

Introduction to ESP32

IS4151/IS5451 – AIoT Solutions and Development
AY 2024/25 Semester 2

Lecturer: A/P TAN Wee Kek

Email: tanwk@comp.nus.edu.sg :: **Tel:** 6516 6731 :: **Office:** COM3-02-35

Consultation: Tuesday, 2 pm to 4 pm. Additional consultations by appointment are welcome.

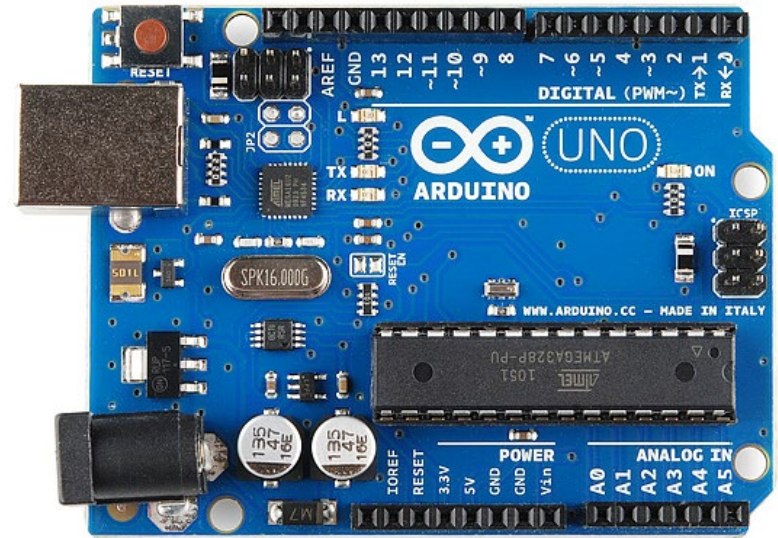
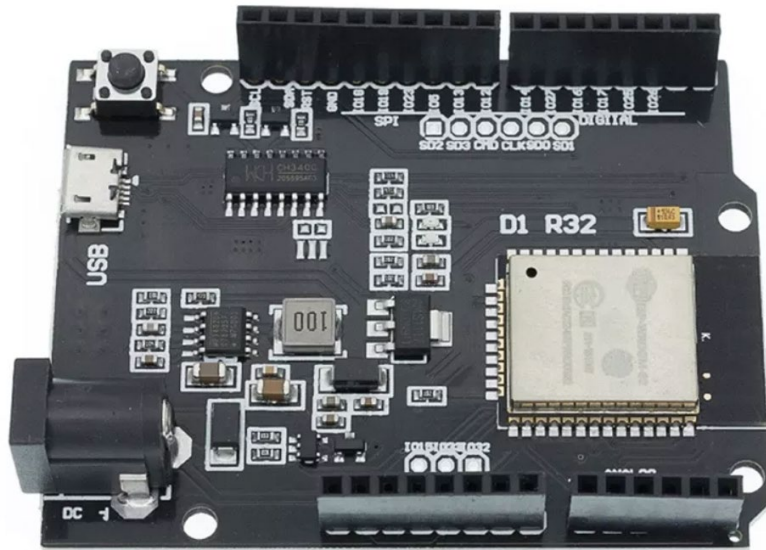
Overview of the ESP32

- ▶ **ESP32** is a series of microcontrollers with integrated WiFi and Bluetooth.
- ▶ ESP32 development boards such as **DI R32** feature an Arduino UNO form factor.
- ▶ General technical specifications:

Dimension	DI R32	Arduino UNO
Processor	Xtensa 32-bit LX6 Dual-core	ATmega328P
Clockspeed	240 MHz	16 MHz
Memory	520KB SRAM, 4 MB Flash	2KB SRAM, 32KB Flash
Pins	14 digital and 6 analog	
Communication	UART, I2C and SPI	
Wireless Communication	WiFi and Bluetooth	None
I/O Voltage	3.3V	5V

