ESP32 mikrovezérlők programozása CircuitPython környezetben

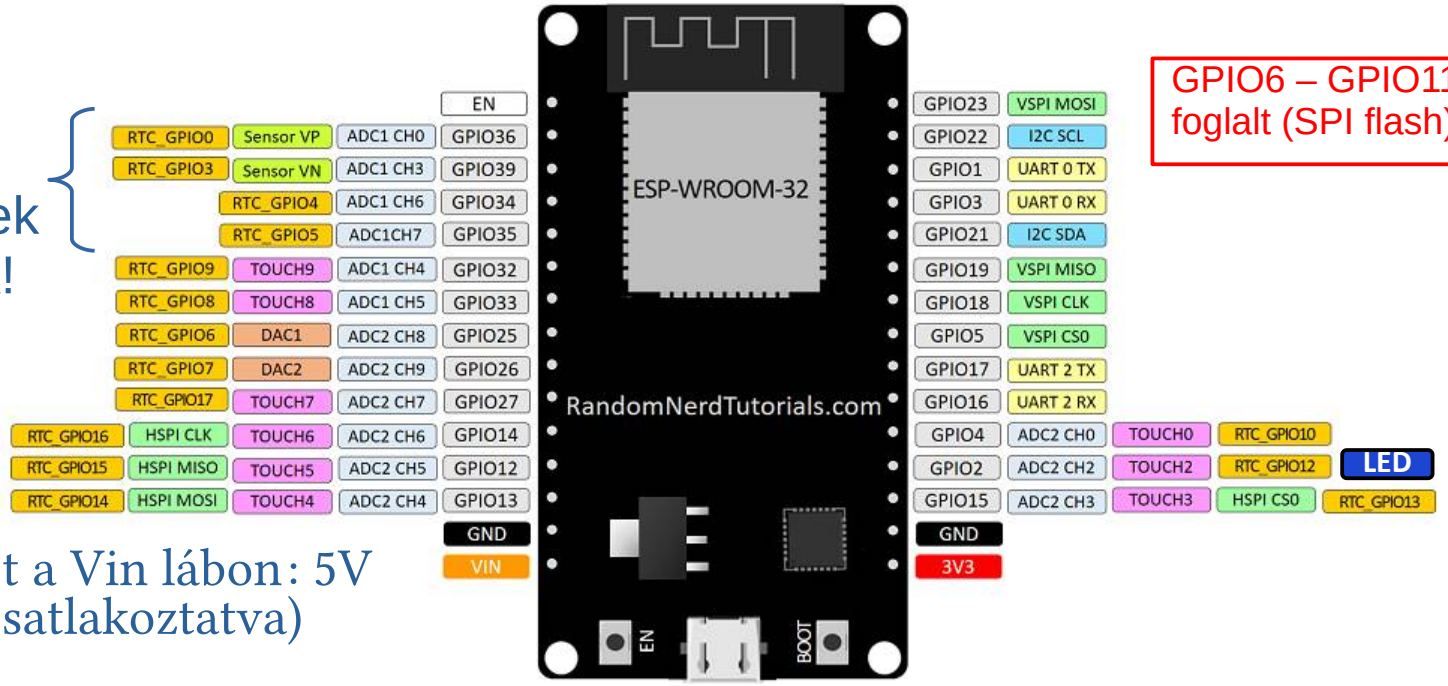
3. Az AHT10 szenzor használata

Felhasznált és ajánlott irodalom

- **Python:**
  - Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!
- **CircuitPython:**
  - Adafruit: https://circuitpython.org/downloads
  - Learn Adafruit: Welcome to CircuitPython
  - Learn Adafruit: CircuitPython Essentials
  - Adafruit: Adafruit CircuitPython API Reference
  - Adafruit: github.com/adafruit/Adafruit CircuitPython Bundle
  - Carter Nelson: CircuitPython on ESP32 Quick Start
  - Anne Barela: Networking in CircuitPython
- **Online eszközök és támogatás:**
  - Adafruit: Adafruit Web Serial ESPTool
  - Adafruit: CircuitPython Code Editor

