EMF camp 2018 Air Quality LoRaWAN workshop

At EMF camp 2018 we ran three workshops in order to improve people's understanding of air quality. The workshop details can be found at https://www.emfcamp.org/line-up/2018/180-air-quality-sensor-devices-build-program-and-test. Here are the details needed to recreate the workshop. The workshop was designed to be suitable for anyone with no prior experience using a Raspberry Pi, however, the setup steps that we performed in advance of the workshop do require more advanced knowledge.

Air quality and low-cost sensors

Some of the slides presented at the beginning of the workshop were infographics taken from the World Health Organization (WHO). They can be found along with other general figures about air quality in the world at http://www.who.int/airpollution/infographics/en/. If you want to learn more about air quality and especially about air quality in UK, we suggest you to read the report published by the Royal College of Physicians and the Royal College of Paediatrics and Child Health in 2016: Every breath we take: the lifelong impact of air pollution.
The sensors used during the workshop are low-cost sensors used to monitor fine particles (PM_{10} and $PM_{2.5}$). We strongly encourage you to read the advice from the Department for Environment Food & Rural Affairs (DEFRA) on the use of low-cost sensors for air pollution to have a better understanding of their limitations. The recommendation can be found at https://uk-air.defra.gov.uk/library/aqeg/pollution-sensors.php

Hardware

The links below are the suppliers we used to purchase the parts – other suppliers are available.

- Raspberry Pi (3B or 3B+) https://cpc.farnell.com/raspberry-pi/2773729/raspberry-pi-3-model-b/dp/SC14882
- SD card
- Plantower Particulate Matter Sensor https://www.ebay.co.uk/itm/PMS5003-G5-PlanTower-High-Precision-Laser-Dust-Sensor-Module-PM1-0-PM2-5-PM10/201903779518
- FTDI USB-serial http://www.hobbytronics.co.uk/ft232rl-breakout-2
- Dragino LoRaWAN HAT https://cpc.farnell.com/seeed-studio/113990254/gps-lora-hat-for-raspberry-pi/dp/SC14509?st=lora
- PiFace RTC (not used in workshop code but in the kits we provided) https://cpc.farnell.com/piface/shim-rtc/real-time-clock-for-raspberry/dp/SC13528
- Battery for RTC https://cpc.farnell.com/gp-batteries/gpca10/battery-lithium-3v-cr1220-1pk/dp/BT00449
- USB Power bank https://cpc.farnell.com/iconbit/ftb7800fx/powerbank-dual-usb-charger-7800mah/dp/BT06432
- 2 * USB cable https://uk.farnell.com/roline/11-02-8310/computer-cable-usb2-0-150mm-black/dp/2444222
- Velcro https://www.rapidonline.com/velcro-brand-vel-ec60236-stick-on-squares-25mm-x-24-sets-black-54-4297

We used heat shrink to attach the USB-serial adapter to the sensor as a single module. The heat shrink is not available in small quantities so a Velcro pad can be used instead to both secure the USB-serial adapter and to prevent accidental shorting of pins. The equipment was mounted on a custom laser cut perspex sheet, however, feel free to create your own mounting solution.

Software

We have written a couple of custom libraries to facilitate this workshop (which will be installed in the setup steps). If you want to dig deeper into the code it can be found on github. If you make any improvements to the code please submit a pull request.

- Draingo LoRaWAN HAT https://github.com/computenodes/dragino
- Cayenne LPP https://github.com/FEEprojects/cayennelpp-python
- Plantower interface https://github.com/FEEprojects/plantower

