ArduECO: cheap air quality control for cities

Voinea Stefan Ciprian
University of Padova, Italy
Department of Pure and Applied Mathematics
stefanciprian.voinea@studenti.unipd.it

***Abstract*—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.**

***Index Terms*—Arduino, embedded, air-quality**

I. Introduction

More and more people around the world have started to understand the importance of air quality and how much having clean and pollution-free air can influence our lives, both on the small and large scale. To have clean air, all of us have to do something to avoid polluting it.
The Government of Luxembourg has set a good example becoming the first country to offer nationwide free public transport to the citizens [1].
For short commutes, vehicles like the classic bike [2], the infamous skateboard and the scooter have started to arise and conquer cities, either in their battery-powered or old-school human-powered.
Since not everyone necessarily owns one of these green methods of transportation, companies like *Mobike* [3] and *MiMoto* [4] have seen the opportunity to enter the market of shared transportation, proposing a *pay-per-use* solution for bikes, electric scooters and other similar vehicles. Each company has its own application, network infrastructure and smart devices aboard their vehicles, gathering data like GPS (Global Positioning System) position, time spent by the user, parking spot, etc. to send it in the cloud and compute information like cost of the ride and charging it to the user.
All this falls under the *IoT* (Internet of Things) paradigm, which has become a well-described market with new ideas and business opportunities being presented every day.
Among the data collected by these companies, none is about air pollution.
In this article, I describe *ArduECO*, a wireless device based on an Arduino-like board capable of gathering data from previously named vehicles and sending it to the cloud, to be processed and displayed.
This paper is organized as follows:

- Section 2: Background and description of the problem
- Section 3: State of the art on smart devices that allow tracking pollution
- Section 4: implementation of the proposed solution
- Section 5: Conclusions

II. Background problem

Background
spiegare quali sono le particelle di inquinamento nell'aria
spiegare quali sono i sensori presenti sul mercato che possono rilevarle
tabella con sensori mq e differenze
come mai la scelta di implementarlo in quel modo

III. State of the art

State of the art non solamente della letteratura ma anche di quello che viene offerto sul mercato

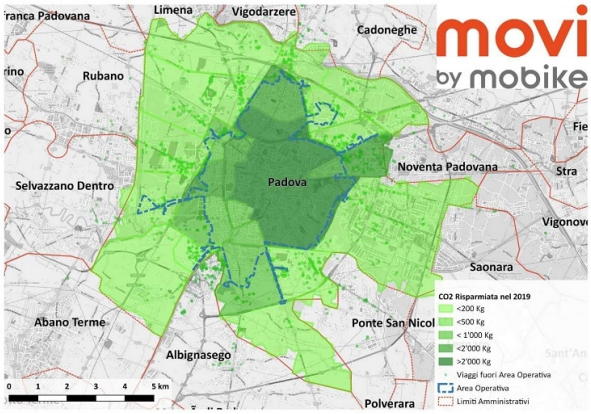

Fig. 1. Example of a figure caption. [**?**]

IV. Proposed solution

ArduECO is a wireless IoT device that represents one of the possible combinations of open source hardware and software. This wireless device is able to detect substances in the air and send the data in the cloud over Wi-Fi, and combined with data from other similar devices detect the level of pollution in the air.
The schematics for this project have been designed using Fritzing, a CAD (Computer-Aided Design) software for the

design of electronics hardware [5] and can be seen in Fig. 2. Both code and schematics for this device are available on the GitHub repository "ArduECO" [6].
Schematics for other components can be easily found online from various sources and vendors: here, for example are the schematics for the MQ7 sensor from SparkFun [7] and the NodeMCU [8].

A. The circuit

ArduECO is composed by four main components:

- *NodeMCU*: it's the hearts and brains of the device, this board is an open-source development kit based on the ESP8266 chip that allows for prototyping of IoT devices;
- *MicroSD card reader*: this allow for collecting and keeping data cached for the current ride and permanently on a removable memory card;
- *GPS sensor*: this allows for localizing the device, it has an onboard LED that blinks when it has established communication with the satellites;
- *MQ sensor*: the MQ sensor used in this project is MQ7 but it can easily be exchanged with other sensors that use the same board and connectors.

