Combining FOSS4G & Open Hardware for Research & Monitoring in Southern Asia

Yann Chemin

International Water Management Institute

University of Moratuwa, Faculty of Architecture

FOSS4G
Free&Open Source Software for Geospatial
2013 Tokyo / Osaka

Consultative Group for International Agricultural Research
Ratified on October 2nd, 2013
Full Open Access & Open Source
Research data and publication

- International Public Goods
- Public Domain
- Publications Open Access
- FOSS models and algorithms

2018: all 15 CG centres, already FOSS4G Lab: (gsl.worldagroforestry.org)

Introduction

PyWPS+MWS
Rationale
MWS
MWS parts
MWS Setup
GRASS GIS
metaModule
pyGRASS
PyWPS

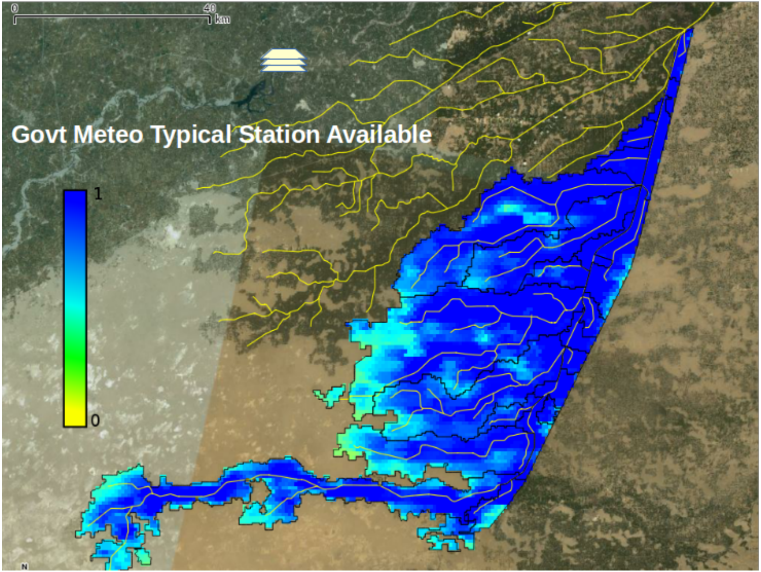
Road condition
Rationale
Components
System