Setup

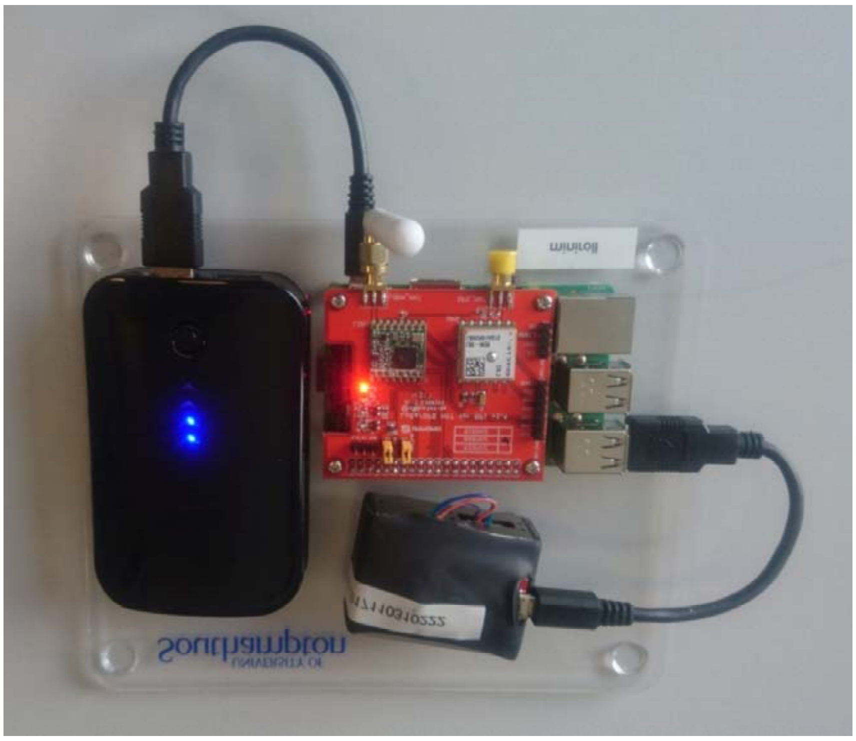
Having purchased / acquired the hardware the following steps have to be carried out to get to the stage where the workshop code can be run. In order for the LoRaWAN part to work you have to be in coverage of The Things Network. For the workshop we deployed 3 gateways on site. Coverage can be checked by using TTN Mapper. If you are not in coverage you can extend the network by running your own gateway. See https://www.thethingsnetwork.org/docs/gateways/ for details.

Hardware setup

In order to assemble the hardware do the following steps:

1. Insert the battery into the RTC module.
2. Attach the stand offs provided in the dragino LoRaWAN HAT box to the Rapsberry Pi.
3. Place the RTC module on to the Pi header connections.
4. Place the dragino LoRaWAN HAT on to the Pi and secure with the screws.
5. Attach the antenna to the LORA_ANT connection on the dragino HAT.
6. Solder the PM5003 to the USB-Serial, the important connections are (Pin 5) TX -> RX, (Pin 4)RX -> TX, (Pin 1)VCC -> 5V and (Pin 2)GND -> GND.
7. Make sure the switch on the USB-Serial is set to 3V3.
8. Connect the USB-serial to the Raspberry Pi using one of the micro-USB leads

Our finished nodes looked like


An example node as used in the workshop.

Software installation

Copy the latest Raspbian image on to the SD card, and then connect to the Pi and run the following script.
The code below will install the required libraries and software to run the workshop. This will take some time.

```
echo "dtparam=spi=on" | sudo tee -a /boot/config.txt
echo "dtoverlay=spi-gpio-cs" | sudo tee -a /boot/config.txt
echo "dtparam=i2c_arm=on" | sudo tee -a /boot/config.txt
sudo raspi-config noint do_serial 2

#get the spi CS pin overlay
wget https://github.com/computenodes/dragino/releases/download/v0.0.1/spi-gpio-cs.dtbo
sudo mv spi-gpio-cs.dtbo /boot/overlays/

#install jupyter
sudo apt update
sudo apt-get install -y python-dev
sudo -H pip install --upgrade pip
sudo -H pip install jupyter
sudo apt-get install -y python-seaborn python-pandas python3-pandas
sudo apt-get install -y ttf-bitstream-vera

sudo pip3 install jupyter
sudo ipython3 kernelspec install-self
```

```
jupyter notebook --generate-config
jupyter notebook password # this will prompt you for a password
#add jupyter to crontab
echo "@reboot pi /usr/local/bin/jupyter notebook --ip=0.0.0.0 --no-browser --notebook-dir=/home/pi &gt;&gt;
/tmp/jupyter.out 2&gt;&gt; /tmp/jupyter.err" | sudo tee -a /etc/cron.d/jupyter
#setup the RTC
echo "i2c-bcm2708" | sudo tee -a /etc/modules
sudo modprobe i2c:mcp7941x
echo "@reboot root echo mcp7941x 0x6f &gt; /sys/class/i2c-adapter/i2c-1/new_device; ( sleep 2; hwclock -s ) &amp;" |
sudo tee -a /etc/cron.d/rtc
#get the cayenne module
wget https://github.com/FEEprojects/cayennelpp-python/releases/download/v1.0.0/simplecayennelpp-1.0.0.tar.gz
#install it
pip3 install simplecayennelpp-1.0.0.tar.gz
#get the plantower module
wget https://github.com/FEEprojects/plantower/releases/download/v0.0.2/plantower-0.0.2.tar.gz
pip3 install plantower-0.0.2.tar.gz
#get the dragino lib and install
wget https://github.com/computenodes/dragino/releases/download/v0.0.2/dragino-0.0.2.tar.gz
pip3 install dragino-0.0.2.tar.gz
sudo reboot
```

