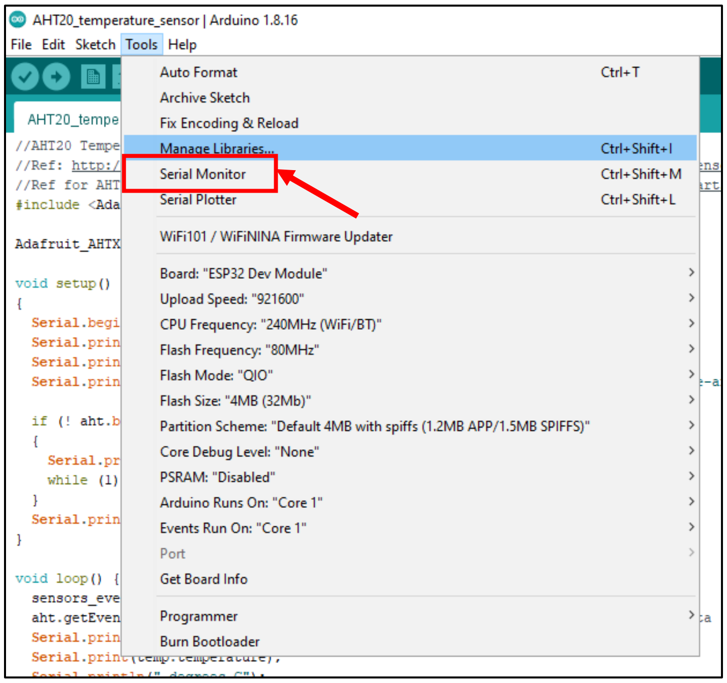
Table of Contents

Part A: AHT20 with I2C

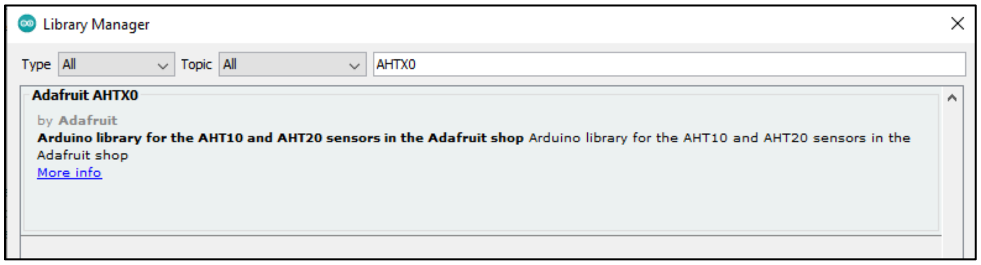
1. In this scenario, AHT20 sensor is used to detect temperature and humidity of the surroundings. And the I2C digital signals of temperature and humidity are checked via the Logic Analyzer and Pulse View.

2. Hardware Required
   a. ESP32
   b. AHT20
   c. Bread board
   d. Micro USB cable
   e. Jumper Wires

3. Software Required
   a. To install Adafruit AHTX0 library, Open the Arduino IDE application, go to Tools > Manage Libraries.

