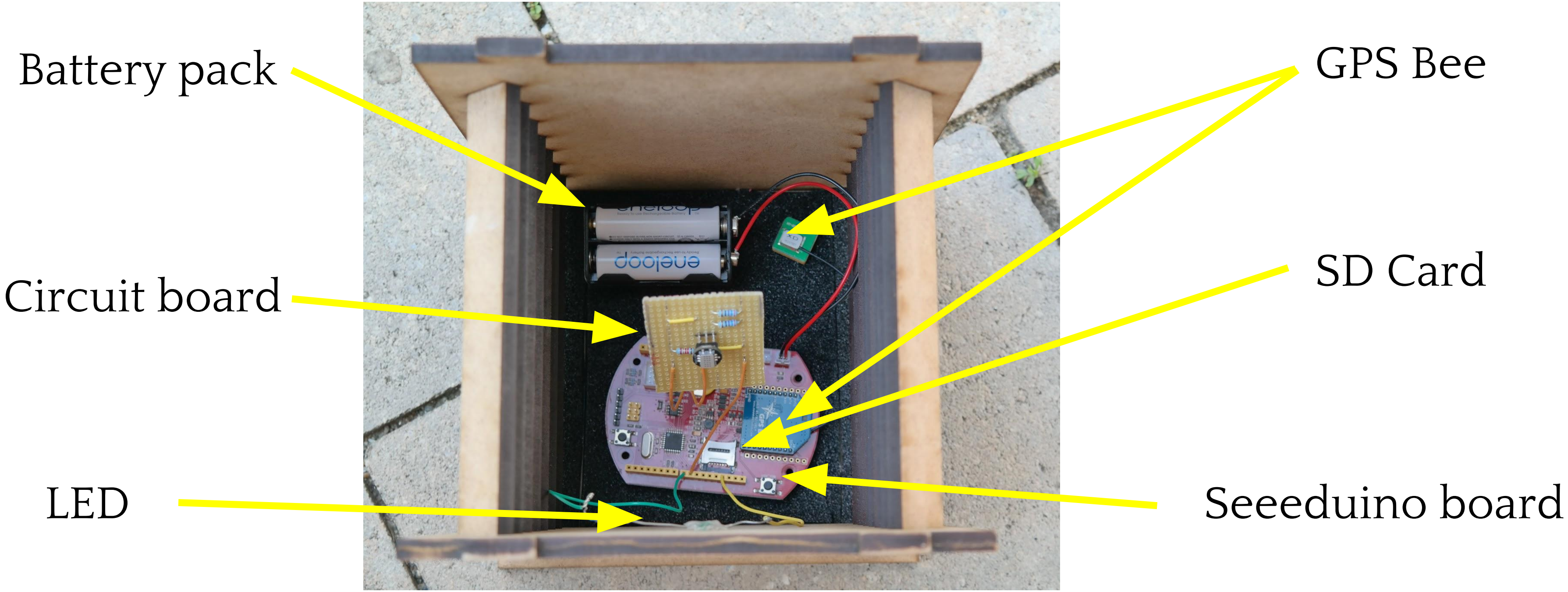
Making the box

- Protection from the elements
- Stable and sturdy
- Ensure airflow
- Laser cut 6 mm plywood





What's inside?



