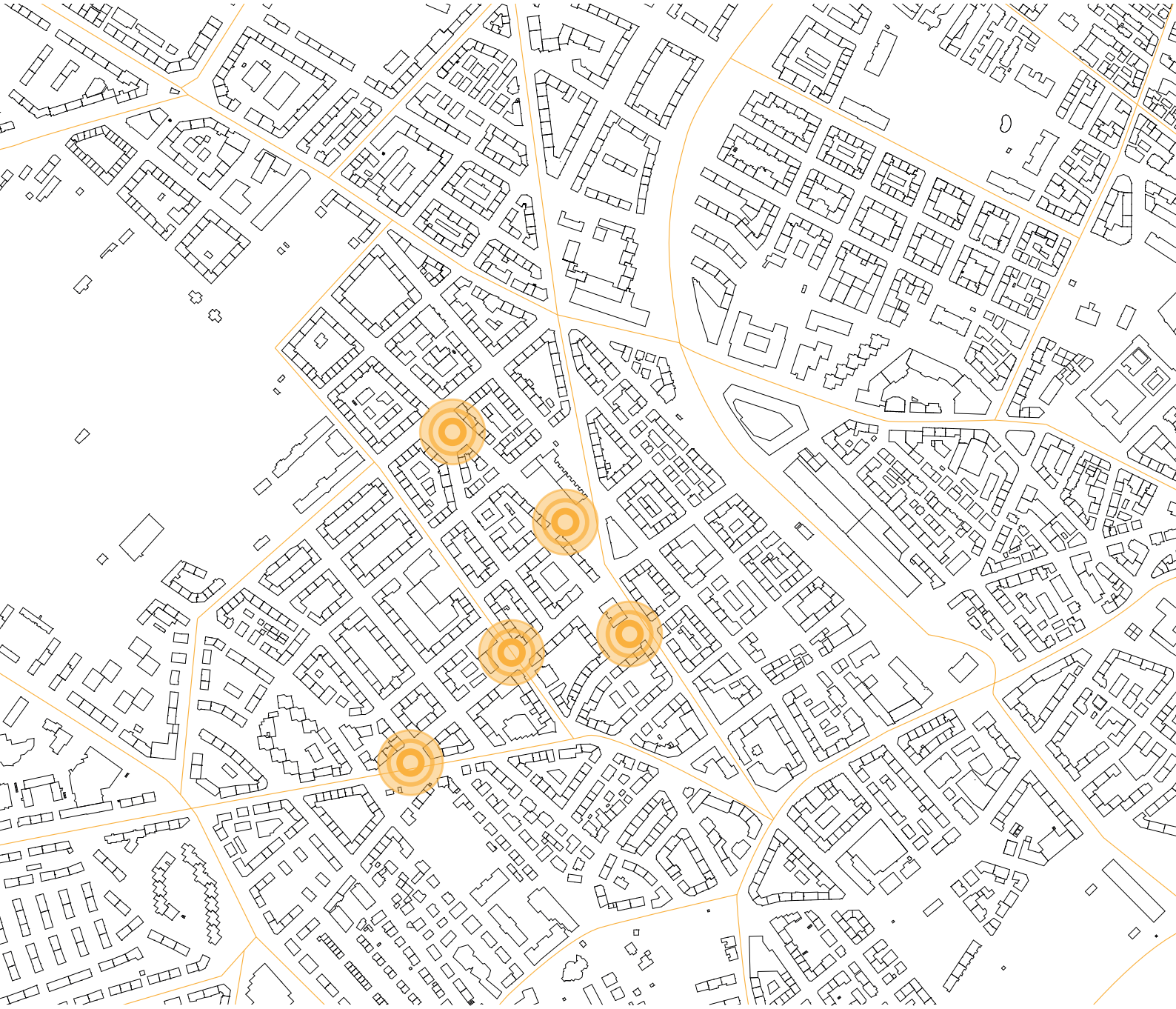
ETH zürich

September 2017

ESUM Urban Sensing Handbook

Component, Assembly and Operational Guide: Sensor backpack & 360° Videos



iA Chair of Information Architecture

ETH zürich

Bauhaus-Universität Weimar

FNSNF Fonds National Suisse Schweizerischer Nationalfonds Fondo Nazionale Svizzero Swiss National Science Foundation

DARCH

Chair of Information Architecture

ESUM Urban Sensing Handbook

Component, Assembly and Operational Guide: Sensor backpack & 360° Videos

iA Chair for Information Architecture

Website
http://esum.arch.ethz.ch/

Published by
Swiss Federal Institute of Technology in Zurich (ETHZ)
Department of Architecture
Chair of Information Architecture
Wolfgang-Pauli-Strasse 27, HIT H 31.6
8093 Zurich
Switzerland

Zurich, September 2017

Layout, Editing & Graphics
Brigitte Clements

Contact
griegod@arch.ethz.ch
standfest@archilyse.com

Cover picture:
Segment Analysis, Wiedikon, Zurich

DARCH

Contents


3
31
27

Overview

1.1 ESUM Project Overview

Authors: Danielle Griego, Saskia Kuliga*, Martin Bielik*, Matthias Standfest, Varun Kumar Ojha,
Sven Schneider*, Reinhard König, Dirk Donath*, Gerhard Schmitt (*Bauhaus University Weimar)

In this research project, we investigate the impacts of urban morphology (UM) on citizen's social potential as a function of accessibility and perception and compare it with the impacts of UM on building energy consumption for parallel case studies in Zürich Switzerland and Weimar Germany. This is of particular interest since urban planning decisions of urban morphology have long–term implications that affect not only the energy performance of the building stock, but also on people's experiential quality of the city environment.

In order to understand, measure and predict the impacts of UM on perception of citizens we developed a three-tiered empirical study, which draws upon the participants' experiences in a real-world experiment, along with two controlled experiments using a 360-degree video and a grey-scaled 3D virtual reality (VR) model of the city. All three experiments are conducted using the same pre-defined path in each of the two cities, for a total of six participant-based experiments. In each experiment, the participants subjectively rate the urban environment through surveys for an understanding of their explicit perception.

We also go beyond subjective impression ratings to gain further insight into the implicit perception of the participants by simultaneously measuring electro dermal activity (EDA) data collected via wearable devices. The EDA data is processed to quantify the frequency, duration and location of arousals along the experimental path to show a measure of excitation. The

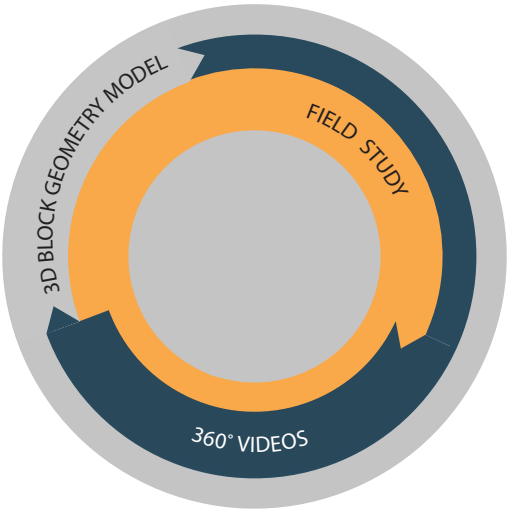


Figure 1: perception of urban morphology is empirically studies using a three tiered experimental approach.

explicit perception data provides the sentiment of the experience such as spacious/narrow, beautiful/ugly, or private/public (among others) while the implicit perception provides the “raw” arousal-based perception from participants.

We also consider the impacts of other “city dynamics”, or non-geometric factors on perception, including temperature, illuminance, sound and dust using data collected via a “sensor-backpack”, specifically designed for the use in the real-world participant studies. The produced from this experiment is highly complex; the seven sensors have varying frequencies and varying temporal dependencies. We therefore developed an information fusion mechanism to combine the spatial-temporal data to produce a structured dataset, which is analyzed using four state-of-the-art Machine Learning algorithms. The quantified physiological responses or “human perception” data are paired with the sensor-based environmental measurements using Machine Learning based knowledge discovery approaches. These include classification, fuzzy rule-based inference, feature selection, and clustering to derive underlying relationships between participants’ physiological response and environmental conditions. The results indicate that the participant’s implicit responses are affected by both urban morphological features measured as participant field-of-view and street width as well as the dynamic environmental features temperature, noise levels, illuminance and traffic speed.

The subsequent experiments are further abstractions of the real-world experiment to emphasize and evaluation the impact potential of urban morphology. For example, the 360-degree video holds all dynamic environmental variables constant and the grey-scaled 3D block geometry VR model purely focuses on participant response to the change in geometry, eliminating the influence of aesthetic qualities. The three experiments enable us to identify the “hot spots” of arousal as a function of pure urban form, thus gaining insight for how human perception is linked to UM.

For building energy demand and social accessibility potential, we draw upon existing methods to evaluate scenarios at the district scale (1km2) for the building stock where the participant experiments are conducted. The energy simulations utilize inputs that highlight the impacts of pure urban form including the building volume, usable floor area and adjacency with neighboring buildings. Accessibility similarly uses urban morphological features as the primary input parameters to simulate select isovist and centrality measures. The results of the energy and accessibility analysis are inherently linked to descriptors of UM and are compared to perception using a methodology that normalizes the data along the street network configuration. As a final outcome, we develop a predictive model to estimate the effects of pure UM on perception and quantitatively compare this with the energy demand and accessibility potential of neighbourhoods.

2. ESUM Sensor Backpack

Authors: Matthias Standfest, Constantinos Miltiadis, Danielle Griego, Alessandro Forino, Brigitte Clements, Ashris Choudhury

The Esum project uses an anthropogenic Urban Sensing Unit (colloqually referred to as the ESUM sensor backpack) to investigate the relationship between environmental data and how urban structures are perceived by inhabitants. The sensor backpack was created as a mobile sensor unit to measure, analyse and correlate dynamic environmental variables with perception measures. We have designed, developed and documented the hardware and software for a stable prototype, which can be replicated by other research labs around the world. This section discusses the various components chosen for the prototype backpack used for the initial experiments, how to assemble and operate the backpack. Troubleshooting and reflections are also included at the end of the section, which highlights potential alternatives to imporove the ESUM sensor. For necessary source codes refer to the Appedix.

In order to make the project portable we needed to find a backpack that could house the devices and sensors, in a way that would be comfortable to carry. Most specialized backpacks are very expensive and difficult to customize so we chose to design our own removable housing for the devices, that could be inserted to a normal backpack. We used the ETHz backpack that can be found at any ETH Store or online. The backpack was selected because it is fairly small, durable, made from waterproof fabric, has padding on the back, branded, does not draw attention, cheap and readily available from any ETH campus store.

This chapter is organised as follows:

- Specifications and guide for individual components
- Backpack and component set up
- Information regarding required software and source codes.
- Discussion of findings from initial prototype.

2.1 Components and Assembly

The ESUM backpack consists of a constellation of easily sourced and assembled components. The software was developed as a Visual Studio C# console application. The case is made from plywood and partly plexiglas, and can be laser cut and assembled in less than 1 hour. The fabrication plans, software source code and an extended setup are documented in this handbook.

