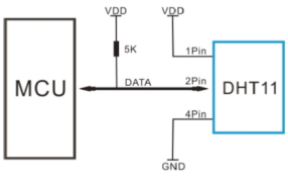
1. DHT11/DHT22 Temperature and Humidity Sensor

Block Diagram:



These sensors contain a chip that does analog to digital conversion and spits out a digital signal with the temperature and humidity.** There are two versions of the DHT sensor:

DHT11

Range: 20-90%

Absolute accuracy: ±5%

Repeatability: ±1%

Long term stability: ±1% per year

Price: $1 to $5

DHT22

Range: 0-100%

Absolute accuracy: ±2%

Repeatability: ±1%

Long term stability: ±0.5% per year

Price: $4 to $10

Pin Description:

Pin	Wiring to Arduino UNO
1st pin - VCC	5V
2nd pin - Data OUT	Digital pin 2
3rd pin	Don't connect
4th pin - GND	GND

Code:

```
DHT dht(DHTPIN, DHTTYPE);

dht.begin();

float h = dht.readHumidity();

float t = dht.readTemperature();

float f = dht.readTemperature(true);
```

How it works and what it does?

DHT11/DHT22 are sensors used to measure **temperature and humidity**.

They work by using a **capacitive humidity sensor** and a **thermistor** (temperature sensor) inside. The sensor detects changes in humidity and temperature and converts them into **digital signals** that a microcontroller (like Arduino) can read.

- **DHT11:** less accurate, works from 0–50°C and 20–80% humidity.
- **DHT22:** more accurate, works from -40–80°C and 0–100% humidity.

2. BMP180 Barometric Sensor

It measures the absolute pressure of the air around it. It has a measuring range from

300hPa to 1100hPa with an accuracy down to 0.02hPa. It can also measure altitude

and temperature.The BMP180 barometric sensor communicates via I2C interface. This means that it communicates with the Arduino using just 2 pins.

Pin Description:

Pin	Wiring to Arduino Uno
Vin	5V
GND	GND
SCL	A5
SDA	A4

** A5 and A4 works as both digital pin and analog pin. Here it works as digital pin.

Code:

```
SFE_BMP180 pressure;

if (pressure.begin())

Serial.println("BMP180 init success");

status = pressure.getTemperature(T);

status = pressure.startPressure(3);
```

3. FC-37 or YL-83 Rain Sensor

The rain sensor is used to detect water and it can detect beyond what a humidity sensor can. The FC-37 rain sensor (or other versions like YL-83) is set up by two pieces: the electronic board (at the left) and the collector board (at the right) that collects the water drops. The rain sensor has a built-in potentiometer for sensitivity adjustment of the digital output (D0). It also has a power LED that lights up when the sensor is turned on and a digital output LED.

How does it work?

In simple terms, the resistance of the collector board varies accordingly to the amount of water on its surface.

When the board is:

Wet: the resistance increases, and the output voltage decreases.

Dry: the resistance is lower, and the output voltage is higher.

Pin Description:

Sensor Pin	Wiring to Arduino Uno
A0	Any analog pin (A5 in this example)
D0	Digital pins
GND	GND
VCC	5V

Code:

```
int sensorValue = analogRead(rainPin);

if(sensorValue < thresholdValue){

Serial.println(" - It's wet");

digitalWrite(greenLED, LOW);

digitalWrite(redLED, HIGH);

}

else {

Serial.println(" - It's dry");
```

4.YL-69 or HL-69 Soil Moisture Sensor

The soil moisture sensor or the hygrometer is usually used to detect the humidity of the soil. So, it is perfect to build an automatic watering system or to monitor the soil moisture of your plants. The sensor is set up by two pieces: the electronic board (at the right), and the probe with two pads, that detects the water content (at the left). The sensor has a built-in potentiometer for sensitivity adjustment of the digital output (D0), a power LED and a digital output LED.

How does it work?

The voltage that the sensor outputs changes accordingly to the water content in the soil. When the soil is:

Wet: the output voltage decreases

Dry: the output voltage increases

The output can be a digital signal (D0) LOW or HIGH, depending on the water content. If the soil humidity exceeds a certain predefined threshold value, the modules outputs LOW, otherwise it outputs HIGH. The threshold value for the digital signal can be adjusted using the potentiometer. The output can also be an analog signal and so you'll get a value between 0 and 1023.