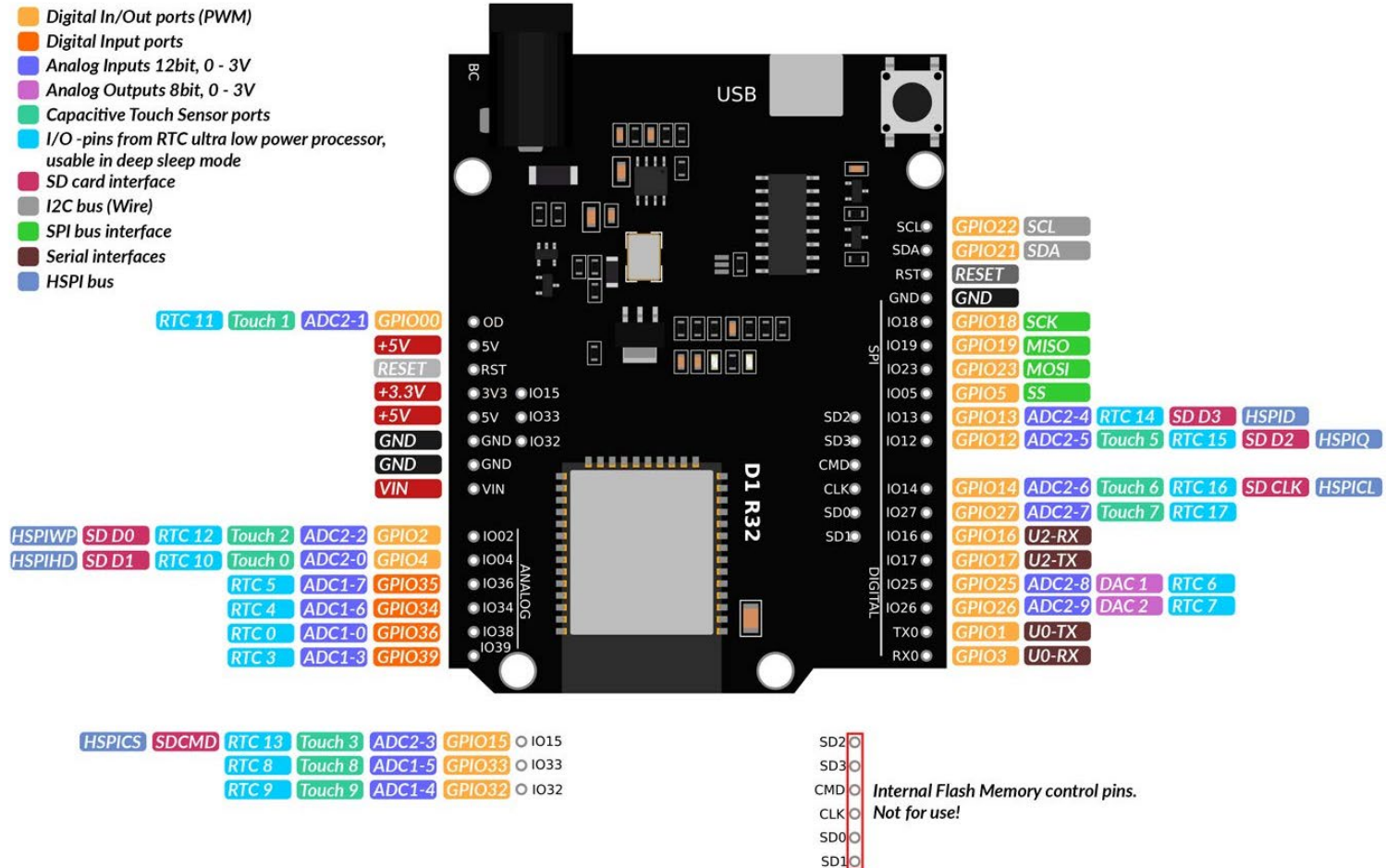
Overview of the ESP32 (cont.)

- ▶ The form factor of D1 R32 (left) is similar to an Arduino UNO (right):



Overview of the ESP32 (cont.)

