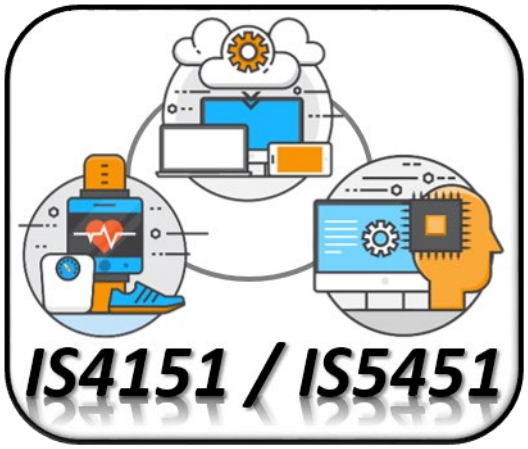


Introduction to ESP32

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.

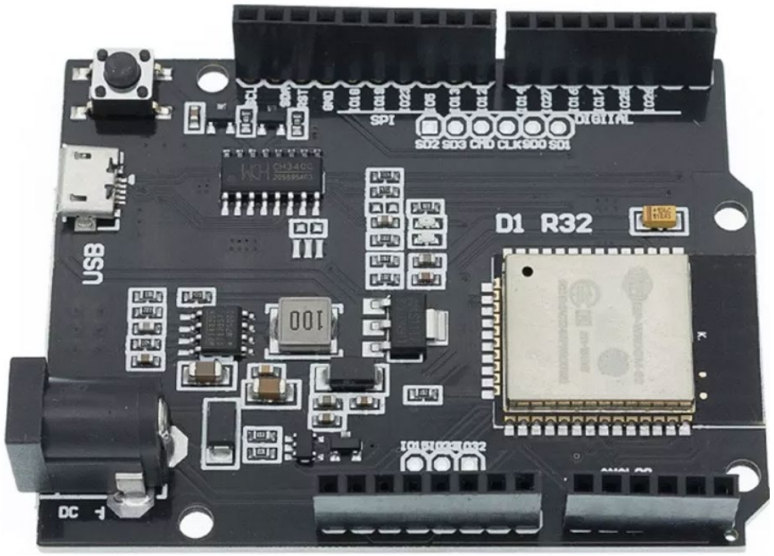
Overview of the ESP32

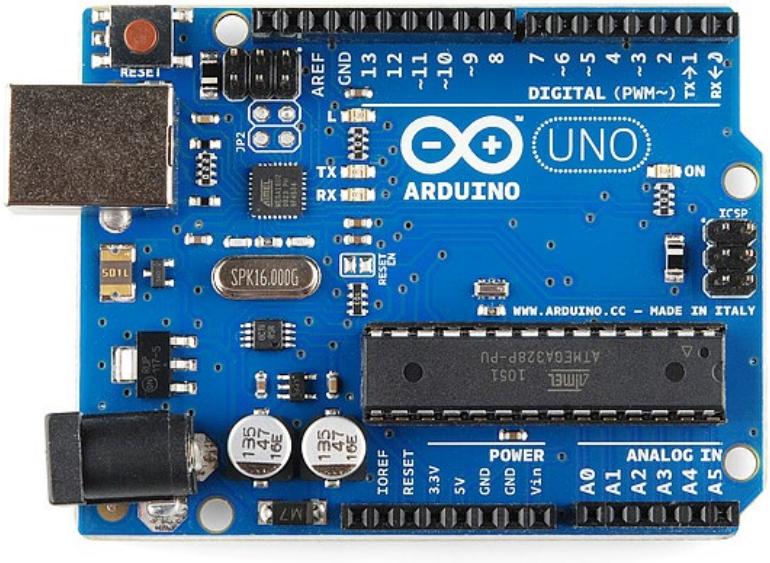
- **ESP32** is a series of microcontrollers with integrated WiFi and Bluetooth.
- ESP32 development boards such as **D1 R32** feature an Arduino UNO form factor.
- General technical specifications:

Dimension	D1 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.)

