CAPUEE 25-26

Communications 2

Adriano Caprara

adriano.caprara@upc.edu

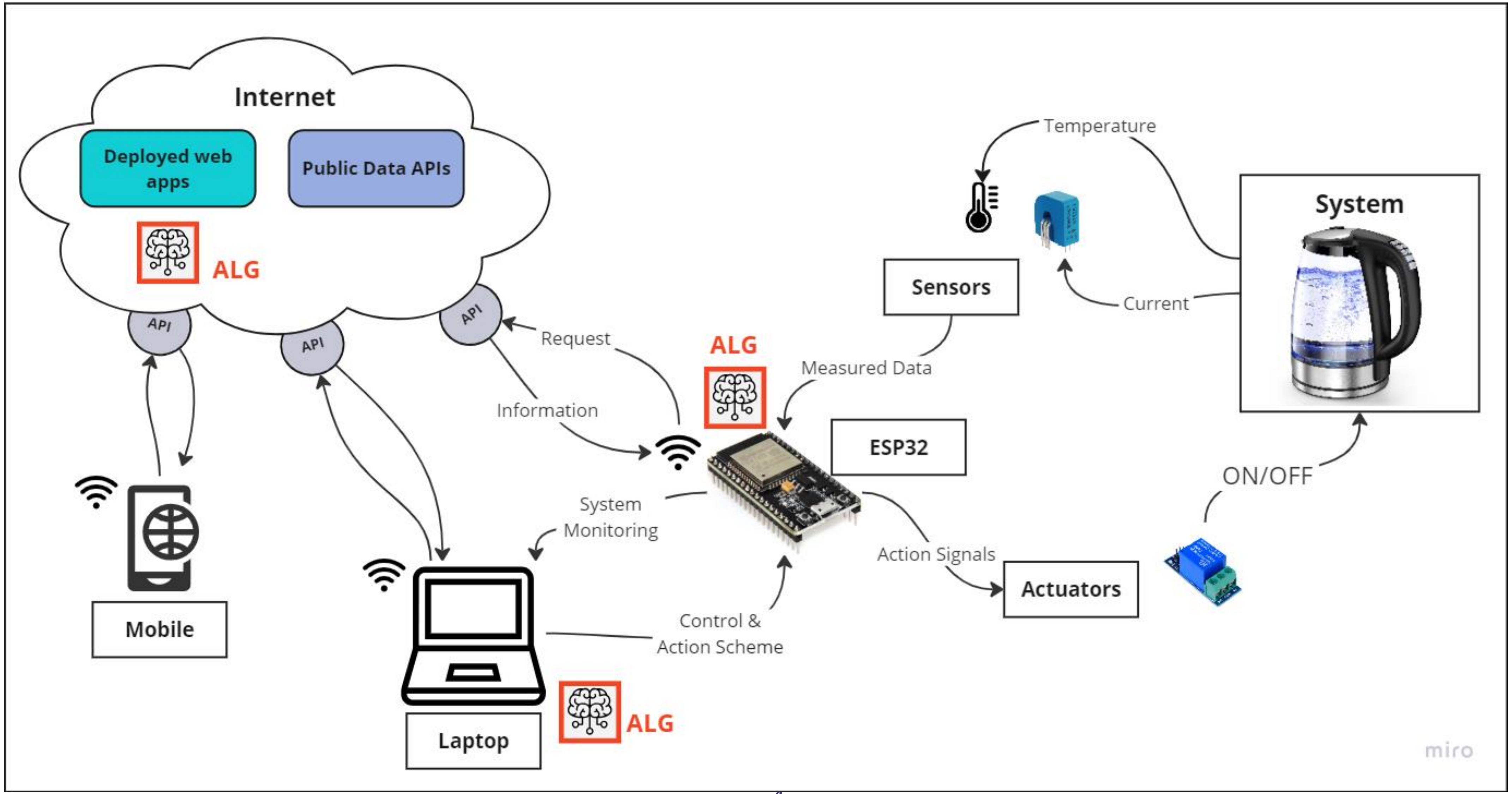
Agenda

CITCEA UPC
Boosting the future of energy

1. Revise Pyserial communications
2. Relay control
3. Wifi connection and telegram app

1. Recap

Communications & data flows

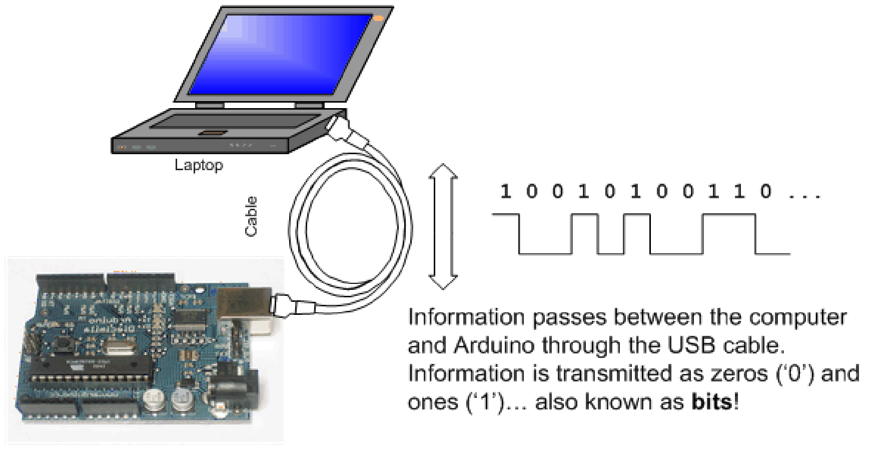