Once the node has rebooted make sure it has the correct time (If it’s connected to a network it will do this automatically). Then run:

```
sudo hwclock –w
```

This sets the time on the hardware RTC so the Pi knows the time when it boots even without GPS or network connections.

The Things Network Setup

We use The Things Network to transfer the data from the device to Cayenne for viewing. This also requires setting up which can be done with the following steps. For more information please read The Things Network documentation.

1. Register at The Things Network https://www.thethingsnetwork.org/.
2. Create a new application
3. On the integrations tab for the application add `Cayenne’.
4. Register a device in the application. Make sure that it generates the `Device EUI’ and `App Key’.
5. Edit the setting for the device to change `Activation Method’ to `ABP’. (You can use OTAA but for simplicity of the workshop we used ABP).
6. Download an example config file for the dragino from https://raw.githubusercontent.com/computenodes/dragino/master/dragino.ini.default save it to `/home/pi/dragino.ini’
7. Fill in the devadddr, nwskey, and appskey from The Things Network

Having done the above steps your config file should look similar but not identical to the following (The devaddr, nwskey and appskey lines will be different:

```
#GPS configuration
gps_baud_rate = 9600
gps_serial_port = "/dev/serial0"
gps_serial_timeout = 1
gps_wait_period = 10

#LoRaWAN configuration
spreading_factor = 7
max_power = "0x0F"
output_power = "0x0E"
sync_word = 0x34
rx_crc = True
#Where to store the frame count
fcount_filename = "/home/pi/.lora_fcount"

##Valid auth modes are ABP or OTAA
##All values are hex arrays eg devaddr = 0x01, 0x02, 0x03, 0x04
auth_mode = "abp"
devaddr = 0x26, 0x01, 0x19, 0xA9
nwskey = 0x4A, 0xE5, 0x3B, 0x8E, 0x1B, 0x4C, 0x24, 0x7C, 0x07, 0xA3, 0x52, 0x38, 0x3A, 0x28, 0x1F, 0xCD
appskey = 0x02, 0x1D, 0x25, 0xF1, 0x92, 0xCE, 0x2B, 0xA2, 0x89, 0x8F, 0x29, 0x18, 0x43, 0xA9, 0x00, 0x1E

#auth_mode = "otaa"
#deveui =
#appeui =
#appkey =
```

Cayenne Setup

In order to visualise the data you will also need an account on Cayenne https://cayenne.mydevices.com/cayenne/dashboard/start. Having created the account you need to add the device. To do this follow these steps:

1. Click Add New… Device/Widget
2. Select `LoRa from the list
3. Select `The Things Network' from the expanded list
4. A new list now appears on the right, scroll down and select `Cayenne LPP'
5. Copy the DevEUI from the things network and change the Name if you wish
6. Click `Add device'.

Workshop Code

Finally we are at the point where we can write the code that was used in the workshop. Connect to the Jupyter notebook, the exact address for this will depend on your network but it may be available at http://raspberrypi:8888.
The code created is as follows:

```
from dragino import Dragino # import the module required for GPS and LoRaWAN
from simplecayennelpp import CayenneLPP # import the module required to pack the format cayenne expects
from plantower import Plantower # import the module to speak to the PM sensor
from time import sleep # import sleep so can wait between polling the sensor
sensor = Plantower()
dataList = []
for i in range(10):
    data = sensor.read()
    dataList.append(data.gr03um)
    sleep(1)
average_reading = sum(dataList) / len(dataList)

D = Dragino("/home/pi/dragino.ini") # Set up the dragino HAT
my_position = D.get_gps()
lpp = CayenneLPP()
lpp.addAnalogInput(1, round(average_reading))
lapp.addGPS(2, my_position.latitude, my_position.longitude, myposition.altitude)
D.send_bytes(list(lpp.getBuffer()))
```