► Pinout of the D1 R32:



Programming the D1 R32

- ▶ The D1 R32 can be programmed using different platforms:
 - ▶ The Arduino platform with C++ is commonly used.
 - ▶ Compatible with the Arduino IDE.
- ▶ **Arduino sketch:**
 - ▶ Sketch is the name of an Arduino program.
 - ▶ A sketch is a unit of code that is uploaded to an Arduino board for execution.
 - ▶ The source file carries the `*.ino` file extension.
- ▶ **Anatomy of an Arduino sketch:**
 - ▶ **Comments** are enclosed within `/*` and `*/` and are ignored by Arduino.

Programming the D1 R32 (cont.)

- ▶ **Variables** are memory spaces for storing data.
- ▶ **Functions** are named module of code that can be used anywhere in a sketch.
- ▶ There are two special functions that every Arduino sketch must have:
 - ▶ `setup()`
 - ▶ This function is called once when the sketch starts.
 - ▶ It is used to perform initialisation tasks such as setting pin modes.
 - ▶ `loop()`
 - ▶ This function is called repeatedly.
 - ▶ It contains the main control logic of the sketch.
- ▶ These two functions must be defined regardless of whether they are used.

Programming the D1 R32 (cont.)

- ▶ The D1 R32 does not have an onboard display panel:
 - ▶ Data and diagnostic messages are written to the serial output.
 - ▶ The Arduino IDE contains a serial monitor to view the output.
- ▶ Commonly used functions:
 - ▶ `Serial.begin()` – Sets the data rate for serial data transmission in bits per second (baud).
 - ▶ `Serial.print()` – Print string data to the serial output.
 - ▶ `Serial.println()` – Print string data to the serial output followed by carriage return and new line.
 - ▶ `delay()` – Pauses the program for the specified amount of time (in milliseconds).



Programming the D1 R32 (cont.)

► Our first “Hello World” Arduino sketch:

The screenshot shows the Arduino IDE interface. The main window displays a sketch named 'src01' with the following code:

```
void setup() {  
  Serial.begin(115200);  
}  
  
void loop() {  
  Serial.println("Hello World!");  
  delay(1000);  
}
```

On the right side, the Serial Monitor window is open, showing the output of the sketch. It displays 'Hello World!' followed by several lines of timestamps and the text 'Hello World!', indicating that the sketch is running successfully.

src01.ino



Programming the D1 R32 (cont.)

- Modifying the “Hello World” sketch to print out a running integer counter value:

The screenshot displays the Arduino IDE interface. The main editor window, titled 'src02 | Arduino 1.8.19', shows the following code:

```
int counter;

void setup() {
  counter = 0;
  Serial.begin(115200);
}

void loop() {
  counter++;
  Serial.println("Hello World: " + String(counter));
  delay(1000);
}
```

The right-hand pane, titled 'COM16', shows the serial output of the sketch. It displays a series of messages, each preceded by a timestamp and a right arrow, indicating the counter value at each second:

```
02:51:00.735 -> Hello World: 1
02:51:01.766 -> Hello World: 2
02:51:02.750 -> Hello World: 3
02:51:03.737 -> Hello World: 4
02:51:04.768 -> Hello World: 5
02:51:05.752 -> Hello World: 6
02:51:06.736 -> Hello World: 7
02:51:07.767 -> Hello World: 8
02:51:08.752 -> Hello World: 9
02:51:09.736 -> Hello World: 10
02:51:10.767 -> Hello World: 11
```

