CAPUEE 25-26

Communications

Adriano Caprara

adriano.caprara@upc.edu

Agenda

CITCEA UPC
Boosting the future of energy

1. Introduction
2. ESP32 -> Laptop (receive data from sensors)
3. Laptop -> ESP32 (send commands to actuators)

Why connect Python and Arduino?

CITCEA UPC
Boosting the future of energy

Arduino/ESP32:

- Reads sensors and control actuators
- Real time interaction with the physical world
- **But low computing power**

Laptop (Python):

- Handles heavy computation (i.e. data processing, ML algorithms, visualizations…)
- Easy access to APIs, databases, …
- Faster to code and more user friendly compared to C/C++

Objectives

CITCEA UPC
Boosting the future of energy

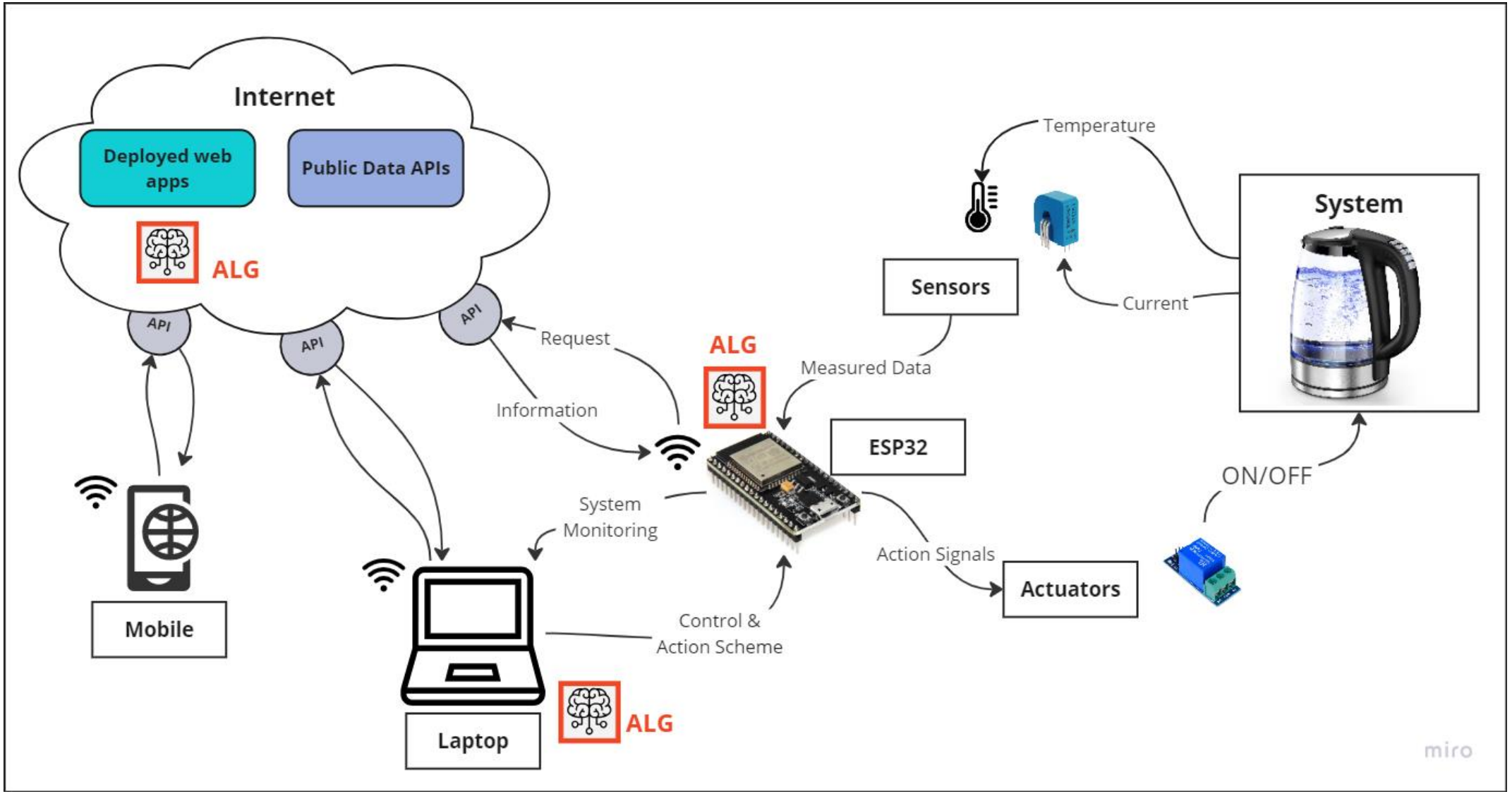
ESP32 -> Laptop

- Verify hardware and sensors connections
- Install pyserial library
- Establish serial communication and send data to laptop
- Store in .csv and visualize measured data

