**Table of Contents**

|  |  |
| --- | --- |
| **Contents** | **Page No.** |
| Part A: AHT20 with I2C | **P2** |
| Part B: RTC with I2C | **P8** |
| Part C: LCD with SPI | **P12** |
|  |  |
|  |  |

***Part A: AHT20 with I2C***

1. In this scenario, AHT20 sensor is used to detect temperature and humidity of the surroundings. And the I2C digital signals of temperature and humidity are checked via the Logic Analyzer and Pulse View.
2. Hardware Required
   1. ESP32
   2. AHT20
   3. Bread board
   4. Micro USB cable
   5. Jumper Wires
3. Software Required
   1. To install Adafruit AHTX0 library, Open the Arduino IDE application, go to Tools > Manage Libraries.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Circuit Schematic
   1. VIN-> 3.3 V
   2. GND -> GND
   3. SCL -> GPIO 22
   4. SDA -> GPIO 21

A picture containing diagram

Description automatically generated

1. Source Code
2. #include <Adafruit\_AHTX0.h>
3. Adafruit\_AHTX0 aht;
4. void setup()
5. {
6. Serial.begin(115200);
7. Serial.println("Adafruit AHT10/AHT20 demo!");
8. Serial.println("AHT20 Temperature sensor demo done on 01/03/2022");
9. Serial.println("Ref: http://www.esp32learning.com/code/aht20-integrated-temperature-and-humidity-sensor-and-esp32-board-example.php");
10. if (! aht.begin())
11. {
12. Serial.println("Could not find AHT? Check wiring");
13. while (1) delay(10);
14. }
15. Serial.println("AHT10 or AHT20 found");
16. }
17. void loop() {
18. sensors\_event\_t humidity, temp;
19. aht.getEvent(&humidity, &temp);// populate temp and humidity objects with fresh data
20. Serial.print("Temperature: ");
21. Serial.print(temp.temperature);
22. Serial.println(" degrees C");
23. Serial.print("Humidity: ");
24. Serial.print(humidity.relative\_humidity);
25. Serial.println("% rH");
26. delay(500);
27. }
28. Connecting the Logic Analyzer with the circuit
    1. SCL -> Ch1 (Logic Analyzer)
    2. SDA -> Ch2 (Logic Analyzer)
    3. GND -> GND

Connect the circuit with your machine and then open the Pulse View application. Then, click the “yellow and green” figure to open the “Decoder Selector” box and, choose I2C.

Graphical user interface, application, Word

Description automatically generated

At the I2C, select SCL as “D0” and SDA as “D1”. The total sample is 1M samples and the clock rate is 1MHz. And then, click Run and zoom out the result.

Graphical user interface, application, Word

Description automatically generated

1. Results

Here is the result after you running the Pulse View. When you see the address “Data Read:1C”, the following hexadecimal values are temperature and humidity values.

Application

Description automatically generated

To read the I2C hexadecimal values are shown in AHT20’s datasheet as follow.

A picture containing graphical user interface

Description automatically generated

The following figure shows the equation how to calculate temperature and humidity from the reading I2C hexadecimal value.

Text

Description automatically generated

According to the datasheet and the above equation,

Graphical user interface, application

Description automatically generated

Humidity

Temperature

The first 20 binary bits, or 5 hexadecimal bits are the humidity data and the following 20 binary bits, or 5 hexadecimal bits are the temperature data.

**Humidity**,

4B5D8 hex -> 308,696 dec

According to Signal Transformation Equation,

Humidity, RH = 308,696 /2^20\*100 = 29.43 %

**Temperature**,

61698 hex -> 399,000dec

According to Signal Transformation Equation,

Temperature, T = 399,000/ (2^20) \*200 – 50 = 26.10 degree Celsius

Here are the results from the Arduino IDE, serial monitor. We can see that the results are approximately correct. Because the sensor has a typical accuracy of +- 2% relative humidity, and +-0.3 °C.

