Table of contents

1. Abstract:

As a new initiative by smart city projects that are going viral in worldwide ICT developments, mostly by governments, IoT sensors and their applications have been exploited and adopted proactively nowadays. A useful but relatively low-tech application is counting human presence by using carbon dioxide sensor. Such CO2 sensors are durable and inexpensive, with their compact sizes they could be deployed anywhere for estimating head counts ubiquitously. In this paper, a case study of applying CO2 sensors in public buses is investigated. Counting passengers in public buses or public transport in general has great economics advantages. However, a few technical challenges include but not limited to the mobility of the bus, the dynamic air flows, and factors such as windows were open, ventilation and even urban pollution etc, would affect the accuracy of occupancy counting. Hardly there would be a simple linear mapping between the number of people in a bus and the measurement of CO2 level. Hence, non-linear machine learning tool is used for inferring the non-linear relation between the two, with the consideration of the mentioned influential factors. Empirical data are collected from experiments conducted in several different buses over different times. The results can point to a promising conclusion that satisfactory accuracy could be achieved.

2. Equipments and tools of the project

2.1 Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital
input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip.
Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial
converter. of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode. of the board has the following new features:
pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR,which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0.

The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in 5 a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmeg5a328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design
EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf

Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use anATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power
The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically
.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The
adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however,the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the
voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (asopposed to 5 volts from the USB connection or other regulated power source). We can supplyvoltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage Wer board. We don't advise it.
3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
GND. Ground pins.

Memory: The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRM and 1KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output
Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e.
1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analogReference().

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8,
168, and 328 is identical.

Communication The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB – to- serial chip and USB connection to the computer (but 7 nort for serial commution on pins 0 and 1). A software serial library allows for serial communication on any of the UNO's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication,
use the SPI library.

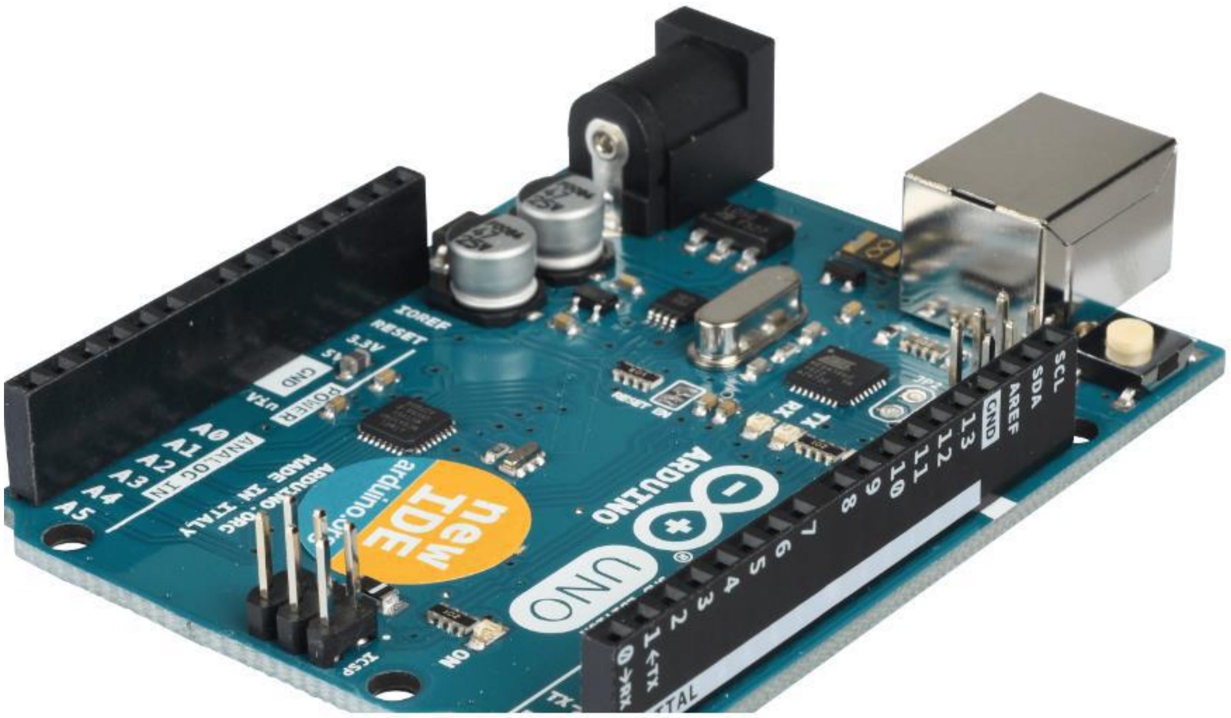
Programming
The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from
the **Tools > Board** menu (according to the microcontroller on Wer board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows We to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).
We can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:
and then resetting the 8U2.
later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.
We can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to
load a new firmware. Or We can use the ISP header with an external programmer (overwriting the
DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset
Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the
ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow We to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following halfsecond or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re enable it. Its labeled “RESET-EN”. We may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line.

USB Overcurrent Protection
The Arduino Uno has a resettable polyfuse that protects our computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics
The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB
connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil
(0.16"), not an even multiple of the 100 mil spacing of the other pins.



