Workshop VI

WiFi & AHT20

© 2024 Open Project Space, Institute of Electrical and Electronics Engineers at the University of California, Irvine. All Rights Reserved.

SECTION I

WiFi Library

MAC Addresses

- MAC address: Media Access Control address
- Unique identifier for devices on a network

- Transmitter needs to know receiving address
  - However, the receiver does not need to know the transmitter’s address
  - Similar to sending mail to your friend
    - You need to know where to send it to but your friend doesn’t need to know your address beforehand.

Getting your MAC address

Copy and flash this code to your ESP32. The Serial Monitor will show its MAC address.

Write it down somewhere.

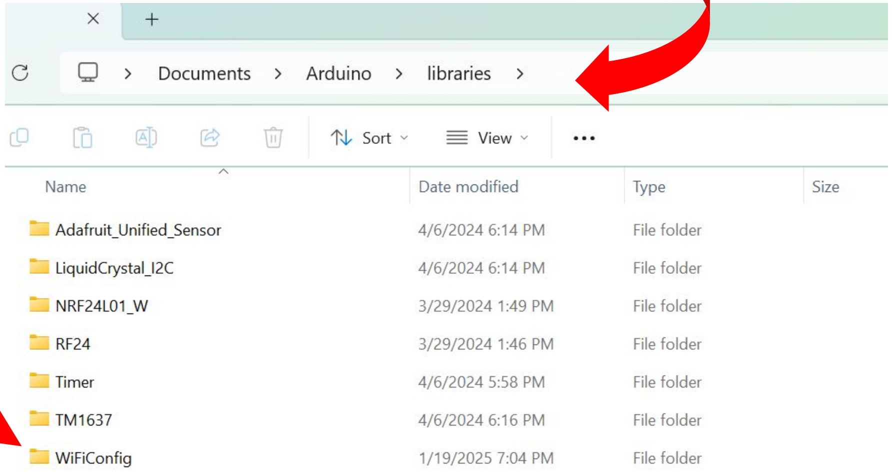
This ESP32 will be your **receiver**.

The other will be your **transmitter**.

```
#include <WiFi.h>
#include <esp_wifi.h>

void readMacAddress(){
  uint8_t baseMac[6];
  esp_err_t ret = esp_wifi_get_mac(WIFI_IF_STA, baseMac);
  if (ret == ESP_OK) {
    Serial.printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                  baseMac[0], baseMac[1], baseMac[2],
                  baseMac[3], baseMac[4], baseMac[5]);
  } else {
    Serial.println("Failed to read MAC address");
  }
}

void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.STA.begin();
  Serial.print("[DEFAULT] ESP32 Board MAC Address: ");
  readMacAddress();
}

void loop(){
  Serial.print("[DEFAULT] ESP32 Board MAC Address: ");
  readMacAddress();
}
```

WiFi Library Installation

- Download the library here
- Extract the zip folder to your Arduino libraries folder
  - Example: C:\Users\username\Documents\Arduino\libraries



Name	Date modified	Type	Size
Adafruit_Unified_Sensor	4/6/2024 6:14 PM	File folder	
LiquidCrystal_I2C	4/6/2024 6:14 PM	File folder	
NRF24L01_W	3/29/2024 1:49 PM	File folder	
RF24	3/29/2024 1:46 PM	File folder	
Timer	4/6/2024 5:58 PM	File folder	
TM1637	4/6/2024 6:16 PM	File folder	
WiFiConfig	1/19/2025 7:04 PM	File folder	

- You should see a folder titled "WifiConfig" in your Arduino libraries folder now

- Include this library in your code with `#include <WiFiConfig.h>`

MAC Address usage in code

Your transmitter needs to send data to the address you just found.

- Open a new sketch for your transmitter ESP32.
- Store as a uint8_t array of hexadecimals, as shown in the example below

```
// Receiver's address: 34:b7:da:f6:3e:78
uint8_t receiverAddress[] = {0x 34, 0xB7, 0xDA, 0xF6, 0x3E, 0x78};
```

- Notes:
  - uint8_t just means unsigned 8-bit integer (0-255 bits)
  - Prefix “0x” is required for hexadecimal values

Setup functions

Call these functions in void setup()

- `wifi_setup()`
  - Puts the ESP32 in WiFi Mode, activating the WiFi module
- `peer_setup(uint8_t receiverAddress[])`
  - Establishes the receiverAddress as the address to send information to
  - receiverAddress is the uint8_t array you created earlier
- wifi_setup() should be called before peer_setup() to avoid any inconsistent program behavior

```
void setup() {
  // Put ESP32 in WiFi Mode
  wifi_setup();

  // Establish the peer
  peer_setup(receiverAddress);
}
```

Weather Report structure

- We will send values to another receiver as a **structure** called **WeatherReport**

- WeatherReport consists of three uint8_t members
  - temperature
  - humidity
  - light

```
struct WeatherReport {
  uint8_t temperature;
  uint8_t humidity;
  uint8_t light;
};
```

Note: you don't have to type this structure code anywhere. It is already in the library.