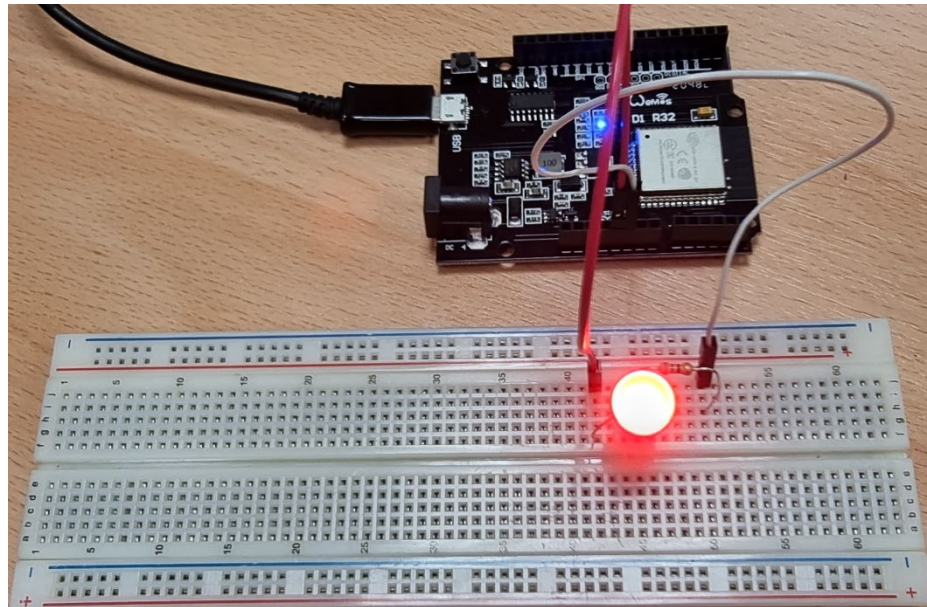
src02.ino



Hands-on with Basic Electronics

► Turn on a LED:

- Connect a LED to your board using two male-male jumper wires.
- The long leg should be connected to a 5V pin.
- The short leg should be connected to a GND pin through a resistor.





Hands-on with Digital Output

- ▶ In the earlier hands-on, the long leg of the LED was connected to a 5V pin.
- ▶ Basic LED blinker:
 - ▶ Reconnect the long leg of the LED to pin 26 (IO26).
 - ▶ Use the `pinMode(pin, mode)` function to set pin 26 to output mode.
 - ▶ Use the `digitalWrite(pin, value)` function to write `HIGH` and `LOW` to turn the LED on and off.
 - ▶ Use the `delay()` function to blink the LED at 1 sec interval.
 - ▶ Refer to `src03.ino` for the sample code.
- ▶ How many times does the LED blink and why?



Hands-on with Digital Input

- ▶ Replace the LED and resistor with a push button on the breadboard:
 - ▶ Use only one side of a push button, i.e., either A-C or B-D.
 - ▶ Connect one button pin to a ground pin.
 - ▶ Connect the other button pin to pin 26 (IO26).
 - ▶ Use the `pinMode(pin, mode)` function to set pin 26 to input mode using the built-in pullup resistor.
 - ▶ Use the `digitalRead(pin)` function to read **HIGH** and **LOW** from the push button and print the digital input to the serial console.
 - ▶ Refer to `src04.ino` for the sample code.



Hands-on with Digital Input (cont.)

The screenshot shows the Arduino IDE interface. The top bar indicates 'src04 | Arduino 1.8.19'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for checking, running, saving, and other functions. The main editor area displays the code for 'src04.ino':

```
int buttonPin = 26;

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  if(digitalRead(buttonPin) == HIGH)
  {
    Serial.println("Button Released");
  }
  else
  {
    Serial.println("Button Pressed");
  }

  delay(100);
}
```

At the bottom of the IDE, a status bar shows 'Done uploading.', 'Leaving...', and 'Hard resetting via RTS pin...'. To the right of the IDE, the serial monitor is open, showing a list of timestamps and messages: '23:11:44.258 -> Button Released', '23:11:44.351 -> Button Released', '23:11:44.445 -> Button Released', '23:11:44.539 -> Button Released', '23:11:44.644 -> Button Released', '23:11:44.738 -> Button Released', '23:11:44.879 -> Button Released', '23:11:44.972 -> Button Released', '23:11:45.066 -> Button Released', '23:11:45.160 -> Button Released', '23:11:45.242 -> Button Released', '23:11:45.336 -> Button Released', '23:11:45.476 -> Button Released', '23:11:45.571 -> Button Released', '23:11:45.665 -> Button Released', '23:11:45.758 -> Button Released', '23:11:45.852 -> Button Released', '23:11:45.946 -> Button Released', '23:11:46.040 -> Button Released', '23:11:46.180 -> Button Pressed', '23:11:46.274 -> Button Pressed', '23:11:46.368 -> Button Pressed', '23:11:46.462 -> Button Pressed', '23:11:46.555 -> Button Pressed', '23:11:46.649 -> Button Pressed', '23:11:46.743 -> Button Pressed', '23:11:46.836 -> Button Pressed', '23:11:46.979 -> Button Pressed', '23:11:47.073 -> Button Pressed', '23:11:47.166 -> Button Pressed', '23:11:47.256 -> Button Pressed', '23:11:47.349 -> Button Pressed', '23:11:47.443 -> Button Pressed', '23:11:47.537 -> Button Pressed', '23:11:47.678 -> Button Pressed', '23:11:47.771 -> Button Pressed', '23:11:47.865 -> Button Pressed', '23:11:47.959 -> Button Released', '23:11:48.052 -> Button Released', '23:11:48.146 -> Button Released', '23:11:48.240 -> Button Released', '23:11:48.381 -> Button Released', '23:11:48.474 -> Button Released'.