Arduino Uno

2.2 Gas Sensor:

The MQ-135 Gas sensors are used in air quality control equipments and are suitable for detecting or measuring of NH3, NOx, Alcohol, Benzene, Smoke, CO2. The MQ-135 sensor module comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when We are only trying to detect one particular gas. If We need to measure the gases in PPM the analog pin need to be used. The analog pin is TTL driven and works on 5V and so can be used with most common microcontrollers. If we are looking for a sensor to detect or measure common air quality gases such as CO2, Smoke, NH3, NOx, Alcohol, Benzene then this sensor might be the right choice.

MQ 135 Gas Sensor

Specifications:

Symbol	Parameter name	Technical condition	Remarks
Vc	Circuit voltage	5V±0.1	AC OR DC
V_H	Heating Voltage	5V±0.1	AC OR DC
R_L	Load resistance	Can Adjust	
R_H	Heater resistance	33Ω+5%	Room Tem
P_H	Heating consumption	Less than 800 mw	

2.3 Arduino IDE:

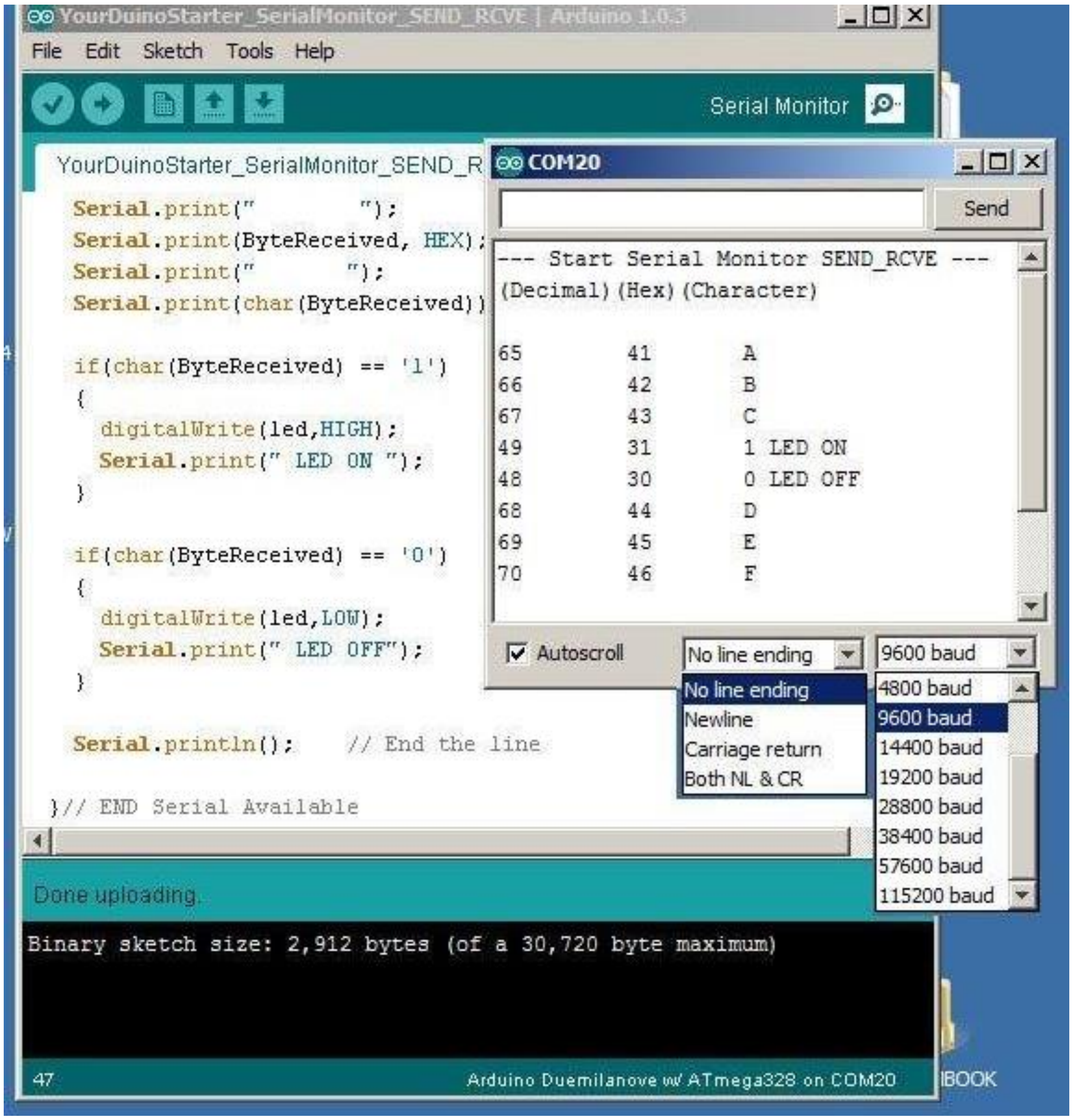
Arduino IDE is a software which is used to compile and upload program on Arduino. In Arduino after creating a new file we can write code for the program for the required project. After writing the code we have to verify the code . If it does not contain any error then we can upload it to the Arduino board and if it contain any error then we have to remove the error from our program. After removing error we can upload the code on the Arduino board.
It also contains a cerial monitor which is a pop-up window that act as a seprate terminal that communicate by sending and receiving serial data.We will use the Serial Monitor to debug Arduino Software Sketches or to view data sent by a working Sketch. We must have an Arduino connected by USB to Wer computer to be able to activate the Serial Monitor.
To get familiar with using the Serial Monitor, Copy and Paste the following example Sketch into a blank Arduino IE window. Then Verify it and if it's OK, Upload it. The next step will show We what to expect.
EXAMPLE SKETCH: CUT and PASTE

```
/* Wer ArDuino Starter_Serial Monitor_SEND_RCVE<br> - WHAT IT DOES:
- Receives characters from Serial Monitor
- Displays received character as Decimal, Hexadecimal and Character
- Controls pin 13 LED from keyboard.
- See the comments after “//” on each line below
- connections:
- None: pin 13 built-in LED

- V1.00 02/11/13
Questions: terry@Werduino.com */
/*-----( Import needed libraries )-----*/
/*-----( Declare Constants and Pin Numbers )-----*/
#define led 13 // built-in LED
/*-----( Declare objects )-----*/
/*-----( Declare Variables )-----*/
int ByteReceived;
void setup() /****** SETUP: RUNS ONCE ******/
{
Serial.begin(9600);
Serial.println("--- Start Serial Monitor SEND_RCVE ---");
Serial.println(" Type in Box above, . ");
Serial.println("(Decimal)(Hex)(Character)");
Serial.println();
}
//--(end setup )---
void loop() /****** LOOP: RUNS CONSTANTLY ******/
{
```

```
if (Serial.available() > 0)
{
ByteReceived = Serial.read();
Serial.print(ByteReceived);
Serial.print(" ");
Serial.print(ByteReceived, HEX);
Serial.print(" ");
Serial.print(char(ByteReceived));
if(ByteReceived == '1') // Single Quote! This is a character.
{
digitalWrite(led,HIGH);
Serial.print(" LED ON ");
}
if(ByteReceived == '0')
{
digitalWrite(led,LOW);
Serial.print(" LED OFF");
}
Serial.println(); // End the line
// END Serial Available
}
}
```

```
//--(end main loop )---
/*-----( Declare User-written Functions )-----*/
/*********( THE END )***********/
```