Pin Description:

Sensor Pin	Wiring to Arduino Uno
A0	Any analog pin (A5 in this example)
D0	Digital pins
GND	GND
VCC	5V

Code:

```
int sensorValue = analogRead(rainPin);
if(sensorValue < thresholdValue){
LIKE ARDUINO? GET 25 ARDUINO STEP-BY-STEP PROJECTS COURSE 30
Serial.println(" - Doesn't need watering");
digitalWrite(redLED, LOW);
digitalWrite(greenLED, HIGH);
}
else {
Serial.println(" - Time to water your plant");
digitalWrite(redLED, HIGH);
digitalWrite(greenLED, LOW);
```

5. DS18B20 Temperature Sensor

The DS18B20 temperature sensor is a 1-wire digital temperature sensor. It communicates on common bus, which means that you can connect several devices and read their values using just one digital pin of the Arduino.

Features

Here's some main features of the DS18B20 temperature sensor:

Communicates over 1-wire bus communication

Operating range temperature: -55oC to 125oC

Accuracy +/-0.5 oC (between the range -10oC to 85oC)

The **DS18B20** is a digital temperature sensor that measures temperature and sends the data as a digital signal through a single data wire. It works by detecting temperature changes inside its chip, converting them to digital form, and communicating with a microcontroller using the one-wire protocol. Each sensor has a unique address, allowing multiple sensors on the same line, and it accurately measures temperatures from **-55°C to +125°C**.

Pin Description:

Vcc – Vcc

Ground – Ground

Datapin – any digital pin of Arduino(D4)

Code:

DallasTemperature sensors(&oneWire);

sensors.requestTemperatures();

Serial.print(sensors.getTempCByIndex(0));

6. DS1307 or DS3231 Real Time Clock (RTC)

The **DS1307** or **DS3231** is a **Real-Time Clock (RTC)** module that keeps track of **time and date** even when the main power is off. It has a **built-in battery** that powers it continuously, so it remembers seconds, minutes, hours, days, months, and years. The module communicates with a microcontroller using the **I2C protocol** (two wires: SDA and SCL). The **DS3231** is more accurate than the DS1307 because it has a **built-in temperature-compensated crystal oscillator (TCXO)**.

Pin Description:

Module Pin	Wiring to Arduino Uno
SCL	A5
SDA	A4
VCC	5V
GND	GND

7. MQ-2 Gas/Smoke Sensor

The MQ-2 sensor detects smoke and flammable gases such as LPG, methane, and alcohol. Its resistance changes with the gas concentration, and a potentiometer lets you set a threshold so the sensor's digital pin (D0) goes HIGH when gas levels exceed that limit.

How does it work?

The voltage that the sensor outputs changes accordingly to the smoke/gas level that exists in the atmosphere. The sensor outputs a voltage that is proportional to the concentration of smoke/gas. In other words, the relationship between voltage and gas concentration is the following:

The greater the gas concentration, the greater the output voltage

lower the gas concentration, the lower the output voltage

Pin Description:

Sensor Pin	Wiring to Arduino Uno
A0	Analog pins
D0	Digital pins
GND	GND
VCC	5V

Code:

```
int analogSensor = analogRead(smokeA0);

if (analogSensor > sensorThres)

{

digitalWrite(redLed, HIGH);

digitalWrite(greenLed, LOW);

tone(buzzer, 1000, 200);

}

else

{

digitalWrite(redLed, LOW);

digitalWrite(greenLed, HIGH);

noTone(buzzer);
```

8. HC-SR04 Ultrasonic Sensor

The HC-SR04 ultrasonic sensor measures distance using sound waves, just like bats. It can detect objects from 2 cm to 400 cm (1 inch to 13 feet) without touching them. The sensor has a transmitter that sends sound waves and a receiver that listens for the echo to calculate the distance accurately.

How does it work?

The **HC-SR04 ultrasonic sensor** works by using **sound waves** to measure distance. It sends out an **ultrasonic pulse** from its **trigger pin**, which bounces off an object and returns to the **echo pin**. The sensor measures the **time** it takes for the sound to come back and uses that to calculate the **distance** with the formula:
Distance = (Time × Speed of Sound) / 2.
It's a simple and accurate way to detect objects without touching them.