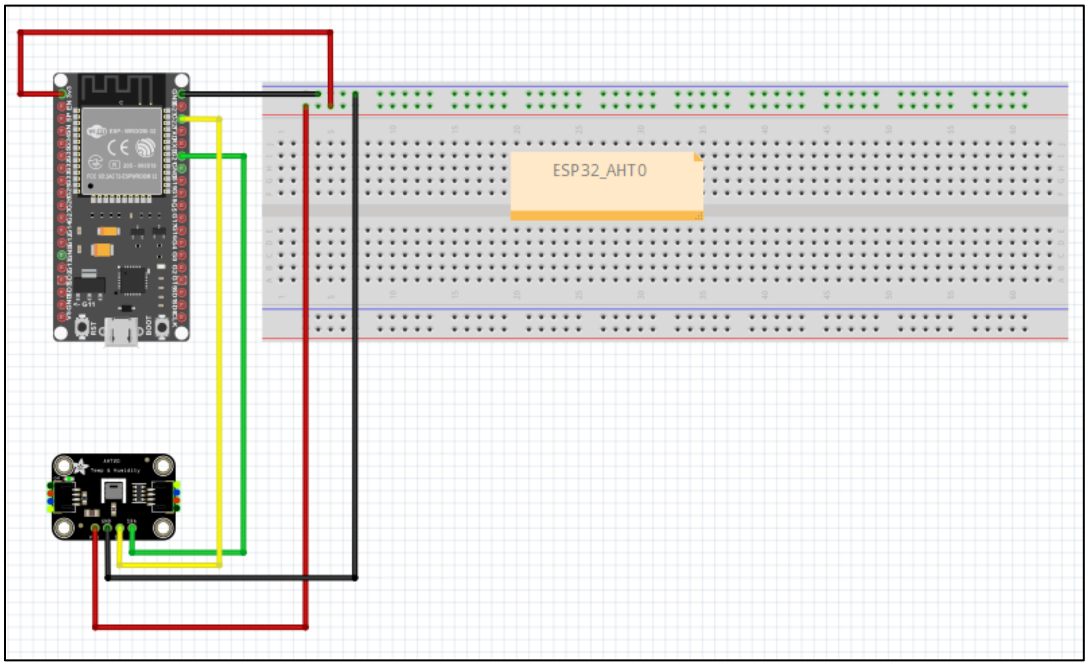

AHT20_temperature_sensor | Arduino 1.8.16
File Edit Sketch Tools Help
AHT20_tempe
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Port
Get Board Info
Programmer
Burn Bootloader

Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"



Adafruit AHTX0
by **Adafruit**
Arduino library for the AHT10 and AHT20 sensors in the Adafruit shop Arduino library for the AHT10 and AHT20 sensors in the Adafruit shop
More info

4. Circuit Schematic
    a. VIN-> 3.3 V
    b. GND -> GND
    c. SCL -> GPIO 22
    d. SDA -> GPIO 21



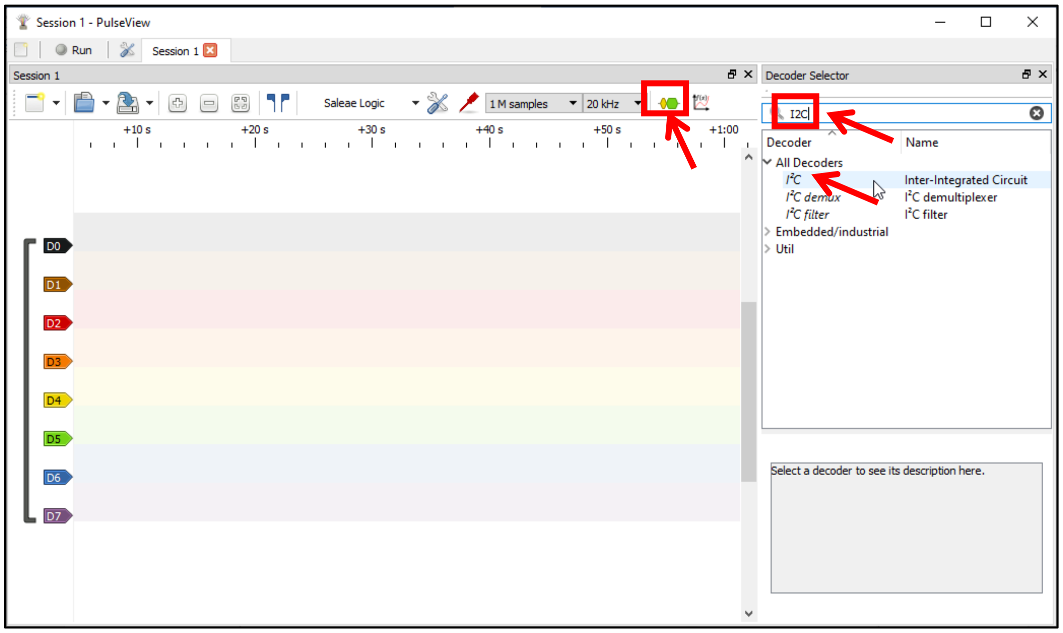
5. Source Code

```
#include <Adafruit_AHTX0.h>
Adafruit_AHTX0 aht;
void setup()
{
Serial.begin(115200);
Serial.println("Adafruit AHT10/AHT20 demo!");
Serial.println("AHT20 Temperature sensor demo done on 01/03/2022");
Serial.println("Ref:      http://www.esp32learning.com/code/aht20-integrated-temperature-and-humidity-sensor-and-esp32-board-example.php");
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);// populate temp and humidity objects with fresh data
      Serial.print("Temperature: ");
      Serial.print(temp.temperature);
```