src04.ino





Hands-on with Digital Input (cont.)

- ▶ The current setup allows us to read digital input signal but is not so useful.
- ▶ Use the same circuit but amend the code to toggle a variable between **HIGH** and **LOW** each time the push button is pressed:
 - ▶ Refer to [src05.ino](#) for the sample code.

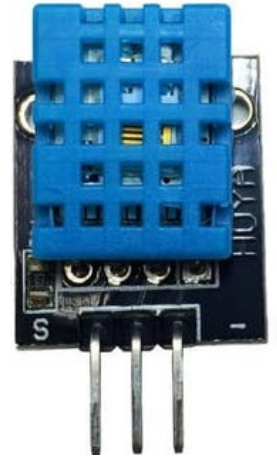


Hands-on with Digital Input/Output

- ▶ Create a simple LED lamp by using a push button to turn a LED on and off with each button press:
 - ▶ Recall that each pin can only be programmed to work in either input or output mode at any one point in time.
 - ▶ Thus, a second pin is required to allow both digital input and output to flow concurrently.
 - ▶ Keep the button pin connected to pin 26 (IO26).
 - ▶ Restore the LED and resistor by connecting the long leg of the LED to pin 25 (IO25).
 - ▶ Amend the code in [src05.ino](#) accordingly.
 - ▶ Refer to [src06.ino](#) for the completed sample code.

Reading Input Data from Sensors

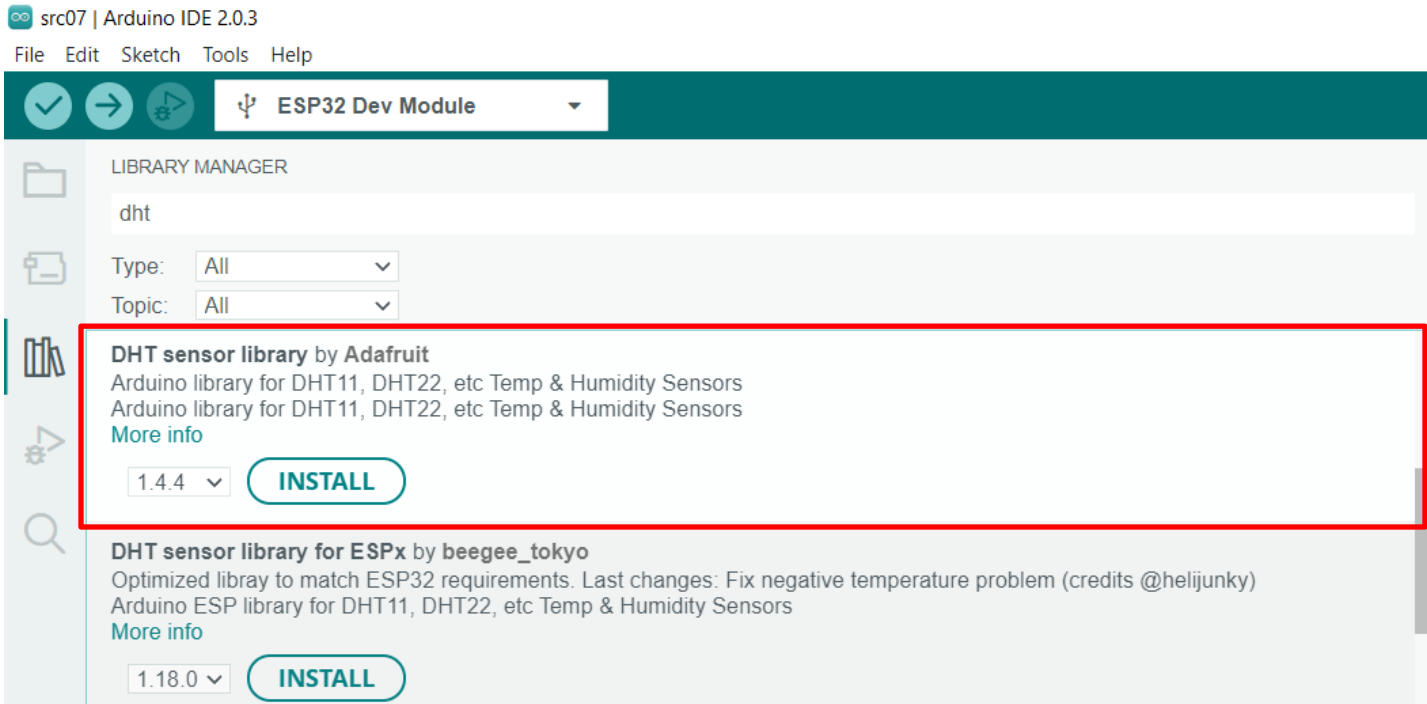
- ▶ Using DHT11 to read temperature:
 - ▶ DHT11 is a basic, ultra low-cost digital temperature and humidity sensor.
 - ▶ Can be used to implement remote weather stations, home environmental control systems, etc.
- ▶ Interacting with sensors such as DHT11 is not a straightforward task of reading digital or analog values:
 - ▶ Need to work with the respective communication protocol, e.g., one-wire, a serial protocol, in the case of DHT11.
 - ▶ It is typically easier to utilise a library or API in a similar fashion as software engineering.
 - ▶ For DHT11, we would be using the “DHT sensor library by Adafruit”.