Features

x Power Supply :+5V DC

x Quiescent Current : <2mA

x Working Current: 15mA

x Effectual Angle: <15°

x Ranging Distance : 2cm – 400 cm/1'' – 13ft

x Resolution : 0.3 cm

x Measuring Angle: 30 degree

x Trigger Input Pulse width: 10uS

x Dimensions: 45mm x 20mm x 15mm

Pin Description:

Sensor Pin	Wiring to Arduino Uno
VCC	5V
Trig	Trigger (INPUT) D9
Echo	Echo (OUTPUT) D12
GND	GND

Code:

```
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
cm = (duration/2) / 29.1;
inches = (duration/2) / 74;
```

9. Tilt Sensor:

The tilt sensor is many times referred to as inclinometer, tilt switch or rolling ball sensor. Using a tilt sensor is a simple way to detect orientation or inclination. The tilt sensor allows to detect orientation or inclination. It detects if the sensor is completely upright or if it is tilted.

This makes it very useful to be used, for example, in toys, robots and other appliances whose working methodology depends on inclination.

Here's how it works:

When the sensor is completely upright, the ball falls to the bottom of the sensor and connects the poles, allowing the current to flow.

When the sensor is tilted, the ball doesn't touch the poles, the circuit is open,

and the current doesn't flow.

This way, the tilt sensor acts like a switch that is turned on or off depending on its inclination. So, it will give digital information to the Arduino, either a HIGH or a LOW signal.

Pin wiring

Wiring the tilt sensor to you Arduino is straightforward. You just need to connect one pin to the Arduino digital pins and the GND to the GND.

Code:

```
int lastTiltState = HIGH;
sensorValue = digitalRead(sensorPin);
if (sensorValue == lastTiltState) {
lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > debounceDelay) {
lastTiltState = sensorValue;
}
digitalWrite(ledPin, lastTiltState);
```

10. Microphone Sound Sensor

The microphone sound sensor, as the name says, detects sound. It gives a measurement of how loud a sound is.

How does it work?

The **Microphone Sound Sensor** works by detecting **sound vibrations** in the air. The built-in **microphone** converts these sound waves into **electrical signals**. When the sound level goes above a certain **threshold**, the sensor outputs a **digital HIGH signal**, and for lower sounds, it stays **LOW**. Some modules also give an **analog output** showing the actual sound intensity.

Pin Description:

Sensor Pin	Wiring to Arduino Uno
A0	Analog pins
D0	Digital pins
GND	GND
VCC	5V

Code:

```
val =digitalRead(sensorPin);
if (val==HIGH) {
digitalWrite(ledPin, HIGH);
}
else {
digitalWrite(ledPin, LOW);
```

11. MRFC522 RFID

RFID means radio-frequency identification. RFID uses electromagnetic fields to transfer data over short distances. RFID is useful to identify people, to make transactions, etc.

You can use an RFID system to open a door. For example, only the person with the right information on his card can enter.

How does it work?

The **MFRC522 RFID** module works by using **radio waves** to read and write data from **RFID cards or tags**. When a card comes close, the module's **antenna** sends out a **13.56 MHz signal** that powers the tag (if it's passive). The tag then sends back its **unique ID** or stored data, which the MFRC522 reads and sends to the **microcontroller** (like Arduino) through the **SPI communication** interface. This allows identification or access control without physical contact.

Pin Description:

Sensor Pin	Wiring to Arduino Uno
SDA	Digital 10
SCK	Digital 13
MOSI	Digital 11
MISO	Digital 12
IRQ	Don't connect
GND	GND
RST	Digital 9
3.3V	3.3V

Code:

```
MFRC522 mfrc522(SS_PIN, RST_PIN);
mfrc522.PCD_Init();
if ( ! mfrc522.PICC_IsNewCardPresent())
{
return;
}
// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial())
{
return;
for (byte i = 0; i < mfrc522.uid.size; i++)
```

12. Relay Module

A relay is an electrically operated switch of mains voltage. It can be turned on or off,

letting the current go through or not.

A **Relay Module** works like a **remote switch** that lets a low-power microcontroller control a high-power device. When the microcontroller sends a **signal** to the relay, it **activates an internal electromagnet**, which **closes (or opens) the switch** inside. This allows **electricity to flow** to devices like lamps, motors, or fans safely, isolating the microcontroller from high voltage.

Pin Description:

GND: goes to ground

IN1: controls the first relay. Should be connected to an Arduino digital pin

IN2: controls the second relay. Should be connected to an Arduino digital

pin if you are using this second relay. Otherwise, you don't need to connect

it.