Figure 6 : ESUM Sensor Backpack

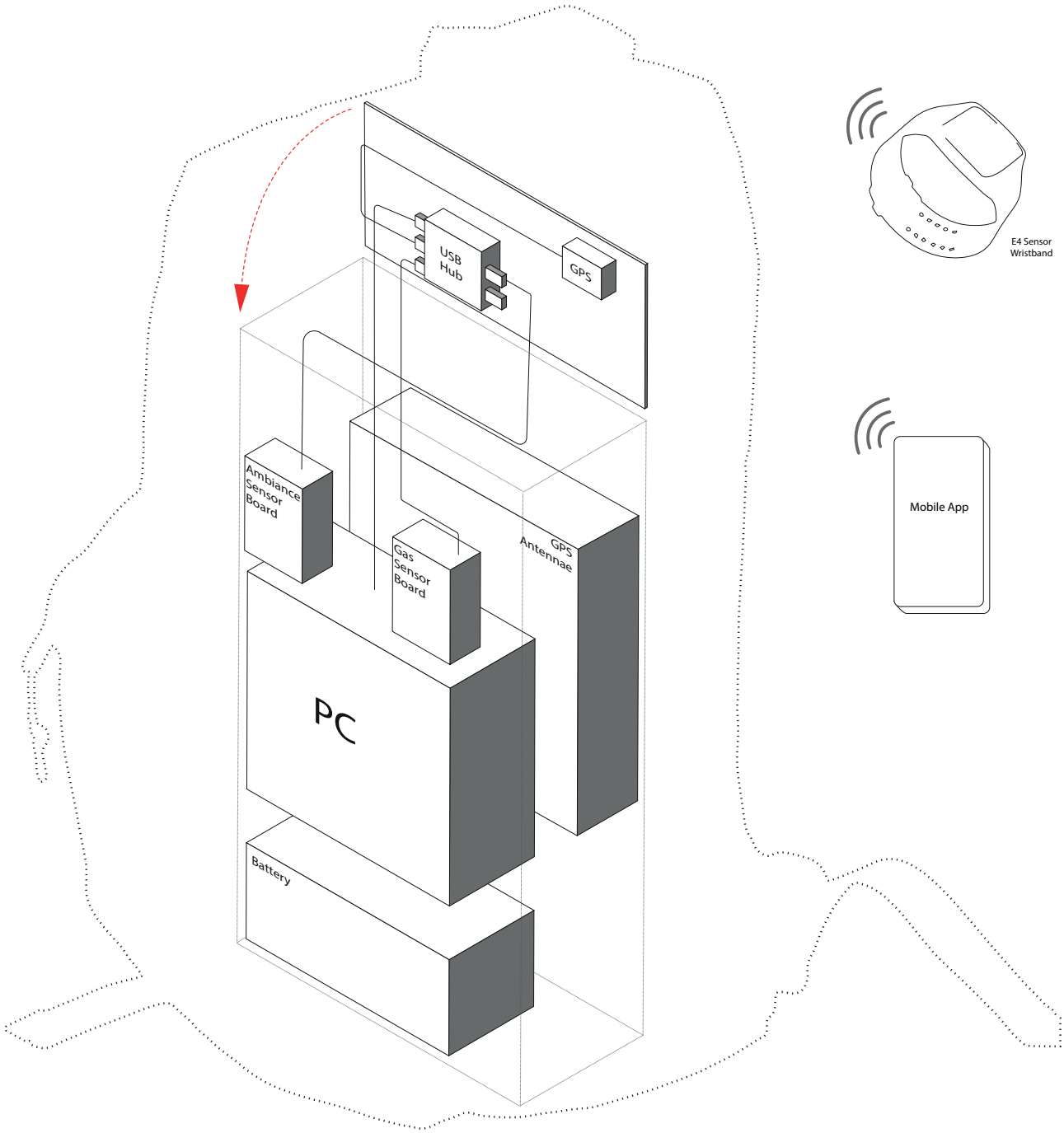


Figure 7: Diagramme showing component structure of the backpack. The various components are placed in a simple wood box and carried in the backpack.

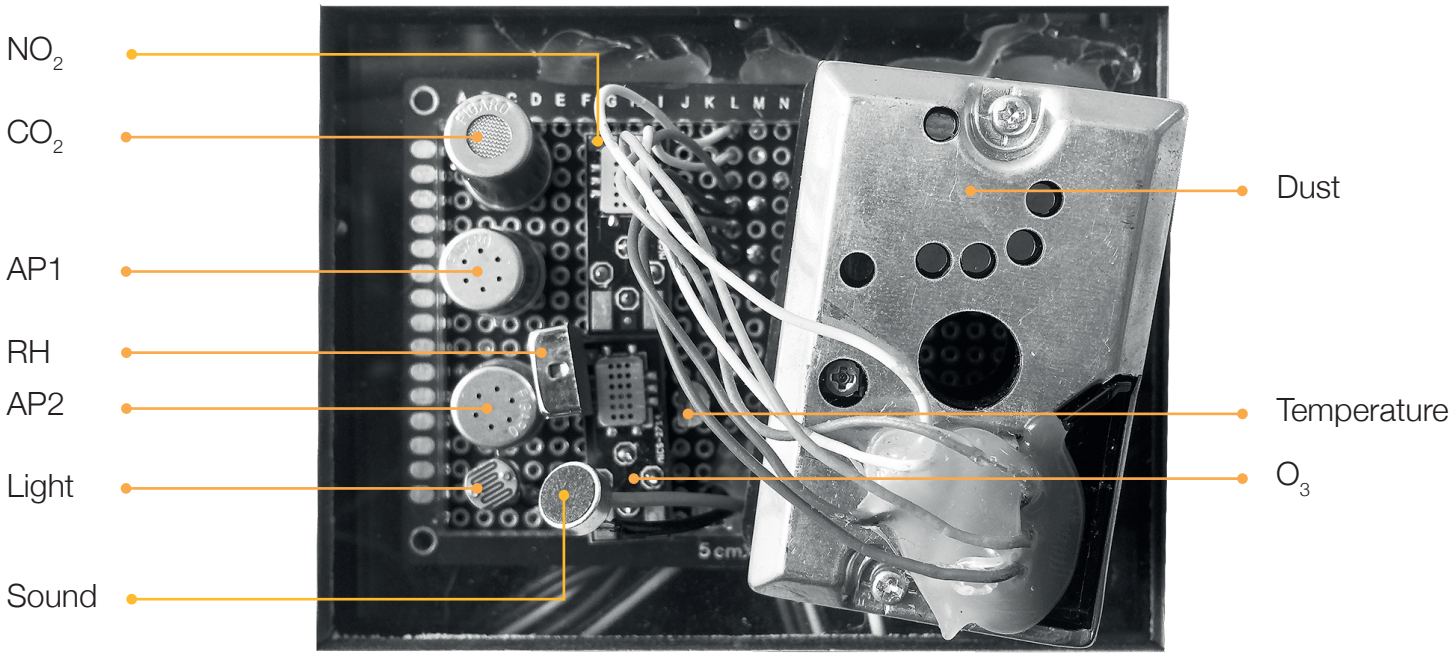


Figure 8 : ESUM Close-up of Environmental Sensors

The Ambiance Sensor Board (also referred to as the Smart Cities Board) measures non-gaseous qualities such as:

- Sound pressure
- Dust
- Illuminance.

The Gas Sensor Board measures:

- Temperature
- Atmospheric Pressure
- Relative Humidity
- CO_2
- O_3
- NO_2
- Air Pollutants (AP1 / AP2)*

Figure 9 : ESUM Sensor Backpack

* AP1 measures particulates that are between 2.5 and 10 micrometers in diameter (SO_2, NO_2, O_3, Co), whereas AP2 measures particulates that are 2.5 micrometers or smaller.

The NUC PC functions as the heart of the ESUM backpack. The PC is charged by a battery and thus works remotely, compiling the sensor data from the various auxilliary components on-the-go. In addition to the sensors and wifi sniffers attached to the PC, there are two remotely run components: an empatica sensor watch, which measures a person's physiological responses and a mobile phone application, which is used by the participant to measure their emotional responses at specific locations. Once an experiment is completed, the the sensor data is stored on the PC.

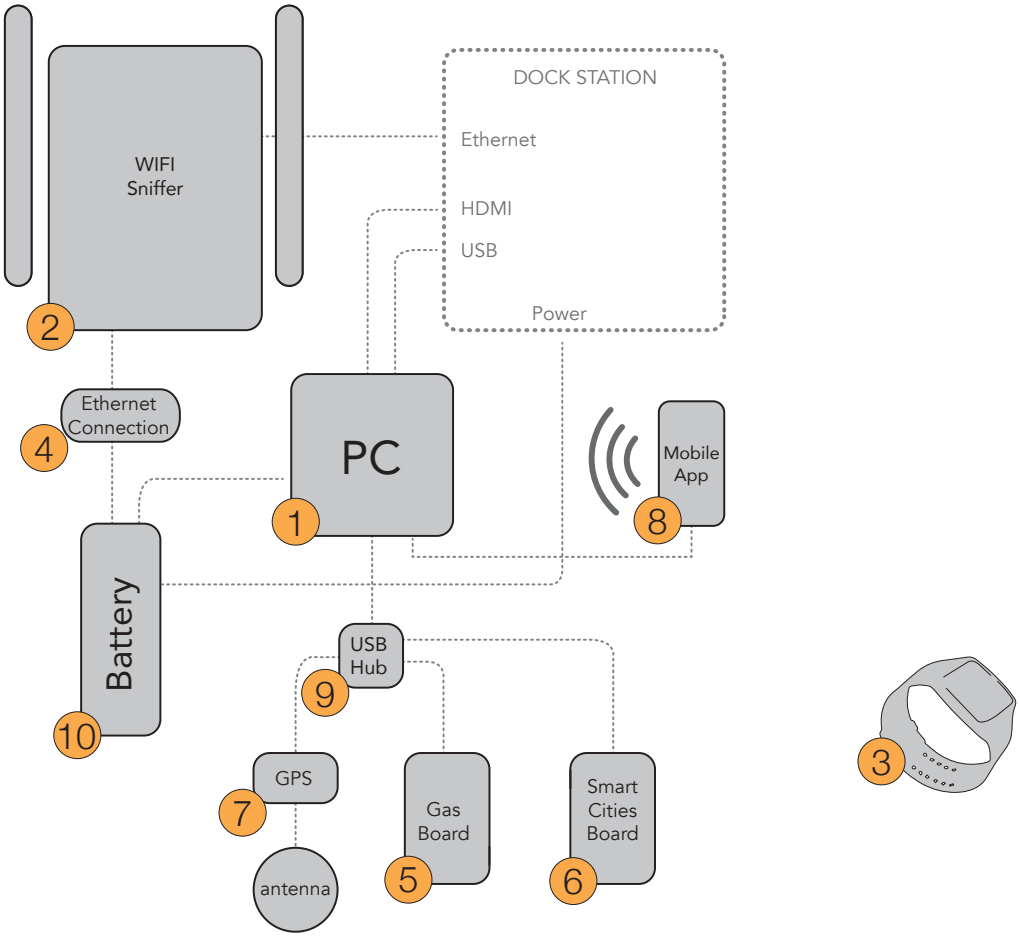
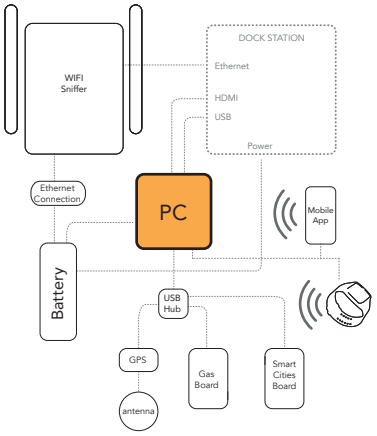


Figure 10 : Diagramme of individual backpack components and how they connect with each other.

Hardware specs used for the assembly of the ESUM Sensory Backpack

* Components housed in rucksack & box (custom made 4mm plywood box)

1 PC. Intel NUC



Manufacturer: Intel
Model: DN2820FYKH

Specifications & Features[1]:

The NUC is the barebone PC from Intel.
Intel Celeron N2820 @ 2.4Ghz
1xDDR3L SODIMM
Intel HD Graphics
HDMI adapter
1x USB3 port (front)
2x USB2 ports (back)
Ethernet port
Intel Wireless-N 7260BN (WiFi / Bluetooth)
Audio Jack
Current draw when Idle : 0.37 Ampere @ 12 Volt DC

Bought separately:

120GB Intel SSD[2]
8GB RAM
120GB Intel SSD[2]. The SSD (Solid State Drive) is significantly faster while consuming less energy. It is based onFlash Chip Technology which does not have moving parts making it less susceptible to impacts.
8GB RAM

Installation: The NUC has a visual BIOS preset for Windows 8. To install Windows 7 you will need to change the respective setting in the PC's Bios.
The latest drivers can be found at Intel's website[3].

*Requires extra RAM & SSD drive

The NUC PC used for the ESUM experiments is a fully comprehenive unit, which transmits information through a USB. The PC is the heart of the ESUM component where data is stored and communicated between the various devices.

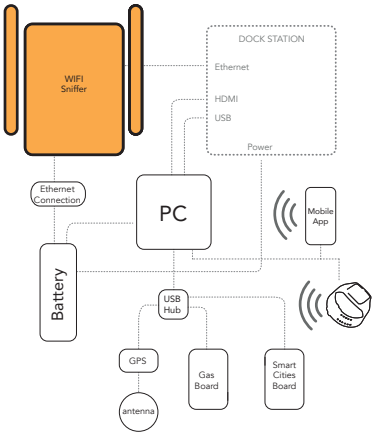
Once the backpack is turned on, a green light will glow. Note: if the green light is blinking it indicates that the PC is in hibernation mode. The PC should then either be reactivated or restarted.

1 http://www.intel.com/content/www/us/en/nuc/nuc-board-dn2820fykh.html

2 We chose an SSD (Solid State Drive) because it is faster and has less moving parts than the traditional HDD and therefore is not susceptible to impacts.