Battery pack

Circuit board

LED

GPS Bee

SD Card

Seeeduino board

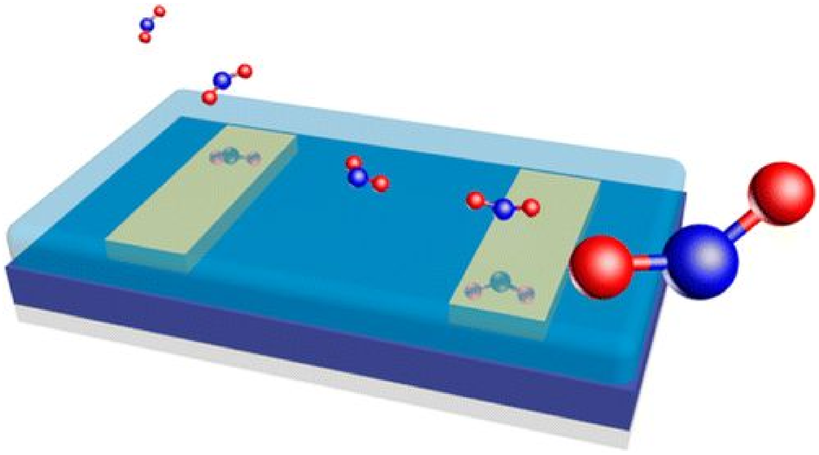
The NO_2 sensor

MiCS-2714

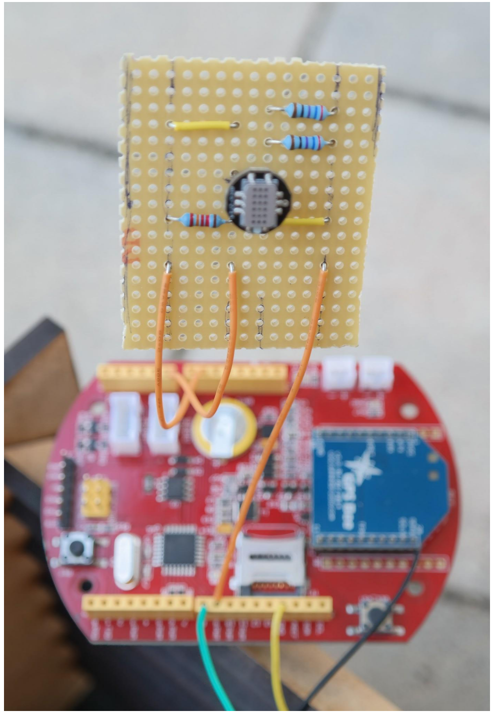
- Semiconductor Sensor
- Sensing layer: Meso-porous Silicon (PS)
- Sensing Resistance: 0.8 -20 k Ω
- NO2 Detection Range: 0.05- 10 ppm

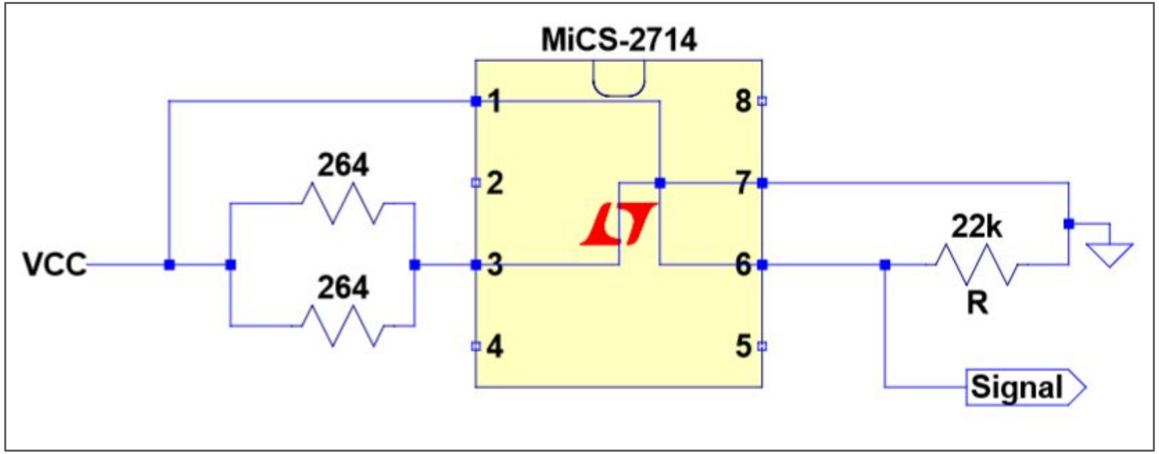
Working of the Sensor Layer

- Molecules of NO2 act as **acceptors**.
- Once adsorbed to PS, there is an **increase in carriers** (holes) leading to **increase in conductivity.**