What the Serial Monitor Looks Like:



```
  Serial.print("        ");
  Serial.print(ByteReceived, HEX);
  Serial.print("       ");
  Serial.print(char(ByteReceived))

  if(char(ByteReceived) == '1')
  {
    digitalWrite(led,HIGH);
    Serial.print(" LED ON ");
  }

  if(char(ByteReceived) == '0')
  {
    digitalWrite(led,LOW);
    Serial.print(" LED OFF");
  }

  Serial.println();    // End the line

}// END Serial Available
```

```
--- Start Serial Monitor SEND_RCVE ---
(Decimal)(Hex)(Character)

65         41        A
66         42        B
67         43        C
49         31        1 LED ON
48         30        0 LED OFF
68         44        D
69         45        E
70         46        F
```

```
Binary sketch size: 2,912 bytes (of a 30,720 byte maximum)
```

When We click on it, the Serial Monitor will pop up in a new window. Above is what our example Serial Monitor Sketch looks like with the Monitor opened.
Look at the Serial Monitor window.
 - The small upper box is where We can type in characters (hit or click "Send")

• - The larger area (Corner can be dragged to enlarge) is where characters sent From Arduino will be displayed.- At the bottom are two pulldowns:

 - One sets the "line ending" that will be sent to Arduino when We or click Send

 - The other sets the Baud Rate for communications. (If this does not match the value set up in Wer sketch in Setup, characters will be unreadable). Example: Serial.begin(9600); Some sketches or other applications may use a different Baud Rate.

DEBUGGING WITH THE SERIAL MONITOR:
If We are testing a new sketch We may need to know what's happening when We try to run it. But **"Software Is Invisible ! "**. So We need the tell the software to tell We what it's doing, and sometimes the value of changing variables. We do this my using the Serial Monitor and adding code to Wer sketch to send characters that We can see.
SETUP: In Setup We need to begin Serial Communications and set the Baud Rate (speed) that data will be transferred at. That looks like this:
Serial.begin(9600); // Other baud rates can be used...
Serial.println("My Sketch has started");
The second line is optional...
LOOP: Here We can print helpful info to the Serial Monitor. Examples:

```
Serial.println("Top of loop");
Serial.println("Reading Temperature Sensor");
Serial.print("LoopCounter value = ");
Serial.println(LoopCounter);
```

For details of how to show different data, see "Serial_Print.html" in the Reference section of Wer Arduino IDE:
Top menu: Help>reference/Serial_Print.html
Experiment with changing the Sample Software Sketch.

2.4 Connecting wires:

We use connecting wires to make the circuit connections.

Connecting Wires

2.4 Power Source and system:

Power supply circuit has evolved to an almost foolproof design. In this project, we will learn about the four different ways in which we can power up the ARDUINO UNO. While making projects, it is necessary to know the following techniques, since there are instances when flexibility with regards to the power supply is required.