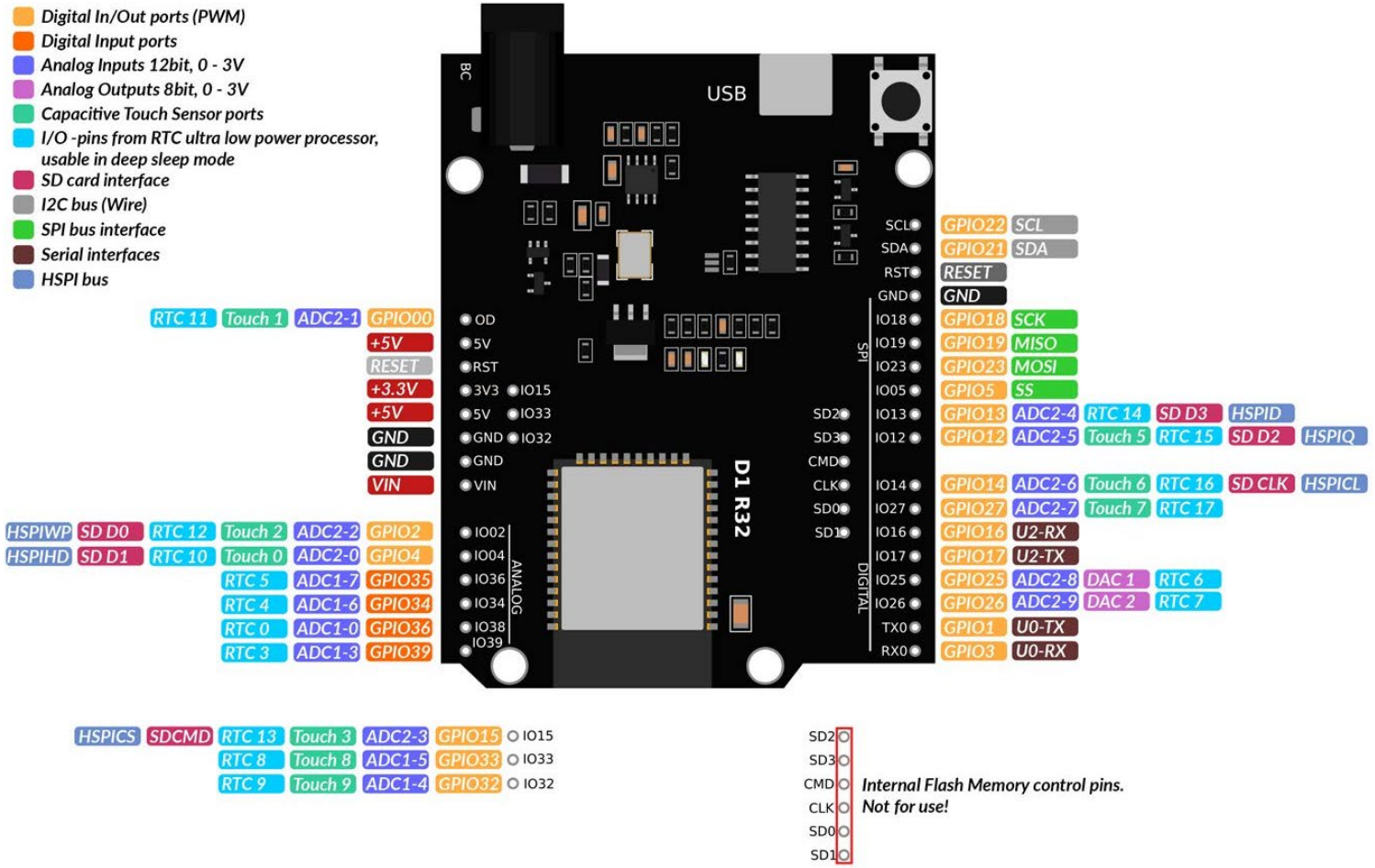
- The form factor of D1 R32 (left) is similar to an Arduino UNO (right):





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.