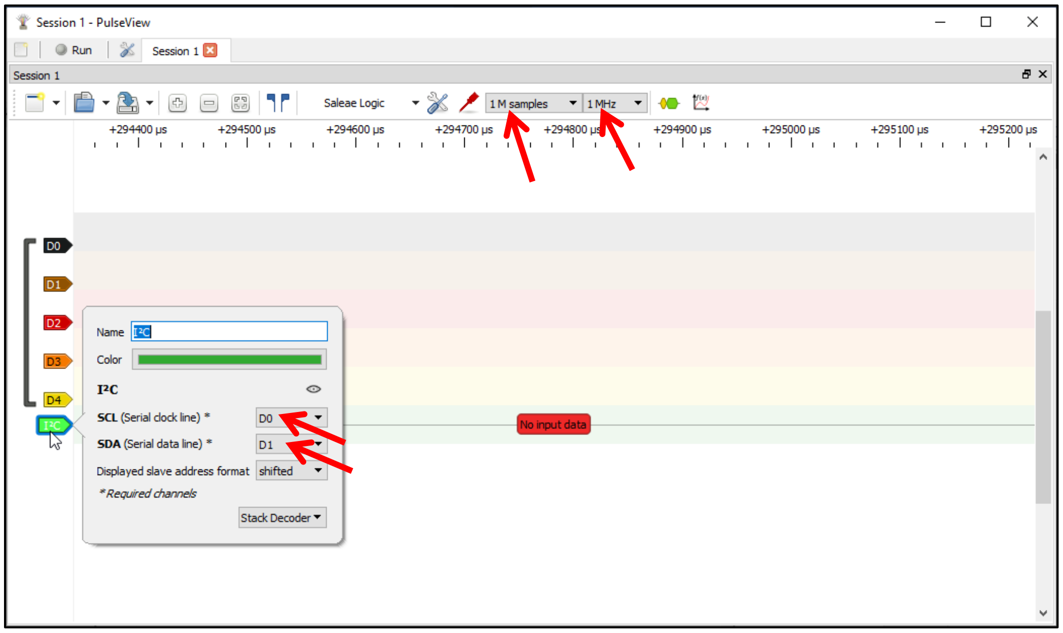
```
        Serial.println(" degrees C");
        Serial.print("Humidity: ");
        Serial.print(humidity.relative_humidity);
        Serial.println("% rH");
        delay(500);
        }
```

6. Connecting the Logic Analyzer with the circuit
   a. SCL -> Ch1 (Logic Analyzer)
   b. SDA -> Ch2 (Logic Analyzer)
   c. 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.