VCC: goes to 5V.

13. 8x8 Dot Matrix

The dot matrix is an 8×8 matrix which means that it has 8 columns and 8 rows, so it contains a total of 64 LEDs.

An **8x8 Dot Matrix** is a grid of **64 LEDs** arranged in 8 rows and 8 columns. Each LED can be **turned on or off** by controlling its **row and column**. By quickly switching LEDs on and off in patterns, you can **display letters, numbers, or simple images**. A **microcontroller** controls which LEDs light up, often using **row-column scanning** to save pins and power.

Pin Description:

Dot Matrix Pin	Wiring to Arduino Uno
GND	GND
VCC	5V
DIN	Digital pin
CS	Digital pin
CLK	Digital pin

setLed(addr, row, col, state)

addr is the address of your matrix, for example, if you have just 1 matrix,

row is the row where the LED is located

col is the column where the LED is located

state

lc.setRow(0,0,sf[0]);

14. Membrane Keypad

How it works?

A membrane keypad is a matrix consisting of rows and columns. Each key is

assigned to a certain row and column (see the picture below).

LIKE ARDUINO? GET 25 ARDUINO STEP-BY-STEP PROJECTS COURSE 115

On a 12 button keypad you have 4 rows and 3 columns. The first key would make a

link between Row 1 and Column 1 (R1C1). 2 would be R1C2, 3 R1C3, * R4C1, 9 R3C3

and so on.

Code:

```
char keys[ROWS][COLS] = {
{'1','2','3'},
{'4','5','6'},
{'7','8','9'},
{'#','0','*'}
};
byte rowPins[ROWS] = {8, 7, 6, 5};
byte colPins[COLS] = {4, 3, 2}; // column pinouts of the keypad C1 = D4, C2 = D3,
C3 = D2
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
char key = keypad.getKey();
if (key != NO_KEY)
Serial.println(key);
```

15. SIM900 GSM GPRS Shield

GSM stands for Global System for Mobile Communications and is the global standard for mobile communications.

GPRS stands for General Packet Radio Service. GPRS is a mobile service on the 2G and 3G cellular communication.

Applications:

The GSM GPRS shield is particularly useful as it allows to:

Connect to the Internet over GPRS network

Send and receive SMS

Place and receive phones calls

Its capabilities make it perfect for projects with Arduino like:

Remote control of electronic appliances – sending an SMS to turn something on;

Receive notifications – send SMS to your cell phone if movement is detected in your house;

Receive sensor data – send periodic SMS to your cell phone with daily weather

data.

LIKE ARDUINO? GET 25 ARDUINO STEP-BY-STEP PROJECTS COURSE 128

Features

Here's some of the most important features of the shield:

Compatible with Arduino and clones

Based on SIM900 module from SIMCOM

Allows you to send SMS, MMS, GPRS and Audio via UART using AT commands.

It has 12 GPIOs, 2 PWMs and built-in ADC of the SIM900 module

Quad Band: 850; 900; 1800 and 1900 MHZ, so it should work in all countries with GSM (2G) networks

Control via AT commands

Supports RTC (real time clock) – it has a holder for a 3V CR1220 battery at the back

Has microphone and headphone jacks for phone calls

16. TCS3200 Color Sensor

The TCS3200 color sensor can detect a wide variety of colors based on their wavelength. This sensor is especially useful for color recognition projects such as color matching, color sorting, test strip reading and much more.

The TCS3200 color sensor – shown in the figure below – uses a TAOS TCS3200 RGB sensor chip to detect color. It also contains four white LEDs that light up the object in front of it.

How does the TCS3200 sensor work?

The TCS3200 has an array of photodiodes with 4 different filters. A photodiode is simply a semiconductor device that converts light into current. The sensor has:

x 16 photodiodes with red filter – sensitive to red wavelength

x 16 photodiodes with green filter – sensitive to green wavelength

x 16 photodiodes with blue filter – sensitive to blue wavelength

x 16 photodiodes without filter

If you take a closer look at the TCS3200 chip you can see the different filters.

By selectively choosing the photodiode filter's readings, you're able to detect the intensity of the different colors. The sensor has a current-to-frequency converter that converts the photodiodes' readings into a square wave with a frequency that is proportional to the light intensity of the chosen color. This frequency is then, read by the Arduino.