- How to create a Weather Report and set values

```
// Create a WeatherReport called report
WeatherReport report;
// Set the values of the report
report.temperature = 15;
report.humidity = 60;
report.light = 55;
```

Sending reports

- `send_report(uint8_t receiverAddress[], WeatherReport report)`
  - Sends a `WeatherReport` to the receiver using its address

```
void loop() {
  // Send message
  send_report(receiverAddress, report);
  delay(500);
}
```

Receiver Code

- Our receiver needs to store the information it receives so it can display the Weather Report values on the serial monitor
- `set_report_receiver(WeatherReport *report)`
  - Configures the ESP32 to receive WeatherReports and store their values in its own WeatherReport structure (at its memory location)
  - **Automatically updates its own report** whenever data is received
  - Note: &report is the memory address of the report.

```
// For storing received reports
WeatherReport report;

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  // Put ESP32 in WiFi mode
  wifi_setup();
  // Configure WeatherReport reception
  set_report_receiver(&report);
}
```

WiFi Module Communication

Write a program that satisfies the following requirements:

- You must write a program in which the **transmitter ESP32 board sends a Weather Report containing the following values:**
  - Temperature: 67
  - Humidity: 52
  - Light: 34
- The receiving ESP32 should **print this information to the serial monitor**.
- Power the transmitter ESP32 with a 9V battery, and power the receiver with USB.
  - (connect red wire to VIN and black wire to GND)
- The program must **run in an infinite loop**.

WiFi Module Communication Code

Transmitter

```
#include <WiFiConfig.h>

// REPLACE WITH YOUR OWN ADDRESS
// Example: 34:b7:da:f6:3e:78
uint8_t receiverAddress[] = {0x34, 0xB7, 0xDA, 0xF6, 0x3E,
0x78};

// Create a WeatherReport called report
WeatherReport report;

void setup() {
  // Put ESP32 in WiFi Mode
  wifi_setup();

  // Establish the other ESP32 as the peer
  peer_setup(receiverAddress);
}

void loop() {
  // Set the values of the report
  report.temperature = 67;
  report.humidity = 52;
  report.light = 34;
  // Send message
  send_report(receiverAddress, report);
  delay(500);
}
```

Receiver

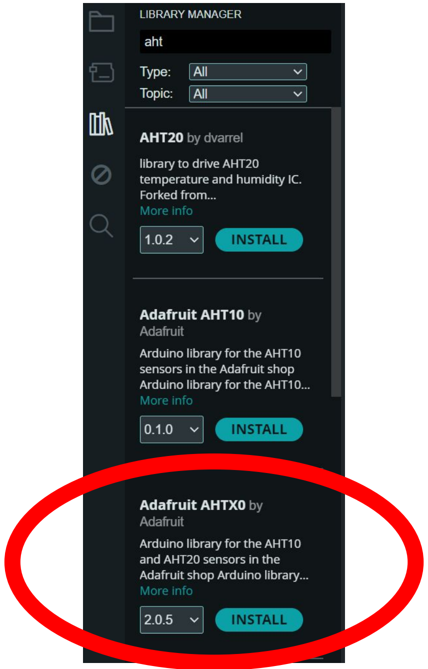
```
#include <WiFiConfig.h>

// Initialize a WeatherReport to store the values
WeatherReport report;

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  // Put ESP32 in WiFi mode
  wifi_setup();
  // Configure this ESP32 to receive WeatherReports
  set_report_receiver(&report);
}

void loop() {
  Serial.print("TEMP: ");
  Serial.println(report.temperature);
  Serial.print("HUM: ");
  Serial.println(report.humidity);
  Serial.print("LIGHT: ");
  Serial.println(report.light);

  delay(500);
}
```

SECTION II

AHT20 Sensor

AHTX0 Library Installation

- Find and install the Adafruit AHTX0 library by Adafruit
- You can include this library with `#include <Adafruit_AHTX0.h>`


LIBRARY MANAGER

aht

Type: All

Topic: All

AHT20 by dvarrel

library to drive AHT20 temperature and humidity IC. Forked from...