Things required to power up the ARDUINO UNO

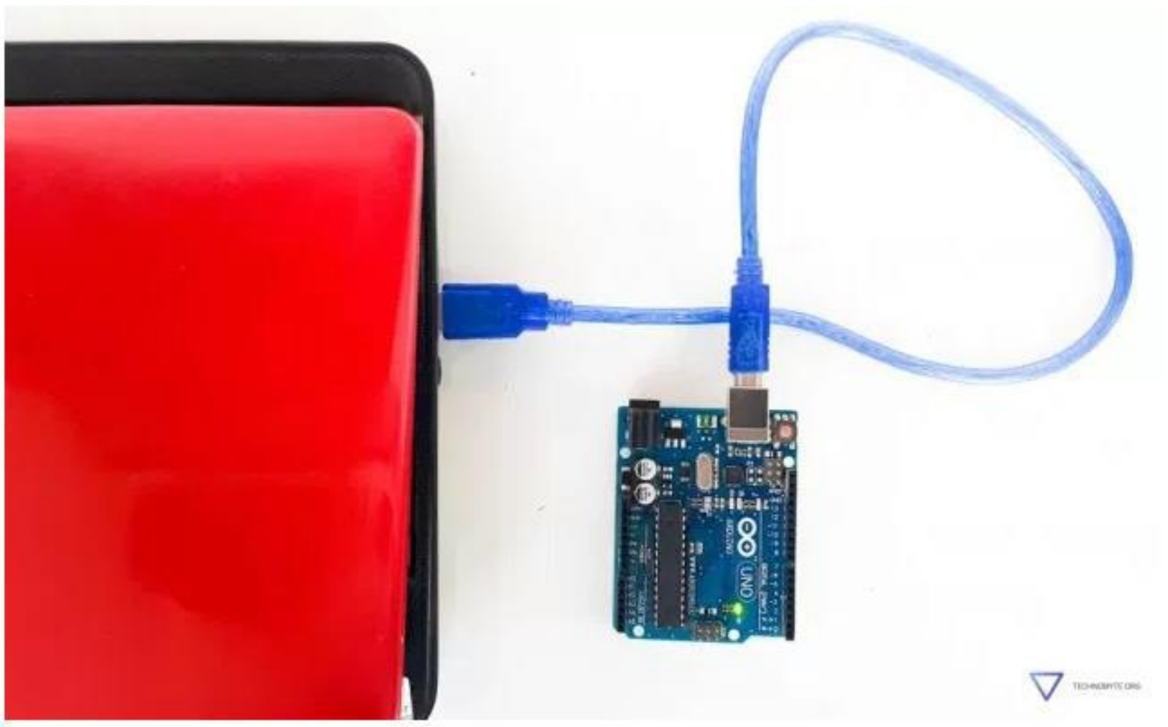
We are going to need the following apparatus to learn how to switch on the ARDUINO UNO.

 An ARDUINO UNO board

 Standard A-B USB cable
 AC to DC Adapter (7-12V)
 Batteries (9V) with a battery connector

- Modern ARDUINO UNO boards allow the board to have more than one source of power to be connected simultaneously. An intelligent switching circuitry ensures that the highest available voltage is selected and sent to the onboard voltage regulator and eventually powers up the board.

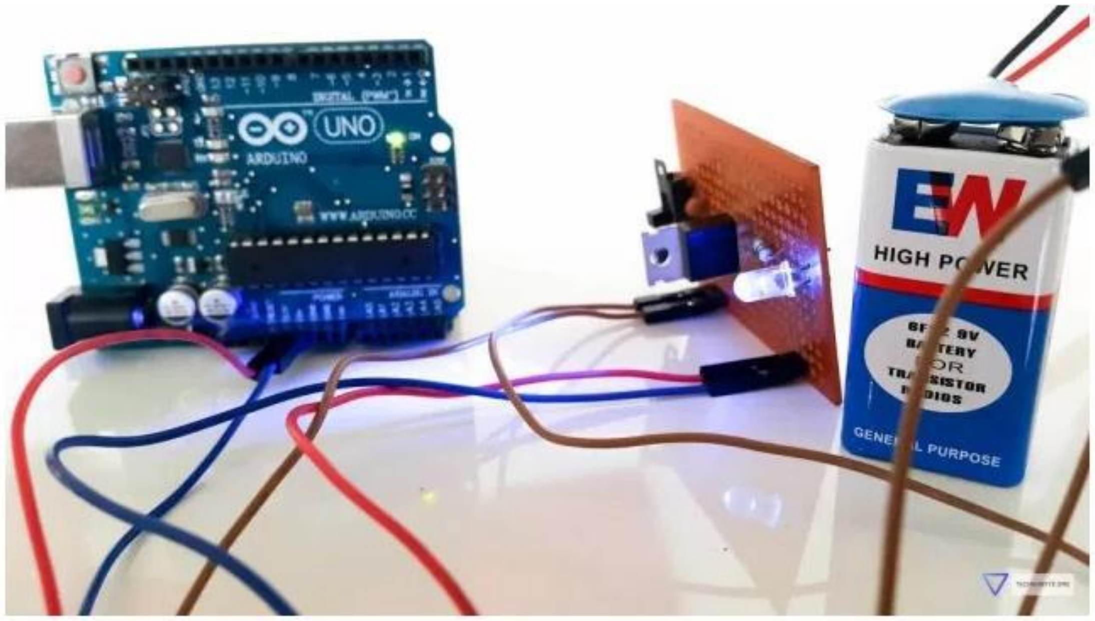
- We can power up the ARDUINO UNO using power supplied from the computer via a USB cable and/or by using external power sources.


