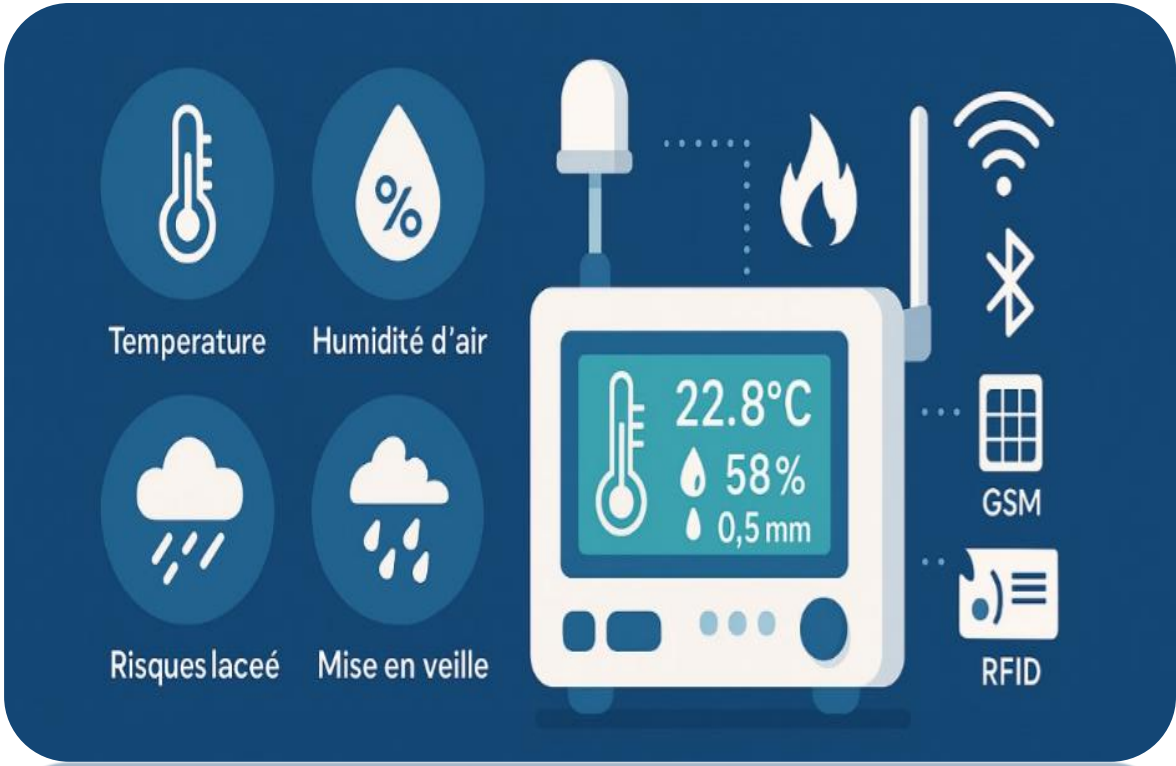
# Station Météo Intelligente Connectée

# Station Météo Intelligente Connectée

- **la température**,
- **L’humidité de l’air**
- **la pluviométrie**
- **Les risques de sécurité**, tels que les **intrusions** ou les **incendies**.