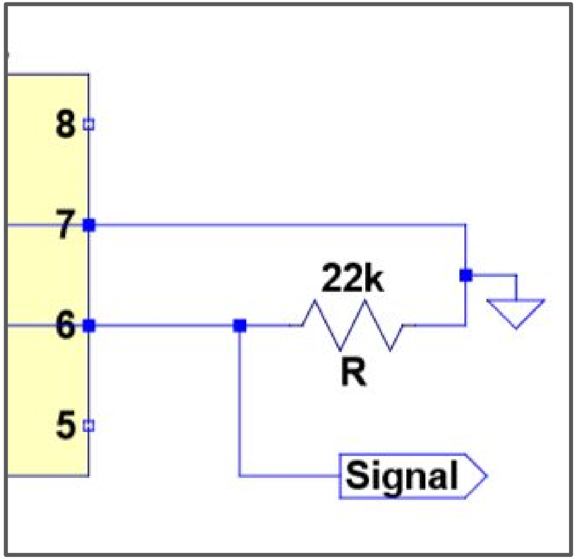
Connecting the Sensor



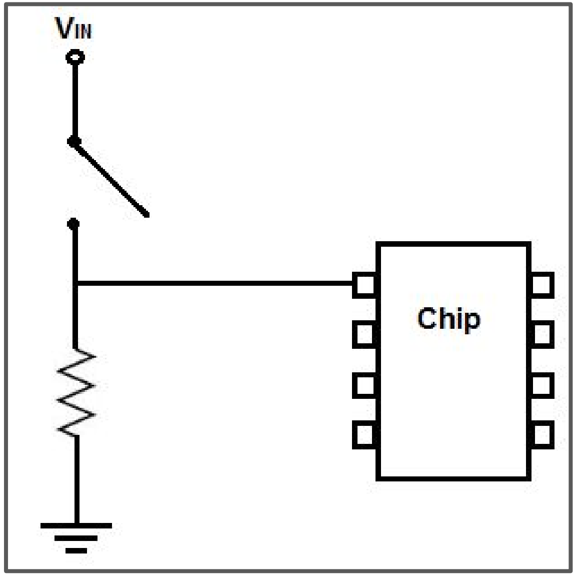


- Heating Layer
- Sensing Layer

Use of a pull-down resistor



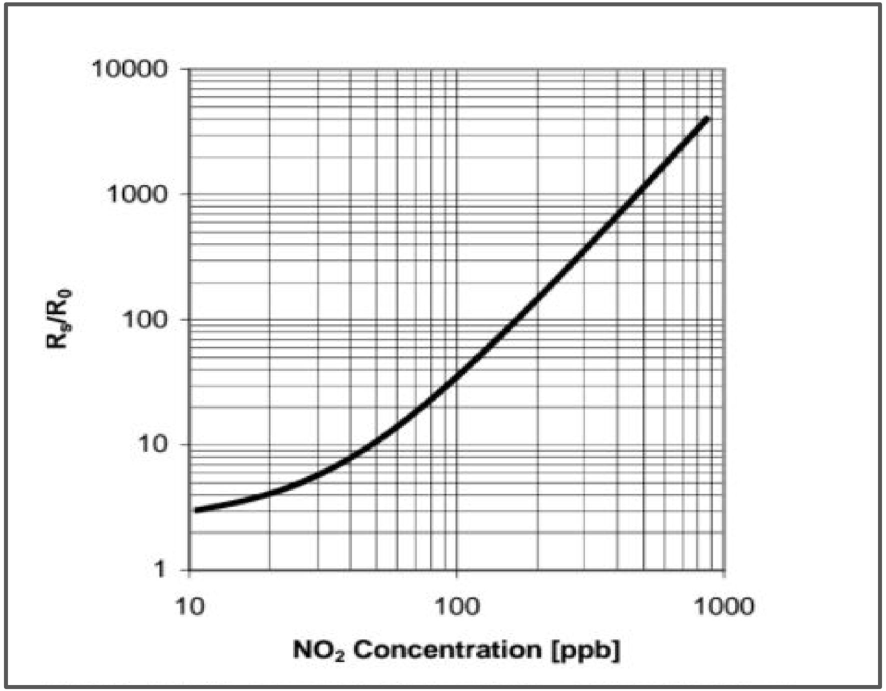
=



- Pulls floating state down to GND

www.learningaboutresistors.cm

Sensor response to NO2



Rs/Ro is a function of NO2 at 40% RH and 25°C

- Standard resistance in air Ro is measured under controlled ambient conditions, i.e. synthetic air at 23 ± 5°C and ≤5% RH.

- Sensitivity factor SR is defined as RS at 0.25 ppm of NO2, divided by RS in air.

The Seeeduino Code

GPS signal processing

NO_2 signal processing

SD card management

State machine

Blinker custom function

GPS signal processing

```
#include <SoftwareSerial.h> //Include SoftwareSerial library for communication
SoftwareSerial GPS(6, 7); //Set pins 6 and 7 as RX and TX for the SoftwareSerial
char c; //Define variable - character
char buff[100]; //Define variable - character array
GPS.begin(9600); //Within the setup start the serial communication at a baud rate of 9600
void loop() {
    if (GPS.available()) //Check if serial communication is working {
      c = GPS.read(); //Read one character of the digital message being received
      if (c == '$') //Check if it is the “Start” character {
        GPS.readBytes(buff, 6); //Store the next 6 characters in an array
        if (buff[2] == 'R') //Check if it is the correct string {
          GPS.readBytes(buff, 99); //Store the next 99 characters in an array
          if ((char)buff[11] == 'A') //Check for valid signal {
            for (int i = 0; i < 99; i++) {
              if ((char)buff[i + 2] == '$') //Check for next line {
                break;
              }
              if (myFile) //Check if SD available{
                myFile.print(buff[i]); //Write message onto SD   ...}...}...}...}...}...}...}
```