Hands-on with Sensor Input

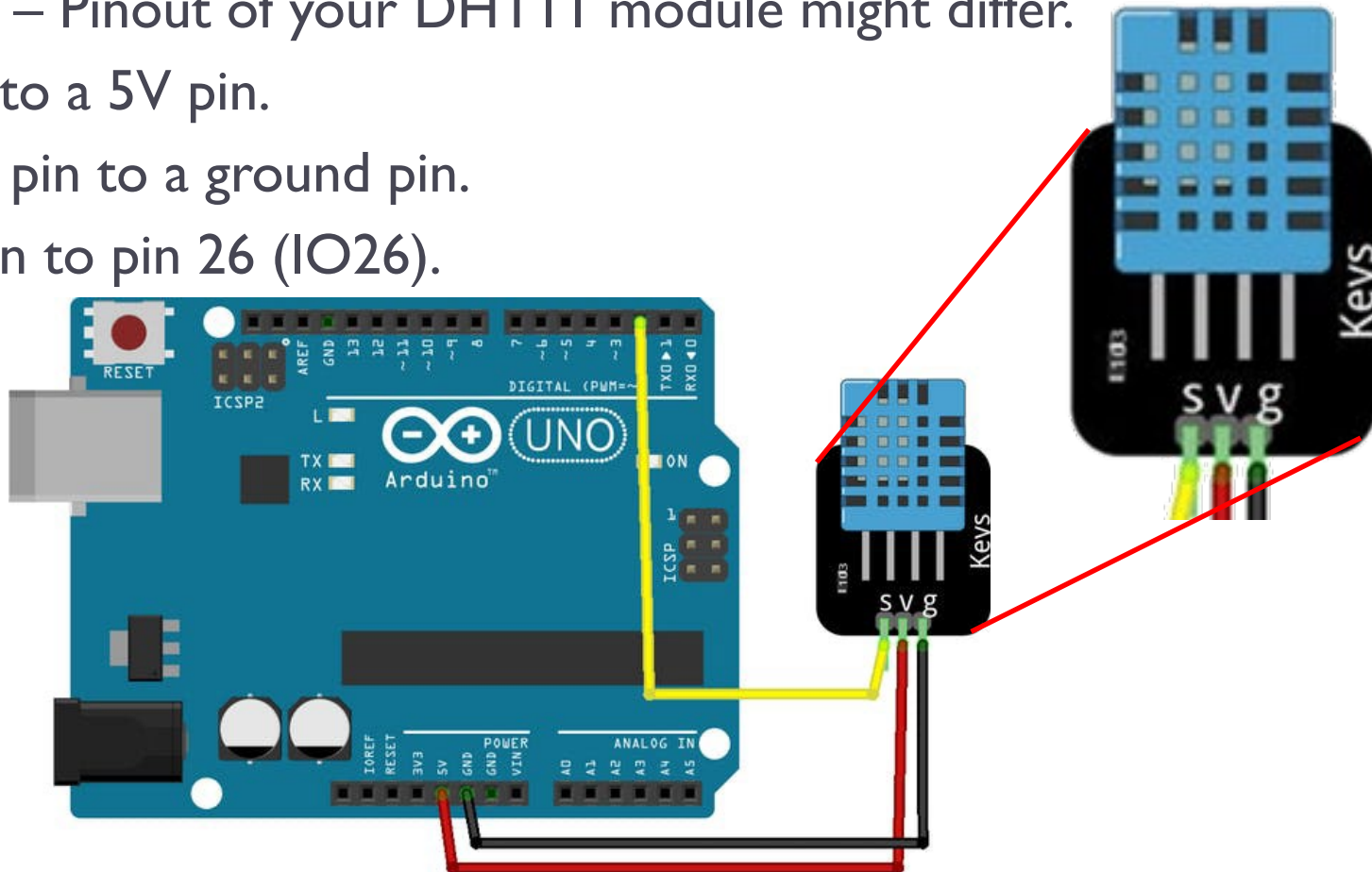
- ▶ Installing the DHT sensor library:
 - ▶ From the main menu of the Arduino IDE, go to “Tools > Manage Libraries” and search for “dht”.
 - ▶ Look for “DHT sensor library” and install it.