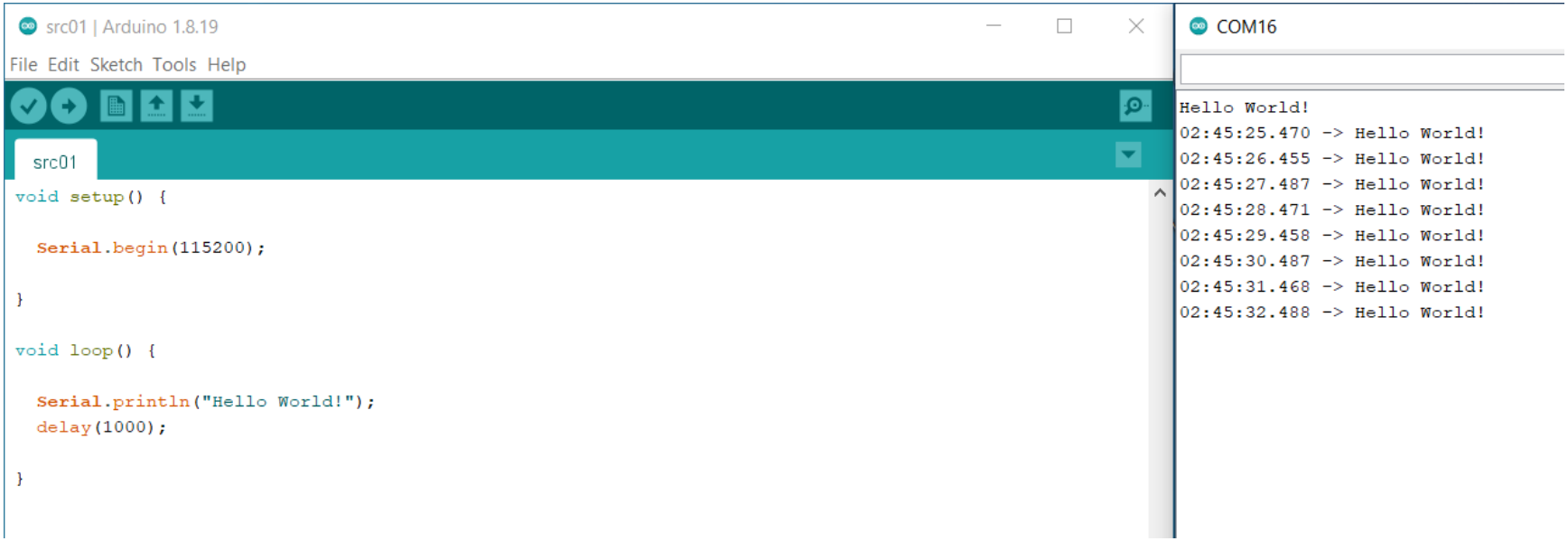
- There are two special functions that every Arduino sketch must have:
  - `setup()`
    - This function is called once when the sketch starts.
    - It is used to perform initialisation tasks such as setting pin modes.
  - `loop()`
    - This function is called repeatedly.
    - It contains the main 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.)

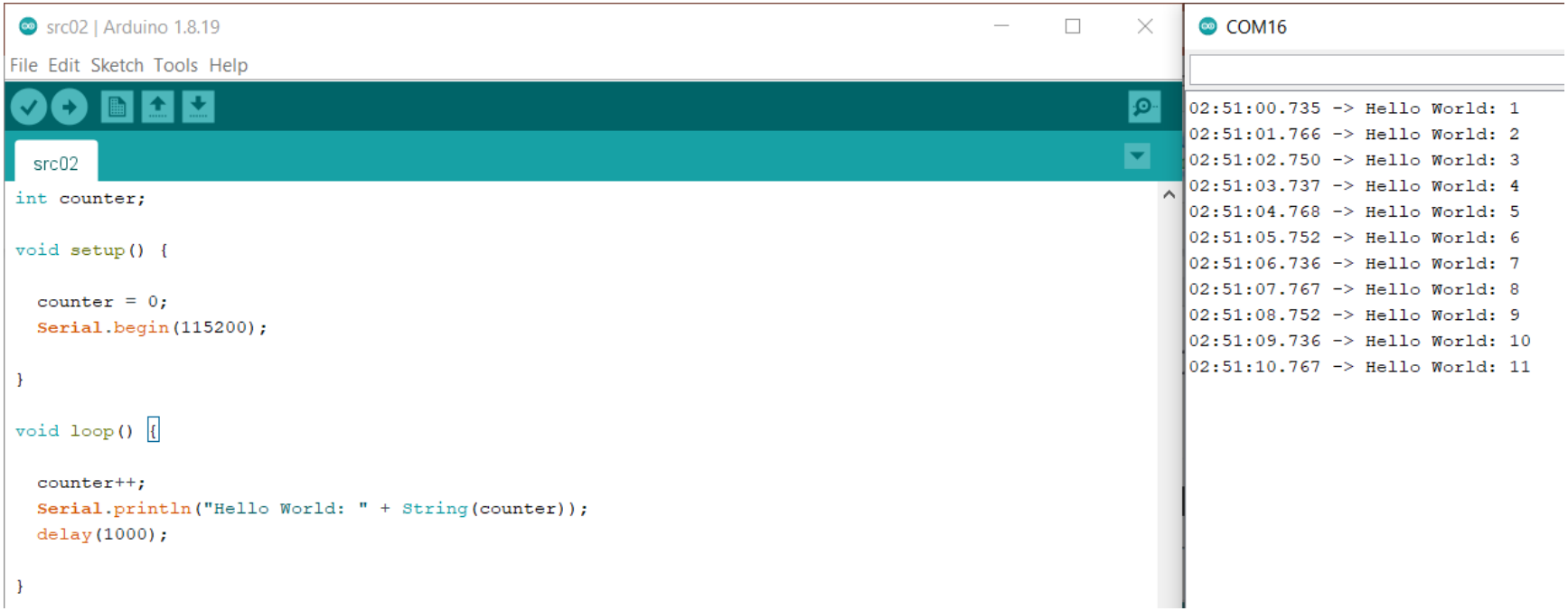
- Our first "Hello World" Arduino sketch:



src01.ino

Programming the D1 R32 (cont.)

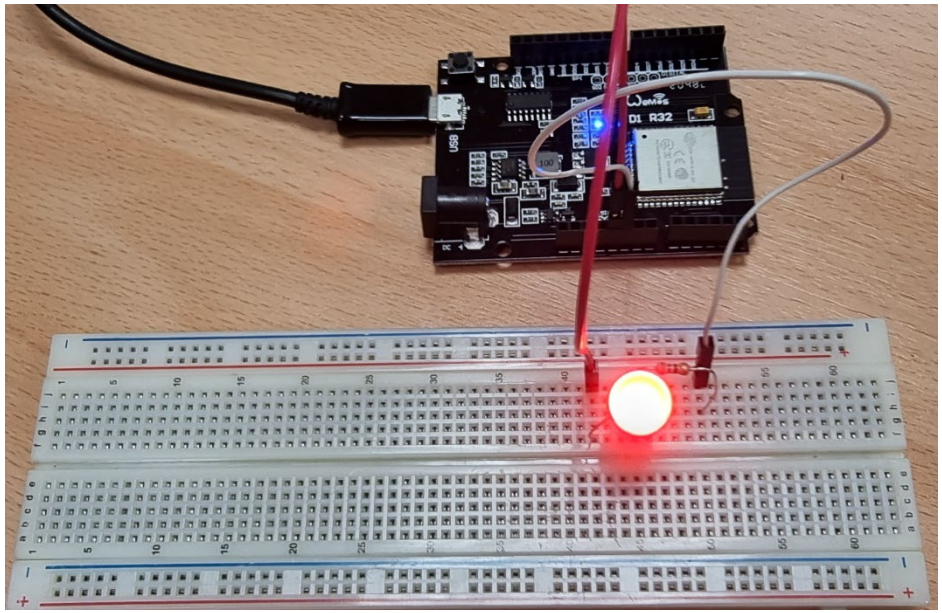
- Modifying the "Hello World" sketch to print out a running integer counter value:



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.