3 https://downloadcenter.intel.com/product/78953/Intel-NUC-Kit-DN-2820FYKH

2 Wifi-sniffer



Manufacturer: Libelium
Model: Meshlium Scanner AP

Specifications & Features[1]:

Meshlium comes in various models depending on the required functionalities: (Wifi, Wifi API, Bluetooth, 3G).ESUM uses the WiFi Scanner AP model. It is used to measure Wifi and Bluetooth activity around it. It runs Debian Linux.

Processor 500MHz (x86)
RAM memory 256MB (DDR)
Disk memory 8GB / 16GB / 32GB *
Power 5W (18V)
Power Source POE (Power Over Ethernet)
Normal Current Consumption 270mA
High Current Consumption 450mA
Enclosure Material: Aluminium
Dimensions: 210 x 175 x 50 mm
Weight: 1,2Kg

Time to start all services: 90 Seconds

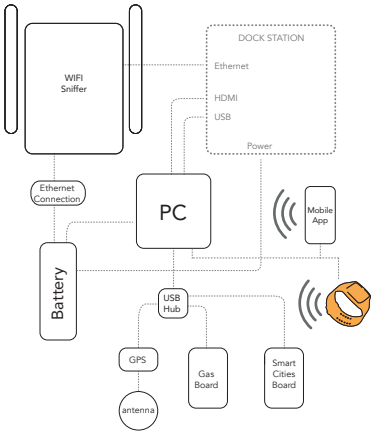
Installation: Connect the required antennas
(the Wifi AP model requires 2).

The Meshlium Wifi Sniffer is a powerful bluetooth and wifi access point used to detect iPhone and Android devices or any device which works with WiFi or Bluetooth interfaces. These devices can be detected without the need of being connected to an specific access point, enabling the detection of devices which comes into the coverage area of Meshlium.

This sensor is used to measure the number of people and cars present in a certain point at a specific time, allowing the study of the evolution of the traffic congestion of pedestrians and vehicles.

1 http://www.libelium.com/uploads/2013/02/meshlium-datasheet_eng.pdf

3 Wearable device



Manufacturer: Empatica
Model: E4 Wristwatch

All of the sensors in the E4 wristband are embedded in the device: the PPG sensor and temperature sensore are on the bottom side of the device while the wristband hold the EDA electrodes. The wristband has sensors for: Photoplethysmography (PPG) - Continuous Heart Rate (Heart Rate Variability, Stress, Relaxation), Electrodermal Activity (EDA) - Skin conductance i.e. Galvanic Skin Response (Arousal, Excitement)Temperature and Heat Flux- Activity, Context Info, 3-axis Accelerometer - Movement, activity. The E4 offers two modes of function: Streaming mode - Sends the data to a mobile app in real-time or record mode, which records the data internally. We use the Record mode to prevent data loss.. The Wristband can hold sensor data for 30 hours on its on-board flash memory.

Specifications & Features:

Flash Memory & 60+ hours of data storage

1 https://www.empatica.com/e4-wristband

https://www.empatica.com/product-e4-download

http://support.empatica.com/hc/en-us/sections/200008246-Empatica-Manager-Application-for-OSX-and-Windows

Setup:

1. Charge the wristband via USB through its USB charger

2. Create an account on the Website of Empatica

3. Install Empatica Manager12 (OSX & Win)

Operation:

1. Communication with the device through upper left indicator button.

- Short Press: Momentary. When On, will log an event mark. When Off will not result in anything
- Standard Press: 1-3 Seconds - Switches the device On/Off
- Long press: >5 Seconds - Resets the device
  * CAUTION * after a Reset the device might need to be connected to a computer over USB to be reactivated.

2. Wear the wristband, and do a Standard Press (until the LED flashes Green - then release it). The wristband searches for a Bluetooth LE connection. If it doesn't find any, in 40 seconds, it goes into Record mode, by blinking a Red LED for 20 seconds.

3. To log an event, make a short press.

4. End the Recording by a Standard Button press.

Data Retrieval

1. Connect the wristband to the computer over USB and start the Empatica Manager, while being connected to the internet.

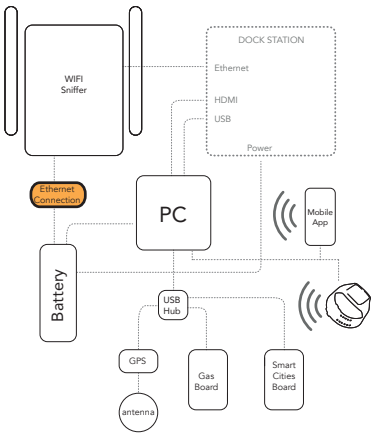
2. Go to the Empatica website and log-in.

3. On the calendar you can see your sessions.

4. By selecting one you can View/Download or Delete it.

5. Download. The zip file will have 6 individual CSV sensor files. The "info.txt" file explains the data format.

4 Microcontroller



Manufacturer: Libelium
Model: Waspmote

Specifications & Features:

The Libelium Waspmote[1] is an "Internet of Things" microcontroller platform developed by Libelium and based on the Arduino microcontrollers, for the purpose of sensor data logging in a wireless sensor network.

Microcontroller: ATmega1281 14MHz
Frequency: 14 MHz
SRAM: 8 KB
EEPROM: 4 KB
FLASH: 128 KB
SD Card: 2 GB
Weight: 20 grams
Dimensions: 73.5 x 51 x 13 mm
Temperature range: -10˚ C - 65˚ C

Harware:

The Waspmote configuration is made of 5 parts: 1. Battery (charge batteries at least 24 hours before use), 2. Antennae, 3. Modules, 4. Sensor Boards, 5. SD card.

Installation & Test:

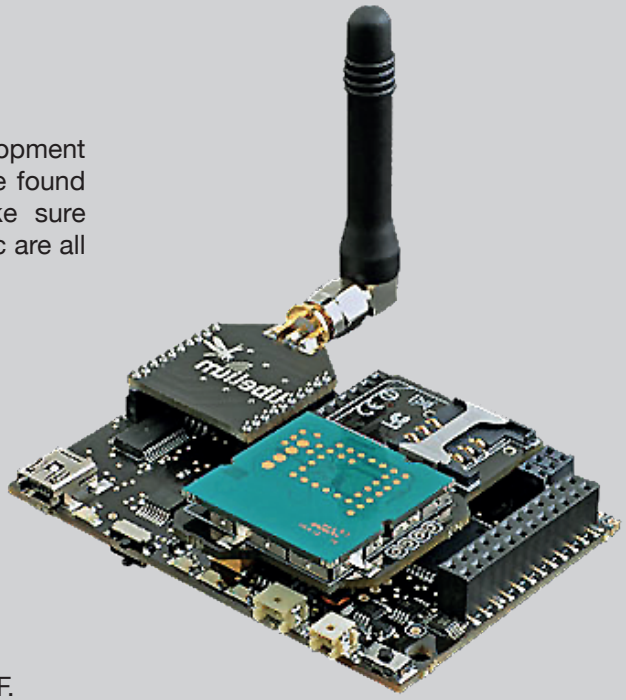
The first step is to install the Waspmote-IDE[2] (Integrated Development Environment) used to program Waspmote. This IDE can be found on: http://www.libelium.com/development/waspmote. Make sure the latest FTDI drivers[3] is installed. Linus, Windows and Mac are all compatible.

Developer Guides Use:

A program for Waspmote is composed of 4 parts:
1. configuration (RTC, sensors, communications module)
2. read sensor(s)
3. communications (XBee, LoRa, WiFi, GPRS, ...)
4. enter sleep mode

Operation:

Push the switches towards the USB connector to power OFF.
Push the other way for switching ON.



1 http://www.cooking-hacks.com/documentation/tutorials/waspmote

2 http://www.libelium.com/products/waspmote/hardware/

3 http://www.libelium.com/development/waspmote/sdk_applications

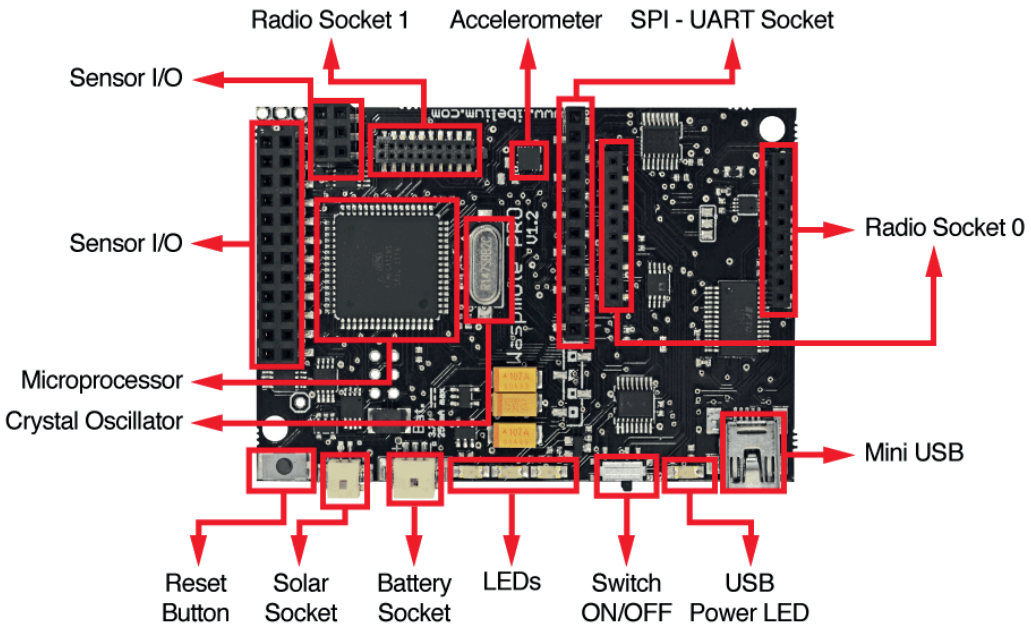


fig. 1.5 Front side of board

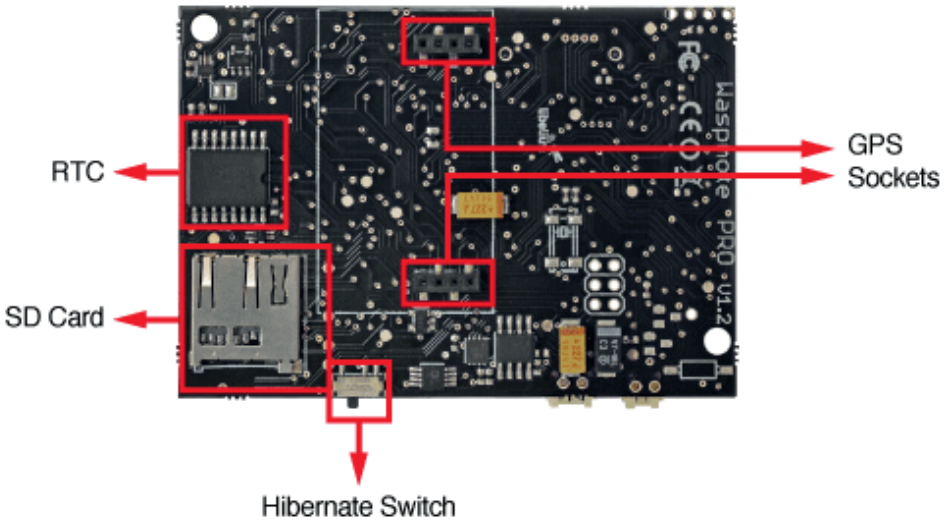
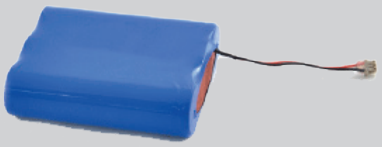


fig. 1.6 Back side of board

Note:

The Waspmote microcontrollers need a battery in order to function, regardless of if they are connected to a power source via USB, otherwise they will output an error (E#). Libelium offers a high capacity 6600MA/H LiPo battery[5] for their Waspmote range. The battery level can be monitored (in percentage) by the Waspmote microcontrollers. It is important to note that the battery pluf is not very common and thus not easy to interchange with other 3rd party batteries. The specific battery needs about 8h to charge fully and it is recommended that the battery is charged at least 24 hours before use. It is not adviced to leave the battery completely empty.