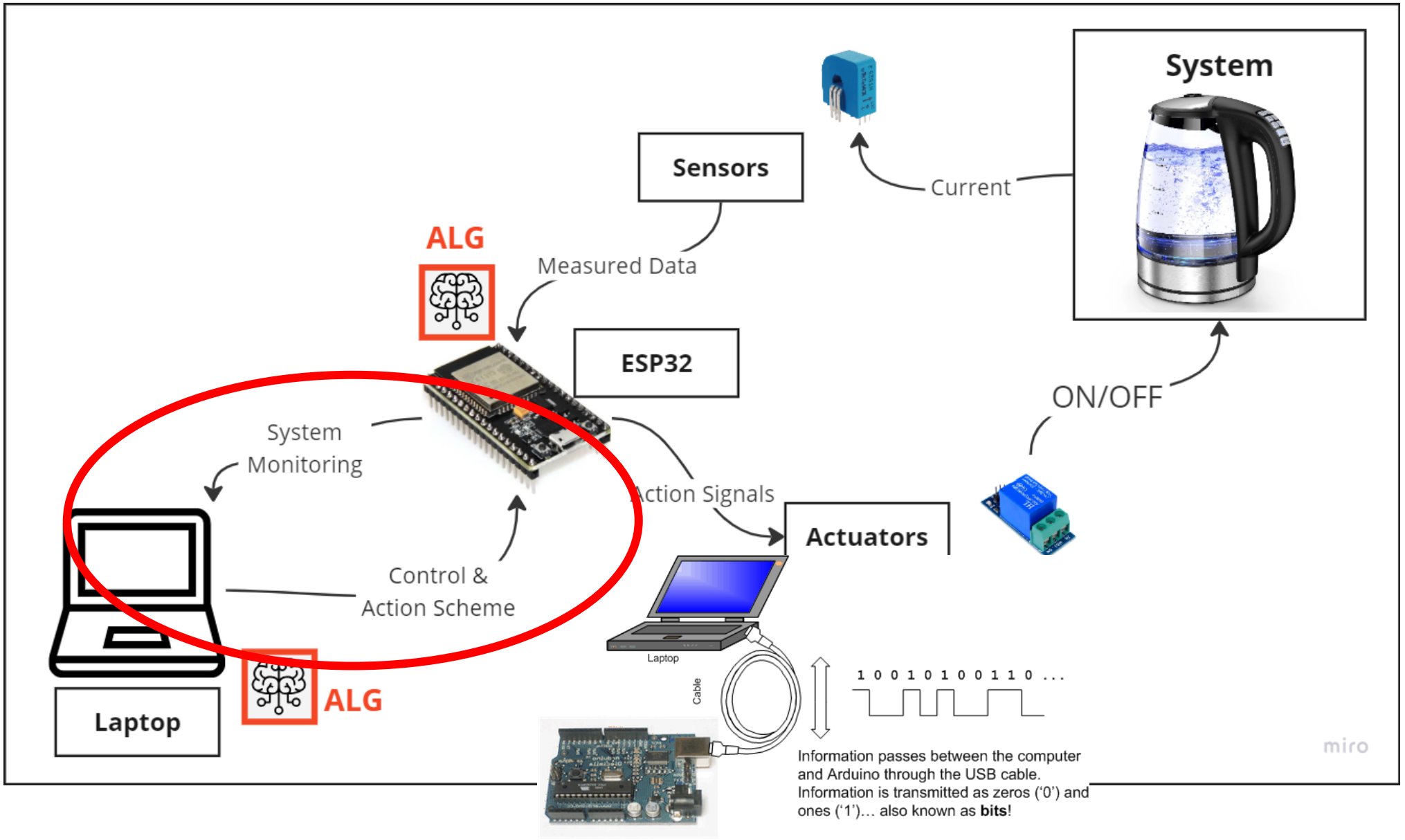
Laptop -> ESP32

- Define action signals (When do we turn ON/OFF?)
- Implement relay actuation

Communications & data flows