Az ESP32 Devkit-1 (DOIT) kártya kivezetései

- A Doit **ESP32 Devkit-1** kártya egy **ESP WROOM-32** modult (ESP32 + 4MB flash) és az alapkártyán egy **CP2102** USB-UART átalakítót, egy 3,3 V-os stabilizátort, egy **Reset** és egy **Boot** nyomógombot tartalmaz



- Tápellátás bemenet a Vin lábon: 5V (ha nincs USB-re csatlakoztatva)

Forrás:
randomnerdtutorials.com/getting-started-with-esp32/

Pin 30 version

Az ESP32 C3 Super Mini kártya

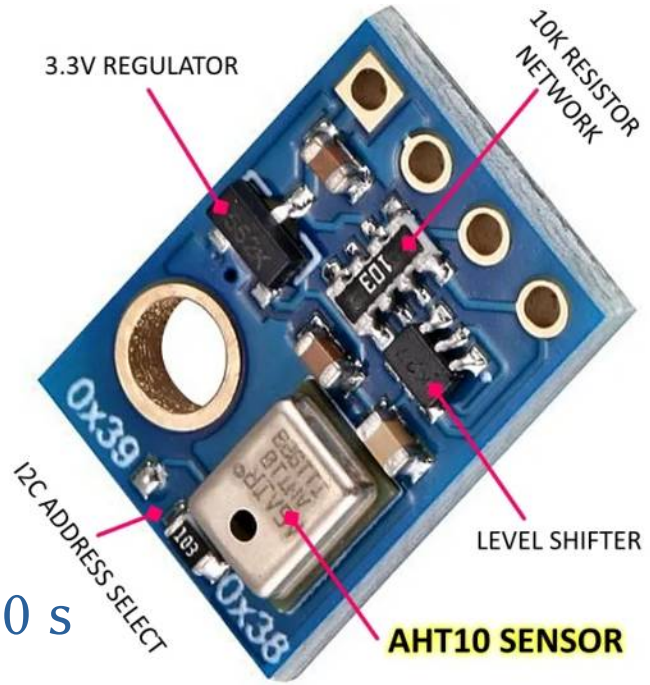
- Az **ESP32-C3 Super Mini** egy kompakt fejlesztői kártya
- Mikrovezérlő: **ESP32-C3**, 32 bites **RISC-V** architektúra
- Órajel: Akár 160 MHz
- Memória: 400 KB SRAM, 384 KB ROM
- Tároló: 4 MB Flash memória
- Wi-Fi: 802.11 b/g/n (2.4 GHz)
- Bluetooth: Bluetooth 5.0 LE
- GPIO: Több mint 20 általános célú bemenet/kimenet (GPIO), melyek közül 13 van kivezetve
- Interfészek: SPI, I2C, UART, ADC, DAC, PWM

Az ESP32 C3 Super Mini kártya kivezetései

Megjegyzés: Az A6 analóg bemenet (ADC2) nem használható, ha a WiFi használatban van!

Az AHT10 szenzor

- Az **AHT10** egy integrált hőmérséklet- és páratartalom-érzékelő, amelyet a Guangzhou Aosong Electronic Co., Ltd. gyárt



- **Főbb jellemzők:**
  - Felbontás: hőmérséklet: 0,01℃
    Páratartalom: 0,024% RH
  - Pontosság: ±3% relatív páratartalom (RH)
    és ±0,5℃ hőmérséklet
  - Működési tartomány: 0-100% RH
    és –40℃-tól 85℃-ig.
  - Válaszidő: Páratartalom 8 s, hőmérséklet 5-30 s
  - Interfész: I²C (alapértelmezett cím: 0x38)
  - I2C sebesség: 100 kHz, vagy 400 kHz
  - Tápfeszültség: 1,8V-tól 3,6V-ig