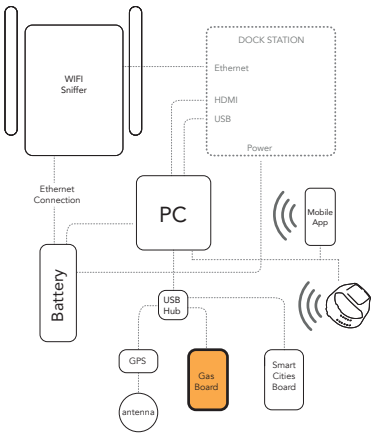


4 http://www.ftdichip.com/Drivers/VCP.htm

5 http://www.cooking-hacks.com/shop/waspmote/accessories/6600ma-h-rechargeable-battery

6 http://www.libelium.com/development/waspmote/documentation

5 Gas Sensor Board



Manufacturer: Liberlium
Model: V2 Gas Sensor Board

Specifications & Features:

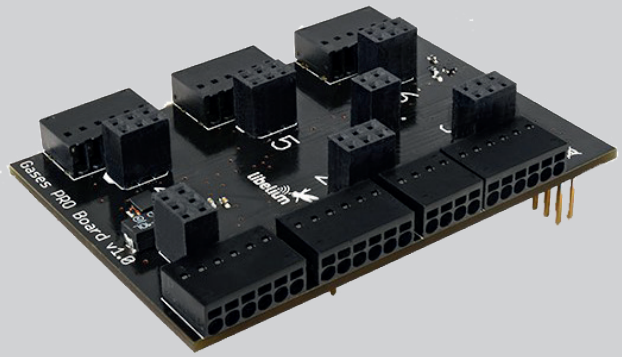
The Waspmote Gases 2.0 board has been designed to monitor environmental parameters such as temperature, humidity, atmospheric pressure and 14 different types of gases. It allows the inclusion of 6 gases sensors at the same time, the regulation of their power through a system of solid state switches and the amplification of the output signal of each one of them through a non-inverting amplification stage of a maximum gain of 101 controlled by a digital potentiometer configurable through the Inter-Integrated Circuit Bus, I2C).

ESUM uses the following sensors:

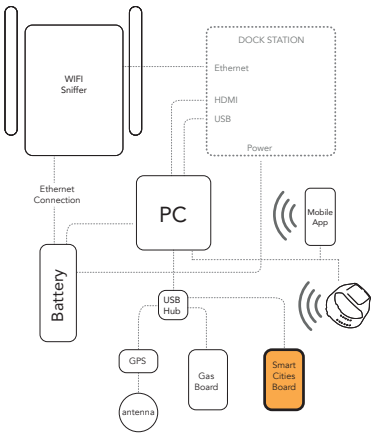
- Temperature
- Atmospheric pressure
- Humidity
- CO_2 - Carbon Dioxide
- O3 - Ozone
- NO_2 - Nitrogen Dioxide
- Air pollutants I : C4H10, CH3CH2OH, H2, CO, CH4
- Air pollutants II : C6H5CH3, H2S, CH3CH2OH, NH3, H2

Note: Both the gas and smart cities boards are standard from the same manufacturere for standard testing.

The Gases 2.0 board does not require any handling of the hardware by the user except for placing the sensors in their corresponding position. In the section dedicated to each connector we can see an image of each of the sensors inserted into the corresponding socket with the reference to the sensor's direction highlighted.



6 Smart Cities Board



Manufacturer: Libelium
Model: V2 Smart Cities Board

Specifications & Features:

The purpose of the Waspmote Smart Cities Board is to extend the monitoring functionalities of the Smart Metering Board from indoor environments to outdoor locations. Apart from the humidity and temperature sensors, present in all theWaspmote boards, other sensors for specific applications have been includedsuch as particle sensors, noise and illuminance.

ESUM uses the following sensors:

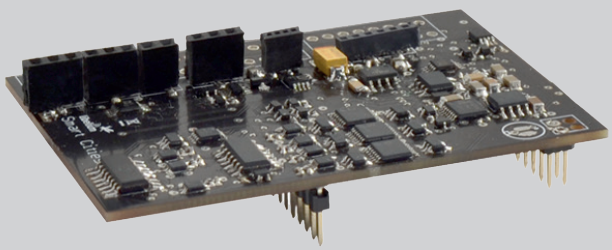
- Illuminance
- Dust
- Sound pressure

Specifications:

Weight: 20gr
Dimensions: 73.5 x 51 x 1.3 mm
Temperature Range: [-20ºC, 65ºC]

Note: You have to charge the local battery and connect it in order to read sensor values

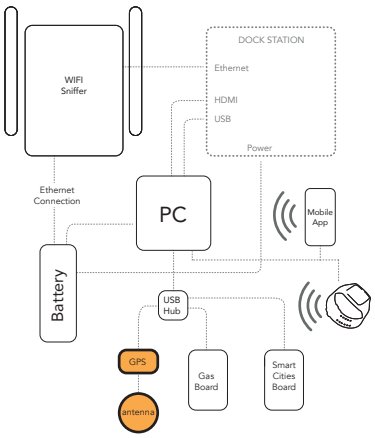
The Smart Cities board does not require of any handling of the hardware by the user except for placing the sensors in their corresponding position. In the section dedicated to each connector we can see an image of the connections between the socket and its corresponding sensor.



1 http://www.cooking-hacks.com/shop/waspmote/sensor-boards/waspmote-smart-cities-sensor-board

2 http://www.libelium.com/development/waspmote/documentation/smart-cities-board-technical-guide/

7 GPS



Manufacturer: Tallysman Wireless Inc
Model: Ducat 10 / Ublox Neo-M8N[1]
Antenna: TW2410[2]

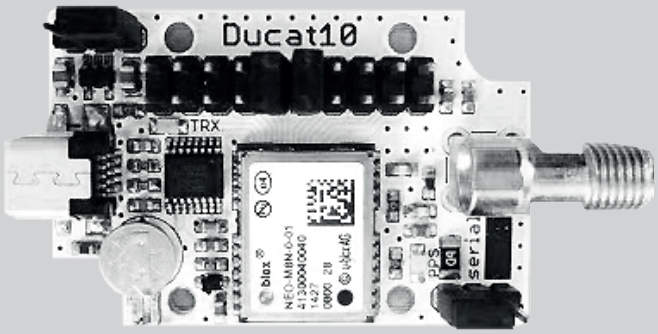
Installation:

To install the module's drivers, you can follow the instructions found on the M8 product page[3]. After installing and restarting your PC, you should be able to receive GPS data through a Serial Port Monitor (eg. Arduino). uBlox offers u-Connect[5] a software designed for their GPS modules, which can monitor the data and also edit the parameters of the module.

Notes:

GPS technology does function only outdoors and only when it has an optical connection to a satellite. Dense urban environments might obstruct its connectivity.

The TW2410/TW2412 features a dual-feed wideband patch element, with a two stage Low Noise Amplifier, comprised of one input LNA per feed, a mid section SAW to filter the combined output, and a final output gain stage. This configuration provides excellent axial ratio that is constant across the full frequency band. An optional tight pre-filter is available with part number TW2412 to protect against saturation by high level sub-harmonics and L-Band signals.



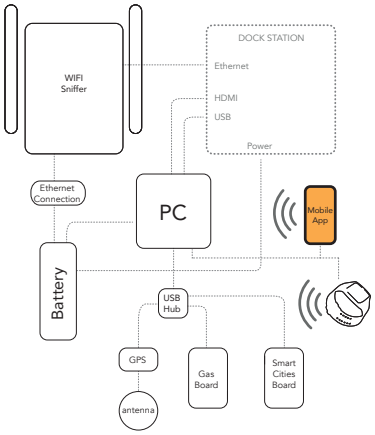
1 http://www.onetalent-gnss.com/ideas/usb-hw-receivers/ducat10

2 http://www.tallysman.com/TW2410.php

3 http://www.u-blox.com/en/drivers-a-middleware/usb-drivers/windows-7-driver.html

4 http://www.u-blox.ch/en/evaluation-tools-a-software/u-center/u-center.html

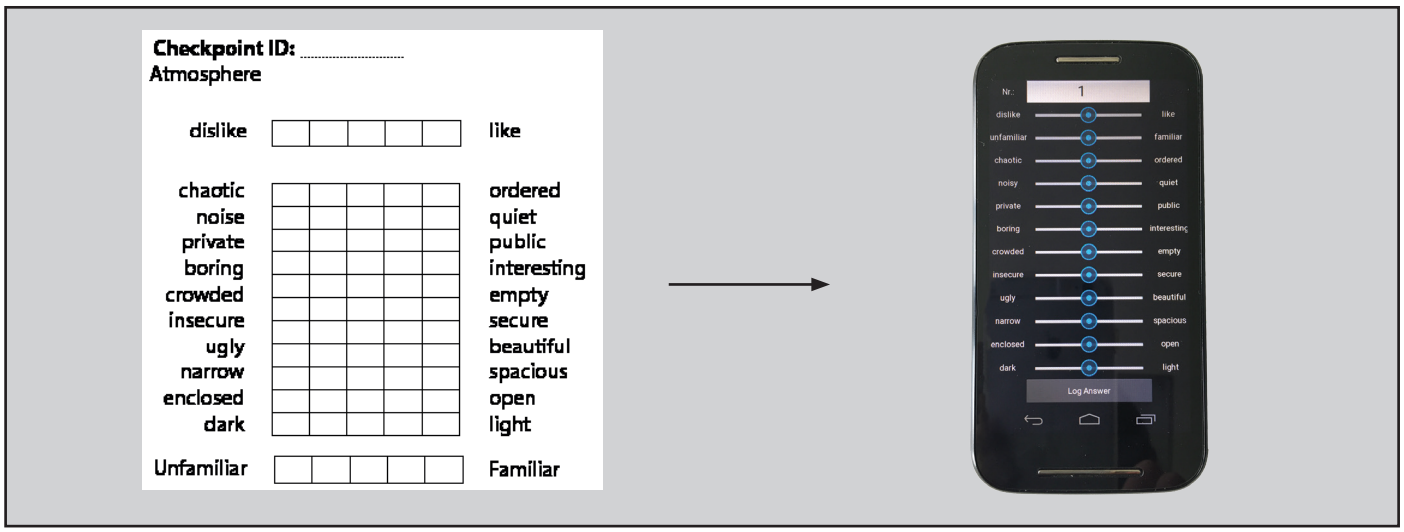
8 Mobile App



Manufacturer: Motorola
Model: Moto E-430

- 4 GB
- black
- Android 4.4
- GPS
- wifi connectivity

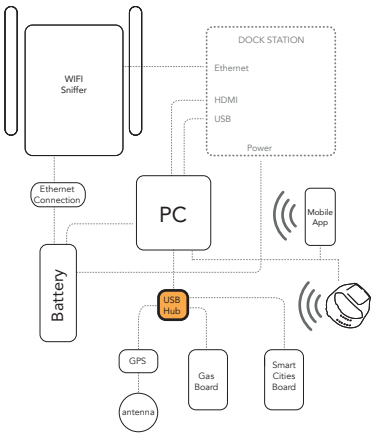
The mobile phone app was programmed to easily record a 12 question survey for the urban experiments. Partipants were asked to rank various qualities in a five-point survey. See below for actual mobile app questionnaire.



1 http://www.libelium.com/uploads/2013/02/meshlium-datasheet_eng.pdf

2 http://www.onetalent-gnss.com/contact-us

9 USB Hub



Manufacturer: Logitec
Model: Premium 4-Port USB Hub

Specifications & Features:

The backpack requires a number of USB connections between sensor devices and the PC. The NUC PC offers already 3 USB Connections.