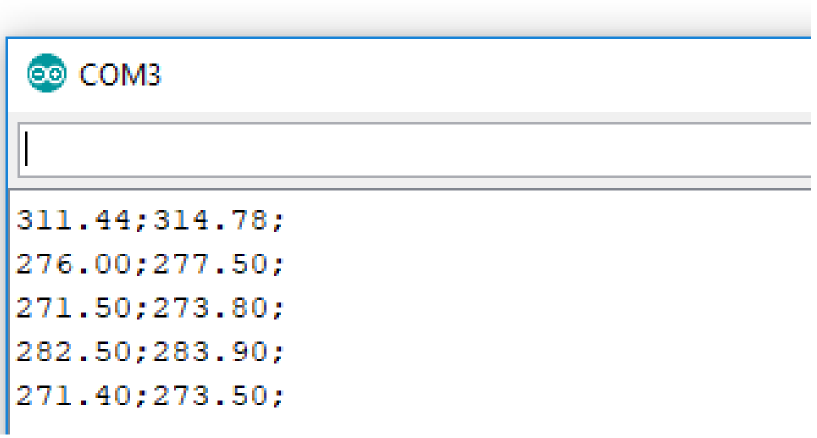
Send data from ESP32 to PC with pyserial

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 1: make sure the serial monitor is closed (exclusive control over serial port taken by pyserial);

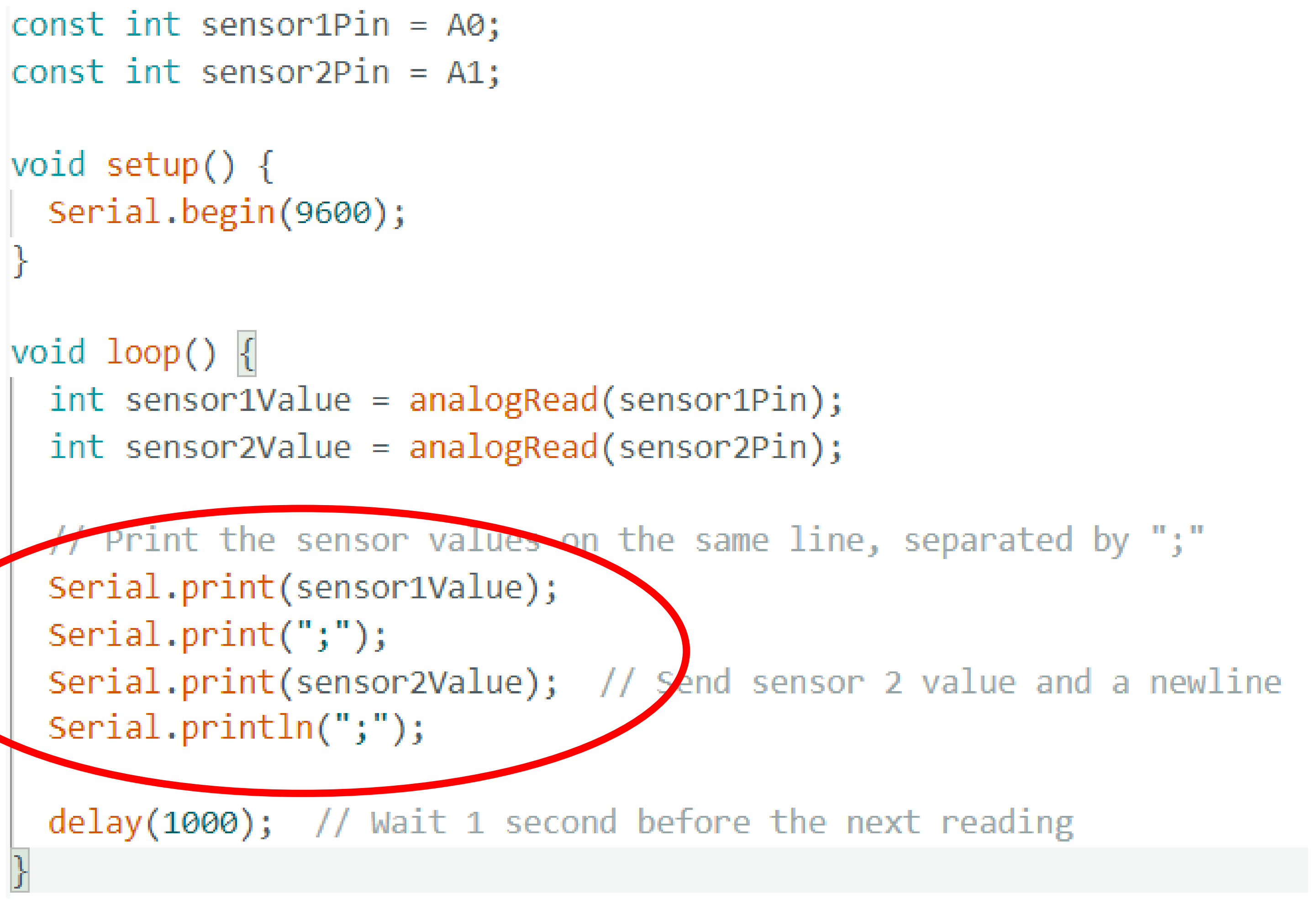
OBS 2: useful to separate values with “;”.