An Arduino Uno powered up using a USB cable

Using an AC to DC adapter plugged into the barrel connector:

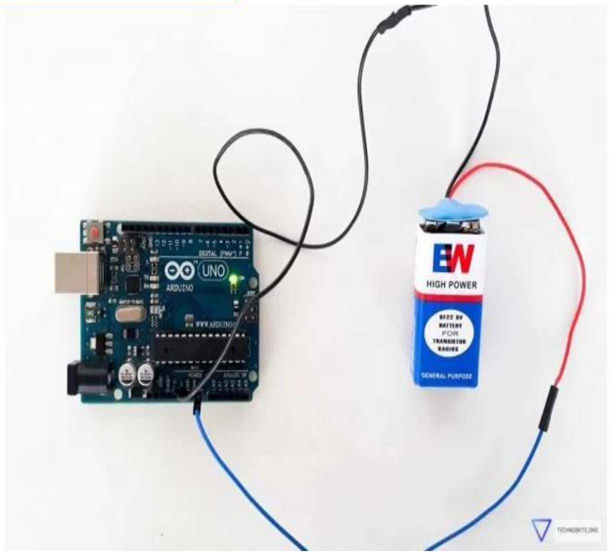
The barrel connector can be supplied an input of 7-12V. This is regulated to 5V by the onboard voltage regulator, and the board is powered on.

Using 5V input:



It is possible to power up the ARDUINO UNO using the 5V and GND pins, provided that the input given is steady and regulated 5V. The 5V pin bypasses the voltage regulator and all the safety measures present on the ARDUINO UNO, so if the input exceeds 5V (5.5 is the maximum upper limit), the board can be damaged. It is generally advised to avoid powering up the ARDUINO UNO using this method.

Using batteries greater than 5V:



Connect a 9V battery with the positive terminal connected to the Vin pin and the negative terminal connected to the GND pin. The Vin port allows an input between 7 and 12 Volts, but we recommend to use a 9V battery. Depending on Wer application We can input 12V too but make sure the current values stay around 500mA.

Precautions to be undertaken before switching on the ARDUINO UNO

- If the barrel connector and an AC-DC adapter are being used to power up the Arduino, make sure that the output of the adapter is between 7-12V. Although the rated input can exceed to as much as 20V, it is safe to stay within the recommended range to protect the voltage regulator from excessive heating. Also, see to it that the GND and Vin pins are not shorted.
- But if we are using the 5V and GND pins to power up the ARDUINO UNO, it is imperative that the 5V input is stable and steady.
- If the Vin/5V and GND pins are being used to power up the Arduino, double check the polarity because if the GND and 5V/Vin pins are mixed up, it can potentially damage the ARDUINO UNO board.

3. Working and Principle:

The MQ135 consists of a surface covered in a thin layer of SnO_2, and a heater resistor which serves to raise the temperature of the SnO_2 surface to several hundred degrees Celsius. SnO_2 is an n-type semiconductor, in which donor electrons are excited to the conduction band at elevated temperatures. However, SnO_2 when exposed to air readily adsorbs oxygen onto its surface. The adsorption reaction consumes electrons from the conduction band to form negatively charged oxygen species. Therefore, SnO2 has low conductivity in clean air. [Shimizu]
In the presence of a flammable gas, the gas will also adsorb onto the sensor surface, where it consumes the adsorbed oxygen species to form CO_2 and H_2O. This reaction releases the donor electrons back into the conduction band, thereby raising the conductivity of the sensor. By quantifying the conductivity response to the presence of various gases, a thin SnO_2 surface can be used to determine the concentration of the gas. [Shimizu]
CO_2, notably, is not a flammable gas. Here, the sensing mechanism is different; instead of reacting with adsorbed oxygen directly, CO_2 sensing relies on water vapor first reacting with adsorbed oxygen to form adsorbed hydroxide (OH^-). CO_2 then in turn reacts with the adsorbed hydroxide, forming carbonate (CO_3^{2-}). This process ultimately also returns electrons to the SnO2 conduction band, and therefore results in increased conductivity.

The MQ-135 alcohol sensor consists of a tin dioxide (SnO2), a perspective layer inside Aluminium Oxide micro tubes (measuring electrodes) and a heating element inside a tubular casing. The end face of the sensor is enclosed by a stainless steel net and the back side holds the connection terminals. Ethyl alcohol present in the breath is oxidized into acetic acid passing through the heat element. With the ethyl alcohol cascade on the tin dioxide sensing layer, the resistance decreases. By using the external load resistance the resistance variation is converted into a suitable voltage variation.