Graphical user interface, application, Word

Description automatically generated

Look at the sample video record. ([PartA\_AHT20\_with\_I2C.mp4)](https://studentmahidolac.sharepoint.com/:v:/s/12022ITCS447EmbeddedSystemsandInternetofThings/EefH1PPoJrFAqKAUKsO5fUQBNQozRZ_emSc8n3cY4VzcmQ?e=v8gSII)

**Your Testing: Take a video record where you should explain and calculate to inspect temperature and humidity data by using I2C decoder measured by the Logic Analyzer.**

***Part B: RTC with I2C***

1. In this scenario, RTC sensor is used to detect date, time, and temperature of the surroundings. And the I2C digital signals of its are checked via the Logic Analyzer and Pulse View.
2. Hardware Required
   1. ESP32
   2. RTC
   3. Bread board
   4. Micro USB cable
   5. Jumper Wires
3. Circuit Schematic
   1. VCC-> 3.3 V
   2. GND -> GND
   3. SCL -> GPIO 22
   4. SDA -> GPIO 21

Diagram, schematic

Description automatically generated

1. Source Code

Note that you have to change at Line No 6 and 7, WiFi SSID and password in the source code to match your selected access point.

1. #define \_SYNC\_NTP //Uncomment this if you want the synchronize RTC with the NTP server
2. #include <WiFi.h>
3. #include <time.h>
4. #include <DS3231.h>
5. #include <Wire.h>
6. const char\* ssid = "";
7. const char\* password = "";
8. const char\* ntpServer = "th.pool.ntp.org";
9. //const char\* ntpServer = "clock.mahidol.ac.th";
10. const long gmtOffset\_sec = 3600 \* 7; //UTC +7.00
11. const int daylightOffset\_sec = 0; //0 means no DST observed; otherwise, 3600.
12. DS3231 rtc;
13. bool h12Format;
14. bool ampm;
15. bool centuryRollover;
16. struct tm timeinfo;
17. void setup()
18. {
19. Serial.begin(9600);
20. Wire.begin();
21. #ifdef \_SYNC\_NTP
22. //connect to WiFi
23. Serial.printf("Connecting to %s ", ssid);
24. WiFi.begin(ssid, password);
25. while (WiFi.status() != WL\_CONNECTED) {
26. delay(500);
27. Serial.print(".");
28. }
29. Serial.println(" CONNECTED");
30. Serial.print("IP Address: ");
31. Serial.println(WiFi.localIP());
32. //init and get the time
33. configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);
34. if(!getLocalTime(&timeinfo)){
35. Serial.println("Failed to obtain time");
36. return;
37. }
38. Serial.println("M:" + String(timeinfo.tm\_mon) + ", Y:" + String(timeinfo.tm\_year));
39. rtc.enableOscillator(true, true, 1);
40. rtc.setClockMode(h12Format); //24-h format
41. rtc.setDoW(timeinfo.tm\_wday);
42. rtc.setHour(timeinfo.tm\_hour);
43. rtc.setMinute(timeinfo.tm\_min);
44. rtc.setSecond(timeinfo.tm\_sec);
45. rtc.setDate(timeinfo.tm\_mday);
46. rtc.setMonth(timeinfo.tm\_mon + 1); //Month from NTP starts from zero
47. rtc.setYear(timeinfo.tm\_year - 100); //Year from NTP is an offset from 1900
48. //disconnect WiFi as it's no longer needed
49. WiFi.disconnect(true);
50. WiFi.mode(WIFI\_OFF);
51. #endif
52. }
53. void loop()
54. {
55. #ifdef \_SYNC\_NTP
56. //Show time from NTP
57. configTime(gmtOffset\_sec, daylightOffset\_sec, ntpServer);
58. if(!getLocalTime(&timeinfo)){
59. Serial.println("Failed to obtain time");
60. return;
61. }
62. Serial.println("M:" + String(timeinfo.tm\_mon) + ", Y:" + String(timeinfo.tm\_year));
63. #endif
64. // Send Day-of-Week
65. Serial.print("DoW:");
66. Serial.print(rtc.getDoW());
67. Serial.print(" ");
68. // Send date
69. Serial.print("-- Date: ");
70. Serial.print(rtc.getDate(), DEC);
71. Serial.print("/");
72. Serial.print(rtc.getMonth(centuryRollover), DEC);
73. Serial.print("/");
74. Serial.print("2"); //This program is still valid until almost the next 1000 years.
75. if(centuryRollover)
76. Serial.print("1");
77. else
78. Serial.print("0");
79. Serial.print(rtc.getYear(), DEC);
80. // Send time
81. Serial.print(" -- Time: ");
82. Serial.print(rtc.getHour(h12Format, ampm), DEC);
83. Serial.print(":");
84. Serial.print(rtc.getMinute(), DEC);
85. Serial.print(":");
86. Serial.print(rtc.getSecond(), DEC);
87. //Temperature
88. Serial.print(" -- RTC Temperature: ");
89. Serial.println(rtc.getTemperature());
90. delay(1000);
91. }
92. Connecting the Logic Analyzer with the circuit
    1. SCL -> Ch1 (Logic Analyzer)
    2. SDA -> Ch2 (Logic Analyzer)
    3. GND -> GND
93. Results

Here is the result after you running the Pulse View. When you see the address “Address Read:68”, the following “Data Read” hexadecimal values are result values.