Pin Description:

S1	Digital pin 5
VCC	5V
S3	Digital pin 6
S4	Digital pin 7
OUT	Digital pin 8

17. PIR Motion Sensor

The PIR motion sensor is ideal to detect movement. PIR stand for "Passive Infrared". Basically, the PIR motion sensor measures infrared light from objects in its field of view. So, it can detect motion based on changes in infrared light in the environment. It is ideal to detect if a human has moved in or out of the sensor range.

Pin wiring

Sensor Pin	Wiring to Arduino Uno
GND	GND
OUT	Arduino digital pin
5V	5V

Code:

```
int state = LOW;
val = digitalRead(sensor); // read sensor value
if (val == HIGH) { // check if the sensor is HIGH
digitalWrite(led, HIGH);
if (state == LOW) {
Serial.println("Motion detected!");
state = HIGH; // update variable state to HIGH
}
}
```

else {

digitalWrite(led, LOW);

if (state == HIGH){

Serial.println("Motion stopped!");

state = LOW;

⚖️ Weighing Load Sensor (Load Cell)

- It measures **weight or force**.
- Inside it has a **small wire** that changes its resistance when you put weight on it.
- This small change turns into a **voltage signal** using a special circuit.
- The signal goes to a **microcontroller**, which shows the **weight**.
- 📊 *Used in digital scales and machines that measure heavy loads*.

⚖️ 1. Weighing Load Sensor (Load Cell + HX711 Module)

Pin (Load Cell / HX711)	Connects To (Arduino)
VCC	5V
GND	GND
DT (Data)	D3
SCK (Clock)	D2
E+ (Red wire)	HX711 E+
E− (Black wire)	HX711 E−
A+ (Green wire)	HX711 A+
A− (White wire)	HX711 A−

Code:

Initialize HX711 module with data pin D3 and clock pin D2

Set calibration factor (for weight conversion)

Loop Forever:

  Read weight value from HX711

  If weight > 0:

    Display weight on serial monitor

  Else:

    Display "No load"

  Wait 500 milliseconds

End Loop

Light Sensor (LDR)

- It measures **light level**.
- When **light increases**, its **resistance decreases**.
- When **dark**, resistance becomes **high**.
- The microcontroller reads these changes to know if it's **bright or dark**.

Used in street lights, cameras, and smart systems.

***Code**:*

Start

Set LDR pin as Analog Input (A0)

Loop Forever:

Read analog value from LDR

If value > threshold:

Print "Bright light detected"

Else:

Print "Dark environment"

Wait 500 milliseconds

End Loop

2. Light Sensor (LDR)

Pin (LDR Circuit)	Connects To (Arduino)	F
One end of LDR	5V	F
Other end of LDR	A0	A
10kΩ Resistor	Between A0 and GND	F r

IR Sensor (Infrared Sensor)

- It finds **objects or movement** using **infrared light**.
- **Active IR:** sends light and catches it when it bounces back from an object.
- **Passive IR (PIR):** senses **body heat** or movement.

Used in robots, alarms, and automatic doors.

***Code**:*

Set IR sensor pin as Digital Input (D4)

Loop Forever:

Read signal from IR sensor

If signal == LOW:

Print "Object detected"

Else:

Print "No object"

Wait 200 milliseconds

👁 3. IR Sensor (Active Type)

Pin (IR Sensor)	Connects To (Arduino)
VCC	5V
GND	GND
OUT	D4

DC motor , Stepper Motor , Servo Motor , Motor Drive

⚙ DC Motor vs Stepper Motor vs Servo Motor

Feature	DC Motor	Stepper Motor	Servo Motor
Type of Motion	Continuous rotation	Moves in small steps	Rotates to specific angle
Control Type	Speed and direction	Position (by steps)	Position (by angle)
Rotation	Free and continuous	Step-by-step (discrete)	Limited (usually 0°–180°)
Accuracy	Low	High (controlled steps)	Very high (feedback-based)
Speed Control	By voltage or PWM signal	By step frequency	By PWM signal
Torque	High at high speed	High at low speed	Moderate
Feedback	No feedback	Open-loop (some closed-loop)	Closed-loop (uses feedback)

Driver Used	L293D, L298N	ULN2003, A4988	Direct Arduino pin or driver
Common Uses	Fans, wheels, pumps	3D printers, CNC, robots	Cameras, robot arms, RC planes

---

🛠️ Motor Driver Overview

A **motor driver** is an **electronic bridge** between your **microcontroller** (like Arduino, Raspberry Pi, or ESP32) and a **motor** (DC, stepper, or servo). Microcontrollers cannot supply the current required by motors, so motor drivers act as **current amplifiers** while also controlling direction, speed, and torque.