Laptop
Cable
1 0 0 1 0 1 0 0 1 1 0 . . .
Information passes between the computer and Arduino through the USB cable. Information is transmitted as zeros (‘0’) and ones (‘1’)… also known as **bits**!



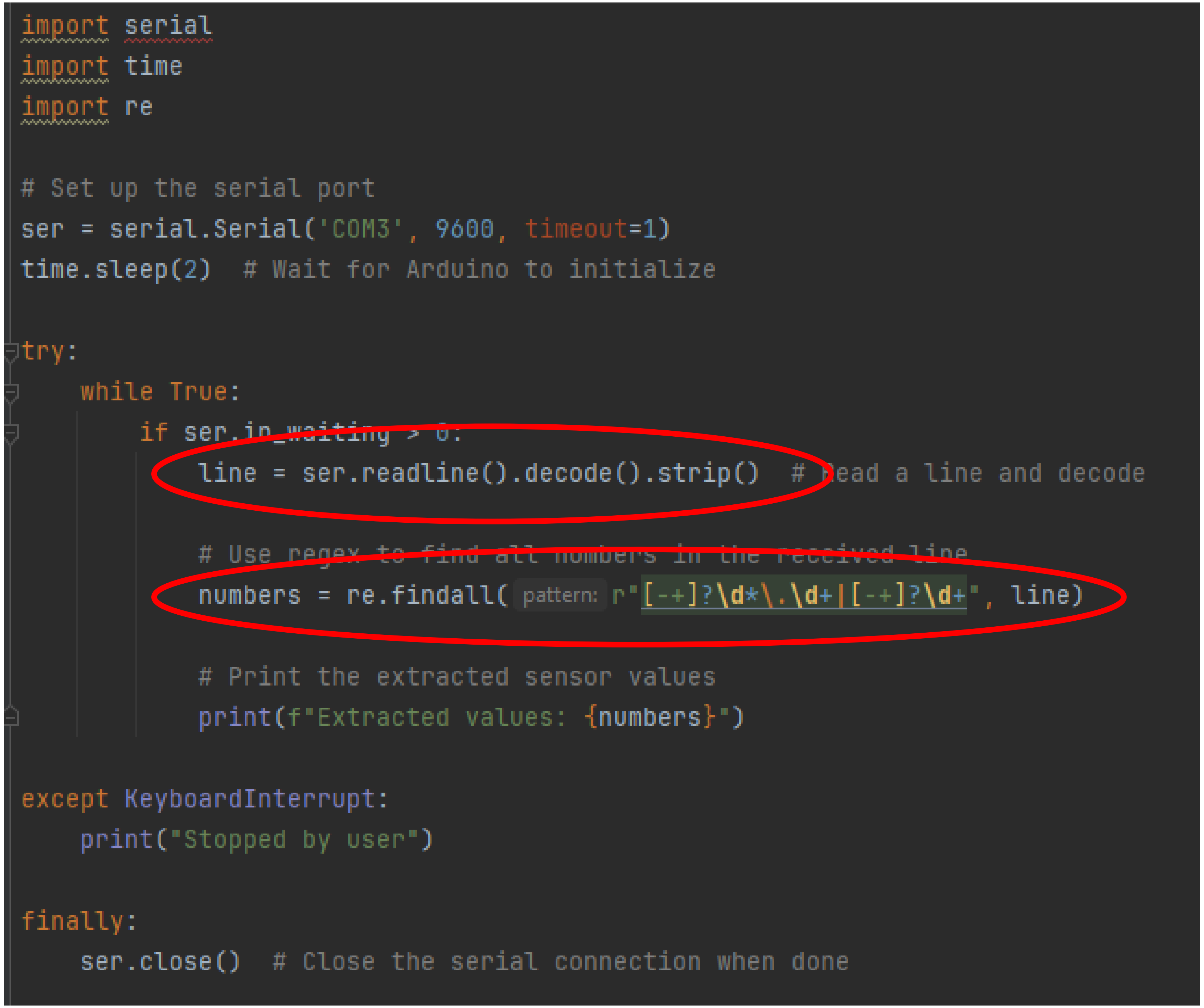
```
311.44;314.78;
276.00;277.50;
271.50;273.80;
282.50;283.90;
271.40;273.50;
```

Arduino: print data on serial



```
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
}
```