NO_2 signal processing

```
int power = 9; //Define pin variable
int NO2pin = A0; //Define pin variable
float NO2resistance; //Define variable - decimal value
int NO2seriesResistor = 22000; //Define variable - integer value
float NO2measure = 0; //Define variable - decimal value
pinMode(NO2pin, INPUT); //Within the setup set pin A0 as input
pinMode(power, OUTPUT); //Within the setup set pin 9 as output
digitalWrite(power, HIGH); //NO2 sensor ON

void loop() {
    if (GPS.available()) { [...]
          if ((char)buff[11] == 'A') //Check for valid GPS signal {
            if (myFile) //Check if SD available {
              int NO2rawInput = analogRead(NO2pin); //Read the voltage at pin A0
              NO2resistance = NO2seriesResistor * ((1023.0 / NO2rawInput) - 1.0); //Calculate
the resistance of the sensor
              NO2measure = NO2resistance / 100; //Make the result more user friendly
              myFile.print(','); //Write ‘,’ onto SD
              myFile.println(NO2measure); //Write value onto SD            [...] }...}...}
```

SD card management

Used to store data

```
#include <SD.h> //Include SD library for communication
File myFile;  //Define variable - file
void setup() {
  pinMode(4, OUTPUT); //Within the setup set pin 4 as output
  digitalWrite(4, LOW); //SD Card ON
  Serial.print("Load SD card..."); //Visual feedback
  if (!SD.begin(10)) //Check if SD card can be initialized {
    Serial.println("SD Card could not be initialized, or not found"); //Visual feedback
    return;         }
  Serial.println("SD Card found and initialized."); //Visual feedback
  myFile = SD.open("GPSlog.CSV", FILE_WRITE); //Open/create a file on the SD card, start
writing        }
void loop() {
    myFile.close(); //Close/save the file on the SD card
    delay(500); //Wait 0.5s
    myFile = SD.open("GPSlog.CSV", FILE_WRITE); //Open/create a file on the SD card, start
writing
    delay(500); //Wait 0.5s        }
```

State machine

```
#define READING 0 //Define constant value
#define CLOSED 1 //Define constant value
byte state; //Define variable - byte
byte times_wrote = 0; //Define variable - byte
  state = READING; //Within the setup set state as READING
void loop() {
  if (state == READING) //Check if the state is READING {
    if (times_wrote > 9) //Check if times_wrote is greater than 9 {
      state = CLOSED; //Set state as CLOSED
      times_wrote = 0; //Reset counter
    }
    if (GPS.available()) { [...]
          if ((char)buff[11] == 'A') //Check for valid GPS signal {
            //Write data onto the SD card
            Times_wrote++; //Increase counter by 1
            [...]}...}
  if (state == CLOSED) //Check if the state is CLOSED {
    //Save the file
    state = READING;      //Set state as READING  ...}...}
```

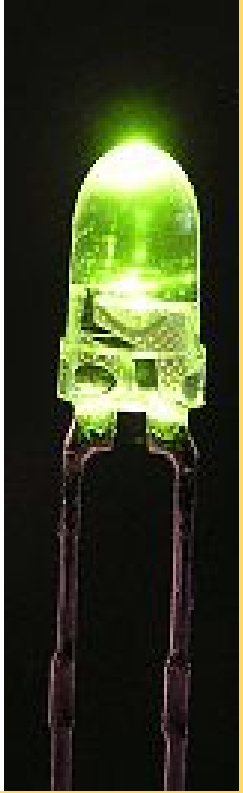
Blinker custom function

Used to indicate when the system is working

```
int LED = 8; //Define pin variable
pinMode(LED, OUTPUT); //Within the setup set pin 8 as output

void blinker (int duration, int npulse) //Define arguments {
  for (int i = 0; i < npulse; i++) {
    digitalWrite(LED, HIGH); //LED ON
    delay(duration / 2);
    digitalWrite(LED, LOW); //LED OFF
    delay(duration / 2);
  }
}
```

SD card not initialized successfully:	blinker(6000, 2); 2 long
Waiting for GPS fix:	blinker(125, 8); 8 short
Logging GPS + NO$_2$ signal:	blinker(1000, 1); 1 medium