Hands-on with Sensor Input (cont.)

- ▶ Setup the required circuit as shown in the figure below:
 - ▶ Caution – Pinout of your DHT11 module might differ.
 - ▶ Vcc pin to a 5V pin.
 - ▶ Ground pin to a ground pin.
 - ▶ Signal pin to pin 26 (IO26).





Hands-on with Sensor Input (cont.)

- ▶ Create a new Arduino sketch:

- ▶ Include the required header file using the command:

```
#include "DHT.h"
```

- ▶ Declare the DHT object:

```
DHT dht(26, DHT11);
```

- ▶ Initialise the DHT object in the `setup()` function:

```
dht.begin();
```

- ▶ Read temperature as Celsius (the default) in the `loop()` function and print it to the serial output:

```
float temp = dht.readTemperature();
```

- ▶ Refer to `src07.ino` for the completed sample code.



Hands-on with Sensor Input (cont.)

The screenshot shows the Arduino IDE interface. The main window displays the sketch 'src07' with the following code:

```
#include "DHT.h"

int dhtPin = 26;
DHT dht(dhtPin, DHT11);

void setup() {
    Serial.begin(115200);
    dht.begin();
}

void loop() {
    float temp = dht.readTemperature();
    Serial.println("Temp: " + String(temp));
    delay(1000);
}
```

The serial monitor on the right, titled 'COM16', shows the output of the sketch. It displays a series of timestamps followed by the temperature reading: 'Temp: 26.10' and then 'Temp: 26.60' for the subsequent 15 readings.

Timestamp	Temp
01:24:04.343	26.10
01:24:05.374	26.60
01:24:06.392	26.60
01:24:07.423	26.60
01:24:08.407	26.60
01:24:09.437	26.60
01:24:10.402	26.60
01:24:11.433	26.60
01:24:12.464	26.60
01:24:13.495	26.60
01:24:14.479	26.60
01:24:15.512	26.60
01:24:16.497	26.60
01:24:17.521	26.60

src07.ino



Consuming REST Web Service in D1 R32

- ▶ We will now integrate our DHT11 sensor code with the REST web service:

- ▶ The ESP32 WiFi library can be used to establish an Internet connection to a web server:

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

- ▶ The HTTPClient can then be used to make a PUT request to our web service:

```
#include <HTTPClient.h>
```

- ▶ The JSON result can be decoded with the Arduino JSON library:

```
#include <Arduino_JSON.h>
```

- ▶ Refer to [src08](#) for the completed sample code.