Python: receive data



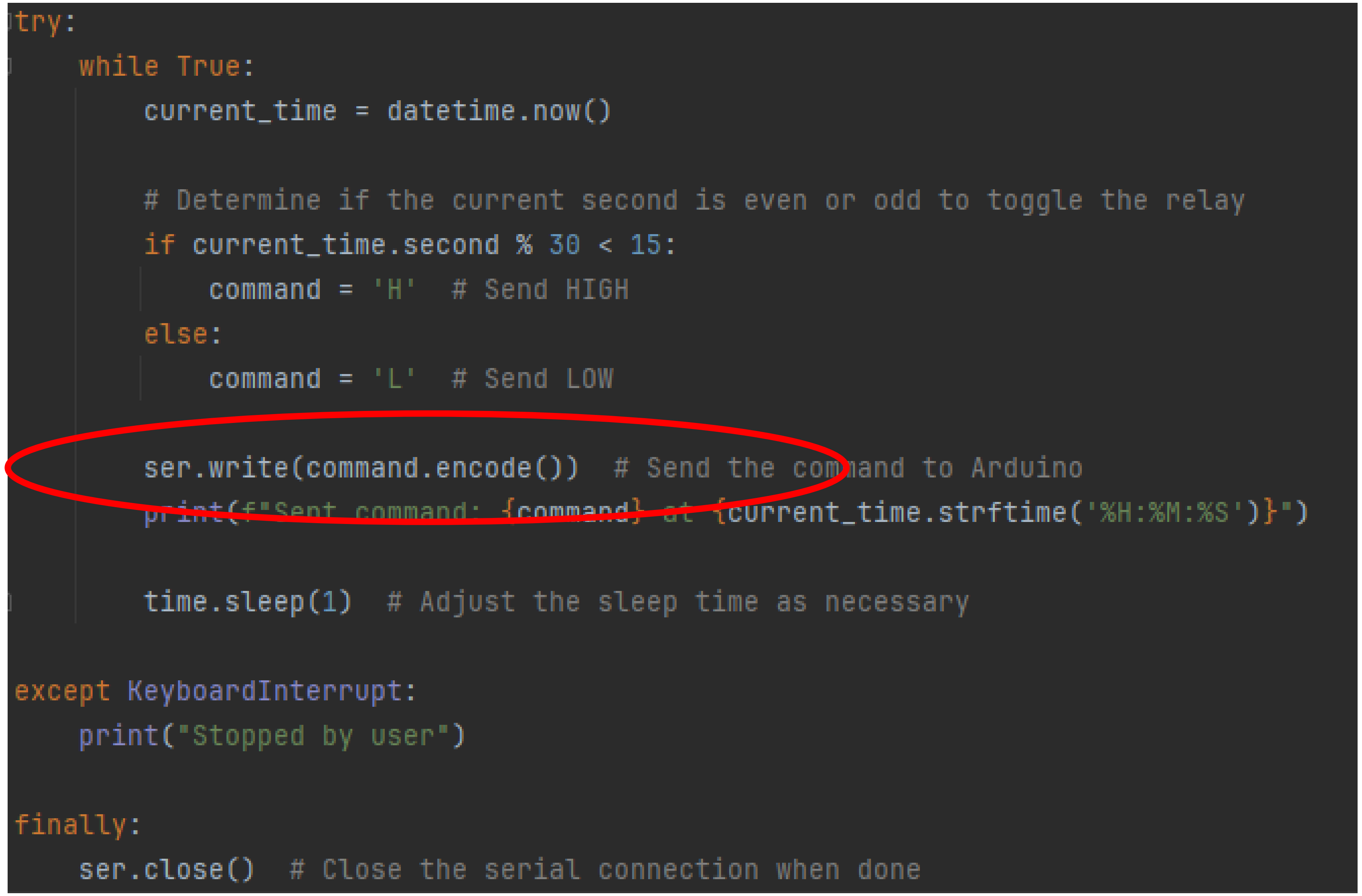
```
import serial
import time
import re

# 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}")

except KeyboardInterrupt:
    print("Stopped by user")

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

2. Send data from PC to ESP32