Communications & data flows



Serial communication

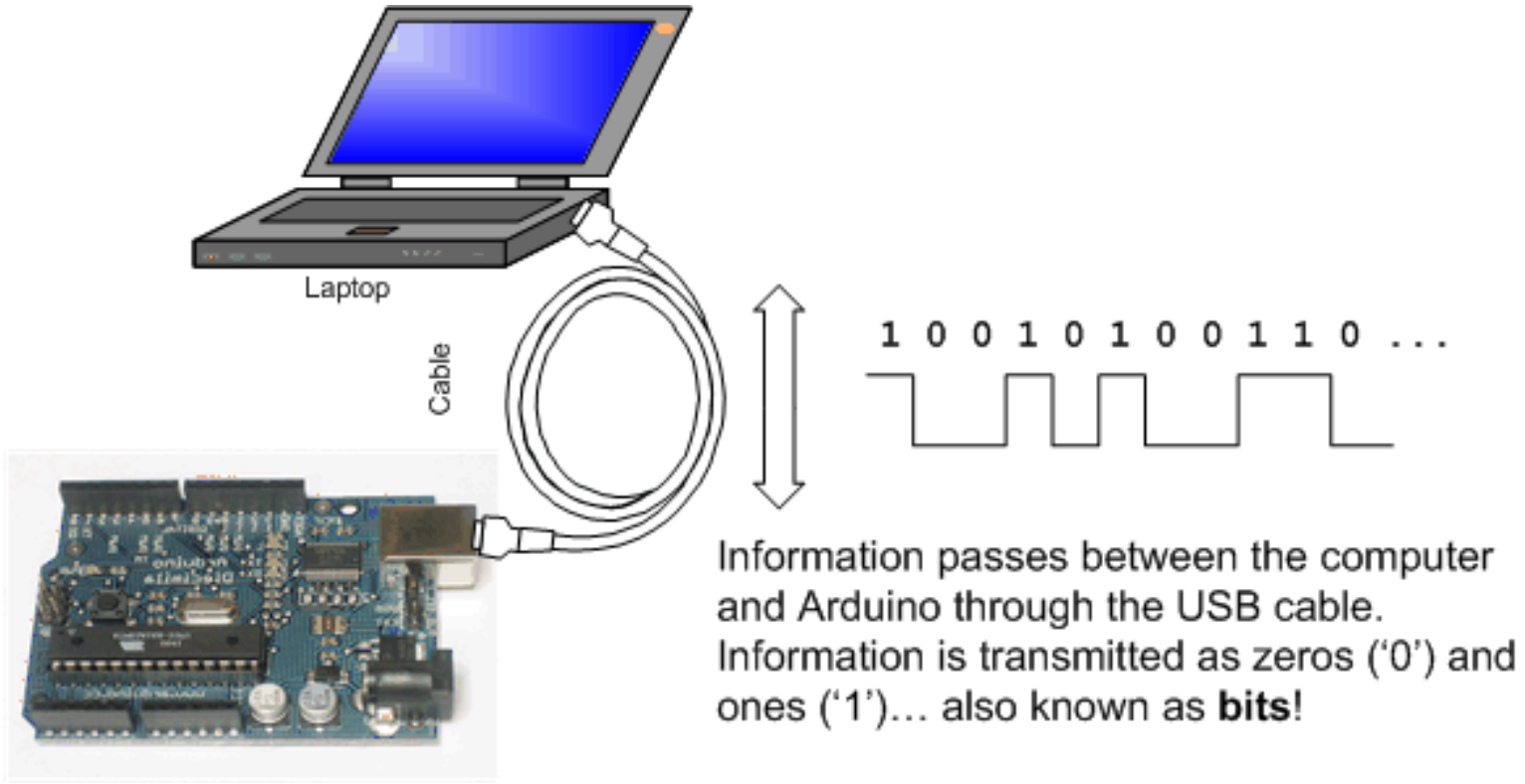
Serial port: information is transferred (Input or Output) sequentially one bit at a time