Small Tanks Monitoring
Rationale
Autoboat
RaspberryPI
Sensors
FOSS4G

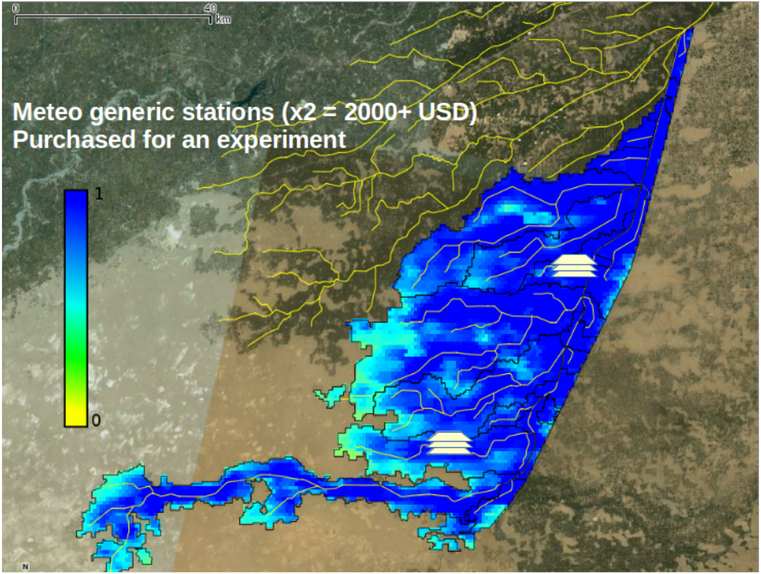
Conclusions

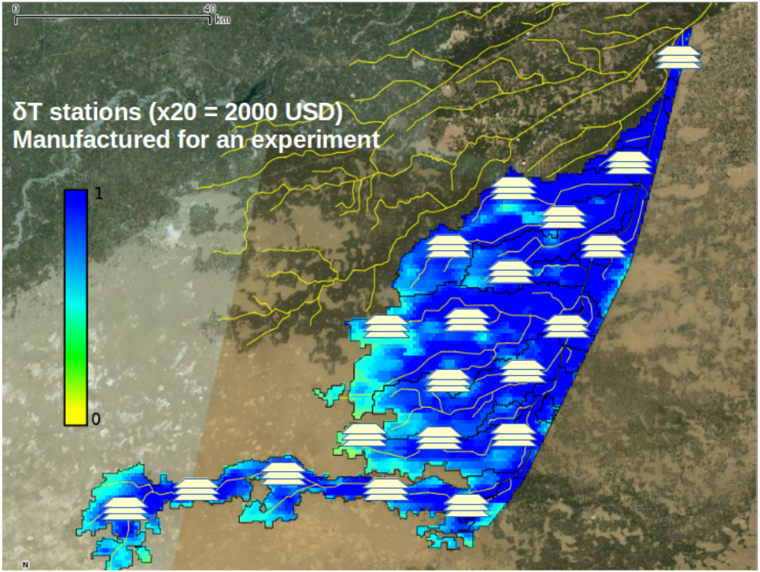
FOSS4G and Open Hardware
Developed together in new avenues

- Evapotranspiration calibration & modeling
- Road condition monitoring
- Rural tanks evaporation modeling



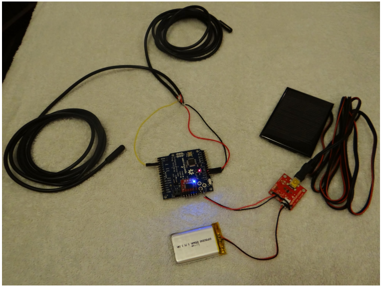
0
40
km
Govt Meteo Typical Station Available
1
0


0
40
km
Meteo generic stations (x2 = 2000+ USD)
Purchased for an experiment
1
0
N

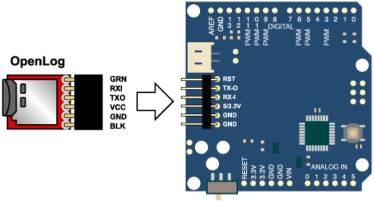

0
40
km
δT stations (x20 = 2000 USD)
Manufactured for an experiment
1
0

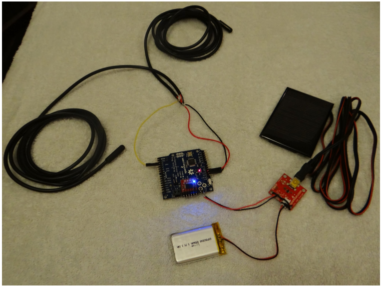
Micro Weather Station v1:
Temperature Profiler for ET models calibration

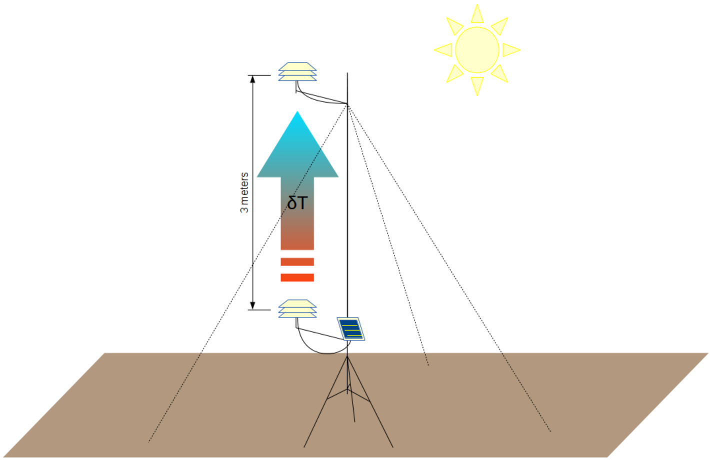
- Arduino Pro 3.3V
- Water-proof Digital Temperature Sensors
- Li-ion Battery + Solar Panel
- OpenLog data logger with SD card
- Cost < 100 USD