Send data from PC to ESP32 (example)

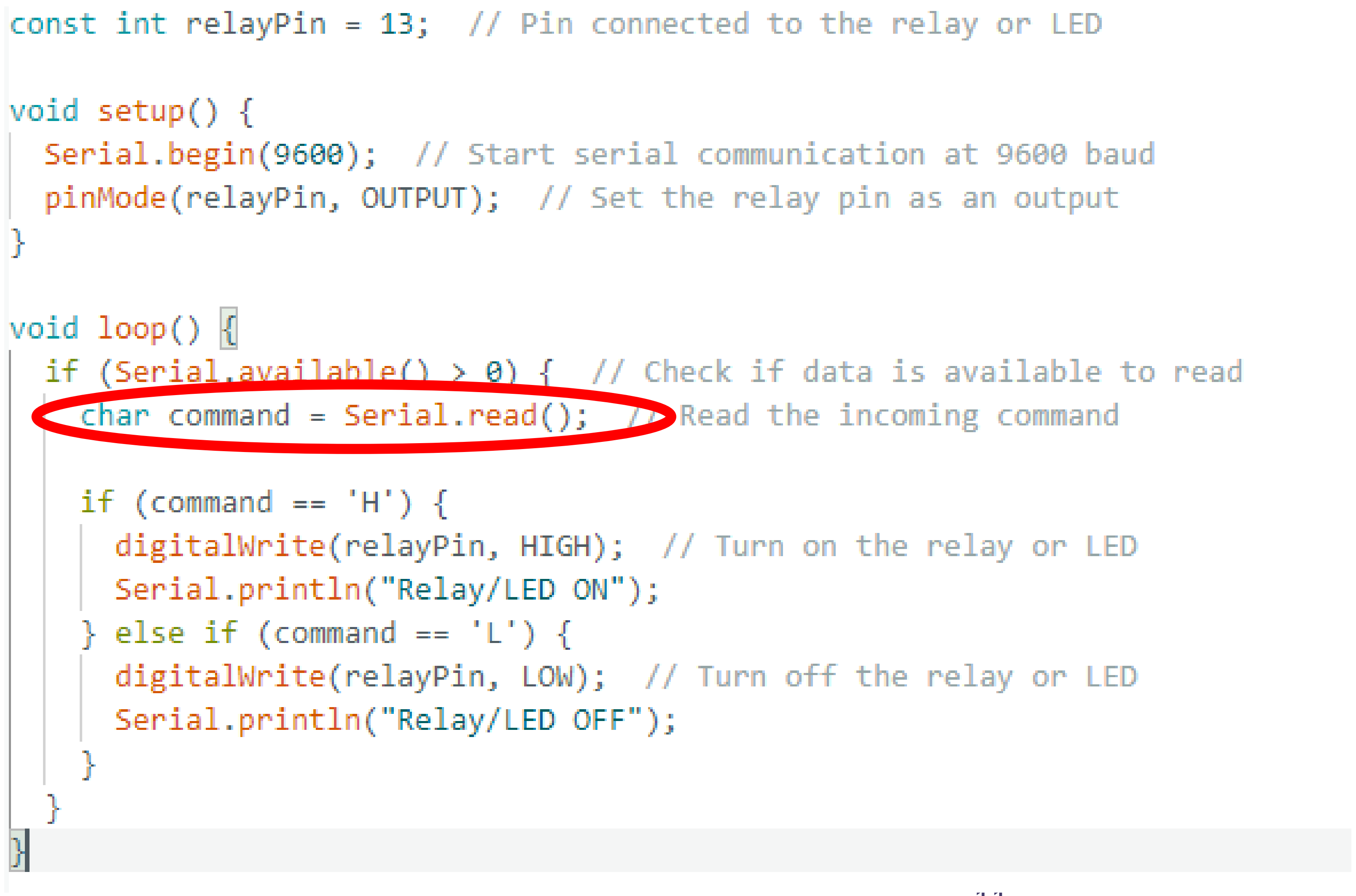
1. Send command from python using pyserial
2. Receive command with Arduino IDE
3. Action signals based on command (i.e. turn ON/OFF)

Python code



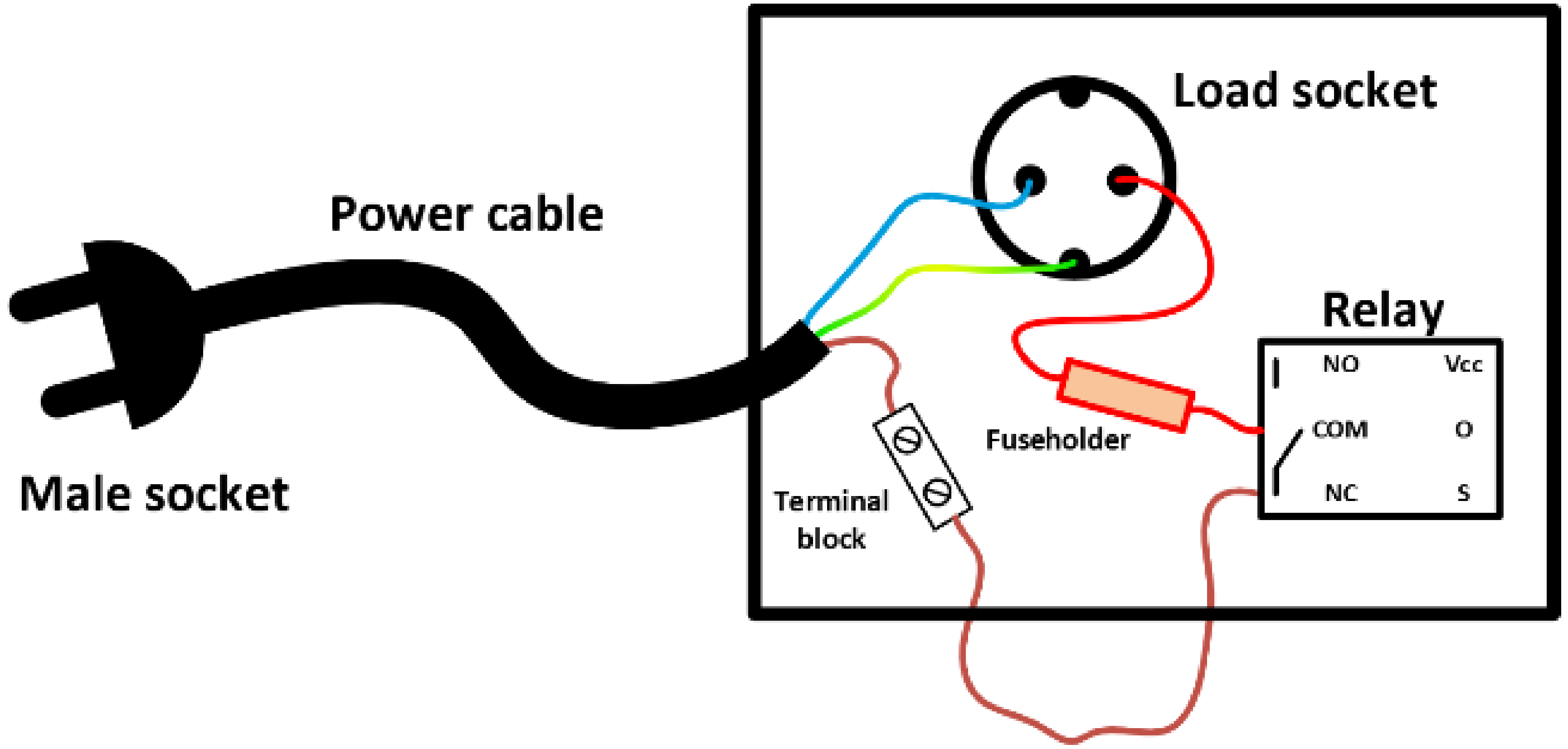
```
try:
    while True:
        current_time = datetime.now()