More info

1.0.2 INSTALL

Adafruit AHT10 by Adafruit

Arduino library for the AHT10 sensors in the Adafruit shop Arduino library for the AHT10...

More info

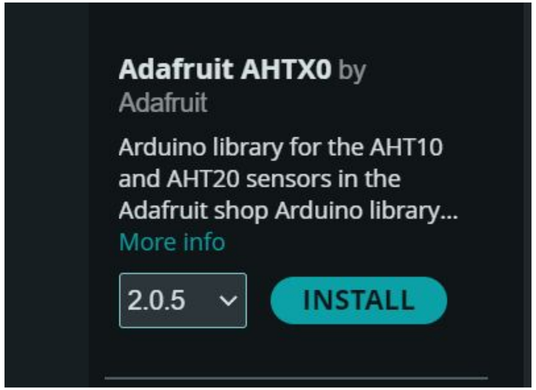
0.1.0 INSTALL

Adafruit AHTX0 by Adafruit

Arduino library for the AHT10 and AHT20 sensors in the Adafruit shop Arduino library...

More info

2.0.5 INSTALL


Adafruit AHTX0 by Adafruit

Arduino library for the AHT10 and AHT20 sensors in the Adafruit shop Arduino library...

More info

2.0.5 INSTALL

Wire Library

- Built-in library that allows us to set our own SDA and SCL pins for I^{2}C
- Include this library with `#include <Wire.h>`
- `Wire.begin(int I2C_SDA, int I2C_SCL);`
  - Initializes the Wire library and allows the ESP32 to participate in I^{2}C as a controller or peripheral.
  - Takes two pin numbers and configures them to be SDA and SCL pins, respectively.
  - Need to connect AHT20's SDA and SCL to the ESP32 pins configured with this function

```
// Define I2C pins
#define I2C_SDA 6
#define I2C_SCL 7

void setup() {
  // configure I2C for pins 6 and 7
  Wire.begin(I2C_SDA, I2C_SCL);
}
```

AHT20 Functions

- `aht.begin()` finds the AHT sensor via the ESP32's SDA and SCL pins
- `aht.getEvent(sensors_event_t *humidity, sensors_event_t *temperature)`
    - Stores the AHT's humidity and temperature readings into two `sensors_event_t` objects (named humidity and temperature in this example)

```
void setup() {
  Serial.begin(115200);

  // configure I2C for pins 6 and 7
  Wire.begin(I2C_SDA, I2C_SCL);

  aht.begin();
}
```

```
void loop() {
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);

  delay(500);
}
```

AHT20 Test

Write a program that satisfies the following requirements:

- You must write a program in which the ESP32 **gets the temperature and humidity from an AHT20 sensor** and stores the values **in a WeatherReport**.
- Connect the AHT20's SDA to pin 6 and SCL to pin 7. Use the Wire library to configure these pins as SDA and SCL.
- The program should print the WeatherReport's temperature, humidity, and lighting to the Serial Monitor.
- The program must **run in an infinite loop**.

AHT20 Test Code

```
#include <WiFiConfig.h>
#include <Adafruit_AHTX0.h>
#include <Wire.h>
// Define I2C pins
#define I2C_SDA 6
#define I2C_SCL 7
Adafruit_AHTX0 aht;
// Create a WeatherReport
WeatherReport report;
void setup() {
  Serial.begin(115200);
  // configure I2C for pins 6 and 7
  Wire.begin(I2C_SDA, I2C_SCL);
  aht.begin();
}
void loop() {
  sensors_event_t humidity, temp;
  aht.getEvent(&humidity, &temp);
  report.humidity = humidity.relative_humidity;
  report.temperature = temp.temperature;
  Serial.print("Temp: ");
  Serial.print(report.temperature);
  Serial.println();
  Serial.print("Hum: ");
  Serial.print(report.humidity);
  Serial.println();
  delay(500);
}
```

Note: we are still including the WiFi library because it is required to use Weather Reports.

FAIR USE DISCLAIMER

Copyright Disclaimer under section 107 of the Copyright Act 1976, allowance is made for “fair use” for purposes such as criticism, comment, news reporting, teaching, scholarship, education and research.

Fair use is a use permitted by copyright statute that might otherwise be infringing.

Non-profit, educational or personal use tips the balance in favor of fair use.

CC BY-NC-SA 4.0

This work by the Institute of Electrical and Electronics Engineers, UC Irvine Branch, is licensed under CC BY-NC-SA 4.0