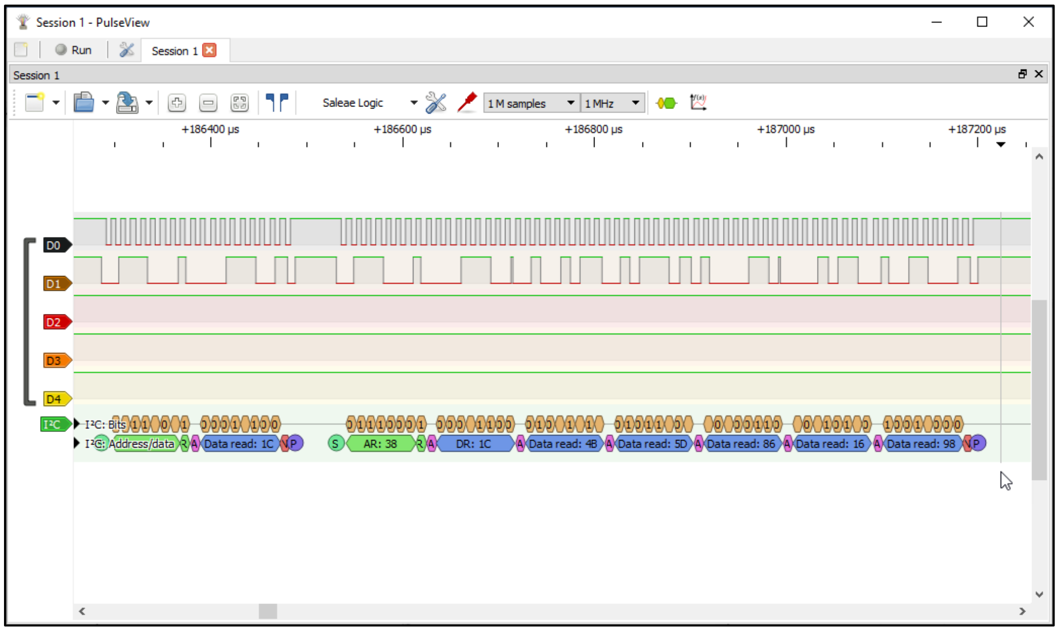


At the I2C, select SCL as "D0" and SDA as "D1". The total sample is 1M samples and the clock rate is 1MHz. And then, click Run and zoom out the result.



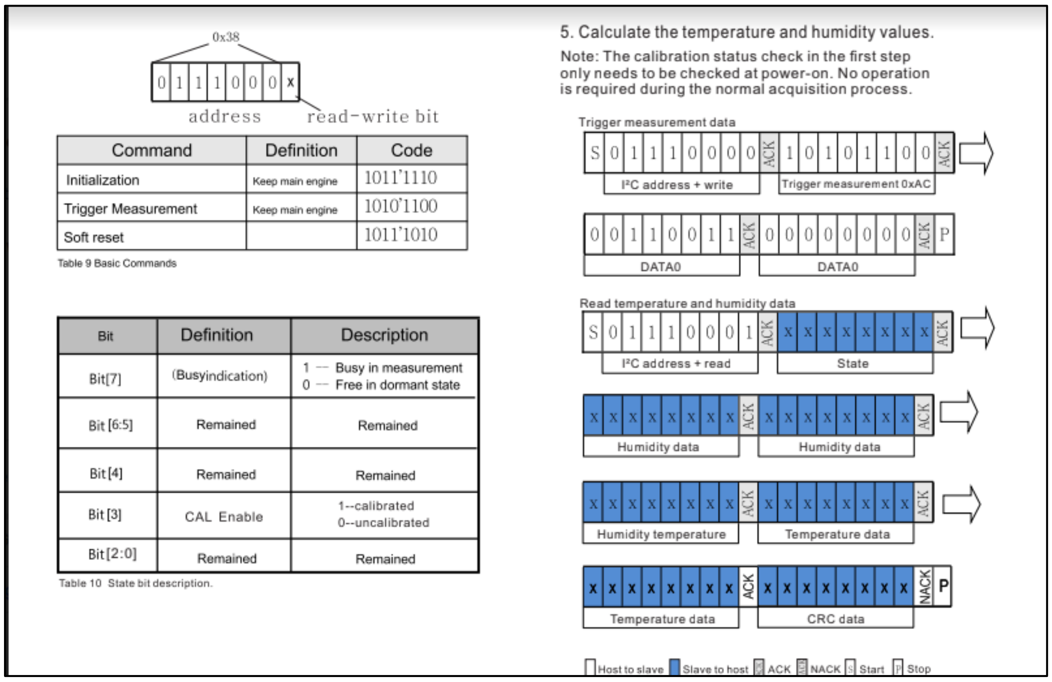
Name: I²C
Color
I²C
SCL (Serial clock line) *: D0
SDA (Serial data line) *: D1
Displayed slave address format: shifted
*Required channels

7. 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.



```
I²C: Bits
I²C: Address/data  R A Data read: 1C N P   S AR: 38 R A DR: 1C A Data read: 4B A Data read: 5D A Data read: 86 A Data read: 16 A Data read: 98 N P
```