        # 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



```
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");
    }
  }
}
```

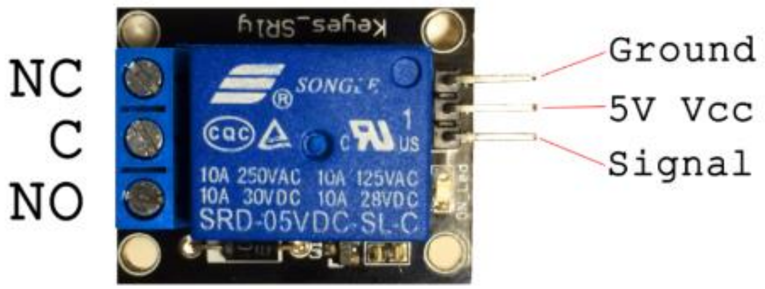
Recap: Electrical scheme



Recap: Relay

A relay has three high voltage terminals (NC, C, and NO) that connect to the device you want to control. The other side has three low voltage pins (Ground, Vcc, and Signal) which connect to the Arduino.

5V Relay Terminals and Pins



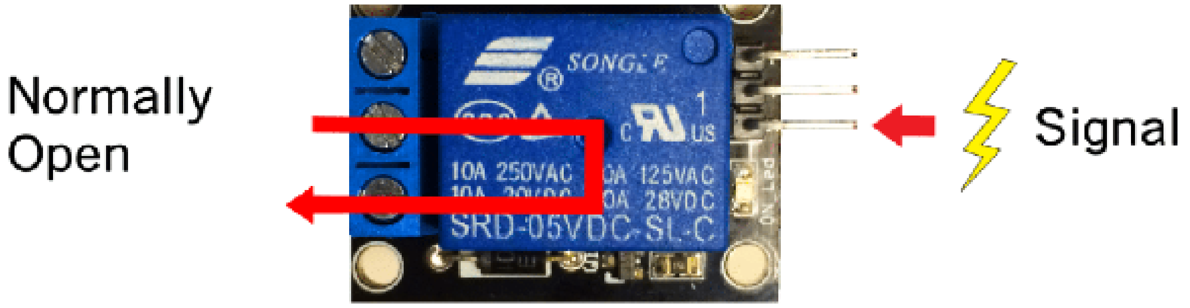
- NC: Normally closed 230V terminal
- NO: Normally open 230V terminal
- C: Common terminal
- Ground: Connects to the ground pin on the Arduino
- 5V Vcc: Connects the Arduino’s 5V pin
- Signal: Carries the trigger signal from the Arduino that activates the relay

Inside the relay is a switch that is connected to an electromagnet. When the relay receives a HIGH signal at the signal pin, the electromagnet becomes charged and moves the contacts of the switch open or closed.

Recap: Relay 2

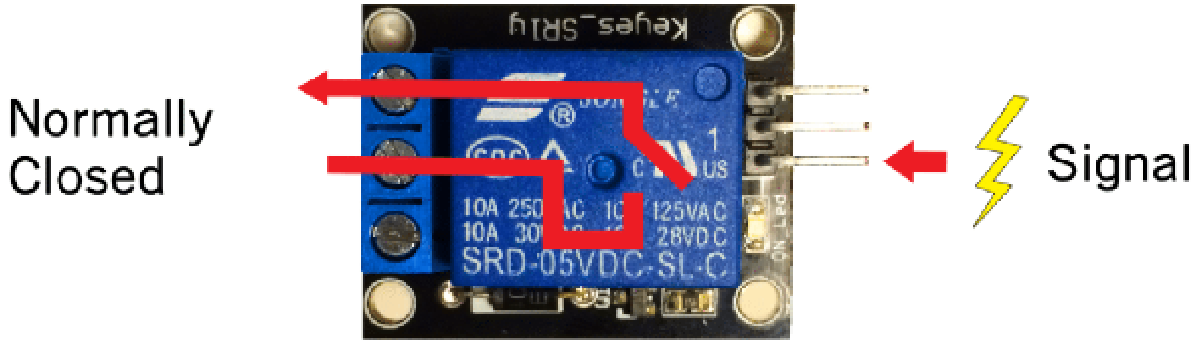
Normally Open

In the normally open configuration, when the relay receives a HIGH signal the 230V switch closes and allows current to flow from the C terminal to the NO terminal. A LOW signal deactivates the relay and stops the current. So if you want the HIGH signal to turn ON the relay, use the normally open terminal:



Normally Closed

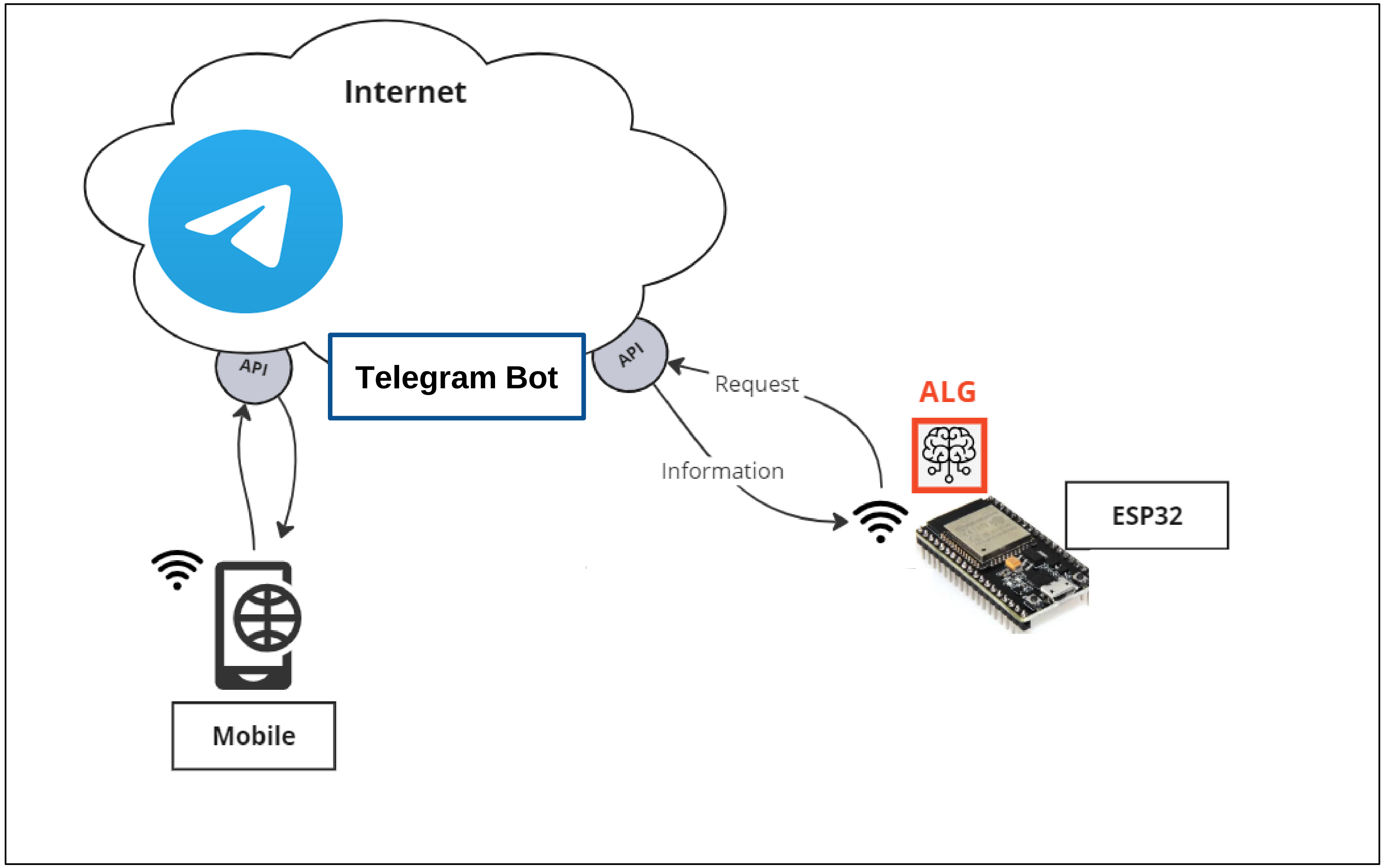
In the normally closed configuration, a HIGH signal opens the switch and interrupts the 230V current. A LOW signal closes the switch and allows current to flow from the C terminal to the NC terminal. Therefore, if you want the HIGH signal to turn OFF the 230V current, use the normally closed terminal:



3. Monitoring with Telegram BOT

Telegram Bot communications scheme

CITCEA UPC
Boosting the future of energy



Telegram and Arduino - introduction

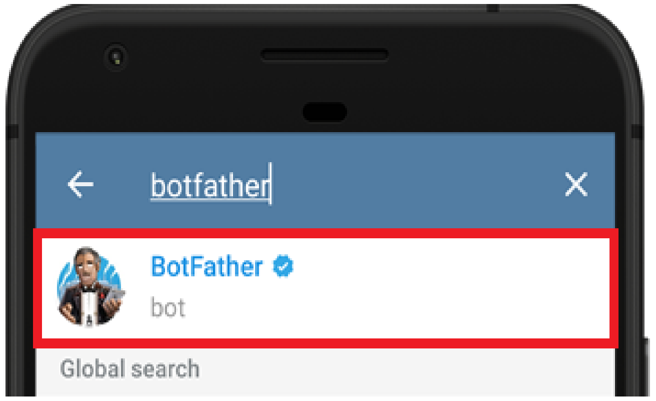
- Use Telegram as a remote controller for hardware
- ESP32 connects to Wi-Fi and communicates securely with Telegram servers
- Commands sent from the phone are received and processed by the ESP32
- ESP32 controls outputs (LED, relay, actuators) and reports sensor readings
- No custom app required — Telegram is the user interface

Monitoring with Telegram Bot

Bots are third-party applications that run inside Telegram. Users can interact with bots by sending them messages, commands, and inline requests. You control your bots using **HTTPS requests to Telegram Bot API**.