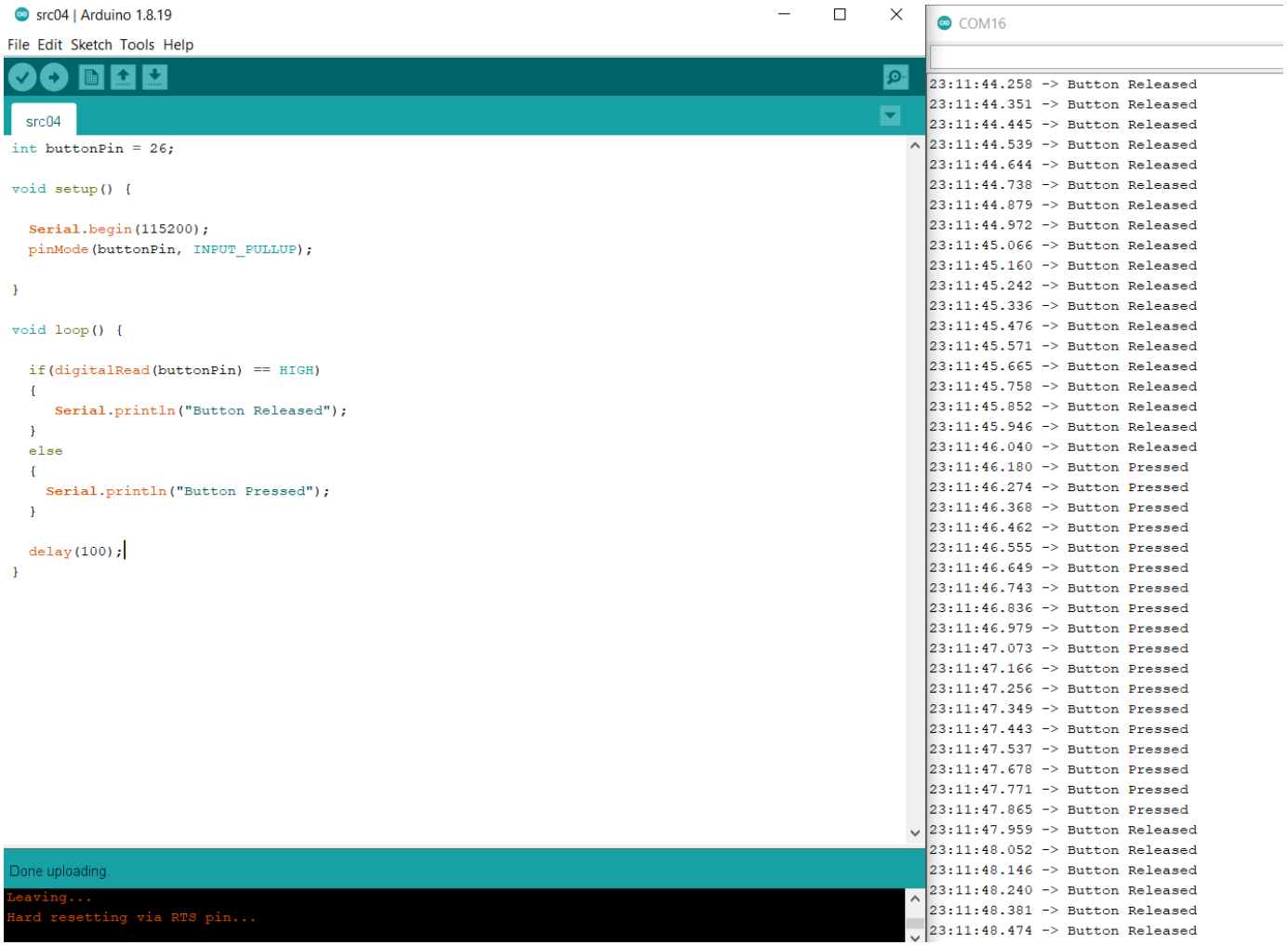
Hands-on with Digital Output

- 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.)



src04.ino

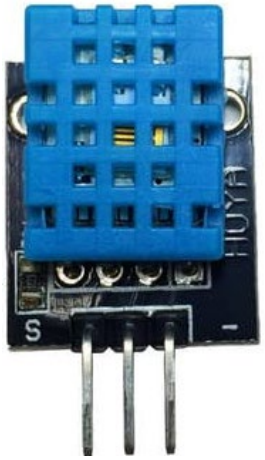
Hands-on with Digital Input (cont.)

- 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.