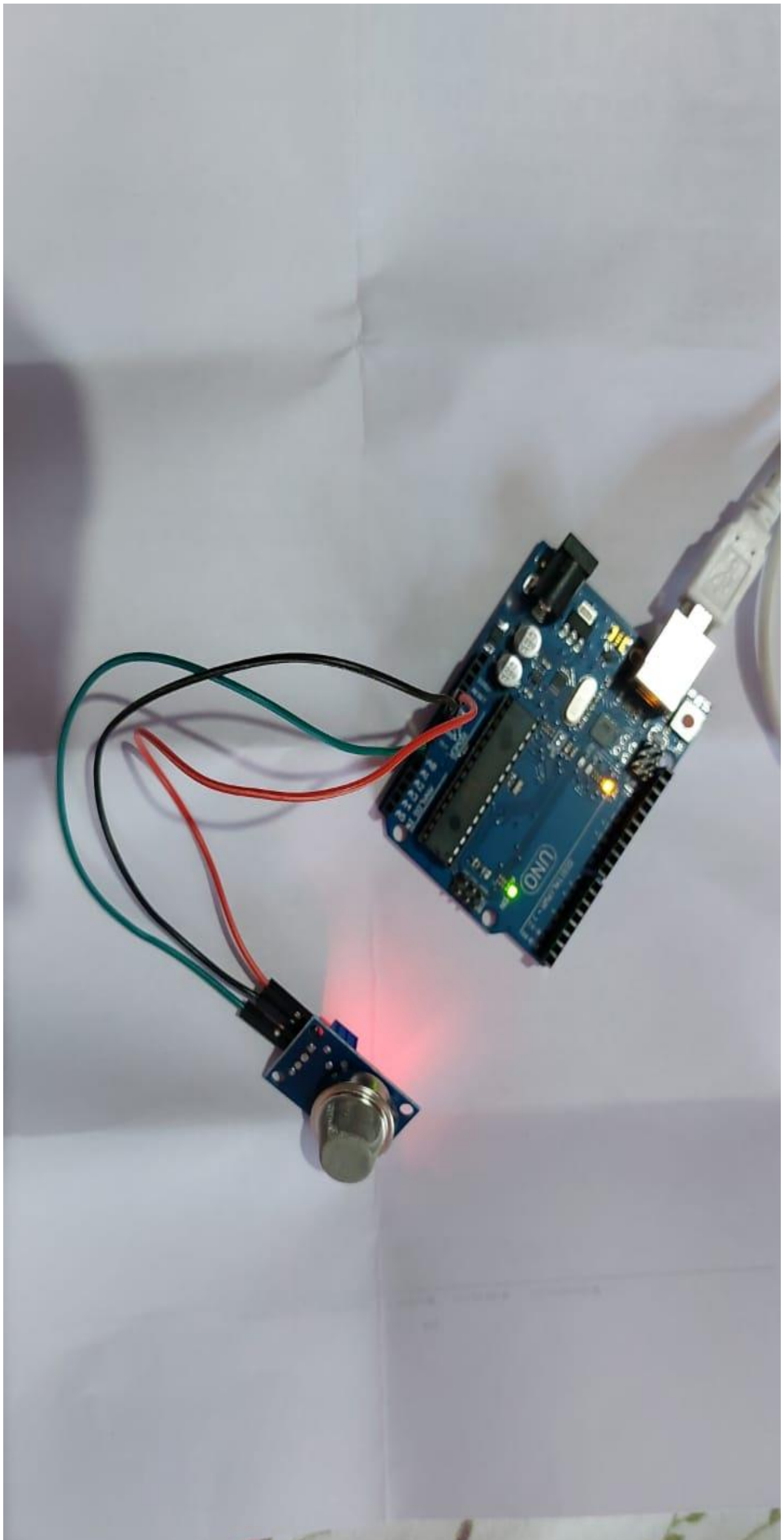
It has lower conductivity compare to clean air and due to air pollution the conductivity is increases. The air quality sensor detects ammonia, nitrogen oxide, smoke, CO2 and other harmful gases. The air quality sensor has a small potentiometer that permits the adjustment of the load resistance of the sensor circuit.

The air quality sensor is a signal output indicator instruction. It has two outputs: analog output and TTL output. The TTL output is low signal light which can be accessed through the IO ports on the microcontroller. The analog output is an concentration, i.e. increasing voltage is directly proportional to increasing concentration. The resistance of the sensor decreases as the concentration of the target gas is increased in PPM while for clean air its resistance remains constant.

The VCC and Ground terminals of the sensor are connected to the common VCC and Ground. The Analog Output pin of the sensor is connected to the A0 pin of the Arduino. The analog output voltage from the sensor can be assumed directly proportional to the concentration of CO2 gas in PPM under standard conditions. The analog voltage is sensed from the sensor and converted to a digital value in range from 0 to 1023 by the inbuilt ADC channel of the controller. The digitized value is hence equal to the gas concentration in PPM.

Circuit Diagram and Code:

The circuit diagram for this project can be designed as below:

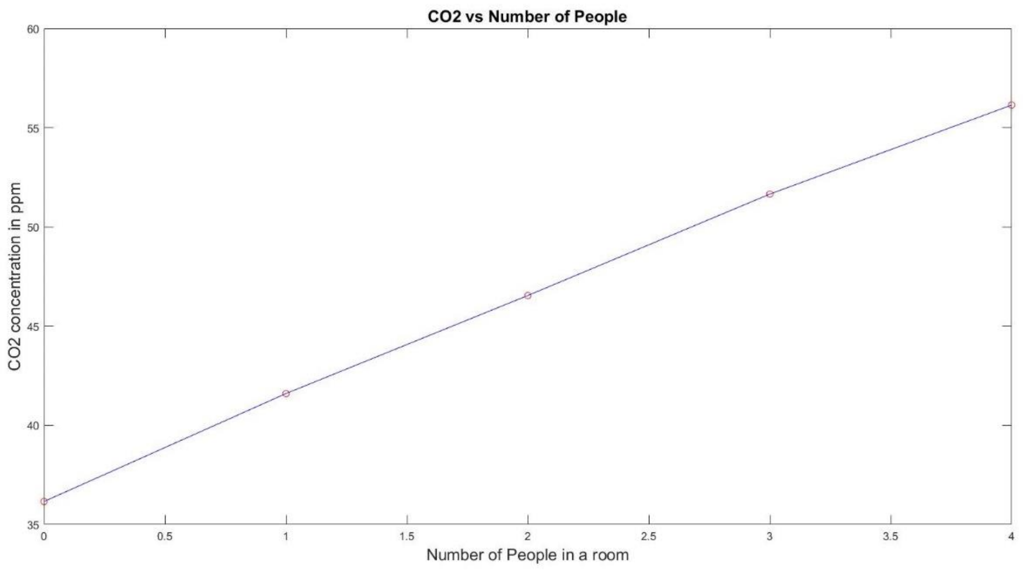

UNO

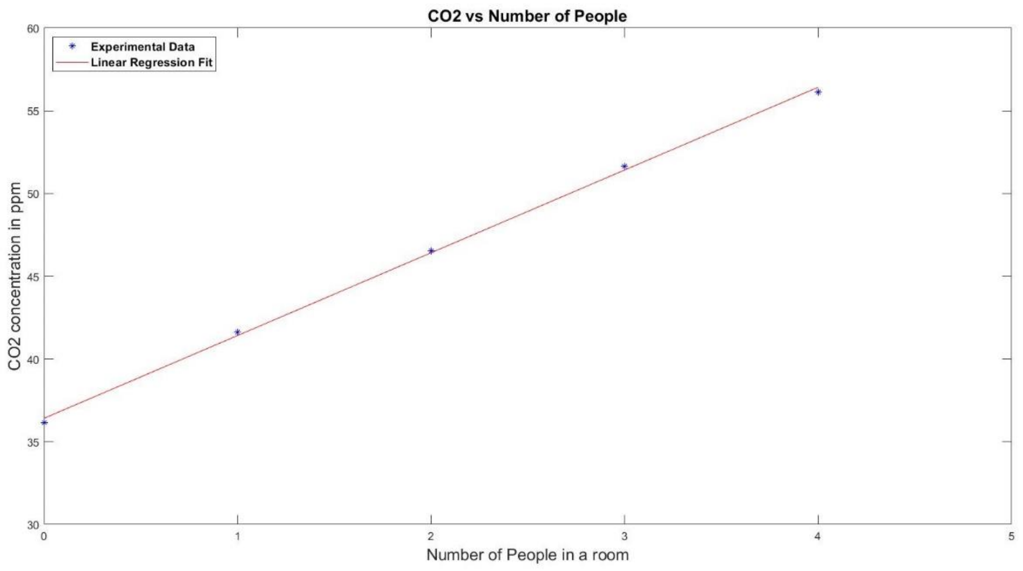
Arduino code for the project:

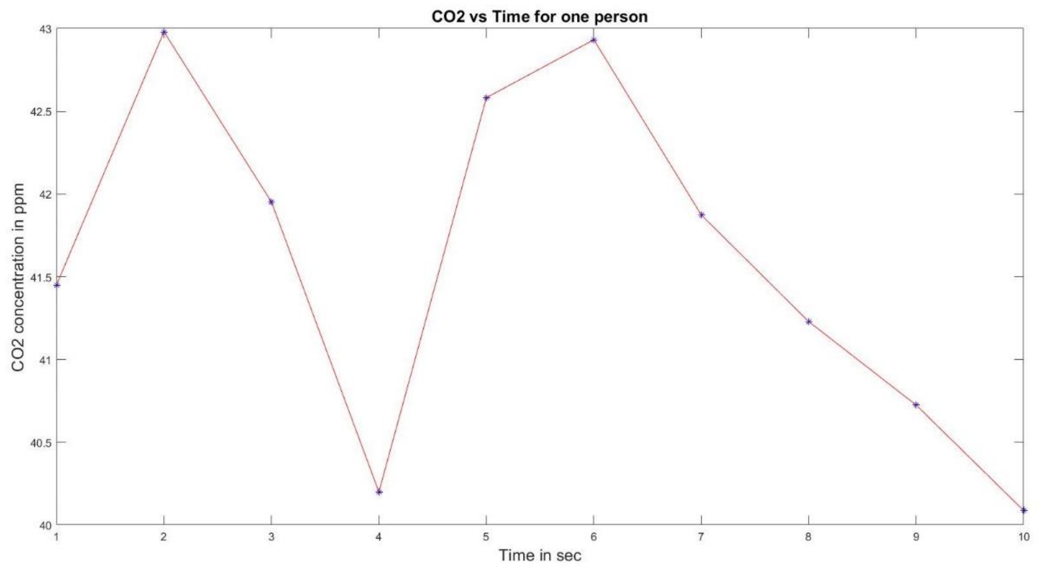
```
int sensorValue;
int avgVal = 0;
int count = 0;

void setup(){
  Serial.begin(9600);                        // sets the serial port to 9600
}
void loop(){
  sensorValue = analogRead(0);                 // read analog input pin 0
  count = count + 1;
  avgVal = sensorValue + avgVal;
  if(count == 100){
    avgVal = avgVal/count;
    Serial.print("AirQua=");
    Serial.print(avgVal, DEC);               // prints the value read
    Serial.println(" PPM");                  // wait 100ms for next reading
    avgVal = 0;
    count = 0;
  }
  delay(100);

}
```