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



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

Table 9 Basic Commands

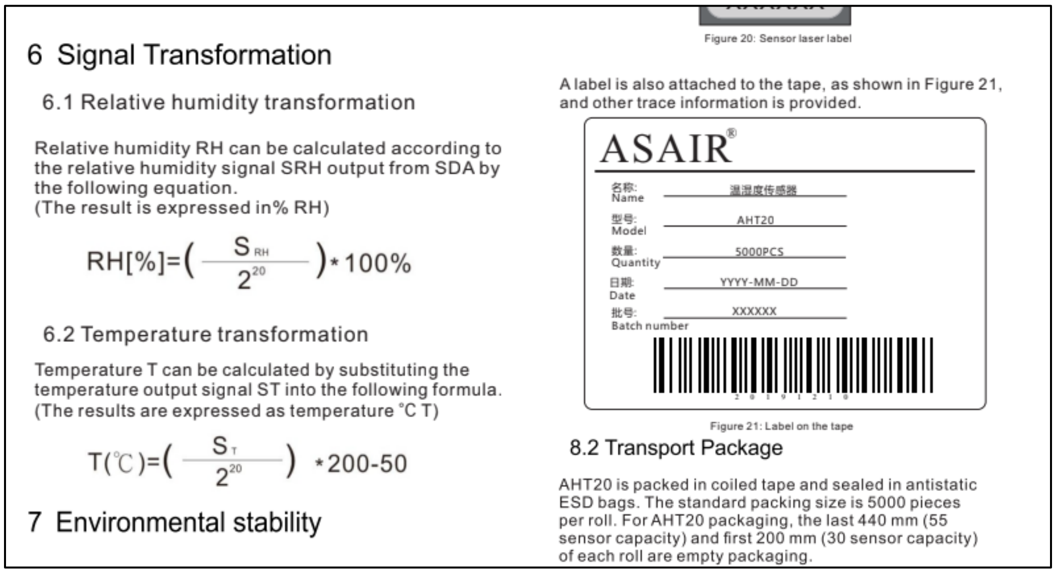
Bit	Definition	Description
Bit[7]	(Busyindication)	1 — Busy in measurement 0 — Free in dormant state
Bit [6:5]	Remained	Remained
Bit [4]	Remained	Remained
Bit [3]	CAL Enable	1--calibrated 0--uncalibrated
Bit [2:0]	Remained	Remained

Table 10 State bit description.

5. Calculate the temperature and humidity values.

Note: The calibration status check in the first step only needs to be checked at power-on. No operation is required during the normal acquisition process.

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



6 Signal Transformation

6.1 Relative humidity transformation

Relative humidity RH can be calculated according to the relative humidity signal SRH output from SDA by the following equation.
(The result is expressed in% RH)

$$RH[\%]=\left(\frac{S_{RH}}{2^{20}}\right)*100\%$$

6.2 Temperature transformation

Temperature T can be calculated by substituting the temperature output signal ST into the following formula.
(The results are expressed as temperature °C T)

$$T(℃)=\left(\frac{S_{T}}{2^{20}}\right)*200-50$$

7 Environmental stability

Figure 20: Sensor laser label

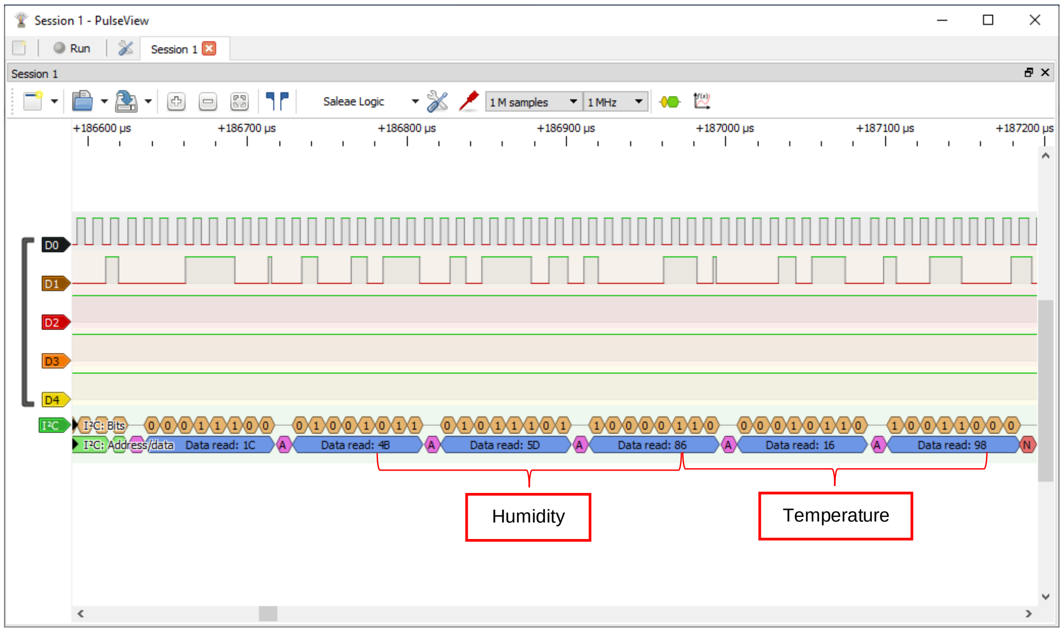
A label is also attached to the tape, as shown in Figure 21, and other trace information is provided.