AHT10 parancsok

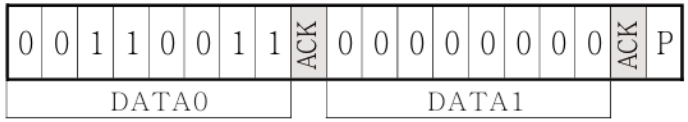
Command	Definition	Code
Initialization	Keep main engine	1110'0001
Trigger Measurement	Keep main engine	1010'1100
Soft reset		1011'1010

Table 9 Basic Commands

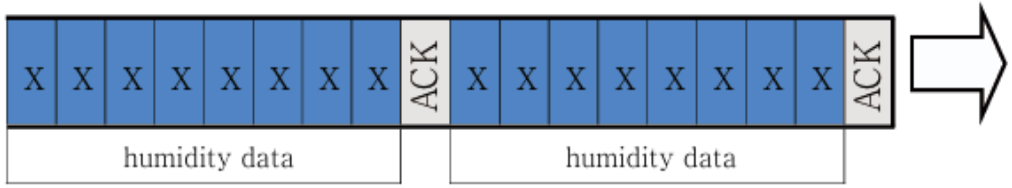
Bit	Definition	Description
Bit[7]	(Busyindication)	1 — Busy in measurement 0 — Free in dormant state
Bit [6:5]	(ModeStatus)	00 in NOR mode 01 in CYC mode 1x in CMD mode
Bit [4]	Remained	Remained
Bit [3]	CAL Enable	1--calibrated 0--uncalibrated
Bit [2：0]	Remained	Remained

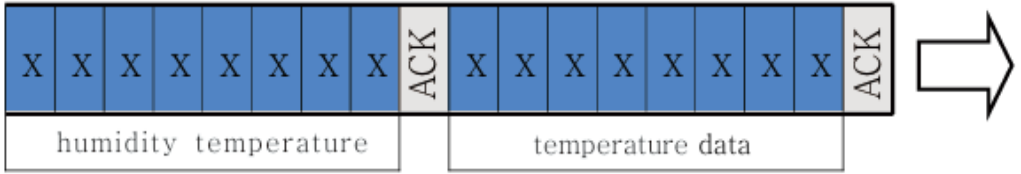
Table 10. State bit description.

Trigger measurement data



Read temperature and humidity data





AHT10 támogatói könyvtárak

- **CircuitPython**
  - **Adafruit_CircuitPython_AHTx0**
- **Tagfüggvények/tulajdonságok:**
  - **reset()** - soft reset
  - **calibrate()** - önkalibrálást indít
  - **status()** - státuszbájt kiolvasása
  - **temperature()** - hőmérséklet
  - **relative_humidity()** - páratartalom
- **Dokumentáció:**