⚡ Key Functions of Motor Drivers

1. **Current Amplification:** Microcontrollers output low current (~20–40 mA). Motors need more (hundreds of mA to several amps). Motor drivers handle this safely.
2. **Direction Control:** Allows forward/reverse motion for DC motors, and precise stepping for stepper motors.
3. **Speed Control:** PWM (Pulse Width Modulation) is used to vary motor speed smoothly.
4. **Protection:** Many drivers include features like overcurrent, overheating, and voltage spike protection.

🧿 In short:

- **DC Motor** → Spins freely — speed-focused.
- **Stepper Motor** → Moves precisely — position-focused.
- **Servo Motor** → Turns to angles — precision and feedback-focused.
- **Motor Driver** → Acts like a bridge — gives power and control signals to motors.

differences between 2D printer and 3D printer

Feature	2D Printer	3D Printer
Output	Flat images or text on paper	Physical 3D objects
Dimension	2D (length × width)	3D (length × width × height)
Material Used	Ink on paper	Plastic filament, resin, metal, or other materials
Printing Method	Inkjet, laser, or dot matrix	Layer-by-layer deposition (FDM, SLA, SLS, etc.)
Purpose	Documents, photos, labels	Prototypes, models, functional parts
Time Required	Seconds to minutes	Hours to days, depending on size/complexity
Complexity	Simple	More complex (requires 3D model)

Software Needed	Basic drivers	CAD software + slicing software
Cost	Low	Higher (printer + material + maintenance)

💡 **In short:**

- **2D printer → paper and ink, flat output**
- **3D printer → real objects, builds layer by layer in space**

📊 Display Module Comparison

Display Module	Power Efficiency	Display Area / Resolution	Data Type	Contrast / Visibility	Cost	Notes
MAX7219 LED Dot Matrix (8×8 per module)	Low (high current draw)	Moderate; scalable by daisy-chaining multiple matrices	Text, numbers, simple animations	Good in bright/dim conditions	Moderate	Great for scrolling text; pixel-based, so limited graphics resolution
TM1637 4-Digit 7-Segment	High (low current)	Small (4 digits only)	Numbers only	Excellent	Low	Simple numeric displays, compact, low power
OLED Graphic Display (e.g., 128×64)	Moderate (depends on brightness; some sleep modes)	Medium; pixel-based, can display graphics & text	Text, numbers, graphics, emojis	Excellent (self-illuminated)	Moderate-High	High flexibility; sharp visuals, supports custom fonts & icons
7-Segment Display (common cathode/anode)	High	Very small (1–8 digits typical)	Numbers only	Good in bright light	Low	Low-cost numeric display; good for clocks

						or counters
I2C LCD (e.g., 16×2 or 20×4)	Moderate	Small to medium	Text, numbers (limited graphics via custom chars)	Good	Low	Easy to interface; limited graphics
16×2 Character LCD Module	Moderate	16×2 characters	Text, numbers	Moderate	Low	Standard, widely used; requires more pins unless using I2C backpack
Nokia 5110 Graphic LCD	Moderate	84×48 pixels	Text, numbers, basic graphics	Moderate	Low	Simple graphic display; low resolution but versatile for small graphics

💡 Key Observations

1. **Power Efficiency:** TM1637 & 7-segment wins for numeric-only displays; dot matrix consumes the most power. OLEDs can be power-efficient if dimmed or sleep modes used.
2. **Display Area / Graphics:** OLED > MAX7219 > Nokia 5110 > 16×2 LCD. 7-segment & TM1637 are purely numeric, tiny area.
3. **Data Versatility:** OLED excels (text + graphics + emojis), dot matrix can scroll text & simple shapes, LCDs are limited to characters.
4. **Contrast & Visibility:** OLED and TM1637 shine; dot matrix can be less visible in bright sunlight.
5. **Cost:** Low-cost options: 7-segment, TM1637, 16×2 LCD; Moderate: MAX7219, OLED; Nokia 5110 also low-cost but limited resolution.