PROS:

- Easy to code and to set up

CONS:

- Requires cable
- Low data transfer capabilities (ok for sensors but not for audio)
- Needs to define how to parse the message

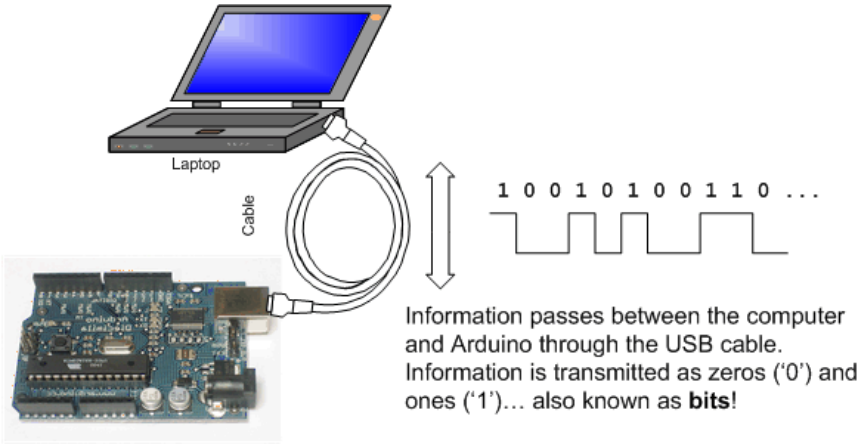


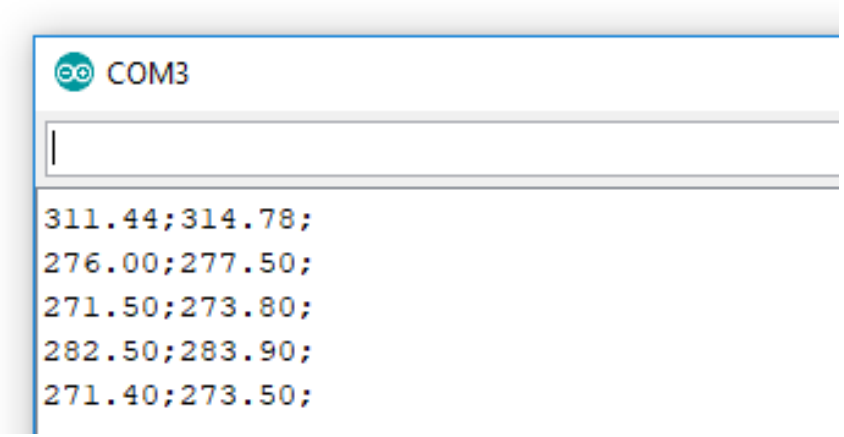
1. Send data from ESP32 to PC

Send data from ESP32 to PC with pyserial

1. Install **Pyserial;**
2. Connect ESP32 (Windows: “**COM3**”, “COM4”…; macOS: ***/dev/ttyUSB0***, /dev/ttyACM0, or similar);
3. (Arduino IDE): print desired variables;
4. (python): initialize serial object, read data lines and decode in desired data structure – i.e. pandas dataframe.

OBS 2: useful to separate values with “;”.