Source: batsocks.co.uk

The GPS Data: NMEA

Raw data in NMEA (National Marine Electronics Association) form

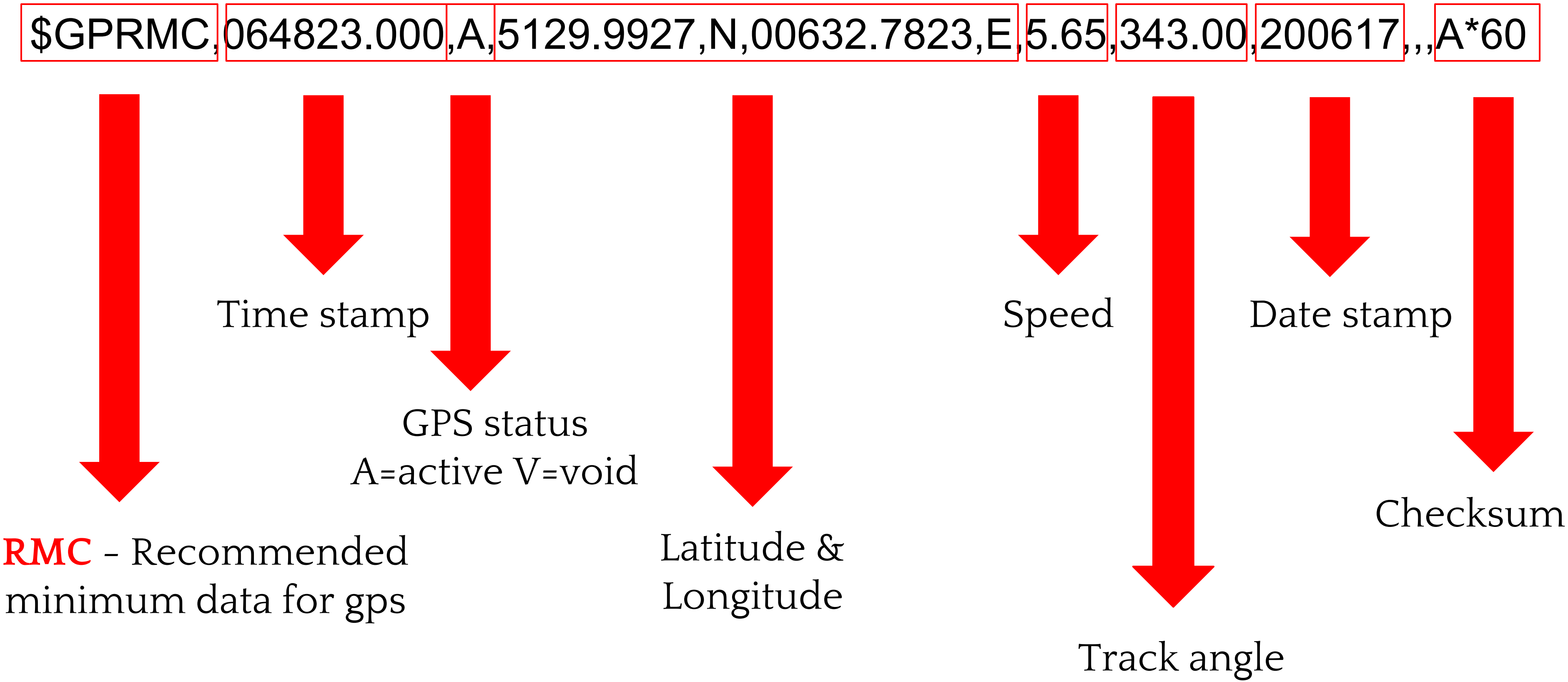
```
$GPGGA,064823.000,5129.9927,N,00632.7823,E,1,7,1.27,49.9,M,47.4,M,,*64
$GPGSA,A,3,12,19,13,15,17,24,18,,,,,,2.62,1.27,2.29*0D
$GPGSV,4,1,13,15,64,207,45,24,59,282,41,13,39,150,40,17,37,085,47*7F
$GPGSV,4,2,13,19,33,116,42,12,30,218,43,33,27,207,27,18,25,284,21*7F
$GPGSV,4,3,13,10,21,312,,28,20,050,,20,07,216,,01,03,031,19*7B
$GPGSV,4,4,13,11,02,018,*40
$GPRMC,064823.000,A,5129.9927,N,00632.7823,E,0.00,343.00,200617,,,A*60

$GPGGA,064824.000,5129.9927,N,00632.7823,E,1,7,1.27,49.9,M,47.4,M,,*63
$GPGSA,A,3,12,19,13,15,17,24,18,,,,,,2.62,1.27,2.29*0D
$GPGSV,4,1,13,15,64,207,45,24,59,282,41,13,39,150,40,17,37,085,47*7F
$GPGSV,4,2,13,19,33,116,42,12,30,218,43,33,27,207,26,18,25,284,21*7E
$GPGSV,4,3,13,10,21,312,,28,20,050,,20,07,216,,01,03,031,18*7A
$GPGSV,4,4,13,11,02,018,*40
$GPRMC,064824.000,A,5129.9927,N,00632.7823,E,0.00,343.00,200617,,,A*67
```