Required connections:

- Waspmote Smart Cities
- Waspmote Gases
- Emotiv Epoc EEG
- GPS module
- USB Clicker
- Extra dock station connections:
- Keyboard
- Mouse

For this reason the ESUM prototype uses a simple 4xUSB 2.0 HUB, that is passive (does not require external power source)

2.2 Operational Guide

Biofeedback wristband setup (in Recording mode)

1. Log-in to empaticaManager application (Need to have internet connection)
2. Connect device to computer;
   a. If no data to upload, indicator light on the wristband should alternate between green and yellow, then go to yellow when it is charging and turn off when it is fully charged
   b. If data to upload, indicator light will show green, blue, yellow, white, purple, etc and the application will show that it is downloading the data from the device and uploading it to the <empatica server.
3. Once the data is downloaded the wristband is ready for the next participant, however make sure that the device is charged

Backpack and Mobile App

1. Turn on the power of the backpack (Red on/off switch, then power button on the computer)
2. Turn on the phone
3. Make sure the phone is connected with the wifi “meshlium”, may need to wait a couple of minutes until the wifi is available
4. Start the app “ESUM Logger” on the phone
5. Connect to meshlium wifi on your laptop in order to connect to the VPN
6. Check the status of all devices via vnc viewer (IP 10.10.10.13)
7. Turn off computer via standard shut down menu option

External environmental sensors[1]:

1. Plug in and load in the HOBO software
2. Launch device
3. Attach to backpack
4. Note that with 1 second measurements, it only logs for 4 hours

Extract the meshlium data from the Mysql database:

1. Double click on „query.sql“
2. Click on database_connect

1 http://www.onsetcomp.com/products/data-loggers/U12-data-loggers

3. Select the local instance in the dialogue field and click „ok“
4. Go back to the „query“ tab
5. Change the date-range from when you want the data
6. Click on the lightning sign to execute the script
7. Click on export above the table to save the file into a csv file

How to tell everything works:

1. The light on the nuc power button glows (no blinking)
   (blinking means it’s in hibernate mode)
2. The owntracks app reports „connected“ on the pull down menu on the phone, if it doesn’t, reconnect to the wifi
3. The ESUM app is starting correctly

Where the data is stored:

1. All the waspmote and on-board gps data: iA/Dropbox/ESUM_data/sensors/$(date and time)
2. The phone gps: iA/Dropbox/ESUM_data/location/$(date and time)
3. The phone logger: : iA/Dropbox/ESUM_dataro/status/$(date and time)

Regarding the battery:

1. Estimated time is around 5-6h (4 hours to be safe and should check via the VPN to be safe)
2. Keep charged when not in use, especially over night or at least 2 hours on the day of use.
   When it is unplugged it can de-charge because of the waspmote batteries

2.3 Reflections

There were complications with dealing with the two sensor boards. The troubleshooting proved complicated because of the lack of flexibility inherent when using a NUS PC. The NUS PC transmits information through a USB and as such is bound to high level electronics, meaning the PC unit is a complete comprehensive unit. One way to bypass this would be to replace the NUS PC with a Raspberry Pi single board computer, which has a directly built-in WIFI. A significant disadvantage of using this method is the extra expertise and fluency in programming knowledge. However, using the Raspberry Pi offers far more flexibility for low level electronics, meaning that individual components and sensors would be individually sourced from various manufacturers and thus be individually calibrated. In addition to the benefits of flexibility, replacing the NUS PC with the Raspberry Pi would allow for far more control and is significantly cheaper. Such a configuration would also make it easier to understand where specific problems lie when troubleshooting.

In addition, there is reason to argue for the replacement of the Wifi sniffer. Despite having minimal problem with the Meshlium Wifi Sniffer, using this product for such a basic purpose is a bit of overkill. The Meshlium Wifi Sniffer is a powerful bluetooth and wifi access point. This particular model is very powerful and expensive. As a result it could be replaced with a simple and inexpensive wifi chip. This would be directly compatible with the Raspberry Pi and thus strengthens the argument for also changing computer systems. For the purpose of solely sniffing out wifi signals, a wifi chip would be plugged into the Raspberry Pi single board computer and could then run on a programme such as “wireshark”. It is important to note that this configuration has not been tested and these alterations are simply suggestions for other possible outfitting based on difficulties experienced in the initial case study.

In regards to the phone and other auxilliary applicationg, a KIVY Framework could be used. KIVY is an open source, cross-platform Python framework used specifically in the development of mobile applications and other multi-touch application software with a NUI (“natural user interface”). It can run on Androis, iOS, Linus, OS X, and Windows. The phone application works as a client and sends MQTT data to the PC. The Python script receives the MQTT data and stores them into files.

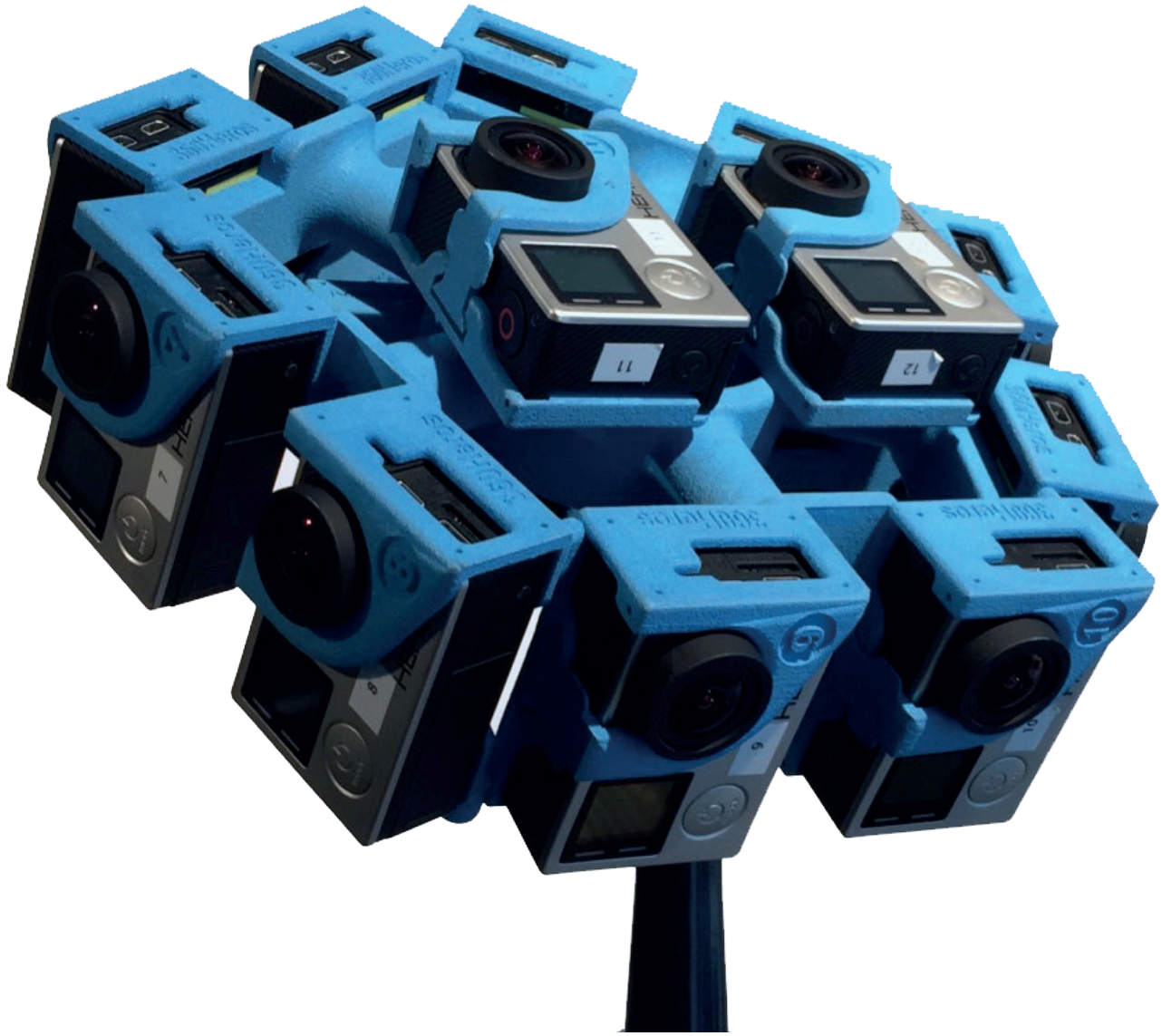
The Gas Sensor board also stopped working only shortly after it was installed and is a result of the delicate nature of the hardware itself. The Smart Cities board and Gas Sensor board are not very durable and need to be carefully connected to the controller and external battery. Furthermore, the Gas Sensor board was inadequate for all sensor measurements including the illuminance, RH and temperature. Since these were crucial measurements for the field studies, a HOBO U12 data logger was purchased and used as the primary measurement device for these variables. A more robust environmental sensor measurement equipment for air pollutants is also recommended for future applications.

3. Stereo-Spherical Video

Authors: Riccardo Joss, Brigitte Clements

This documentation provides information to record, edit and playback stereo-spherical videos with a GoPro rig of 14 cameras using Autopano Video and HTC Vive. It contains a list of all hardware and software equipment and provides additional know-how for troubleshooting.

The Autopano Video documentation (http://www.kolor.com/wiki-en/action/view/Autopano_Video_Documentation) is recommended https://wistia.com/blog/producing-360m-video. It offers a quick overview how to work with a similar camera setup.



3.1 Components and Assembly

This is the list of all hardware and software used for the ESUM Project.

3.1.1 EQUIPMENT USED FOR RECORDING

Hardware:

-14 x GoPro Hero 3+ Silver Edition

-14 x SanDisk 64 GB MicroSDXC Cards

-1 x 360Hero 3DH3Pro14H

-1 x GoPro Smart Remote

-1 x Clicker (usually used for dog training, here as acoustic syncing tool)

- optional 1 x Camera Flash (visual syncing tool)

-1 x Steadicam - Merlin 2, with arm and vest

-2 x powered 7-Port USB 3.0 Hubs

To charge the GoPros and download the movies from the camera we used two 7 Port USB 3.0 Hubs. Make sure to have enough electrical power to charge all cameras at once. Recommended is also a Laptop with SD Card reader (this is only needed if you want to check the files right after recording). Software: Kolor Autopano Video Pro 2.3 (if you want to use the wizard to set-up and assign the GoPros)

Image 1 : 360 Hero 3DH3Pro14H

Image 2 : GoPro SmartRemote

Image 3 : Camera Flash

Image 4 : Clicker

Image 5 : USB Hub

Image 6 : Steadicam Merlin Arm-Vest

3.1.2. EQUIPMENT USED FOR EDITING

The hardware required for editing is a computer suited for VR-Setup. For the HTC Vive the following was used:

(Check if you have the right number of USB and HDMI according to your VR-Setup and the recording setup)

Asus G11

Processor: Intel Core i7-6700

Memory: DDR4 32 GB

Graphics: NVIDIA GeForce GTX980 4GB

Storage: HDD 2TB and SSD 256 GB

Image 7 : Asus G11

Software:

-Kolor Autopano Video Pro 2.3 (main video editing software)

-Kolor Autopano Giga 4.2 (plugin in software to edit the stitching of panoramas)

-optional: Streamclip (free download by Squared 5, used to merge two videos into one)

Prior software (not used any more):

-360 CamMan

-Videostitch

-PTGui

3.1.3 EQUIPMENT USED FOR PLAYBACK

Hardware:

-Computer

-HTC Vive (or another video-goggle like Oculus Rift, Samsung Gear, Google Cardboard. For video playback head-tracking or a virtual cage is not needed. A high screen resolution is the most important feature)

Software:

-SteamVR (included in HTC Vive)

-Virtual Desktop (for Video playback on HTC Vive, not included)

-Kolor Eyes (for Oculus Rift)

Image 8 : HTC Vive

3.2 Operational Guide

It has to be stressed that it helps a lot to work precisely and in a well organized process. With 14 Cameras running simultaneously it can easily get very messy. If just one camera doesn't work they way you want, it will affect the whole recording. Since you don't have a visual display feedback, you will not notice any issue until much later in the process compared to usual camera setups. This means that if there is a failure, you will have to repeat the steps 14 times; therefore the simplest and safest way to avoid this is to stubbornly stick to a running order of each of the fourteen steps.

3.2.1 Preparation of the recording equipment

Set all cameras to the correct date and time. This will ensure the correct files go together. For the *GoPro Hero 3+ Silver Edition* set the settings for each camera to a resolution of „960", FPS (frames per second) to „60" and FOV (Field of View) to „Wide". The aspect ratio has to be 4:3.

Give each SD Card a number that corresponds to the camera number to avoid mixing up different SD Cards. Write the number as well on the SD Card with a waterproof pen. Keep the same camera in the same holder of the rig.

There is a GoPro import wizard which is an app that helps you to organize and manage the video files and SD Cards. This can be very handy if the number of used files is huge and if you frequently record things. For ESUM Zurich it was actually not used, therefore it's not covered in this documentation.

Make sure all 14 cameras and the GoPro Remote are fully charged and equipped with the correct SD Card (check numbers again if you are not sure). Switch on the WiFi Mode on all 14 cameras and connect them to the WiFi GoPro Remote. If this works you can switch all cameras on and off with the remote and of course also start and stop the recording. For further information there are numerous tutorials and vidoes on the internet. Simply check “GoPro” in Youtube.

3.2.2 Further preparation before recording

It is helpful to tape the remote to the blue camera rig, so you can always look at the display to make sure that all 14 cameras are connected. Simply use some sticky tape and take care to not cover any camera view.