Figure 21: Label on the tape

8.2 Transport Package

AHT20 is packed in coiled tape and sealed in antistatic ESD bags. The standard packing size is 5000 pieces per roll. For AHT20 packaging, the last 440 mm (55 sensor capacity) and first 200 mm (30 sensor capacity) of each roll are empty packaging.

According to the datasheet and the above equation,



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 $_{\text{hex}}$ -> 308,696 $_{\text{dec}}$

According to Signal Transformation Equation,

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

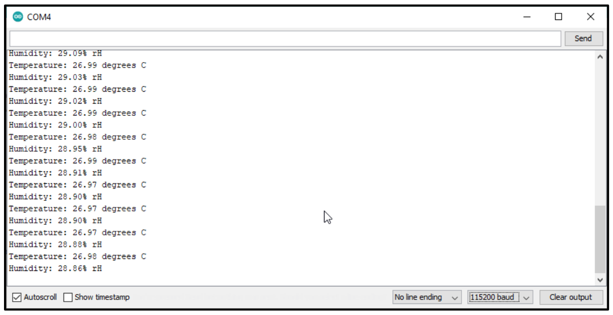
Temperature,

61698 $_{\text{hex}}$ -> 399,000$_{\text{dec}}$

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.



```
Humidity: 29.09% rH
Temperature: 26.99 degrees C
Humidity: 29.03% rH
Temperature: 26.99 degrees C
Humidity: 29.02% rH
Temperature: 26.99 degrees C
Humidity: 29.00% rH
Temperature: 26.98 degrees C
Humidity: 28.95% rH
Temperature: 26.99 degrees C
Humidity: 28.91% rH
Temperature: 26.97 degrees C
Humidity: 28.90% rH
Temperature: 26.97 degrees C
Humidity: 28.90% rH
Temperature: 26.97 degrees C
Humidity: 28.88% rH
Temperature: 26.98 degrees C
Humidity: 28.86% rH
```

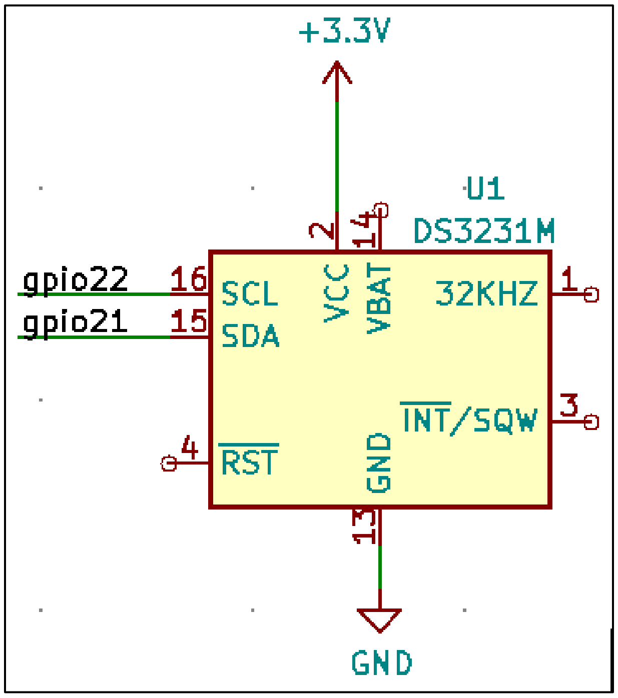
Look at the sample video record. (PartA_AHT20_with_I2C.mp4)