RMC - Recommended minimum data for gps

- Latitude
- Longitude
- Time
- Date
- Speed
- Track angle

Interpreting the NMEA data



$GPRMC,064823.000,A,5129.9927,N,00632.7823,E,5.65,343.00,200617,,,A*60

RMC - Recommended minimum data for gps

Time stamp

GPS status
A=active V=void

Latitude &
Longitude

Speed

Track angle

Date stamp

Checksum

The Python Parser

```
import csv //Include csv library to handle the files
originalfile = open('GPSlog.csv') //Define variable - file
csv_f = csv.reader(originalfile) //Read the csv file generated by the Seeeduino
temp_csv = [] //Define variable - array
newfile = open('GPSlogParsed.csv', 'w') //Create new file
wr = csv.writer(newfile) //Start writing on the new file
wr.writerow(["N","E","Measurement","Time","Date","Speed","Track Angle"]) //Write headers for values
for row in csv_f: //Open a loop to access every row in the original file once
    a = int(row[2][:2])+((float(row[2][2:]))/60) //Parse N coordinate - conversion to degrees
    a2 = "{:.8f}".format(a) //Convert N coordinate to a string up to the 8th decimal
    b = int(row[4][:3])+((float(row[4][3:]))/60)//Parse E coordinate - conversion to degrees
    b2 = "{:.8f}".format(b) //Convert E coordinate to a string up to the 8th decimal
    c = row[0][:2]+":"+row[0][2:4]+":"+row[0][4:] //Reformat time
    d = row[8][:2]+"/"+row[8][2:4]+"/"+row[8][4:] //Reformat date
    e = row[12] //Parse $NO_2$ measurement
    f = row[6] //Parse speed
    g = row[7] //Parse track angle
    temp_csv.append(a2) //Add N coordinate to last slot of array
    temp_csv.append(b2) //Add E coordinate to last slot of array
    temp_csv.append(e) //Add $NO_2$ measurement to last slot of array
    temp_csv.append(c[:8]) //Add first 8 characters of time to last slot of array
    temp_csv.append(d) //Add date to last slot of array
    temp_csv.append(f) //Add speed to last slot of array
    temp_csv.append(g) //Add track angle to last slot of array
    wr.writerow(temp_csv) //Insert the array as a new row in the new csv file
    temp_csv = [] //Clear the array
newfile.close() //Close the new csv file
originalfile.close() //Close the csv file generated by the Seeeduino
```

The Result

Input:

```
163709.000,A,5129.9548,N,00632.7603,E,2.40,44.88,070717,,,A*5C,167.37
```

Output:

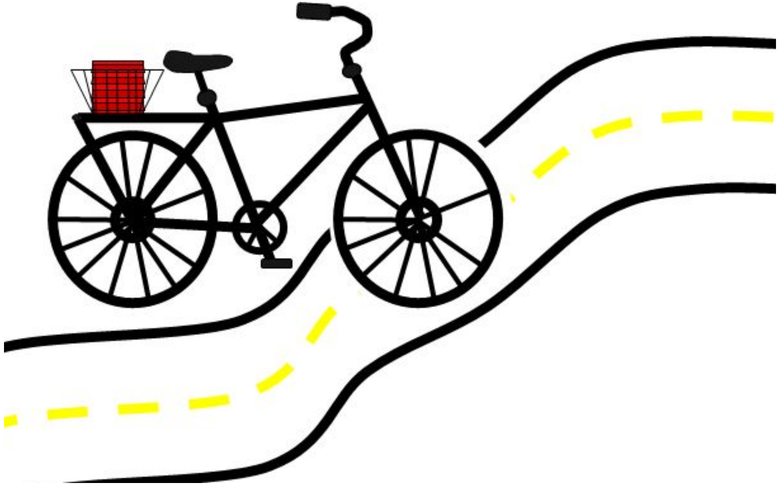
```
N,E,Measurement,Time,Date,Speed,Track Angle

51.49924667,6.546005,167.37,16:37:09,07/07/17,2.40,44.88
```

Testing our idea!!!