4. Data and Observations:

The plots obtained from the sensor data are mentioned below:






CO2 vs Time for one person
CO2 concentration in ppm
Time in sec
43
42.5
42
41.5
41
40.5
40
1
2
3
4
5
6
7
8
9
10

Applications:

- Modified atmospheres
- Indoor air quality
- Stowaway detection
- Cellar and gas stores
- Marine vessels
- Greenhouses
- Landfill gas
- Confined spaces
- Aerospace
- Healthcare
- Horticulture
- Transportation
- Cryogenics
- Ventilation management
- Mining
- Rebreathers (SCUBA)
- Decaffeination
- For indoor human occupancy counting
- For HVAC(Heating,Ventilation and Air conditioning) applications, CO_2 sensors can be used to monitor the quality of air and the tailored need for fresh air, respectively. Measuring CO_2 levels indirectly determines how many people are in a room, and ventilation can be adjusted accordingly.

6. Conclusion:

Using gas sensor we can count the approximate number of passengers in a bus or in metro or other vehicles. In the same manner we can count the number of passenger in a particular compartment, using this we can control the rush and make efficient use of the resources used in transportation services.

7. References:

1. Jiang, C., Masood, M. K., Soh, Y. C., & Li, H. (n.d.). Indoor occupancy estimation from carbon dioxide concentration. Retrieved July 20, 2016.

2. M. A. ul Haq, M. Y. Hassan, H. Abdullah, H. A. Rahman, M. P. Abdullah, F. Hussin, and D. M. Said, "A review on lighting control technologies in commercial buildings, their performance and affecting factors," Renew. Sustain. Energy Reviews, vol. 33, 2014.

3. T. Teixeira, G. Dublon, and A. Savvides, "A survey of human-sensing: Methods for detecting presence, count, location, track, and identity,"ACM Comput. Surveys, vol. 5, 2010.

4. Stitnimankarn, P.; Siripoorikan, T.; Thanomkul, N.; De Silva, P.; Pungetmongkol, P.; Staubs, J. Transient CO2 Diffusion from Vehicle Cabin Micro-environment in Hot and Humid Climates. *IJESD* **2020**, *11*, 273–277, doi:10.18178/ijesd.2020.11.5.1262.

5. Luangprasert, M.; Vasithamrong, C.; Pongratananukul, S.; Chantranuwathana, S.; Pumrin, S.; De Silva, I.P.D. In-vehicle carbon dioxide concentration in commuting cars in Bangkok, Thailand. *J. Air Waste Manag. Assoc.* **2017**, *67*, 623–633, doi:10.1080/10962247.2016.1268983.

6. Monthly Average Mauna Loa CO2; Global Monitoring Laboratory, National Oceanic and Atmospheric Administration (NOAA), US Department of Science, USA. Available online: https://www.esrl.noaa.gov/gmd/ccgg/trends/ (accessed on 09 December 2020).

Eng. Proc. **2020**, *2*, 88 5

7. Lindsey, R. *Climate Change: Atmospheric Carbon Dioxide*; National Oceanic and Atmospheric Administration: US Department of Science, USA. Available online: https://www.climate.gov/newsfeatures/ understanding-climate/climate-change-atmospheric-carbon-dioxide (accessed on 09 December 2020).

8. Lüthi, D.; Le Floch, M.; Bereiter, B.; Blunier, T.; Barnola, J.-M.; Siegenthaler, U.; Raynaud, D.; Jouzel, J.; Fischer, H.; Kawamura, K.; et al. High-resolution carbon dioxide concentration record 650,000–800,000 years before present. *Nature* **2008**, *453*, 379–382, doi:10.1038/nature06949.

9. ASHRAE Standards *Ventilation for Acceptable Indoor Air Quality*; ASHRAE Standing Standard Project Committee 62.1; American Society of Heating refrigerating and Air-Conditioning Engineers Inc.: Atlanta, GA, USA: 2003; pp. 1–15.

10. ASHRAE Standards. *Ventilation for Acceptable Indoor Air Quality - Addenda 2016*; ANSI/ASHRAE Addendum b to ANSI/ASHRAE Standard 62.1-2016; American Society of Heating refrigerating and Air-Conditioning Engineers Inc.: Atlanta, GA, USA, pp. 1–6.

11. *Review of In-Cabin Carbon Dioxide Levels*; Roads and Maritime Services: Sydney, NSW, Australia; p. 57.

12. ASHRAE Standards *ASHRAE standard 62-1981: Ventilation for Acceptable Indoor Air Quality*; ASHRAE Standing Standard Project Committee 62.73R; American Society of Heating refrigerating and Air- Conditioning Engineers Inc.: Atlanta, GA, USA, 1989; pp. 1–11

13. Persily, A.; de Jonge, L. Carbon dioxide generation rates for building occupants. *Indoor Air* **2017**, *27*, 868–879, doi:10.1111/ina.12383.