Consuming REST Web Service in D1 R32 (cont.)

The screenshot displays the Arduino IDE interface with the 'src08' project open. The sketchbook shows 'src08.ino' with the following code:

```
39 }
40
41 void loop() {
42
43     // float temp = dht.readTemperature();
44
45     float temp = random(2000, 4000) * 0.01;
46     Serial.println("Temp: " + String(temp));
47
48     WiFiClient client;
49     HTTPClient http;
50     http.begin(client, "http://" + host + "/api/sensor?temperature=" + String(temp));
51     http.addHeader("Content-Type", "text/plain");
52     int httpResponseCode = http.PUT("");
53
54     if(httpResponseCode>0)
55     {
56         String response = http.getString();
57         Serial.println(httpResponseCode);
58
59         JSONVar myObject = JSON.parse(response);
60         JSONVar value = myObject["temperature"];
```

The Serial Monitor shows the output of the sketch, displaying the temperature values generated by the random function and the corresponding HTTP PUT requests to the REST web service.

```
19:41:00.251 -> 200
19:41:00.251 -> 20.75
19:41:01.253 -> Temp: 39.27
19:41:01.393 -> 200
19:41:01.393 -> 39.27
19:41:02.388 -> Temp: 38.47
19:41:02.512 -> 200
19:41:02.512 -> 38.47
19:41:03.509 -> Temp: 32.71
19:41:03.635 -> 200
19:41:03.635 -> 32.71
```

The Command Prompt window shows the output of the Python script 'src08.py', which simulates the REST web service. It displays the HTTP PUT requests and the corresponding temperature values returned by the service.

```
HTTP/1.1" 200 -
***** temperature 26.52
192.168.137.220 - - [16/Mar/2023 19:40:47] "PUT /api/sensor?temperature=26.52
HTTP/1.1" 200 -
***** temperature 38.04
192.168.137.220 - - [16/Mar/2023 19:40:48] "PUT /api/sensor?temperature=38.04
HTTP/1.1" 200 -
***** temperature 33.9
192.168.137.220 - - [16/Mar/2023 19:40:50] "PUT /api/sensor?temperature=33.90
HTTP/1.1" 200 -
***** temperature 39.67
192.168.137.220 - - [16/Mar/2023 19:40:51] "PUT /api/sensor?temperature=39.67
HTTP/1.1" 200 -
***** temperature 35.63
192.168.137.220 - - [16/Mar/2023 19:40:52] "PUT /api/sensor?temperature=35.63
HTTP/1.1" 200 -
***** temperature 37.05
192.168.137.220 - - [16/Mar/2023 19:40:53] "PUT /api/sensor?temperature=37.05
HTTP/1.1" 200 -
***** temperature 39.15
192.168.137.220 - - [16/Mar/2023 19:40:54] "PUT /api/sensor?temperature=39.15
HTTP/1.1" 200 -
***** temperature 26.88
192.168.137.220 - - [16/Mar/2023 19:40:55] "PUT /api/sensor?temperature=26.88
HTTP/1.1" 200 -
***** temperature 20.75
192.168.137.220 - - [16/Mar/2023 19:40:59] "PUT /api/sensor?temperature=20.75
HTTP/1.1" 200 -
***** temperature 39.27
192.168.137.220 - - [16/Mar/2023 19:41:01] "PUT /api/sensor?temperature=39.27
HTTP/1.1" 200 -
***** temperature 38.47
192.168.137.220 - - [16/Mar/2023 19:41:02] "PUT /api/sensor?temperature=38.47
HTTP/1.1" 200 -
***** temperature 32.71
192.168.137.220 - - [16/Mar/2023 19:41:03] "PUT /api/sensor?temperature=32.71
HTTP/1.1" 200 -
***** temperature 30.58
```

src08