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
   a. ESP32
   b. RTC
   c. Bread board
   d. Micro USB cable
   e. Jumper Wires
3. Circuit Schematic
   a. VCC-> 3.3 V
   b. GND -> GND
   c. SCL -> GPIO 22
   d. SDA -> GPIO 21



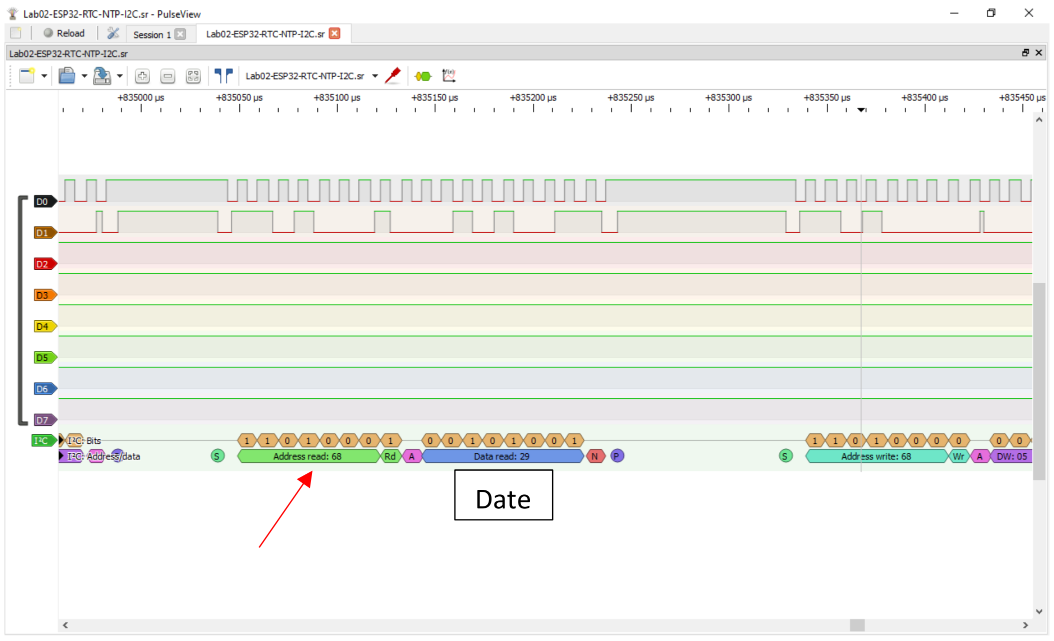
4. Source Code
   Note that you have to change at Line No 6 and 7, WiFi SSID and password in the source code to match your selected access point.

```
1.  #define _SYNC_NTP //Uncomment this if you want the synchronize RTC with the NTP
    server
2.  #include <WiFi.h>
3.  #include <time.h>
```

```
#include <DS3231.h>
#include <Wire.h>
const char* ssid       = "";
const char* password   = "";
const char* ntpServer = "th.pool.ntp.org";
//const char* ntpServer = "clock.mahidol.ac.th";
const long  gmtOffset_sec = 3600 * 7; //UTC +7.00
const int   daylightOffset_sec = 0; //0 means no DST observed; otherwise, 3600.
DS3231  rtc;
bool h12Format;
bool ampm;
bool centuryRollover;
struct tm timeinfo;
void setup()
{
Serial.begin(9600);
Wire.begin();
#ifdef _SYNC_NTP
//connect to WiFi
Serial.printf("Connecting to %s ", ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println(" CONNECTED");
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());
//init and get the time
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
if(!getLocalTime(&timeinfo)){
Serial.println("Failed to obtain time");
return;
}
Serial.println("M:" + String(timeinfo.tm_mon) + ", Y:" + String(timeinfo.tm_year));
rtc.enableOscillator(true, true, 1);
rtc.setClockMode(h12Format); //24-h format
rtc.setDoW(timeinfo.tm_wday);
rtc.setHour(timeinfo.tm_hour);
rtc.setMinute(timeinfo.tm_min);
rtc.setSecond(timeinfo.tm_sec);
rtc.setDate(timeinfo.tm_mday);
rtc.setMonth(timeinfo.tm_mon + 1); //Month from NTP starts from zero
rtc.setYear(timeinfo.tm_year - 100); //Year from NTP is an offset from 1900
//disconnect WiFi as it's no longer needed
WiFi.disconnect(true);
WiFi.mode(WIFI_OFF);
#endif
}
void loop()
{
#ifdef _SYNC_NTP
//Show time from NTP
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
if(!getLocalTime(&timeinfo)){
Serial.println("Failed to obtain time");
return;
}
Serial.println("M:" + String(timeinfo.tm_mon) + ", Y:" + String(timeinfo.tm_year));
#endif
// Send Day-of-Week
Serial.print("DoW:");
Serial.print(rtc.getDoW());
Serial.print(" ");
// Send date
Serial.print("-- Date: ");
Serial.print(rtc.getDate(), DEC);
Serial.print("/");
Serial.print(rtc.getMonth(centuryRollover), DEC);
Serial.print("/");
Serial.print("2"); //This program is still valid until almost the next 1000 years.
if(centuryRollover)
Serial.print("1");
else
Serial.print("0");
Serial.print(rtc.getYear(), DEC);
// Send time
Serial.print(" -- Time: ");
Serial.print(rtc.getHour(h12Format, ampm), DEC);
Serial.print(":");
Serial.print(rtc.getMinute(), DEC);
Serial.print(":");
Serial.print(rtc.getSecond(), DEC);
//Temperature
Serial.print(" -- RTC Temperature: ");
Serial.println(rtc.getTemperature());
delay(1000);
}
```