Source: D'Souza

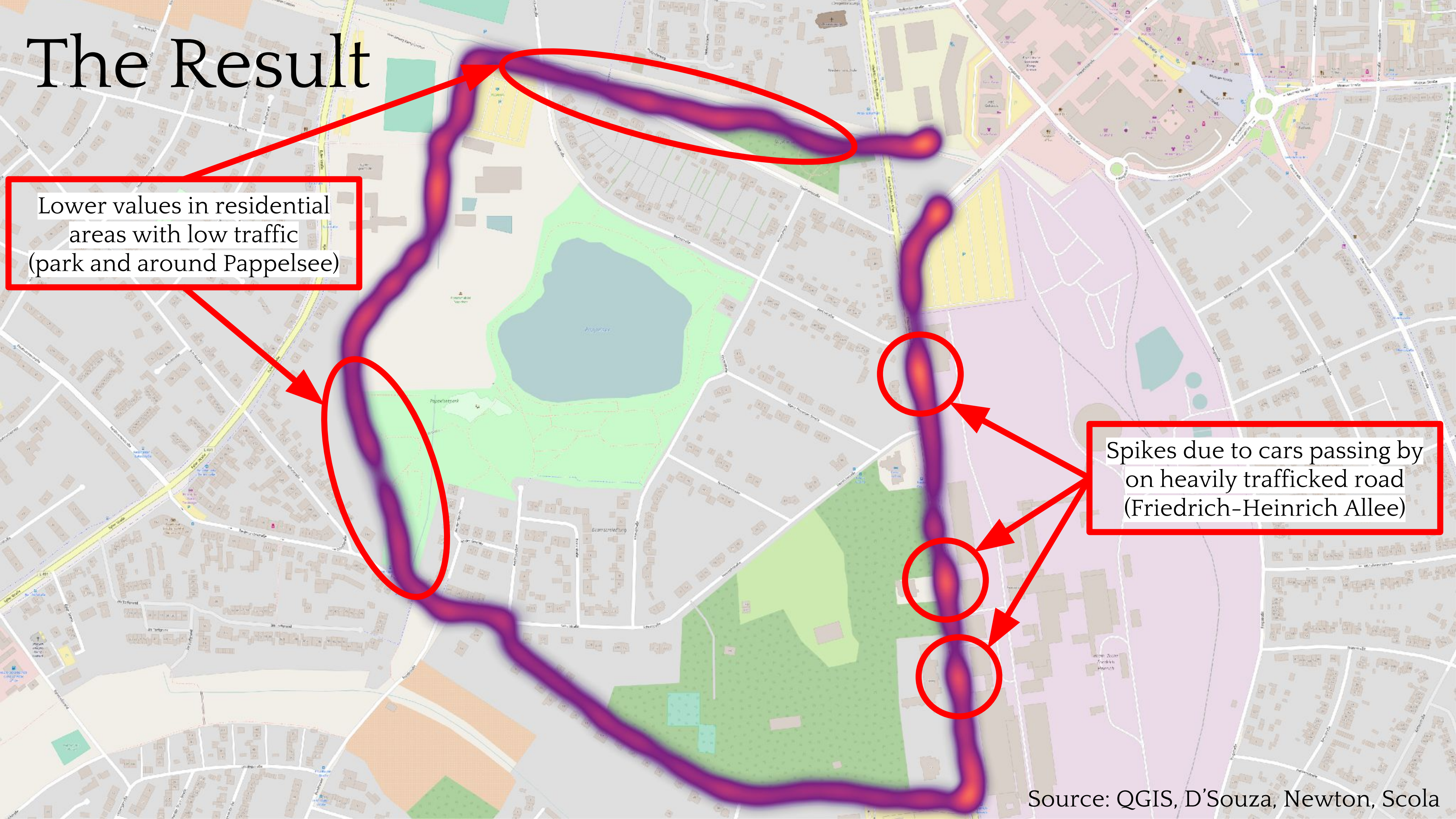
20 minute bike ride through a residential areas as well heavily trafficked areas



Visual representation of the data collected

Source: qgis.org

The Result



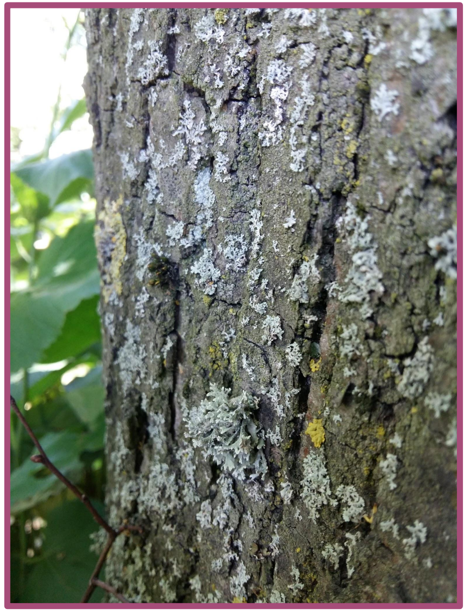
Lower values in residential areas with low traffic (park and around Pappelsee)

Spikes due to cars passing by on heavily trafficked road (Friedrich-Heinrich Allee)