To avoid noise remove coins and keys (etc.) from your pockets and wear shoes with soft soles. Depending on sunlight and if you want to hide your face a little bit, sunglasses might help you. Your face will be very visible in the video (even though strangely split in half because it is to close to the cameras to be stitched together).

People on the streets can react strangely because they are not familiar with this kind of recording gear. Have somebody behind you walking who can either help you or explain to confused or angry people what this gear is for. I would also recommend to write a sign on both your back and front stating that you are video recording. The reactions were more positive when people had an idea of what is going on.

3.2.3 Recording

The camera should be held as level as possible and with the front cameras (Number 9 and 10) pointing towards the walking direction. Grab the camera at the black tube which is directly attached to the blue GoPro rig. Do not use the grip below the ball joint as the camera would swing too much while walking. Walk slowly and gently. Reducing vibrations and swings is very important to both avoid the sensation of motion sickness of the viewing audience and as well as ensuring better stiching results. Practive walking and handle the setup before using it on real shots. Whilst recording make sure you can see the remote display showing the number of connected cameras.

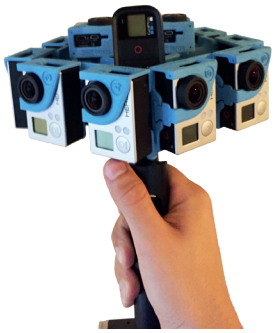


Image 9 : GoPro Rig with Remote

When shooting long shots it can be tiresome using the same hand, I recommend to practice switching hands a few times before the real shot. Directly after starting the recording use the clicker to make an audio cue for the later synchronization.

Unfortunately during recording I experienced unexpected stops of one or two cameras on a few separate occassions. Possible reasons could have been the WiFi connection with the remote, or that the charging cable or battery were not functioning properly. There was also an incident that the remote doesn't show all cameras as active, even though they actually are. This could also be attributed to the WiFi connetion. It helps to keep the remote rather close to the cameras or sometimes rotate the cameras until the remote finds a link to all 14 cameras. I recommend to check on a laptop right after shooting if all 14 SD Cards recorded a file of the same length or size. So you can check if all cameras where running as intended before going back to the editing computer.

3.3 EDITING

First of all have a look at the documentation of *Autopano Video Pro*. It is useful for almost any issue. They provide a full walkthrough of every step in the process in easily comprehensive videos on YouTube. It is more helpful than any written document. This document just explains additional knowledge for this camera setup and city street shootings.

3.3.1 Name your files

Download the video files from the SD Card onto your hard drive. Use a file name that makes sense to you, which includes the number of the camera. I recommend to put the camera number first, so you have a clean order in Autopano which will help you later in the process. It is important to make sure you keep the files very organised with corresponding details of each camera. This can ensure that complications are easy to resolve or understand later on in the latter steps.

3.3.2 Merge video sequences

The GoPro Hero 3+ Silver Edition splits a file in two parts when it becomes larger than 2GB. This has the result that a shot of more than approx. 20 minutes will be saved into 2 files (if longer, even more). Autopano has an option to merge the two videos before importing but it only works with the GoPro connected to the computer. Since transfer speed is much slower that reading out SD Card by SD Card with a card reader this option was not used. To merge two videos you have to use another software like Streamclip. Before you import the files to Autopano you have to make sure that you merge your sequence to one file per camera. Using Streamclip by Squared 5 works fine. It is a very simple software and for free. Even though it is from 2008 it still works fine on Windows 10. To merge the videos simply import all files of a sequence into Streamclip, make sure the running order is right and then „save as“ with a meaningful name. Repeat this procedure 14 times. Of course you can as well use any other video editing software (e.g. Adobe Premiere, Final Cut, etc.).

3.3.3 Import to Autopano Video Pro

Simply drag and drop the 14 files into Autopano Video Pro.

3.3.4 Autopano step 1: Synchronize Videos

Even if you used the GoPro remote to start the recording, the videos will be slightly out of sync (around 8-12 frames). Therefore you first need to synchronize all videos. If you used the clicker to make an audio-cue to synchronize the videos with the „use audio to synchronize“ button and set range on around 30 seconds.

Alternatively you can use motion to synchronize. Try what works the best, usually there is no perceivable difference. This step is very important and probably the most mysterious. During a longer shot it can occur that it looks sometimes well synced and other times you see wobbly images, either one or more cameras are running out of sync. Professional video producers (360 Concept and Wistia, for example) don't recommend working with stereo-spherical videos since the quality achieved is clearly not as good as with mono-spheres; Geometry as well as syncing can cause problems. The more cameras are in play, the more difficult it is to keep them in perfect sync. Static shots of course would be much more forgiving because none or just a few parts of the image show movements. When walking around, the whole content of the sphere is in motion and minor syncing inaccuracies will be visible.

The people at Wistia started with the exact same setup (14 GoPro 3+ Silver and the same rig) but later switched to a monoscopic setup with six GoPros due to lack of quality. They also didn't use the syncing of Autopano Video since they achieved better results by doing it manually in Adobe Premiere. When I tried this approach it was not possible to define a crisp common starting point since the resolution of time was too low. Some of the acoustic peaks were at the start of the 1/60 seconds frame, others in the middle some at the end. Therefore the starting point has a variance of 1/60 of a second. Acoustically this is very clearly audible. Instead of one clicking sound, five to ten can instead be distinguished and of course phasing effects occur as well. With a moving camera this is too much to achieve in continuously seamless stitches. Within 1/60 of a second one video of the sphere could moves back while it's neighbor video moves forward. At the stitch it will „collide" causing the stitching artifacts.

To improve stitching problems I see no other solution than a better camera setup. The cameras could be wired together with a pace keeper that triggers the frame rate for all cameras. Another solution is simply shooting with higher frame rates so syncing can be handled more accurately. The used *GoPro 3+ Silver* is limited to 60 FPS, the *GoPro 4 Black* can go up to 240 FPS at 720p. Or a bit lower in FPS but even higher image resolution. This should improve sync problems, since a variance of 1 frame at 60 FPS seems to be long when seen next to each other in the same video-sphere. A higher frame rate has a lower variance in time. Autopano recommends to use as high FPS as possible. But others warn of shooting with too high frame rates because of rolling shutter effects.

Using multiple cameras is obviously the main source of problem with synchronization, thus the cleanest solution is a camera that renders out one file. Giroptic has a such monoscopic camera on the market with three lenses and microphones for 500 Euros. For professional use *Nokia Ozo* has eight 2K x 2K lenses and eight microphones for spherical surround sound at a cost of 45'000 USD. Except for the announced Vuze camera there is no stereo-spherical one-in-a-box camera on the market.

3.3.5 Autopano step 2: Assign to stereo

Instead of going to the next steps (stitching, stabilization, etc.) switch to the „stereo" tab and check the box for stereo mode. The Autopano Video documentation recommends to assign cameras 1-10 to „horizontal split" and cameras 11-14 to „left and right". It might work better with other settings. This is something to you have to test. The logic of the rig would lead to set all odd numbers to „right" and all even numbers to „left". It works as well but might have more artifacts. Trying with cameras 1,3,5,7,9 assigned to „right", cameras 2,4,6,8,10 to „left" and cameras 11-14 to „left and right" I got the best results, but this might not be the same for other shots.

To find what works best, take a difficult sequence (with lots of artifacts) and try the different settings. It can be quite hard to compare since sometimes other artifacts appear.

Below the camera assignment you can choose what is rendered. To do the final render, choose „over / under". Whilst working you might switch to „left" or „right" to see better what is going on. It also helps a lot to switch the preview resolution from „512" to „4096" to see a sharp image, even though playback speed slows down almost to zero.

There is as well the possibility to put the stereo-layout „side-by-side" (or left and right) but I think this results in a lower horizontal resolution. Usually in video spheres the horizontal resolution is more important than the vertical since most of the content is rather about looking left and right than up and down. In general a stereo-spherical video has half the resolution of a mono-spherical video since the full resolution is split in half and blown up again for each eye. The finally perceivable resolution depends of course on the final rendering resolution and the display resolution of the head mounted display. The difference in resolution comparing

mono (4096x2048px) to stereo(2160x2160px) on a HTC Vive was not noticeable. The slightly grainy resolution of the HTC Vive (2160x1200px) is more dominant (Oculus Rift has the same resolution as HTC Vive).

Image 10 : Stereo Assignment

3.3.6 Autopano step 3: Stitch

Now you can go back to the stitch tab. Stitch as GoPro Hero 3+/4 and stitch at current selection. A number around 15 positions (for 20 minutes video) should be fine. Calculate the RMS curves to improve stitching. This will take a long time to compute (probably 2-5 hours for 20 minutes of video with 14 cameras, but it can even take up to 12 hours or more). The stitching can be edited with Autopano Giga in a lot of ways. It opens up if you click „Edit“ in the preview window. See the according tutorials for further information.

3.3.7Autopano step 4.1: Stabilization

Since the video is shot while walking, it will never be very smooth and free of vibrations. To maximize stability, choose the 100% compensation level. By increasing stability you might have a bit less of immersion but on the other hand, a more stable video which seems to cause less motion sickness. For ESUM Zurich the level was set to 100%. This calculation can take up to a half a day or more (for 20 minutes and 14 cameras). Unfortunately it regularly results in very twisted orientation which you have to fix afterwards manually. At least this is the case with city street shots, it might be much better for o<yy<utdoor shots with a visible flat horizon since it tries to detect automatically where the horizon is.

3.3.8 Autopano step 4.2: Orientation

As recommended in the Autopano documentation re-orient the video by adjusting at the beginning, then the end and then in the half of the video, further in the half of the half and so on. Depending on how twisted the orientation was, you can get easily more than 150 correction points. This can take a few hours of work. As a first step adjust the horizon. Then move the video to the left or right until the aluminum cap of the steadicam is right (top video) and left (bottom video) of the vertical axis. To see the aluminum cap you might change the camera assignment under the „stereo“ tab. For camera 13 choose „right“ and for camera 14 choose „left“ and set rendering to „over / under“. Don’t forget to switch back the assignment before rendering. If you press „shift“ on the keyboard it helps to move the video just lateral without messing up the fixed horizon. Remember some points and their position in the grid, so you know that the orientation will be correct.

The intersection of the main vertical and the main horizontal axis should point towards the walking direction on eye level. This makes sure that the default viewing direction is corresponding with the walking direction. Be specially careful where the walk leads around corners and curves, usually there are much more manual fixes needed than on straight paths. #Image: Adjust Orientation

3.3.9 Autopano step 5: Color

Depending on weather and light this step can take more or less time to work on. In general it is useful to compute an automatic color correction approximately every 5 seconds with all options ticked „on“. With Autopano Giga you can adjust difficult parts manually. Be careful when in the paths changes from dark to bright or vice versa occur (canopies, trees, arcades).

3.3.10 Autopano step 6: Blend

This part of the Autopano documentation is hard to understand, therefore I approached it by trial and error. The best results where achieved with custom settings choosing multiband on Level 10 and weighting on „Iso cutting“. #Image: Blend

3.3.11 Autopano step 7: Render

The following render settings are recommended: As audio-source any video can be chosen. In most of the cases the one of the front cameras makes sense, since the viewer is mostly oriented towards the walking direction, thus rendering the audio from camera 9 or 10. Render time with the mentioned computer for a video of 20 minutes is about 3 hours. It can take 15 or more seconds without any notice that the render will start. This can be quite confusing since it appears that nothing is working as expected. If you choose your file to render and then click on the green arrow shaped button it will start sooner or later. Wait for a couple of seconds, and then it will show the progress of the rendering process.

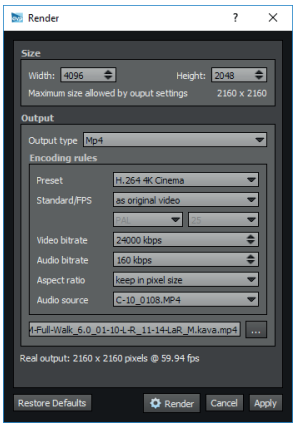


Image 11 : Render Settings

3.4 REFLECTIONS

Keep in mind that the technical development is very fast and this setup (September 2016) is by now already outdated in terms of price and quality. There are a lot of different camera setups out in the market (just google images for „VR camera“ or „spherical video camera“). Stereoscopic setups are less pushed and therefore more expensive and more difficult to work with. Nonetheless Vuze estimates to ship by October 2016 a stereo-spherical camera (without zenith and nadir cameras though) shooting in 4K including stitching software and a VR headset for a smartphone at a price of 800$. This is the first consumer grade product for stereo applications. Until now professionals are not recommending to work with a stereoscopic setup because they couldn’t achieve satisfying results (360concept.ch, wisitia.com).