The other components, LEDs and button, are used as I/O for interacting with the user.
In VI I explain how the project can be improved both in hardware and software. In Fig. 2 there is no indication of

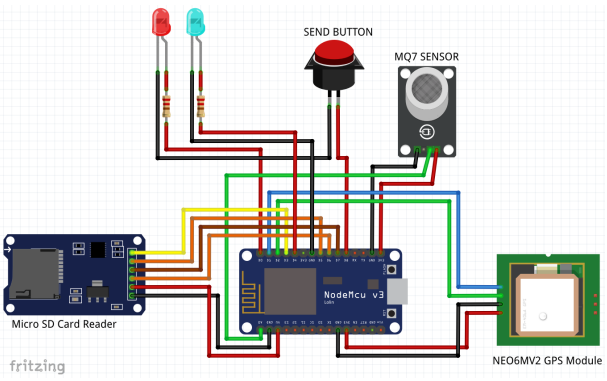

Fig. 2. Example of a figure caption.

a power supply since for this prototype version it has been powered using an external battery with a standard output of 5V and 1A.

B. The Arduino software

The reasons behind the choice of using Arduino's programming language instead of micropython [**?**] because of the large community behind and interoperability with other boards. The main functions in the Arduino programming language are "*void setup()[]*" and "*void loop()[]*". The first one runs once at the boot of the device to initialize and set the initial values while the second one does precisely what its name suggests, and loops consecutively, allowing the program to change and respond. The main file that contains these functions and that board executes is ArduECO.ino. In here the setup function contains the instructions in order to correctly initialize the serial communication with the GPS module, create the files in the sd card, initialize the pinouts, gathering the certificates for communicating with its cloud component and creating a random id for identifying the ride.

An important part of the configuration of the device is the file parametets.json

```
{
        "ssid" : "",
        "password" : "",
        "in_topic" : "",
        "out_topic" : "",
        "endpoint" : ""
}
```

If this file is not present in the sd card at boot-time the device will not end the boot sequence returning an error and rebooting after 10 seconds. Here we set the network ssid and password it is supposed to connect to, the topics for communicating with the mqtt server and its endpoint. In the loop function the board acquires a reading of its surrundings every 5 seconds by first detecting if the GPS has a fix (which can be seen on the board when the led blinks blue) and by reading the MQ sensor. Each read is then appended on two separate files in json format: cache_log.txt and perm_log.txt. The firts one is created each time the board boots and it regards each single session while the second one contains the information of all the rides.

```
{
        "id":"127",
        "air":"18.22",
        "lt":"45.39641",
        "lg":"11.88527",
        "dt":"50320.12173900",
        "tpc":"arduecо_proto_01_OUT"
}
```

The two leds that are attached to the board are used not only to communicate when the board has finished the setup sequence or in case of errors but for indicating wheter the ssid specified in the params file is in range and whether the board is transmitting data or not. When the button is pressed the board searches the list of networks in range for the specified one and, if present, it connects to it and establishes communication with the MQTT server specified by sending it a message with the number of reads that will be sent. Each row of the cache file is then sent to the server and at the end of this sequence the cache file is deleted and created again awaiting new reads. The other files that implement this project contain specific functions for either setting up the device or connecting the the server. All of them can be found in the github repo associated with this project.

C. The cloud

The cloud part of this project is divided between AWS (Amazon Web Services) and Altervista. From AWS the services that are used are IoT and lambda which are easy

to set up and are highly interconnectable with multiple architectures. At first when the button on the ArduECO is pressed the message is sent to an MQTT server in AWS IoT. To connect to this server the devices need to use certificates downloaded and buned into the devices memory when the code is sent in it using the arduino IDE. The MQTT (Message Queuing Telemetry Transport) server designed as a lightweight messaging protocol that uses publish/subscribe operations to exchange data between clients and the server So, its easy to implement in software and fast in data transmission (https://1sheeld.com/mqtt-protocol/) The MQTT server in the AWS IoT console is then connected to a serverless lambda function, triggered each time a new message arrives from the ardueco. This python function receives in input the message and sends it to a php file on the webserver that inserts it in the database, hosted on the free website hosting altervista.org under the url ardueco.altervista.org.

The webpage of the website displays the routes that are stored and each marker indicates a reading.

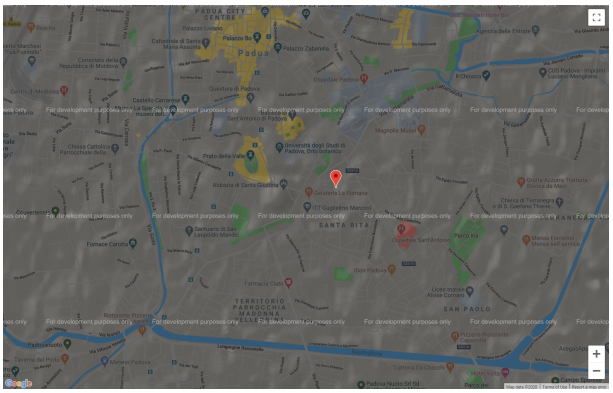

Fig. 3. Example of a figure caption.