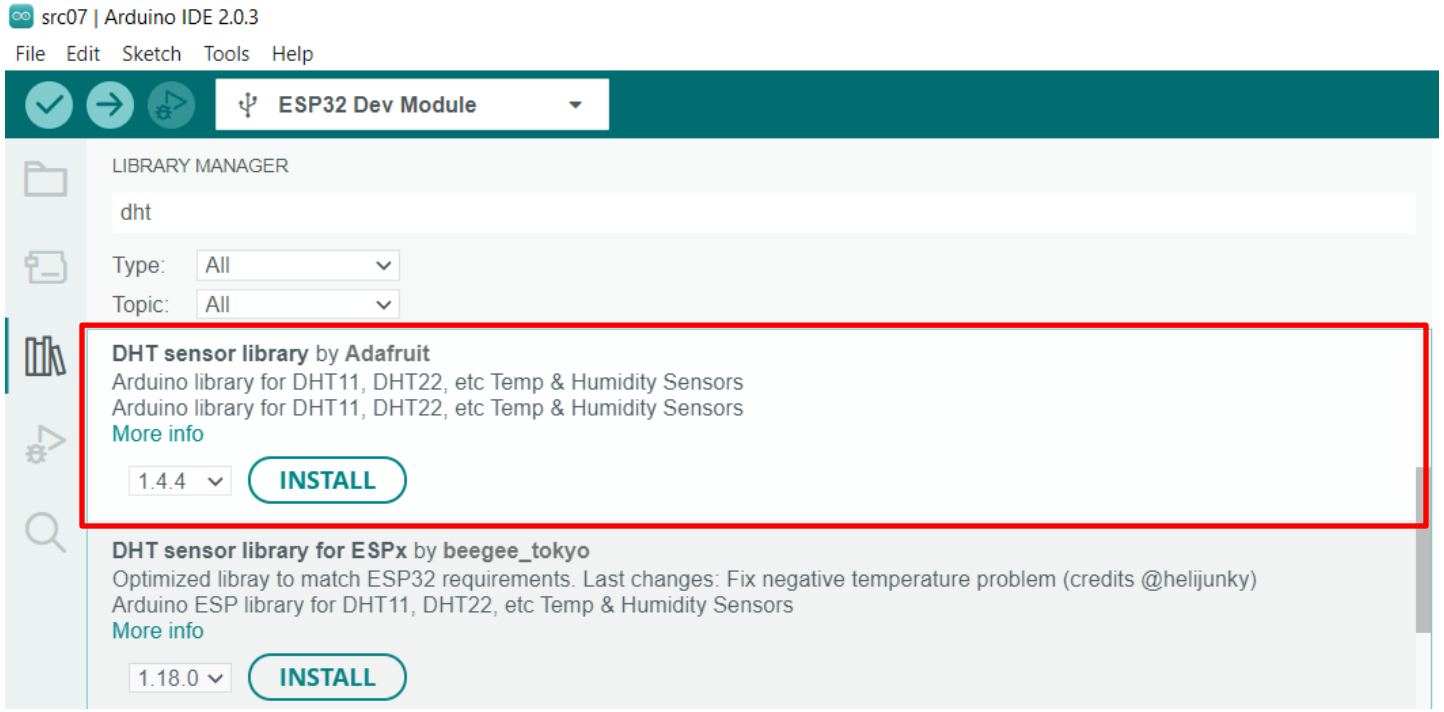
Reading Input Data from Sensors



- 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".