Source: QGIS, D'Souza, Newton, Scola

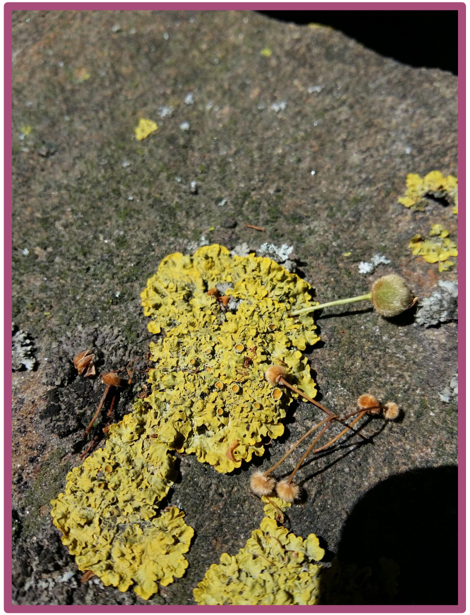
Lichens as natural air quality sensors

Close to the road



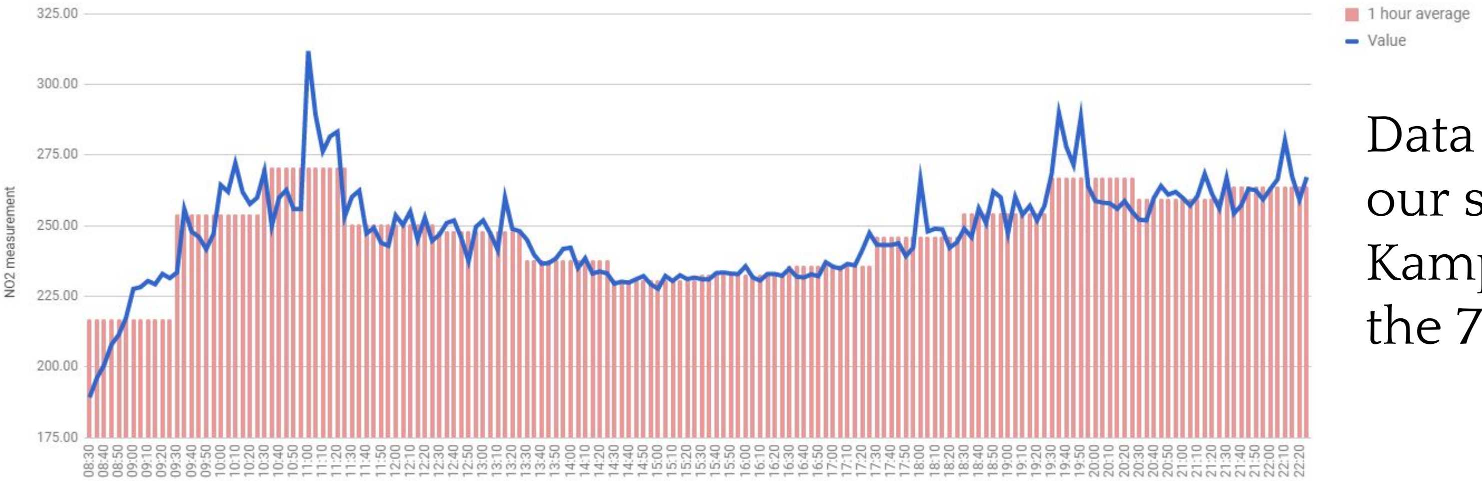
- Lichens that are dust resistant & nitrogen-loving tend to grow
- Sparse growth
- Dry and brittle

100m from the road

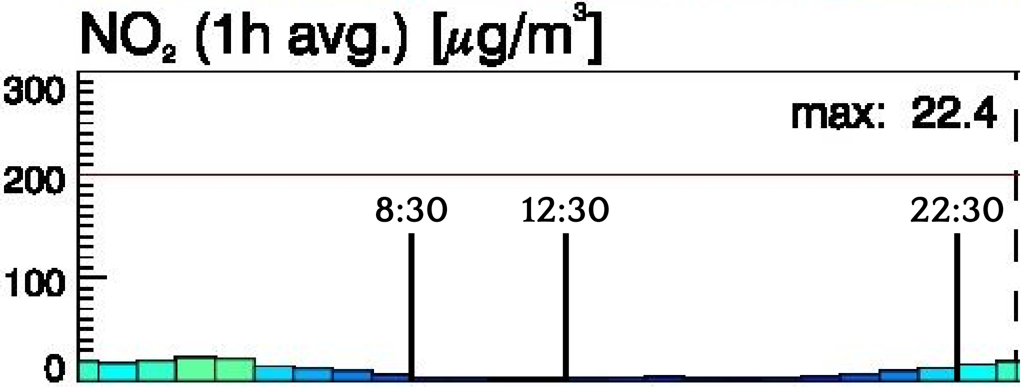


- More dense growth pattern
- More varieties of lichens
- “Furry” growth indicating good air quality

Reliability of our relative measurements



Data measured by our sensor in Kamp-Lintfort on the 7th of July



Data measured by LANUV in Moers on the 7th of July

And because we want the CANARY to live on..

instructables

https://www.instructables.com/id/The-CANARY-Arduino-Based-NO2-Sensor-and-Mapper/

References

- https://www.qgis.org/it/site/forusers/visualchangelog218/index.html
- http://www.bom.gov.au/climate/cdo/about/airtemp-measure.shtml
- http://www.batsocks.co.uk/img/XMega/LED_blink_320.gif
- Very sensitive porous silicon NO2 sensor, L. Pancheri et al.
- https://www.sgxsensortech.com/content/uploads/2014/08/1107_Datasheet-MiCS-2714.pdf