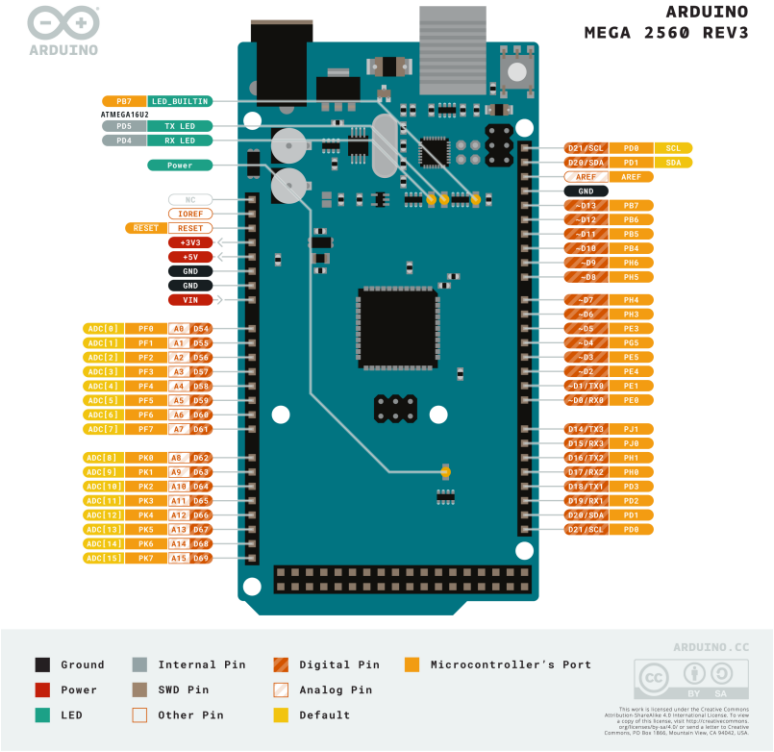



ElectroGrow

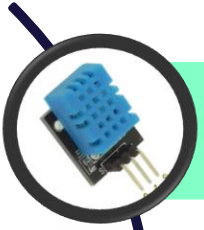
# Composants:

## Arduino Mega2560 R3

- **Microcontrôleur : ATmega2560**
- **Tension de fonctionnement : 5V**
- **Tension d’entrée (recommandée) : 7-12V**
- **Entrées/Sorties numériques :** 54 (dont 15 PWM)
- **Entrées analogiques :** 16
- **Mémoire Flash :** 256 KB (8 KB utilisés par le bootloader)
- **SRAM :** 8 KB
- **EEPROM :** 4 KB



# Capteur



**Capteur DHT11 (température et humidité)**

- ✓ **Tension de fonctionnement :** 3.3 à 5V
- ✓ **Plage de mesure (température) :** 0 à 50 °C ±2°C
- ✓ **Plage de mesure (humidité) :** 20 à 90% RH ±5%
- ✓ **Fréquence d'échantillonnage :** 1 Hz (1 lecture par seconde)
- ✓ **Sortie :** Numérique

**Capteur de pluie**

- ✓ **Tension de fonctionnement** : 3.3 à 5V
- ✓ **Sortie** : Analogique et numérique
- ✓ **Principe de fonctionnement** : Détection de l'humidité sur une plaque PC

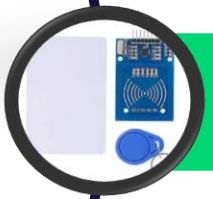
**Capteur de présence PIR**

- ✓ **Tension de fonctionnement** : 4.5 à 20V
- ✓ **Plage de détection** : 3 à 7 mètres (typique)
- ✓ **Angle de détection** : ~120°
- ✓ **Sortie** : Numérique (HIGH/LOW)
- ✓ **Temps de déclenchement** : ajustable

# Capteur

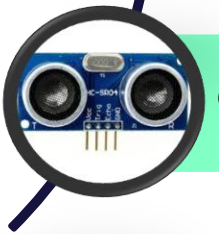
**Écran LCD I2C 16x2**

- ✓ **Affichage** : 2 lignes de 16 caractères
- ✓ **Interface** : I2C (2 fils : SDA et SCL)
- ✓ **Tension** : 5V
- ✓ **Adresse I2C par défaut** : 0x27 ou 0x3F (selon le module)
- ✓ **Avantage** : Moins de fils comparé à un LCD parallèle



**Module RFID MFRC522**

- ✓ **Tension de fonctionnement** : 3.3V (ne pas connecter en 5V directement)
- ✓ **Fréquence de fonctionnement :** 13.56 MHz
- ✓ **Communication** : SPI
- ✓ **Distance de lecture** : 2 à 5 cm



**Capteur ultrason HC-SR04**

- ✓ **Tension de fonctionnement** : 5V
- ✓ **Plage de mesure** : 2 cm à 400 cm
- ✓ **Précision :** ±3 mm
- ✓ **Fréquence** : 40 kHz

# Outils logiciels

**Arduino IDE**

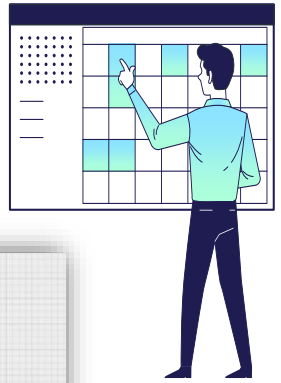
Logiciel utilisé pour programmer la carte ESP32 avec un langage simple basé sur C/C++.