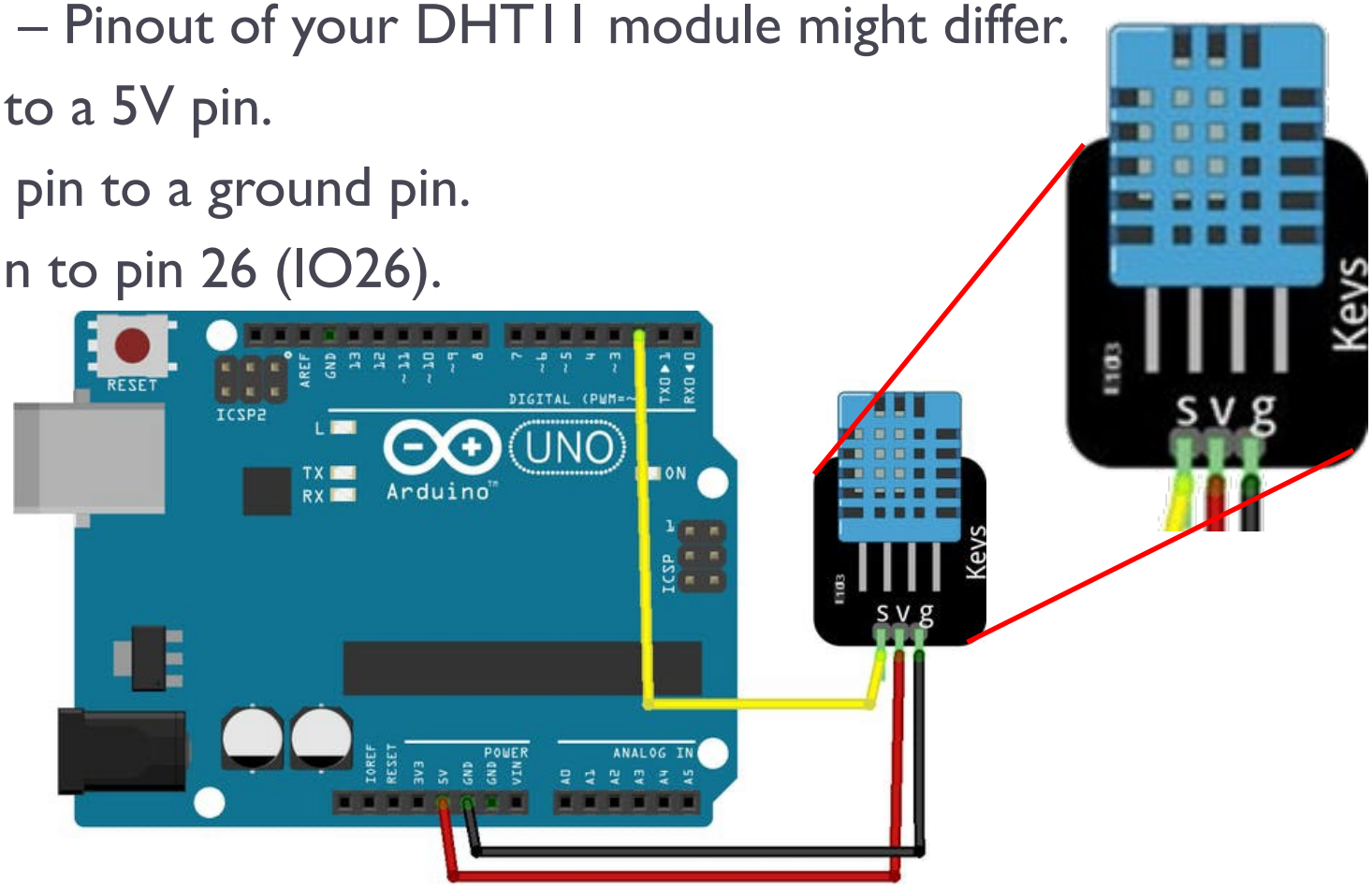
Hands-on with Sensor Input

- 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.)