Graphical user interface, application

Description automatically generated

Date

Look at the sample video record. ([PartB\_RTC\_with\_I2C.mp4)](https://studentmahidolac.sharepoint.com/:v:/s/12022ITCS447EmbeddedSystemsandInternetofThings/ETr6Vm0Y8I1NukUjWQBNBzkB8VTv2K82CPuPTYITFj6whA?e=KcWGzk)

**Your Testing: Take a video record where you should explain and calculate to inspect RTC data by using I2C decoder measured by the Logic Analyzer.**

***Part C: LCD with SPI***

1. In this scenario, LCD is used to show real time clock and SPI digital signals of its are checked via the Logic Analyzer and Pulse View.
2. Hardware Required
   1. ESP32
   2. LCD
   3. Bread board
   4. Micro USB cable
   5. Jumper Wires
3. Circuit Schematic

A picture containing diagram

Description automatically generated

* **Install** TFT\_eSPI **library from** Tools -> Manage Libraries...
* **Go to the location of the installed** TFT\_eSPI **library**
* **On Windows,** My Documents -> Arduino -> libraries -> TFT\_eSPI
* **On Mac,** Documents -> Arduino -> libraries -> TFT\_eSPI
* **Go to** User\_Setups **directory and create file** Setup0\_ESP32\_ILI9341.h**.**
* **Add the following lines to the file and save it.**

#define ILI9341\_DRIVER

#define TFT\_MISO 19

#define TFT\_MOSI 23

#define TFT\_SCLK 18

#define TFT\_CS   15 // Chip select control pin

#define TFT\_DC    2 // Data Command control pin

#define TFT\_RST   4 // Reset pin (could connect to RST pin)

#define LOAD\_GLCD   // Font 1. Original Adafruit 8-pixel font needs ~1820 bytes in FLASH

#define LOAD\_FONT2 // Font 2. Small 16-pixel high font, needs ~3534 bytes in FLASH, 96 characters

#define LOAD\_FONT4 // Font 4. Medium 26-pixel high font, needs ~5848 bytes in FLASH, 96 characters

#define LOAD\_FONT6 // Font 6. Large 48 pixel font, needs ~2666 bytes in FLASH, only characters 1234567890: -.apm

#define LOAD\_FONT7 // Font 7. 7 segment 48 pixel font, needs ~2438 bytes in FLASH, only characters 1234567890:

#define LOAD\_FONT8 // Font 8. Large 75 pixel font needs ~3256 bytes in FLASH, only characters 1234567890: -.

#define LOAD\_GFXFF // FreeFonts. Include access to the 48 Adafruit\_GFX free fonts FF1 to FF48 and custom fonts

#define SMOOTH\_FONT

* **Open** User\_Setup\_Select.h
* **Comment the line** #include <User\_Setup.h>
* **Add line** #include <User\_Setups/Setup0\_ESP32\_ILI9341.h>

1. Look at Sample video record. ([PartC\_LCD\_with\_SPI.mp4](https://studentmahidolac.sharepoint.com/:v:/s/12022ITCS447EmbeddedSystemsandInternetofThings/Efp8pcI0JxZHvnqc1i6CVH0BbuZF3cZN9dnFMzgE4eOH-w?e=vVnOvg))

**Your Testing: Take a video record where you should test as follows**

**Try the programs in** Files -> Examples -> TFT\_eSPI -> 320 x 240

* **TFT\_Rainbow\_one\_lib**
* **Free\_Font\_Demo**
* **TFT\_Clock**
* **TFT\_Pong**

**And SPI data by using I2C decoder measured by the Logic Analyzer.**

Connecting the Logic Analyzer with the circuit to check SPI result.

1. SCK -> CH1
2. MISO -> CH2
3. MOSI -> CH3