5. Connecting the Logic Analyzer with the circuit
    a. SCL -> Ch1 (Logic Analyzer)
    b. SDA -> Ch2 (Logic Analyzer)
    c. GND -> GND

6. 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.

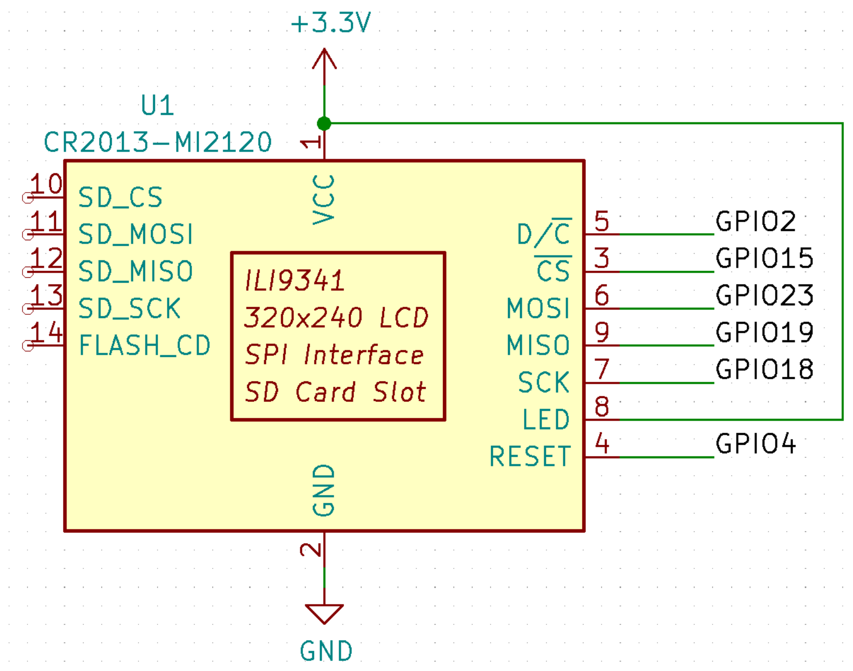


Look at the sample video record. [(PartB_RTC_with_I2C.mp4)]

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
   a. ESP32
   b. LCD
   c. Bread board
   d. Micro USB cable
   e. Jumper Wires
3. Circuit Schematic



- **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>

4. Look at Sample video record. (PartC_LCD_with_SPI.mp4)

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