FIRST STEP: Creating a Telegram Bot

- Download and install Telegram.
- Inside Telegram, search for “BotFather”.
- Type /start.
- Type /newbot. You will have to define a name and a username for your bot.
- If successfully created, you will receive a link to access the bot and the bot token. **SAVE IT!**



Monitoring with Telegram Bot 2

SECOND STEP: Sending a Message to the Bot

- You must send a message to your Bot from your Telegram account before it can send you messages. Go to the chats tab, type the username of your bot, and start a conversation.

THIRD STEP: Get your Telegram User ID

- Search for “*myidbot*”, start a conversation, and type /getid. Save that ID.

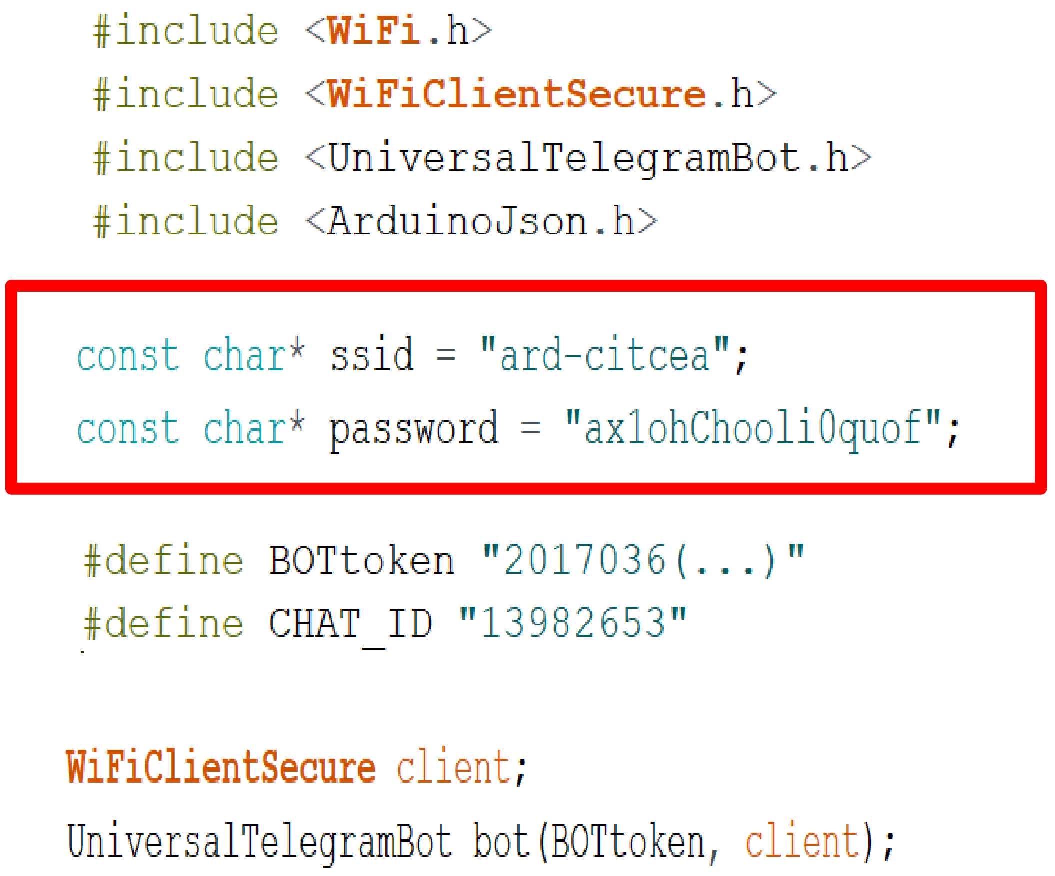
FOURTH STEP: Universal Telegram Bot Library and ArduinoJson Library

- Download the Universal Telegram Bot Library.
- Go to Sketch > Include Library > Add .ZIP Library and add the library you’ve just downloaded.
- Go to Sketch > Include Library > Manage Libraries, search for “*ArduinoJson*” and install it.

Monitoring with Telegram Bot 3

FIFTH STEP (1): Arduino IDE code.

- Include the required libraries.
- Insert your SSID and password in the following variables so that the ESP32 can connect to the internet.
- Insert your Telegram Bot Token and your chat ID.
- Create a new Wi-Fi client with *WifiClientSecure*.
- Create a bot with the token and client defined earlier.



```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

const char* ssid = "ard-citcea";
const char* password = "ax1ohChooli0quof";

#define BOTtoken "2017036(...)"
#define CHAT_ID "13982653"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
```

Monitoring with Telegram Bot 4

FIFTH STEP (2): Arduino IDE code.

- Initialize the Wi-Fi in the *setup()*:
    - Connect to Wi-Fi.
    - Add root certificate for api.telegram.org
    - Check if Wi-Fi is connected and send first message.
    - Apply states logic and send messages when the power load (or other chosen variable) changes from low to high.

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
```

```
client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
```

```
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  bot.sendMessage(CHAT_ID, "Bot started up", "");
```

```
bot.sendMessage(CHAT_ID, "Attention! Large power consumption. Power: " + Power + " W", "");
```

Monitoring with Telegram Bot 5

Check online tutorials!

https://randomnerdtutorials.com/telegram-control-esp32-esp8266-nodemcu-outputs/

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