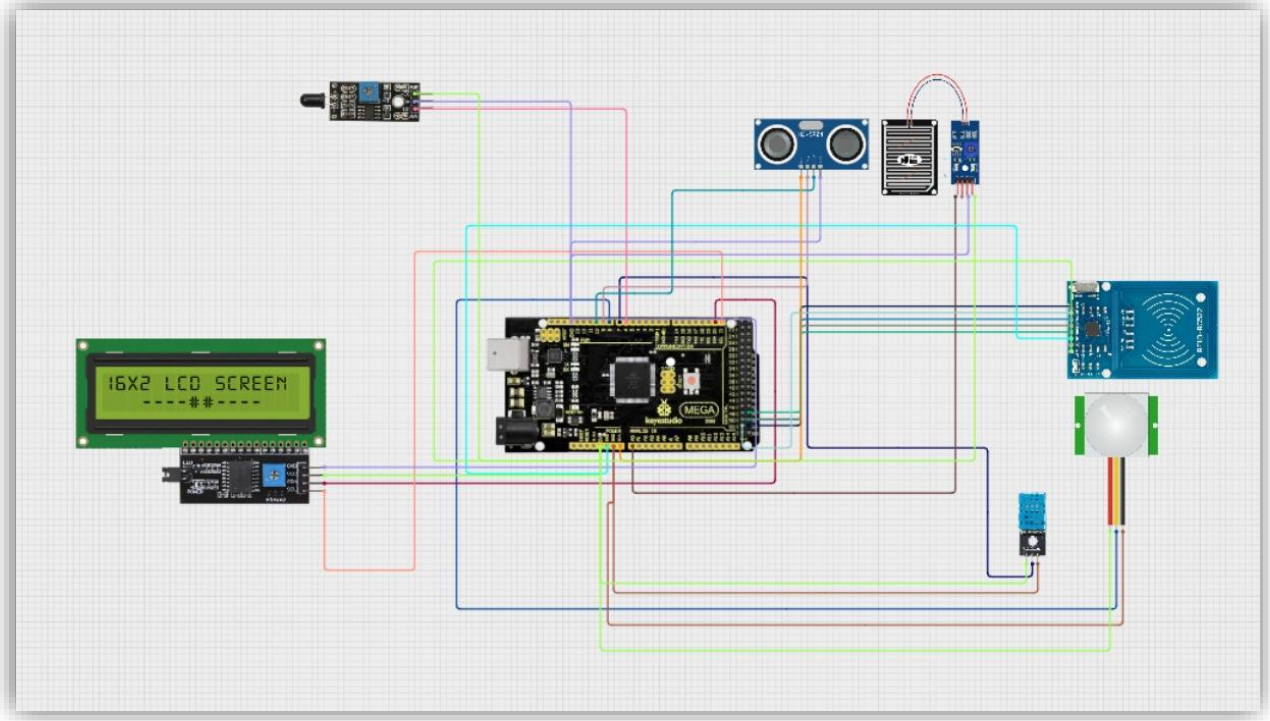
# Code source (extraits importants)

```
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include <EEPROM.h>

#define DHTPIN 2
#define DHTTYPE DHT11
#define RAINPIN A0
#define PIRPIN 3
#define FLAME_PIN A1
#define SS_PIN 53
#define RST_PIN 9
#define TRIG_PIN 6
#define ECHO_PIN 7


DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 rfid(SS_PIN, RST_PIN);

volatile bool intrusion = false;
int intrusionCount = 0;
bool authorized = false;
unsigned long authorizedTime = 0;
const unsigned long AUTH_TIMEOUT = 60000;  // 60s
byte authorizedUIDs[2][4] = { { 0x04, 0x75, 0x42, 0x92 }, { 0x04, 0x33, 0x91, 0x15
} };
unsigned long lastDisplayUpdate = 0;
int displayState = 0;
unsigned long lastActivity = 0;
bool lowPowerMode = false:
```

```
void updateDisplay(float temp, float hum, int rain, float distance)
  if (millis() - lastDisplayUpdate < 3000) return;
  lcd.clear();
  switch (displayState) {
    case 0:
      lcd.setCursor(0, 0);
      lcd.print("T:");
      lcd.print(temp);
      lcd.print("C H:");
      lcd.print(hum);
      lcd.print("%");
      displayState = 1;
      break;
    case 1:
      lcd.setCursor(0, 0);
      lcd.print("Pluie:");
      lcd.print(rain < 500 ? "Oui" : "Non");
      lcd.setCursor(0, 1);
      lcd.print("Dist:");
      lcd.print((int)distance);
      lcd.print("cm");
      displayState = 2;
      break;
    case 2:
      lcd.setCursor(0, 0);
      lcd.print("Intrusions:");
      lcd.print(intrusionCount);
      displayState = 0;
      break;
  }
  lastDisplayUpdate = millis();
}
```

# Schéma de câblage du système Station Météo Intelligente






# Réalisation