In the improbements section i'll explain some modifications that could improve the system

D. Hardware cost

Because most of these parts are open source they can be found sold on various websites from many vendors. Here are some examples of the prices taken from the site aliexpess at the time of writing.

- NodeMCU [1]: $\sim$ 2€
- MicroSD car reader [2]: $\sim$ 0.50€
- GPS sensor [3]: $\sim$ 3.5€
- MQ gas sensor [4]: $\sim$ 1€

In total the main components $\sim$ 7€, considering the shipping and the cost of the other components (cables, leds, button, sd card) it can amount to less than 15€. Keep in mind that these prices are valid for a single piece and that in case of bulk purchase it would be much lower.

[1]https://it.aliexpress.com/item/32879925927.html
[2]https://it.aliexpress.com/item/32572793362.html
[3]https://it.aliexpress.com/item/32836015224.html
[4]https://it.aliexpress.com/item/4000575957425.html

V. Results and data analysis

A. Test scenarios

1) Prototype test:

2) Real life implementation: State of the art

It make detection by method of cycle high and low temperature, and detect CO when low temperature [15]

VI. Future improvements

In this section

A. Hardware improvements

There are various improvements that could be made to the circuit and the hardware used, for example in this project using one single air quality sensor could be seen as a drawback since we could take advantage of the cheapness of these sensors

It is possible to add multiple sensors by expanding the analog inputs using an *Analog to Digital Converter* (ADC)

that converts an analog voltage on a pin to a digital number. By converting from the analog world to the digital world, we can begin to use electronics to interface to the analog world around us. [16]

In case we would want to use another transmission medium instead of wifi we could think about switching boards and use an arduino leonardo or an arduino uno or whatever iteration possible

5gh board

Given the small dimensions of the components, this project could be easily fitted inside of a custom made box. A solution could be to 3D print one and designing it based on the and considering that the air sensor should remain outside of the plastic and that the whole project could be exposed to high levels of humidity (in the winter season e.g.).

B. Software improvements

Supposed we use the same architecture we can improve the software such that instead of looking for the ap and snding the data only when the button is pressef it could be set such that

C. Cloud services improvements

In the development of this project I was forced by the we server host altervista.org to connect to the database only from within

connections within its domain

i was not able to allow the lambda to insert the data directly in the database, so I was forced to create an intermediate webpage to insert that data

for this reason a possible improvement could be

VII. Conclusions and future work

This paper describes the implementation of a wireless IoT system capable to gather data from air and send it to the cloud in order to be analyzed and displayed.

REFERENCES

[1] Mobilititszentral official website: https://www.mobiliteit.lu/en/tickets/free-transport/

[2] *The Vehicle of the Future Has Two Wheels, Handlebars, and Is a Bike*: https://www.wired.com/story/vehicle-future-bike/

[3] Mobike official global website: https://mobike.com/global/

[4] Mimoto official website: https://mobike.com/global/

[5] Fritzing official website: https://fritzing.org/home/

[6] ArduECO GitHub repository: https://github.com/cipz/ArduECO

[7] test https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-7 Ver1.3 - Manual.pdf

[8] https://github.com/nodemcu/nodemcu-devkit

[9] *In bicicletta al lavoro: a Milano risparmiate 27,5 tonnellate di CO2*: https://www.ilsole24ore.com/art/in-bicicletta-lavoro-milano-risparmiate-275-tonnellate-co2-ACYBZ1FB

[10] Official Arduino website: https://www.arduino.cc/

[11] Official Raspberry Pi website: https://www.raspberrypi.org/

[12] Lua based interactive firmware for ESP8266, ESP8285 and ESP32 https://github.com/nodemcu/nodemcu-firmware
http://www.padovaoggi.it/attualita/dati-mobike-padova-10-ottobre-2019.html

[13] MicroPython tutorial for ESP8266: https://docs.micropython.org/en/latest/esp8266/tutorial/

[14] In arrivo anche a Padova le E-Bike a pedalata assistita: l'annuncio di Mobike In arrivo anche a Padova le E-Bike a pedalata assistita: l'annuncio di Mobike: http://www.padovaoggi.it/attualita/mobike-e-bike-dati-padova-21-febbraio-2020.html
https://arduinojson.org/

[15] *MQ-7 Semiconductor Sensor for Carbon Monoxide* specifications: https://www.pololu.com/file/0J313/MQ7.pdf

[16] *What is the ADC?*: https://learn.sparkfun.com/tutorials/analog-to-digital-conversion/