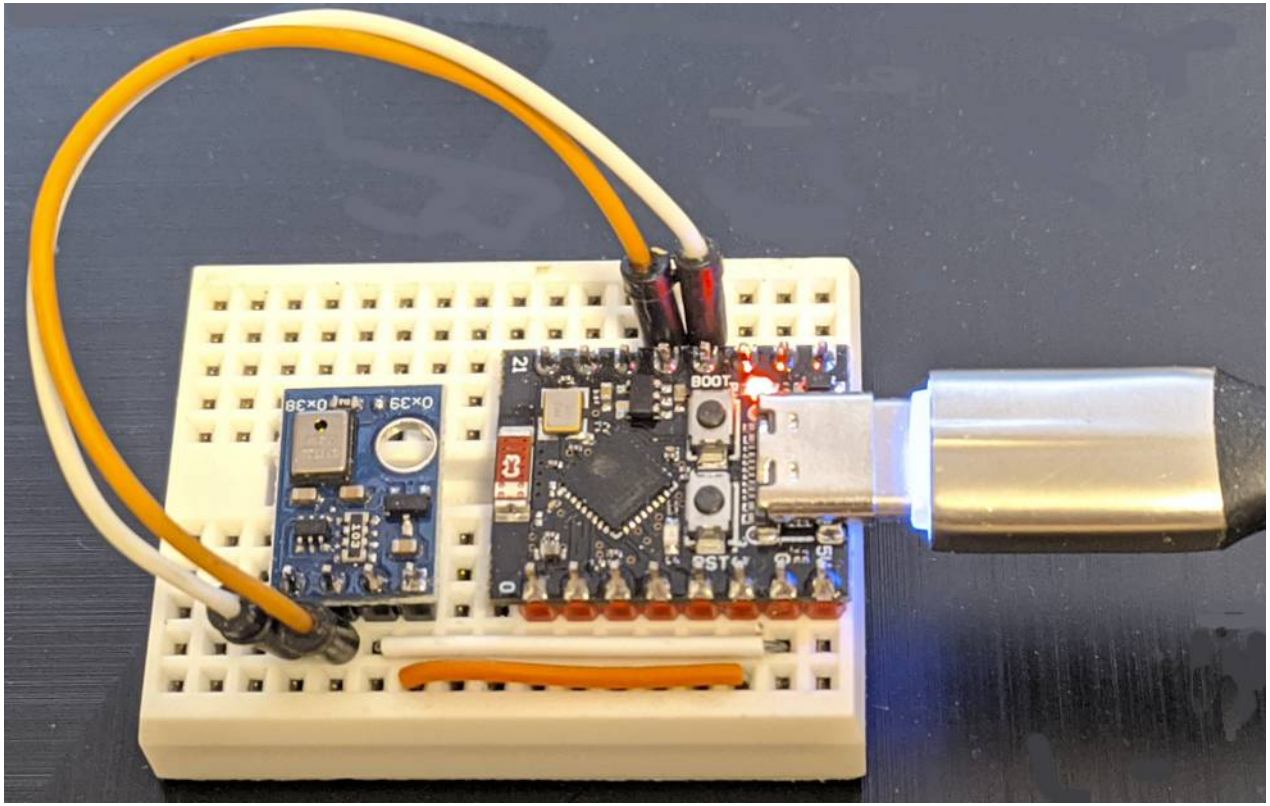
  docs.circuitpython.org/projects/ahtx0

- **Arduino**
  - **Adafruit_AHTx0**
- **Tagfüggvények/tulajdonságok:**
  - **begin()** - inicializálás
  - **getEvent()** - kiolvasást indít
  - **getStatus()** - státuszbájt kiolvasása
- **Dokumentáció:**

  Adafruit AHT10 Library

Bekötési vázlat

- A Vin bemenetre 3,3 V-ot kötöttünk, SCL ⟶ GPIO9, SDA ⟶ GPIO8



ahtx0_simpletest.py

- **CircuitPython**-ban így olvashatjuk ki a szenzort:

```
"""
Basic `AHTx0` example test
"""
import time
import board
import adafruit_ahtx0

# Create sensor object, communicating over the board's
default I2C bus
i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = adafruit_ahtx0.AHTx0(i2c)
sensor.calibrate()

while True:
    print("Temperature: %0.1f C" % sensor.temperature)
    print("Humidity: %0.1f %%" % sensor.relative_humidity)
    time.sleep(2)
```

code.py output:
Temperature: 24.0 C
Humidity: 46.6 %
Temperature: 24.0 C
Humidity: 46.6 %
Temperature: 24.0 C
Humidity: 46.6 %
Temperature: 24.0 C
Humidity: 46.6 %
Temperature: 24.0 C
Humidity: 46.6 %
Temperature: 24.0 C
Humidity: 46.5 %
Temperature: 24.0 C
Humidity: 46.5 %
Temperature: 24.0 C
Humidity: 46.6 %