Apart from a head mounted display like HTC Vive or Oculus Rift a swivel chair is an important piece of equipment. It keeps the person watching in a save position with minimal movement (specially in Z-axis) and maximal freedom to rotate

with the whole body to adjust the body position to the displayed content. For spherical videos (a thing like VR-videos actually don't exist yet, it's just a misleading name) the head should theoretically just rotate around the focal point and never move, since movement does not affect what the display shows and this causes a disconnection from our daily perception where head movement always changes your perspective slightly. Some people though are much more sensitive than others to such phenomenas.

3.4.1 Operation on different platforms

Surprisingly there was no free software available to watch 360 Videos on **HTC Vive**. But you can buy VirtualDesktop for 15 USD at the Steam VR store. There you can playback 360 photos and videos. Open the video from your hard drive or past the URL from YouTube and set the video mode to „over / under" and project it as sphere.

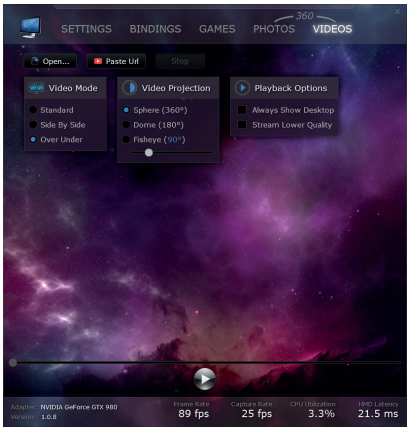


Image 12 : Virtual Desktop

For the **Oculus Rift** DK2 there is / was Kolor Eyes, a free software where you could drag and drop a 360 Video and watch it with a plug in Oculus Rift DK 2. Now they provide an updated software called GoPro VR Player which supports the Oculus Rift VR (3rd generation of Oculus).

3.4.2 RECORDING

Cameras switching off

Even with loaded batteries it occurred that one or two cameras switched off. Probably it was caused by a lost connection

to the remote. Or a close WiFi network could have an influence. This happened very seldom. While recording the remote shows on the display how many cameras are connected to it. So you see if there is a problem while recording.

Lost connection on remote

Sometimes one or more cameras lose connection to the remote and the display shows less than 14 cameras connected. This can be just a temporarily loss or they actually quit recording or even switched off. First keep on recording for maybe a minute and see if the lost cameras reconnect. If they don't you have to choose if you finish the shot with the possible risk of not having a complete set of recordings or you abort the recording session and start from the beginning.

3.4.3 VIDEO EDIT AND PLAYBACK

Wobbly buildings or spongy videos

Unfortunately sections of the sphere can wobble around. This is a typical effect when one of the videos is running out of sync. It just needs one of the 14 videos to be a bit late or early to cause this effect. Strangely the wobbly effect occurs and disappears sometimes in the middle of a sequence, it might be linked to the intensity of movement in the same sequence. I guess that the longer a sequence the more likely the cameras are not totally perfectly synced. A solution is to edit the long sequence and cut out bad parts after rendering.

Too dark and / or too bright

Recording under sunlight always has the effect that some cameras are facing the sun and others the opposite shadow side. If possible record with good light conditions at daytime with a rather cloudy sky, then you have the easiest and most evenly lit material to edit. The automatic color correction (exposure, color, signeting and gradient options marked) does a very good job but of course has it's limits in extreme situations.

Cut objects

Close object will always be cut due to geometrical reasons (parallax error). Cuts in more distant object can occur due to bad „syncing“ or a suboptimal setting in the blending mode.

Motion sickness

This problem is highly individual and it is hard to understand what really is the solution to this problem, if there is one to find. A high resolution head mount display, a swivel chair and as smooth as possible shots with good stitching reduce motion sickness. In Autopano Video I recommend to compute stabilization and set the level to 100%. This takes a long time to compute and most probably you'll have to manually re-orientate the whole shot but it smoothens the video substantially.

Wrong playback speed

If the playback speed seems to be wrong, check under „Render" / Standard/FPS and choose „as original video". Now the speed should be correct.

In the preview window the speed is as well affected by the chosen resolution. The higher the resolution the slower the preview playback will be. This does not affect the final rendered video. This parameter depends a lot on the computational power of the used graphic card.

Strange orientation

After computing stabilization the orientation is mostly or always messed up. Autopano Video tries to detect the horizon automatically. This might be good with outdoor shots in plain nature with a clear horizon but doesn't work well or not at all in inner city shots. Follow the instructional video of the Autopano documentation. It needs a bit of practice and patience to adjust the orientation but is always solvable.

Black and white artifacts

Probably due to high dynamic difference (direct sunlight and shadow) sometimes sections of the video sphere appears slightly black and white. If it's not to extreme it can be corrected in Autopano Giga.

3.4.4 SOFTWARE COMPARISON

At the beginning of the project Videostitch was used to stitch the videos, later Kolor Autopano Video was chosen to continue. The main reason for the change was the built in option for stereo video. In contrast the Autopano Video allows to produce stereo video as direct output. Videostitch just allows to render monoscopic videos. The idea with Videostitch was to render first the left eye and later the right eye and afterwards mount them over / under and render it again. This approach is much more work intensive and needs an additional professional video edit software to do the over / under mounting. And even with this, it would be quite tricky to make sure that the orientation of both videos is exact. Theoretically this approach could work but it is a very fragile process not recommendable for practice. Since Autopano Video has this process included, it works quite smooth to produce stereoscopic-videos with just one software.

There are some features in Videostitch (if just used for monoscopic videos) like the interactive preview and synchronizing with flash that are not offered in Autopano Video. But these benefits are minor compared to those of Autopano Video. Autopano Video allows much deeper control for stitching, stabilization, color correction and blending. They are also partners with GoPro and provide a smooth workflow from hardware to software like the import wizard (actually not used in this project) and GoPro templates for stitching. For monoscopic videos GoPro and Autopano released in August 2016 GoPro Omni, a full setup including all hard- and software needed. Videostitchs development seems to be substantially slower.

Appendix

I. Source Code Library

The applications and devices are programmed individually using a simple IDE (integrated development environment) for the WASPMOTE devices and the KIVY open source Python platform for the mobile phone application.

The source codes used for the ESUM project are included in this section.

1.0 Gas Board_.pde

Waspmote microcontroller code for Gases board (Waspmote IDE)

Source Code: ESUM_Waspmote_Gases/BackpackGasBoard.pde
Available at: https://github.com/cmiltiadis/ESUM_Waspmote_Gases/blob/master/BackpackGasBoard.pde#L1

This program was written by Constantinos Miltiadis for The Chair of Information Architecture, ETH Zurich
http://www.ia.arch.ethz.ch
It uses the Libelium Waspmote v2.0 and the GAS board.

Make sure that the sensors are attached on the right connectors, otherwise you could damage them.
For that consult the libelium PDF guide for this specific board:
http://www.libelium.com/development/waspmote/documentation/gases-board-technical-guide/

For the board to function you should connect the supplied 3.7v battery regardless if the board is connected with USB (a USB cannot provide enough current for this).

Sensors used:
Atmospheric Pressure
Temperature
Humidity
O3
Air pollutants 1 , Air pollutants 2 ,
CO2: S1A, NO2: S3B

```
//LIBS
#include <WaspSensorGas_v20.h>
//CONSTANTS
#define FRAME_DELAY_TIME 1000 //delay time between readings (milliseconds)
#define STABILIZATION_TIME 1000 // delay for the board to get measurements
(milliseconds)
//SENSORS
bool readTemp=true;
bool readPressure = true;
bool readHumidity=true;

bool readAccelerometer=false;
bool isAccelInitialized=false;

//CO2
bool readCO2=true;
//NH3
//bool readNH3=true;
bool readO3=true;
//NO2
bool readNO2 = true;
//Air pollutants
bool readAP1=true;
bool readAP2=true;


//FLAGS
bool isRTCon=false;
bool isBoardOn= false;


//SENSOR CONSTANTS
#define CO2_GAIN  7  //GAIN of the sensor stage
#define O3_GAIN  1      //GAIN of the sensor stage
#define O3_RESISTOR 20  //LOAD RESISTOR of the sensor stage
#define NO2_GAIN 1
#define NO2_RESISTOR 3
#define AP1_GAIN  1      // GAIN of the sensor stage
#define AP1_RESISTOR 20  // LOAD RESISTOR of the sensor stage
#define AP2_GAIN  1      // GAIN of the sensor stage
#define AP2_RESISTOR 20  // LOAD RESISTOR of the sensor stage
//
float initDelay; //CALCULATED INITIALIZATION DELAY

//parse token
char token='$';

void setup() {
  // put your setup code here, to run once:
  USB.ON();
  USB.println(F("ESUM_GAS_BOARD"));
  //battery
  getBatteryReading();
  //
  initDelay= calculateInitializationDelay();
  printSensorSetup();
  //
  RTC.ON();
  isRTCon=true;

  delay(1000);

  if (readPressure){
   initializePressure();
  }
  if (readO3){
    initializeO3();
  }
  if (readCO2){
    initializeCO2();
  }
  if (readNO2){
    initializeNO2();
  }
  if (readAP1){
    initializeAP1();
  }
  if (readAP2){
    initializeAP2();
  }
  //
  doInitializationDelay();
  //
  //
  if (readAccelerometer){
    ACC.ON();
    isAccelInitialized=true;
  }

  USB.println("STARTING");
}

void loop() {

  /*
   ************************************************************************
   *  GET SENSOR READINGS
   ************************************************************************
   */

  //switch board on
  switchBoardOn();

  //WAIT FOR THE SENSORS TO STABILIZE
  delay(STABILIZATION_TIME);

  //TEMPERATURE
  float tempVal=0;
  if (readTemp){
    tempVal =SensorGasv20.readValue(SENS_TEMPERATURE);
  }
  //PRESSURE
  float pressureVal=0;
  if(readPressure){
    pressureVal = SensorGasv20.readValue(SENS_PRESSURE);
    //set sensors off
    SensorGasv20.setSensorMode(SENS_OFF, SENS_PRESSURE);
  }
   //HUMIDITY
  float humidityVal=0;
  if (readHumidity){
    humidityVal = SensorGasv20.readValue(SENS_HUMIDITY);
  }

  //O3
  float o3Val_volt=0;
  float  o3Val_kohms;
  if (readO3){
    o3Val_volt = SensorGasv20.readValue(SENS_SOCKET2B);
    // Conversion from voltage into kiloohms
    o3Val_kohms = SensorGasv20.calculateResistance(SENS_SOCKET2B, o3Val_volt, O3_GAIN, O3_RE-
SISTOR);

  }
```

```
//CO2
float co2Val=0;
if (readCO2){
  // Read the sensor
  co2Val = SensorGasv20.readValue(SENS_CO2);
}
//NO2
float no2Val=0;
if (readNO2){
  no2Val = SensorGasv20.readValue(SENS_SOCKET3B);
}
//AP1
float ap1Val_volt=0;
float ap1Val_kohms=0;
if (readAP1){
  // Read the sensor
  ap1Val_volt = SensorGasv20.readValue(SENS_SOCKET4A);
  // Conversion from voltage into kiloohms
  ap1Val_kohms = SensorGasv20.calculateResistance(SENS_SOCKET4A, ap1Val_volt, AP1_GAIN, AP1_RESISTOR);
}
//AP2
float ap2Val_volt=0;
float ap2Val_kohms=0;
if (readAP2){ //@S3
  // Read the sensor
  ap2Val_volt = SensorGasv20.readValue(SENS_SOCKET3A);
  ap2Val_kohms = SensorGasv20.calculateResistance(SENS_SOCKET3A, ap2Val_volt, AP2_GAIN, AP2_RESISTOR);

}
//Commented off to have less frame calculations
//switch off board
//switchBoardOff();
//

/*
 ********************************************************************************
 *   OUTPUT READINGS
 ********************************************************************************
 */

  //SYNCH TOKEN DO NOT REMOVE / gives a sign that this is the start of the frame
  USB.print(token);

//temperature
if (readTemp){

  USB.print(F("Temprature: "));
  USB.print(tempVal);
  USB.println(F(" Celcius"));
}
//pressure
if (readPressure){
  USB.print(F("Pressure: "));
  USB.print(pressureVal);
  USB.println(F(" kPa"));
}

if (readHumidity){
  USB.print(F("Humidity: "));
  USB.print(humidityVal);
  USB.println(F("%RH"));
}

if (readAccelerometer && isAccelInitialized){
  getAccelValue();
}
  if (readO3){
    // Print the result through the USB
    USB.print(F("O3 @S2B: "));
    USB.print(o3Val_volt);
    USB.print(F("V - "));


    USB.print(o3Val_kohms);
    USB.println(" kohms");
  }

  if (readCO2){
    // Print the result through the USB
    USB.print(F("CO2: "));
    USB.print(co2Val);
    USB.println(F("V"));
  }

  if (readNO2){
    // Print the result through the USB
    USB.print(F("NO2 @S3B: "));
    USB.print(no2Val);
    USB.print(F("V - "));

    // Conversion from voltage into kiloohms
    float no2Val_kohms = SensorGasv20.calculateRe-
sistance(SENS_SOCKET3B, no2Val, NO2_GAIN, NO2_RE-
SISTOR);
    USB.print(no2Val_kohms);
    USB.println("kohms");
  }
  if (readAP1){
    // Print the result through the USB
    USB.print(F("AP1 @ S4: "));
    USB.print(ap1Val_volt);
    USB.print(F("V - "));

    USB.print(ap1Val_kohms);
    USB.println(F("kohms"));
  }
  if (readAP2){
    // Print the result through the USB
    USB.print(F("AP2 @S3A: "));
    USB.print(ap2Val_volt);
    USB.print(F("V - "));

    USB.print(ap2Val_kohms);
    USB.println(F("kohms"));
  }
  //
  delay(FRAME_DELAY_TIME);
}
```