Install pySerial

Install with package manager of choice

```
pip install pyserial
```

When we use the code then we import the module it with:

```
import serial
```

Somewhat uncommon (but not impossible) for a **module** to have a different name from the **package** that installs it. (in most cases, the package name and module name are the same to avoid confusion)isn’t

pySerial how to use

CITCEA UPC
Boosting the future of energy

- Same baud rate on Arduino and Python side
- Make sure the serial monitor is closed (exclusive control over serial port taken by pyserial);
- Wait after opening port
- Use **println()** on the Arduino side and read line by line in Python using **.ser** command

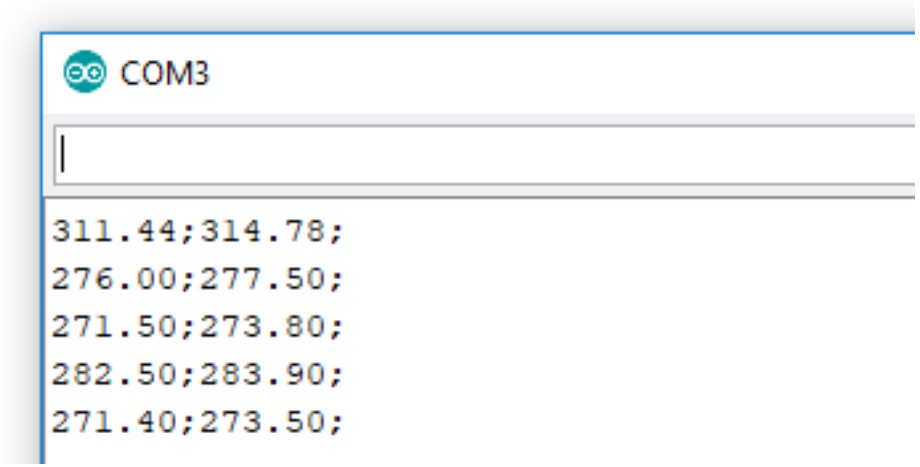
```
line = ser.readline().decode().strip()
```

- Read the line passed from the serial port, decode it and strip it of blanks

Aduino code: print() & println()

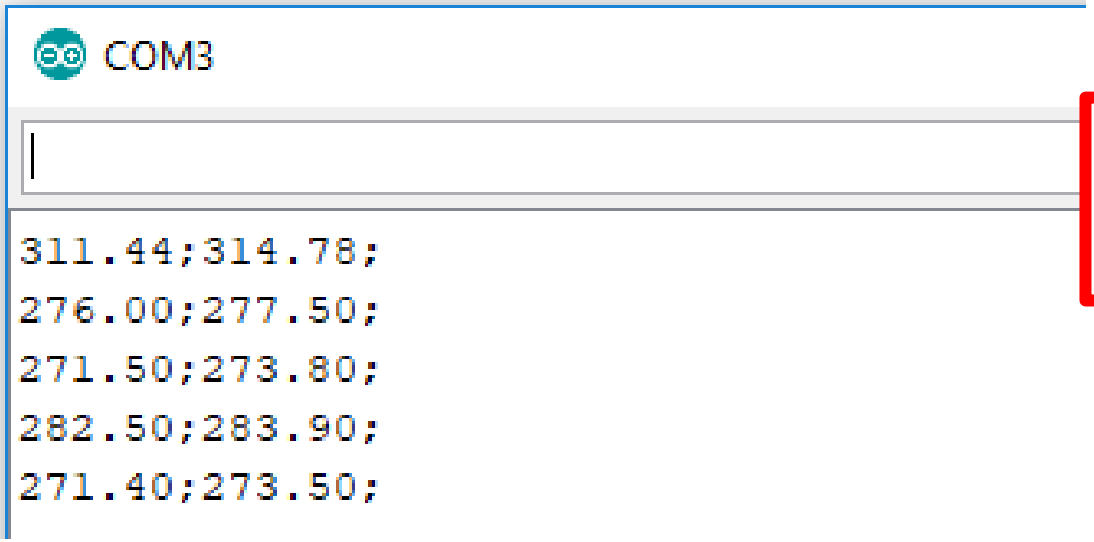
- **print():** print to serial monitor (or **serial output**);
- **println():** print to serial monitor and create new line afterwards.

```
const int sensor1Pin = A0;
const int sensor2Pin = A1;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensor1Value = analogRead(sensor1Pin);
  int sensor2Value = analogRead(sensor2Pin);

  // Print the sensor values on the same line, separated by ";"
  Serial.print(sensor1Value);
  Serial.print(";");
  Serial.print(sensor2Value);  // Send sensor 2 value and a newline
  Serial.println(";");

  delay(1000);  // Wait 1 second before the next reading
}
```