aht10_simpledemo.ino – 2/1. oldal

- Arduino-ban, csupán a beépített I2C könyvtár használatával így inicializálhatjuk és olvashatjuk ki a szenzort:

```
#include <Wire.h>
#define AHT10_ADDRESS 0x38

void setup() {
  Serial.begin(115200);
  // I2C kommunikáció inicializálása az új GPIO lábakkal
  Wire.begin(8, 9); // SDA = GPIO 8, SCL = GPIO 9
  delay(5000);
  Serial.println("AHT10 szenzor inicializálás...");
  // Automatikus kalibrálás
  Wire.beginTransmission(AHT10_ADDRESS);
  Wire.write(0xE1); // Inicializálás parancs (kalibrálás)
  Wire.write(0x08); // Paraméter bájt 1
  Wire.write(0x00); // Paraméter bájt 2
  Wire.endTransmission();
  delay(20); // Várakozás a kalibrálás befejezésére
  Serial.println("AHT10 szenzor inicializálva és kalibrálva.");
}
```

aht10_simpledemo.ino – 2/2. oldal

```ino
void loop() {
  Serial.println("Új ciklus");
  float temperature, humidity;
  readAHT10(&temperature, &humidity);

  Serial.print("Hőmérséklet: ");
  Serial.print(temperature);
  Serial.println(" °C");

  Serial.print("Páratartalom: ");
  Serial.print(humidity);
  Serial.println(" %");

  delay(5000);
}
```

```ino
void readAHT10(float *temperature, float *humidity) {
  Wire.beginTransmission(AHT10_ADDRESS);
  Wire.write(0xAC); // Mérés indítása parancs
  Wire.write(0x33); // Paraméter bájt 1
  Wire.write(0x00); // Paraméter bájt 2
  Wire.endTransmission();
  delay(100); // Várakozás a mérés befejezéséhez
  Wire.requestFrom(AHT10_ADDRESS, 6);
  if (Wire.available() == 6) {
    uint8_t data[6];
    for (int i = 0; i < 6; i++) {data[i]=Wire.read();}
    uint32_t rawHumidity = ((uint32_t)data[1] << 12) |
    ((uint32_t)data[2] << 4) | (data[3] >> 4);
    uint32_t rawTemperature = ((uint32_t)data[3] &
    0x0F) << 16 | ((uint32_t)data[4] << 8) | data[5];
    *humidity = ((float)rawHumidity/1048576.0) * 100.0;
    *temperature = ((float)rawTemperature / 1048576.0)
    * 200.0 - 50.0;
  }
}
```

adafruit_aht_test.ino

Egy másik Arduino példaprogram az Adafruit **AHTx0** könyvtár felhasználásával

```
#include <Adafruit_AHTX0.h>
Adafruit_AHTX0 aht;

void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit AHT10/AHT20 demo!");
  Wire.begin(8, 9);       // SDA = GPIO8, SCL = GPIO9
  if (! aht.begin()) {
    Serial.println("Could not find AHT? Check wiring");
    while (1) delay(10);
  }
  Serial.println("AHT10 or AHT20 found");
}

void loop() {
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);
  Serial.print("Temperature: "); Serial.print(temp.temperature);
  Serial.println(" degrees C");
  Serial.print("Humidity: ");
  Serial.print(humidity.relative_humidity); Serial.println("% rH");
  delay(5000);
}
```

```
AHT10 or AHT20 found
Temperature: 24.63 degrees C
Humidity: 44.41% rH
Temperature: 24.63 degrees C
Humidity: 44.84% rH
Temperature: 24.63 degrees C
Humidity: 44.94% rH
Temperature: 24.62 degrees C
Humidity: 44.84% rH
Temperature: 24.63 degrees C
Humidity: 44.83% rH
Temperature: 24.61 degrees C
Humidity: 44.73% rH
Temperature: 24.60 degrees C
Humidity: 44.74% rH
Temperature: 24.60 degrees C
Humidity: 44.56% rH
Temperature: 24.63 degrees C
Humidity: 45.28% rH
```