1.1 Gas Board_general

Source Code: ESUM_Waspmote_Gases/General.wasp
Available at: https://raw.githubusercontent.com/cmiltiadis/ESUM_Waspmote_Gases/master/General.wasp

```
//outputs battery
void getBatteryReading(){
  // Show the remaining battery level
  USB.print(F("Battery Level: "));
  USB.print(PWR.getBatteryLevel(),DEC);
  USB.print(F(" %"));
  // Show the battery Volts
  USB.print(F(" | Battery (Volts): "));
  USB.print(PWR.getBatteryVolts());
  USB.println(F(" V"));
}
//prints On for true Off for false
void printBool(boolean val){
  if (val) USB.println("ON");
  else USB.println("OFF");
}

//outputs the settings for the readings
void printSensorSetup(){
  USB.println(F("SENSOR SETUP"));

  USB.print(F("Temperature:"));
  printBool(readTemp);
  //
  USB.print(F("Pressure: "));
  printBool(readPressure);
  //
  USB.print(F("Humidity:"));
  printBool(readHumidity);
  //
  USB.print(F("CO2:"));
  printBool(readCO2);
  //
  USB.print(F("Accelerometer: "));
  printBool(readAccelerometer);
  //
  USB.print(F("O3: "));
  printBool(readO3);
  //
  USB.print(F("NO2: "));
  printBool(readNO2);
  //
  USB.print(F("AP1: (C4H10, CH3CH2OH, H2, CO, CH4) : "));
  printBool(readAP1);
  //
  USB.print(F("AP2: (C6H5CH3, H2S, CH3CH2OH, NH3, H2) :"));
  printBool(readAP2);
  //output rate
  USB.print(F("Output rate:"));
  USB.print(FRAME_DELAY_TIME);
  USB.println(F(" Seconds"));
  //Init time
  USB.print(F("Initialization time: "));
  USB.print((int)initDelay);
  USB.println(F(" Seconds"));

}
//calcualte the time needed for initialization of the sensors
float calculateInitializationDelay(){
  float result =1;
  if (readCO2 || readNO2 || readO3 || readAP1||readAP2){
    result+=40;
  }
  return result;
}

//performs the initialization delay with output every second
void doInitializationDelay(){
  USB.println(F("Wait for intialization"));
  for (int i=0;i<initDelay;i++){
    delay(1000);
    USB.print(F("."));
  }
  USB.println(F(""));
  USB.println(F("Initialization done"));
}

//ACCELEROMETER READING
void getAccelValue(){
  byte check = ACC.check();

  //----------X Value----------
  int x_acc = ACC.getX();

  //----------Y Value----------
  int y_acc = ACC.getY();
  //----------Z Value----------
  int z_acc = ACC.getZ();

  //--------------------------------
  USB.print(F("ACC: "));
  USB.print(x_acc, DEC);
  USB.print(F(" / "));
  USB.print(y_acc, DEC);
  USB.print(F(" / "));
  USB.println(z_acc, DEC);
}

// Turn on the sensor board
void switchBoardOn(){
  if (isBoardOn==false){
    SensorGasv20.ON();
    isBoardOn= true;
  }
}

void switchBoardOff(){
  if (isBoardOn==true){
    SensorGasv20.OFF();
    isBoardOn= false;
  }
}
```

1.2 Gas Board_sensor

Source Code: ESUM_Waspmote_Gases/SensorInit.wasp
Available at: https://raw.githubusercontent.com/cmiltiadis/ESUM_Waspmote_Gases/master/SensorInit.wasp

```
void initializeO3(){

  switchBoardOn();

  // Configure the 2B sensor socket
  SensorGasv20.configureSensor(SENS_SOCKET2B, O3_GAIN, O3_RESISTOR);

//  // Turn on the RTC
//  RTC.ON();

  // Turn on the sensor on socket 2B and wait for stabilization and
  // sensor response time
  SensorGasv20.setSensorMode(SENS_ON, SENS_SOCKET2B);

}

void initializeNO2(){
 switchBoardOn();

//  // Turn on the RTC
//  RTC.ON();

  // Configure the 3B sensor socket
  SensorGasv20.configureSensor(SENS_SOCKET3B, NO2_GAIN, NO2_RESISTOR);


  // Turn on the sensor on socket 3B and wait for stabilization and
  // sensor response time
  SensorGasv20.setSensorMode(SENS_ON, SENS_SOCKET3B);
}

void initializeCO2(){
//  // Turn on the sensor board
switchBoardOn();

//  // Turn on the RTC
//  RTC.ON();

  // Configure the CO2 sensor socket
  SensorGasv20.configureSensor(SENS_CO2, CO2_GAIN);

  // Turn on the CO2 sensor and wait for stabilization and
  // sensor response time
  SensorGasv20.setSensorMode(SENS_ON, SENS_CO2);
  //delay(40000);
}

void initializeAP1(){ // at Connector S4

//  // Turn on the RTC
//  RTC.ON();

  // Configure the 4A sensor socket
  SensorGasv20.configureSensor(SENS_SOCKET4A, AP1_GAIN, AP1_RESISTOR);
  // Turn on the sensor on socket 4A and wait for stabilization and
  // sensor response time
  SensorGasv20.setSensorMode(SENS_ON, SENS_SOCKET4A);
}

void initializeAP2(){

//  // Turn on the RTC
//  RTC.ON();

  // Configure the 3A sensor socket
  SensorGasv20.configureSensor(SENS_SOCKET3A, AP2_GAIN,  AP2_RESISTOR);


  // Turn on the sensor on socket 3A and wait for stabilization and
  // sensor response time
  SensorGasv20.setSensorMode(SENS_ON, SENS_SOCKET3A);
}

void initializePressure(){
   if(readPressure){
    SensorGasv20.setSensorMode(SENS_ON, SENS_PRESSURE);
  }
}
```

2.0 SmartCities_.pde

Waspmote microcontroller code for Smart Cities board (Waspmote IDE)

Source Code: ESUM_Waspmote_Cities
Available at: https://github.com/cmiltiadis/ESUM_Waspmote_Cities/blob/master/General.wasp#L1

This program was written by Constantinos Miltiadis for The Chair of Information Architecture, ETH Zurich
http://www.ia.arch.ethz.ch
It uses the Libelium Waspmote v2.0 and the Smart Cities board.

Make sure that the sensors are attached on the right connectors, otherwise you could damage them.
For that, consult the libelium PDF guide for this specific board:
http://www.libelium.com/development/waspmote/documentation/smart-cities-board-technical-guide/

For the board to function you should connect the supplied 3.7v battery regardless if the board is connected with USB (a USB cannot provide enough current for this).

Sensors used:
Luminosity sensor
Sound sensor (Comes preconnected to board)
Dust sensor: Use the special bus connector to attach to the board

```
// Put your libraries here (#include ...)
#include <WaspSensorCities.h>

// CONSTANTS
#define FRAME_DELAY_TIME 1000 //delay time between readings (milliseconds)
//SENSORS
bool readAudio= true;
bool readLuminosity = true;
bool readDust= true;

//parse token
char frameToken ='$';



void setup() {
  // Turn on the USB and print a start message
  USB.ON();
  USB.println(F("ESUM_Smart_Cities_Board"));

  getBatteryReading();
  delay(100);

  // Turn on the sensor board
  SensorCities.ON();
  // Turn on the RTC
  RTC.ON();

  //print sensor setup
  printSensorSetup();

  /*
   * Initialize sensors from the beginning
   * Otherwise the have to be switched on and off on each frame
   * which makes everything slower
   */

  if (readLuminosity){
    // Turn on the sensor and wait for stabilization and response time
    SensorCities.setSensorMode(SENS_ON, SENS_CITIES_LDR);
  }
  if (readAudio){
    SensorCities.setSensorMode(SENS_ON, SENS_CITIES_AUDIO);
  }
  if (readDust){
     SensorCities.setSensorMode(SENS_ON, SENS_CITIES_DUST);
  }
}

void loop() {

  /*
   *  GET SENSOR READINGS
   */

  float luminosityVal=0;
  if(readLuminosity){
    // Read the LDR sensor
    luminosityVal = SensorCities.readValue(SENS_CITIES_LDR);

  }

  float audioVal=0;
  if (readAudio){
   // Read the audio sensor
    audioVal = SensorCities.readValue(SENS_CITIES_AUDIO);
  }

  float dustVal =0;
  if (readDust){
    // Read the Dust sensor
  dustVal = SensorCities.readValue(SENS_CITIES_DUST);
  }

  /*
   * OUTPUT READINGS
   */
  USB.print(frameToken);  //the token is used to get the first value of each frame

  if (readAudio){
    USB.print(F("Sound pressure: "));
    USB.print(audioVal);
    USB.println(F(" dBA"));
  }
  if (readLuminosity){
    //USB.print(token);
    USB.print(F("Luminosity: "));
    USB.print(luminosityVal);
    USB.println(F(" %"));
  }

  if (readDust){
     USB.print(F("Dust: "));
    USB.print(dustVal);
    USB.println(F(" mg/m3"));
  }

  //do a delay for the sensors to power down
  delay(FRAME_DELAY_TIME);
}
```

2.1 SmartCities_general

Source Code: ESUM_Waspmote_Cities/General.wasp
Available at: https://raw.githubusercontent.com/cmiltiadis/ESUM_Waspmote_Gases/master/SensorInit.wasp

---

```
//outputs battery
void getBatteryReading(){
  // Show the remaining battery level
  USB.print(F("Battery Level: "));
  USB.print(PWR.getBatteryLevel(),DEC);
  USB.print(F(" %"));
  // Show the battery Volts
  USB.print(F(" | Battery (Volts): "));
  USB.print(PWR.getBatteryVolts());
  USB.println(F(" V"));
}


//prints On for true Off for false
void printBool(boolean val){
  if (val) USB.println("ON");
  else USB.println("OFF");
}

//outputs the settings for the readings
void printSensorSetup(){
  USB.println(F("SENSOR SETUP"));

  USB.print(F("Sound pressure:"));
  printBool(readAudio);

  USB.print(F("Luminosity:"));
  printBool(readLuminosity);

   USB.print(F("Dust Particles:"));
  printBool(readDust);
}
```

3.0 ESUM_logger

Visual Studio C# program for sensor data logging

Source Code: ESUM_Threaded_Logger.csproj
Available at: https://github.com/cmiltiadis/ESUM_Logger

This program was written by Constantinos Miltiadis for The Chair of Information Architecture, ETH Zurich
http://www.ia.arch.ethz.ch
It uses the Libelium Waspmote v2.0 and the GAS board.

Visual Studio C# program for sensor data logging

Developed by Constantinos Miltiadis, for the Chair for Information Architecture ETH Zurich info at c.miltiadis [at] gmail.com

Chair of iA http://www.ia.arch.ethz.ch

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="12.0" DefaultTargets="Build" xmlns="http://schemas.micro-
soft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Com-
mon.props" Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\
Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{9A95EAC3-BB62-444E-92EE-5EFF978421F3}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>ESUM_Logger</RootNamespace>
    <AssemblyName>MultiThreaded_Serial</AssemblyName>
    <TargetFrameworkVersion>v4.5</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Data" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Device.cs" />
    <Compile Include="Program.cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
    <Compile Include="Util.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="App.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and
uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```