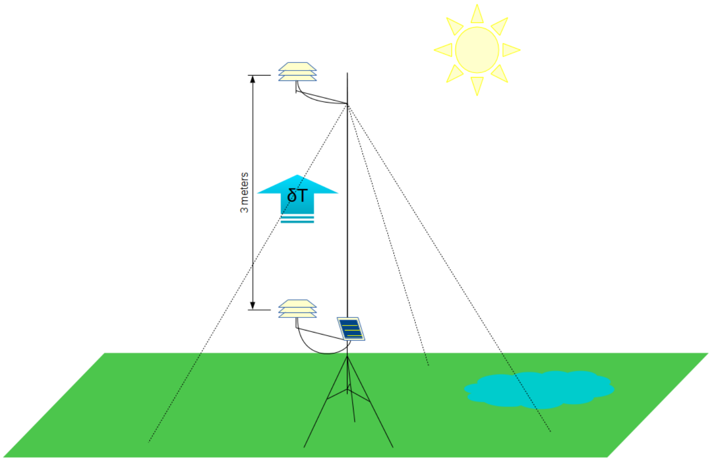
OpenLog + Arduino Pro

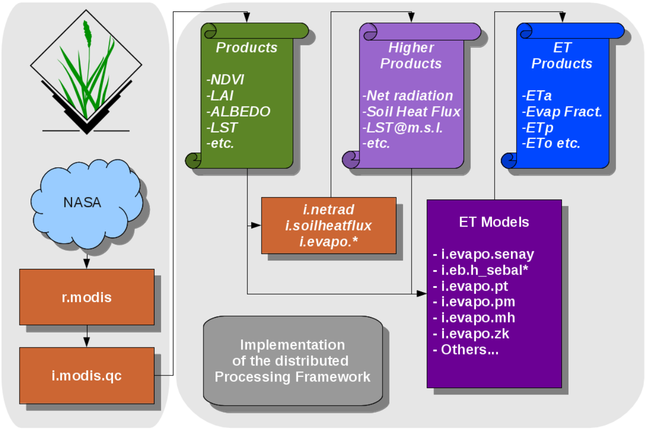






3 meters
δT


3 meters
δT


Products
-NDVI
-LAI
-ALBEDO
-LST
-etc.
Higher Products
-Net radiation
-Soil Heat Flux
-LST@m.s.l.
-etc.
ET Products
-ETa
-Evap Fract.
-ETp
-ETo etc.
NASA
r.modis
i.modis.qc
i.netrad
i.soilheatflux
i.evapo.*
ET Models
- i.evapo.senay
- i.eb.h_sebal*
- i.evapo.pt
- i.evapo.pm
- i.evapo.mh
- i.evapo.zk
- Others...
Implementation
of the distributed
Processing Framework

Pythonizing GRASS:
From Shell commands to Python functions

metaModule concept

1. **GRASS GIS:** Specific image processing modules
2. **PyWPS:** G modules called by Python
3. **GRASS script:** G mod. called by Python: metaModule
4. **pyGRASS:** G mod. called as Python fun.: metaModule
5. **PyWPS v4:** pyGRASS metaModule used directly **(TODO)**