src07 | Arduino 1.8.19

File Edit Sketch Tools Help

src07

```
#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);
}
```

COM16

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

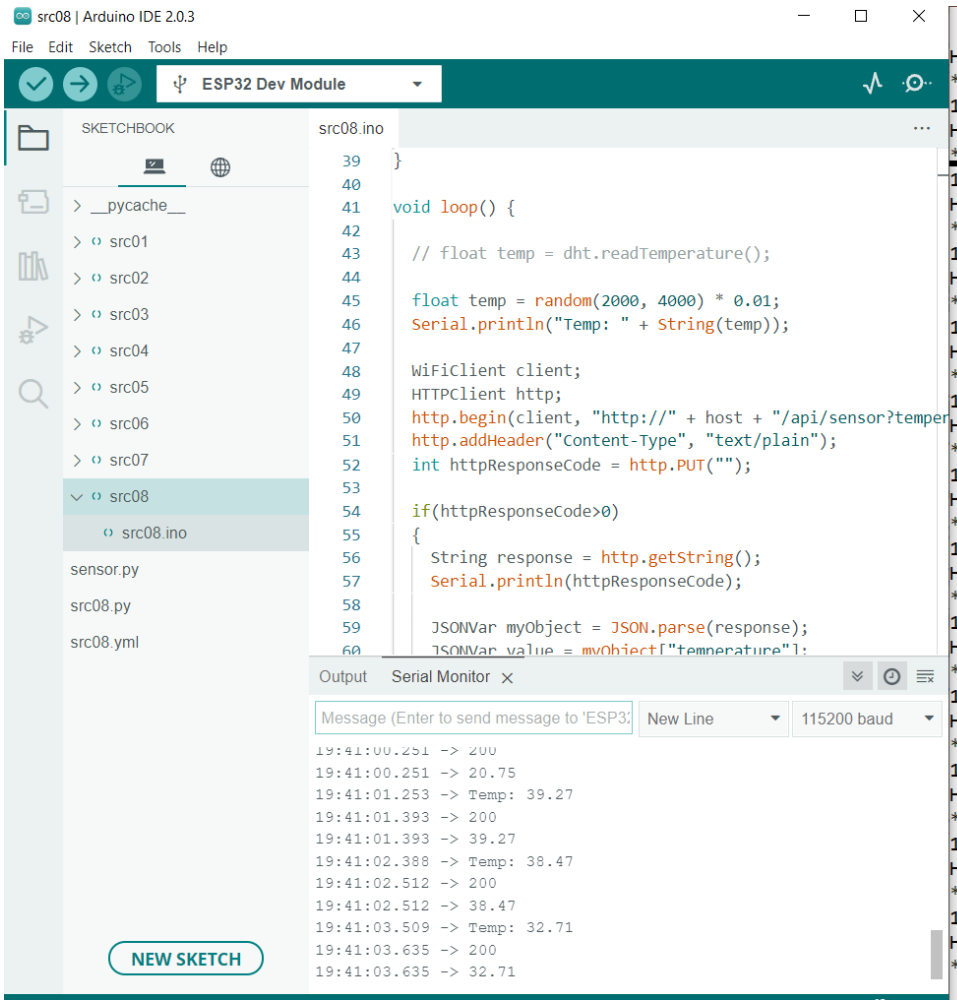
```
#include <HTTPClient.h>
```

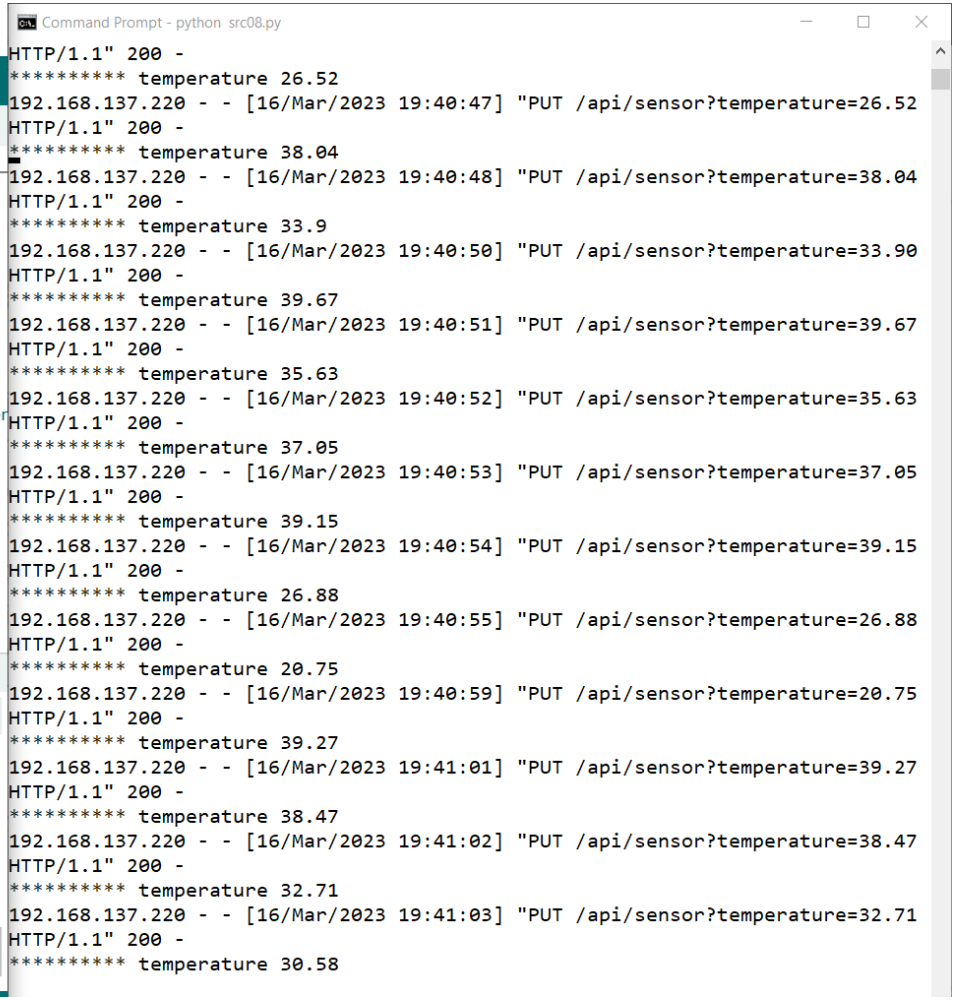
  - The JSON result can be decoded with the Arduino JSON library:

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

  - Refer to `src08` for the completed sample code.

Consuming REST Web Service in D1 R32 (cont.)





src08