Parsing data

parsing: take a raw string and extract useful meaningful values



Understand the format

float + “;” + float + “;” + newline

Parsing with Python functions

CITCEA UPC
Boosting the future of energy

```python
import serial

ser = serial.Serial("COM3", 115200)

while True:
    line = ser.readline().decode().strip()  # read a full line
    if not line:
        continue

    parts = line.split(";")  # split at semicolon
    parts = [p for p in parts if p]  # remove empty elements

    # convert to floats
    values = [float(p) for p in parts]

    print(values)
```

Or with regex

CITCEA UPC
Boosting the future of energy

```python
import serial
import time
import re

# Set up the serial port
ser = serial.Serial('COM3', 9600, timeout=1)
time.sleep(2)  # Wait for Arduino to initialize

try:
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode().strip()  # Read a line and decode

            # Use regex to find all numbers in the received line
            numbers = re.findall( pattern: r"[-+]?\d*\.\d+|[-+]?\d+", line)

            # Print the extracted sensor values
            print(f"Extracted values: {numbers}")
```

Parsing data: REGEX

CITCEA UPC
Boosting the future of energy

To find and extract specific text from a string

```python
import re

data = "Price: 42, Discount: 5.5%"
numbers = re.findall( pattern: r"[-+]?\d*\.\d+|[-+]?\d+", string=data)
print(numbers)  # Output: ['42', '5.5']

data2 = "32; 32.44; 375.67"
numbers2 = re.findall(pattern=r"[-+]?\d*\.\d+|[-+]?\d+", string=data2)
print(numbers2) # Output: ['32', '32.44', '375.67']
```

Floating-Point Numbers: [-+]?\d*\.\d+: **[-+]?** optional + or – sign, **\d*** zero or more digits before the decimal point, **\.** decimal point, **\d+** one or more digits after the decimal (e.g., 3.14, -982.22).

Integers: [-+]?\d+: **[-+]?,** optional + or -, **\d+** one or more digits (e.g., 42, -15).

2. Send data from PC to ESP32

Send data from PC to ESP32 (example)

CITCEA UPC
Boosting the future of energy

1. Send command from python using pyserial
2. Receive command with Arduino IDE
3. Action signals based on command (i.e. turn ON/OFF)

Python code

CITCEA UPC
Boosting the future of energy

```
try:
    while True:
        current_time = datetime.now()

        # Determine if the current second is even or odd to toggle the relay
        if current_time.second % 30 < 15:
            command = 'H'  # Send HIGH
        else:
            command = 'L'  # Send LOW

        ser.write(command.encode())  # Send the command to Arduino
        print(f"Sent command: {command} at {current_time.strftime('%H:%M:%S')}")

        time.sleep(1)  # Adjust the sleep time as necessary

except KeyboardInterrupt:
    print("Stopped by user")

finally:
    ser.close()  # Close the serial connection when done
```

Arduino code

CITCEA UPC
Boosting the future of energy

```ino
const int relayPin = 13;  // Pin connected to the relay or LED

void setup() {
  Serial.begin(9600);  // Start serial communication at 9600 baud
  pinMode(relayPin, OUTPUT);  // Set the relay pin as an output
}

void loop() {
  if (Serial.available() > 0) {  // Check if data is available to read
    char command = Serial.read();  // Read the incoming command

    if (command == 'H') {
      digitalWrite(relayPin, HIGH);  // Turn on the relay or LED
      Serial.println("Relay/LED ON");
    } else if (command == 'L') {
      digitalWrite(relayPin, LOW);  // Turn off the relay or LED
      Serial.println("Relay/LED OFF");
    }
  }
}
```

Exercise

Exercise

CITCEA UPC
Boosting the future of energy

1) Write an Arduino code to send a line to python (sensor data, “hello world”, ...) and read it with python

2) Do the opposite. Write a python code that writes piece of information (I,e, “HIGH” or “LOW”) and send it to Arduino

THANK YOU
FOR YOUR ATTENTION

CITCEA UPC
Boosting the future of energy

CIT UPC TECHNOLOGY CENTER
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

tecnio catalonia
ACCIÓ Generalitat de Catalunya