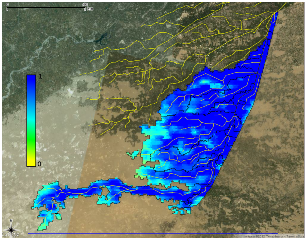
Summary for Landsat pyGRASS metaModule

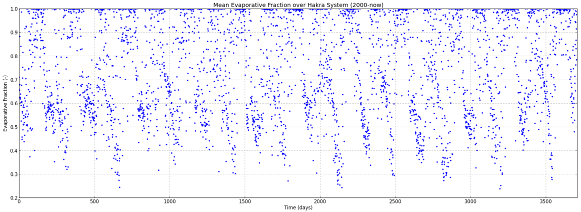
```
from grass import script as g
from grass.script import setup as gsetup
gisbase=os.environ['GISBASE']
gsetup.init(gisbase,gisdb,location,mapset)
from grass.pygrass.modules.shortcuts import raster as r
from grass.pygrass.modules.shortcuts import imagery as i
from grass.pygrass.modules.shortcuts import display as d

r.mapcalc(expression="vis=18",overwrite=OVR)
r.in_gdal(input=L7f,output=L7r,flags="e",overwrite=OVR)
r.proj(input="dem",location="Myanmar",memory=10000,resolution=90.0,overwrite=OVR)

i.landsat_toar(input_prefix=pref,output_prefix=outpref,
        metfile=metadata[0],sensor=LSENSOR,quiet=QIET,overwrite=OVR)

i.atcorr(input=b, elevation="dem", visibility="vis", parameters=prm,
        output=b_out, flags="ra", range=[0,1],quiet=QIET,overwrite=OVR)

i.landsat_acca(input_prefix=b_in,output=b_clouds,overwrite=OVR)
r.mask(raster=b_clouds,flags="i",overwrite=True)

i.vi(red=b3,nir=b4,output=b_ndvi,viname="ndvi",quiet=QIET,overwrite=OVR,finish_=False)
i.albedo(input=b_in,output=b_albedo,flags="lc",quiet=QIET,overwrite=OVR,finish_=False)
i.emissivity(input=b_ndvi, output=b_emissivity,quiet=QIET,overwrite=OVR,finish_=False)
```

http://grasswiki.osgeo.org/wiki/Python/pygrass

Irrigation water monitoring & management

- Map: Uniform colour is equity of water distribution
- Graph: Irrigation system equity (mm/d, daily, 12 years)





Developed by Jachym Cepicky (http://les-ejk.cz/)

- OGC WPS standard
- Server side
- Written in Python Language
- Version 4 in the making
- v4 Low-level API: integration with GRASS GIS
- v4 Possible pyGRASS support

PyWPS v2 style

WPS_hakra_ef.py (~/wps_processes/evapfr) - gedit

Fichier Édition Affichage Rechercher Outils Documents Aide

Ouvrir Enregistrer Annuler

WPS_hakra_ef.py

```
        # EF processing
        if os.system("i.eb.evapfr lst=lst ouput=hakra_ef_%s >&2" % (self.Inputs[0]['value'])):
            return    """Could not process Hakra EF map"""

        #Mask non Hakra Command Area
        if os.system("""r.mapcalc hakra_ef_%s"="if(isnull(MASK),null(),hakra_ef_%s)" >&2""" % (self.Inputs[0]
['value'])):
            return    """Could not clip Hakra Command Area"""

        # export
        if os.system("r.out.gdal in=hakra_ef_%s out=hakra_ef_%s.tif type=Float32 >&2" % (self.Inputs[0]
['value'],self.Inputs[0]['value'])):
            return    """Could not export Hakra EF map"""

        #clean the mess 2
        os.system("rm -f %s" % tmpfilelist)
        del rnd, tmpfilelist, f, lstfiles, wildcard, tmpdir


if __name__ == "__main__":

    p = Process()
    p.Inputs[0]['value'] = "2012-09-01"
```

Python ▾ Largeur des tabulations: 8 ▾ Lig 67, Col 9 INS

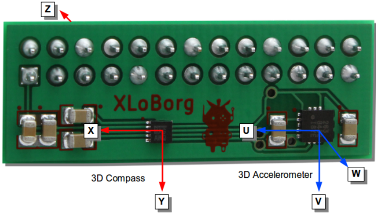
University of Moratuwa, F. of Archit., Urban Planning

- **Road condition:** chronic issue in Sri Lanka
- **RDA:** few IMU Vehicles (V. Expensive)
- **Challenge:** OSHW+FOSS4G $<$ 100 USD/vehicle
- **Solution:** GDAL/OGR + RaspberryPI

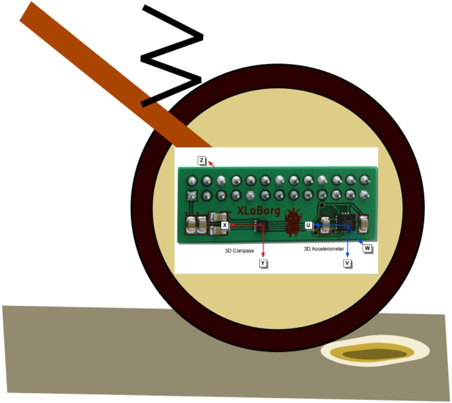
DEPARTMENT OF
TOWN & COUNTRY PLANNING

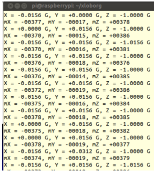
System setup on a vehicle:

- RaspberryPI
- + XloBorg Accelerometer
- + GPS
- + Python-OGR



Python-OGR reporting Z-axis anomalies into road Shapefiles by integrating Xloborg and GPS data





Water Resources Monitoring in Sri Lanka
Trans-basin water, Jaffna city pipeline, etc.

Characteristics

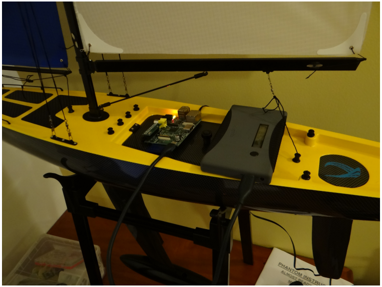
- Rural tanks (several thousands!)
- Cascade systems (interconnected)
- Water Storage capacity changes regularly
- Evaporative losses less known

Calibration of evaporative losses
and regular monitoring are much needed

Amitomi is a 1m-class autonomous sailing boat
Designed to survey small tanks temperature gradient
for calibrating Evaporation models

https://sites.google.com/site/amitomiautoboat

RaspberryPI as AmiTomi

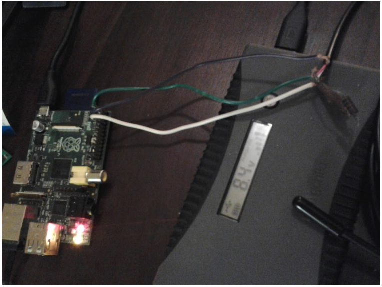


Boat itself

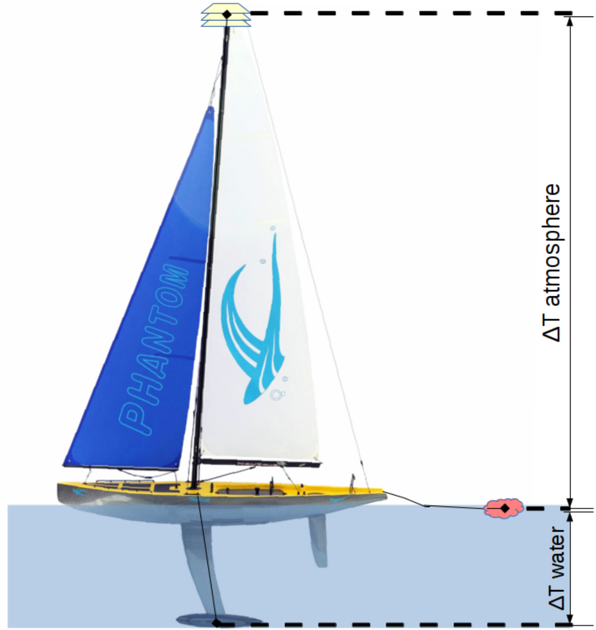
AmiTomi's brain is the RaspberyPI python code:

- Skipper: the captain/navigator software
- Waypoint sorter: optimizer for route
- Sensor datalogger: simultaneous sensing
- Mapper: import data and 3D interpolation

RaspberryPI GPIO connecting to temperature sensor



Temperature digital sensors (2m cables)



PHANTOM
ΔT atmosphere
ΔT water

- Python-gps (GPS data)
- Python-i2ctools (Compass/Temperature data)
- Python-XloBorg (Compass data)
- Python-openopt (Waypoints downwind sorting openopt.org)
- Python-MotorPiTX (servo control for sails & rudder)
- (py)GRASS (live processing of 3D GIS data)
- If online: PyWPS, SOS/network reporting.

FOSS4G natural extension is Open Source Hardware

- **RaspberryPI:** Small PC (ARM v8, Linux)
- **Arduino:** Micro-controller
- **OpenLog:** Data Logger
- **GDAL/OGR:** Flexible sensor raw data manipulation
- **GRASS GIS:** Mobile FOSS4G powerhouse
- **PyWPS:** Online GRASS GIS processing
- **Together:** Flexible all-in-one sensor-to-map solutions

International Water Management Institute

DEPARTMENT OF TOWN & COUNTRY PLANNING