

Arduino

- [Arduino's overview](#)
- [Temperatuur en luchtvochtigheid](#)
- [Display](#)
- [Thermometer bewegingsmelder en clock](#)
- [Arduino projects ideas](#)
- [PageESP32-WROOM-32S 32d is](#)

Arduino's overview

Entry-Level Boards

1. Arduino UNO R4 Minima

- **Price:** ~\$20
- **Main Purpose:** Beginner-friendly board for learning electronics and programming.
- **Capabilities:** HID support (keyboard/mouse emulation), no built-in Wi-Fi or Bluetooth.
- **Memory:** 32 KB SRAM, 256 KB Flash, 8 KB EEPROM.

2. Arduino UNO R4 WiFi

- **Price:** ~\$27
- **Main Purpose:** Enhanced UNO for IoT and creative projects with wireless features.
- **Capabilities:** Wi-Fi, Bluetooth (via ESP32-S3), HID support, 12x8 LED matrix, Qwiic connector.
- **Memory:** 32 KB SRAM (RA4M1), 512 KB SRAM (ESP32-S3), 256 KB Flash (RA4M1).

3. Arduino Nano Every

- **Price:** ~\$13
 - **Main Purpose:** Compact, affordable board for small projects and prototyping.
 - **Capabilities:** No Wi-Fi or Bluetooth, no HID support natively.
 - **Memory:** 6 KB SRAM, 48 KB Flash.
-

Enhanced Boards

4. Arduino Mega 2560 Rev3

- **Price:** ~\$48
- **Main Purpose:** Advanced projects needing many I/O pins and higher memory.
- **Capabilities:** No Wi-Fi or Bluetooth, no HID support natively, 54 digital I/O pins.
- **Memory:** 8 KB SRAM, 256 KB Flash, 4 KB EEPROM.

5. Arduino Due

- **Price:** ~\$45
 - **Main Purpose:** High-performance projects requiring 32-bit processing.
 - **Capabilities:** No Wi-Fi or Bluetooth, HID support, DAC for analog output.
 - **Memory:** 96 KB SRAM, 512 KB Flash.
-

IoT-Focused Boards

6. Arduino Nano 33 IoT

- **Price:** ~\$25
- **Main Purpose:** Small IoT projects with wireless connectivity.
- **Capabilities:** Wi-Fi, Bluetooth, HID support, onboard crypto chip.
- **Memory:** 32 KB SRAM, 256 KB Flash.

7. Arduino Nano 33 BLE

- **Price:** ~\$28
- **Main Purpose:** IoT and wearable projects with Bluetooth focus.
- **Capabilities:** Bluetooth Low Energy (BLE), HID support, no Wi-Fi, onboard IMU.

- **Memory:** 256 KB SRAM, 1 MB Flash.

8. Arduino Nano ESP32

- **Price:** ~\$20
- **Main Purpose:** Compact IoT with high-performance ESP32 capabilities.
- **Capabilities:** Wi-Fi, Bluetooth, HID support, MicroPython support.
- **Memory:** 520 KB SRAM, 16 MB Flash (external).

9. Arduino MKR WiFi 1010

- **Price:** ~\$40
- **Main Purpose:** IoT projects with secure wireless communication.
- **Capabilities:** Wi-Fi, Bluetooth, HID support, onboard crypto chip.
- **Memory:** 32 KB SRAM, 256 KB Flash.

10. Arduino GIGA R1 WiFi

- **Price:** ~\$75
- **Main Purpose:** Advanced, large-scale projects with multimedia and connectivity.
- **Capabilities:** Wi-Fi, Bluetooth, HID support, dual-core MCU, camera/display support.
- **Memory:** 576 KB SRAM (M7 core) + 256 KB (M4 core), 2 MB Flash.

11. Arduino Portenta H7

- **Price:** ~\$110
- **Main Purpose:** Industrial-grade, high-performance IoT and AI applications.
- **Capabilities:** Wi-Fi, Bluetooth, HID support, dual-core MCU, high-speed I/O.
- **Memory:** 2 MB SRAM (total), 16 MB Flash (external).

Temperatuur en luchtvochtigheid

Temperatuur en luchtvochtigheid

Temperatuur.jpg
Image not found or type unknown

DHT11

image.png
Image not found or type unknown

DHT11; van links naar rechts: *signaal* (naar port 2), *3.3V*, *GRND*.

Let op er zijn meerdere varianten en bij de meeste zit de data in het midden.

(<https://elektronicavoorjou.nl/product/dht11-temperatuur-en-vochtigheid-sensor/>)

Output

```
1
Temperature: 20.5 °C
Humidity: 45 %
2
Temperature: 20.5 °C
Humidity: 45 %
...
```

Code

```
#include <DHT.h>

#define DHTPIN 2 // Pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // Specify DHT11 sensor
```

```
DHT dht(DHTPIN, DHTTYPE);

int count;

void setup() {
  Serial.begin(9600);
  dht.begin();
  count = 0;
  delay(5000);
}

void loop() {
  count++;
  delay(2000); // Wait 2 seconds between readings (DHT11 needs time)

  float temp = dht.readTemperature(); // Read temperature as a float
  int humidity = dht.readHumidity();

  if (isnan(temp)) {
    Serial.println("Failed to read from DHT sensor!");
  } else {
    Serial.println(count);

    Serial.print("Temperature: ");
    Serial.print(temp, 1);
    Serial.println(" °C");

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");

    Serial.println("");
  }
}
```

Display

1602 LCD Module Display Bundle with I2C interface 2x16 Characters

image.png and or type unknown

Aansluitschema

image.png and or type unknown

Let op: de pin configuratie verschilt per type Arduino

Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL) of SDA1, SCL1

Links

Arduino Lessen: <https://arduino-lessen.nl/>

EBook: <https://azde.ly/TF12MBFS>

Voorbeeld code: [Arduino-Beginners-NL/E11-I2C-LCD/i2c-lcd-deel-1.ino at master · BasOnTech/Arduino-Beginners-NL · GitHub](#)

Thermometer bewegingsmelder en clock

Thermometer bewegingsmelder en clock

Het display laat de temperatuur en luchtvochtigheid zien en toont de trend met een pijltje omhoog of naar beneden.

Op de tweede regel staat de datum en tijd.

Het display gaat alleen 'aan' als er beweiging wordt geconstateerd.

Gemeten stroom is ongeveer 40 mA voor de gehele schakeling.

[Screenshot 2025-03-22 110222.jpg](#)

Aansuitschema

PIR Bewegingsmelder

[shopping.webp](#)

Van onder gezien van links naar rechts

(Let op er zijn verschillende pin configuraties. Verwijder lens/cap om de pin configuratie te zien).

- Plus 5V
- Signaal naar Pin 12 Arduino Leonardo.
- Min

Signaal naar Digitaal pin 12 Arduino Leonardo

Clock DS 3231

[Untitled.jpg](#)

Screenshot 2025-03-22 18:41:22.jpg

<http://www.rinkydinkelectronics.com/library.php?id=73>

Van links naar rechts van onderkant (niet batterij kant) gezien en *laatste* 4 pootjes.

- SCL naar SCL op Arduino (op Leonarde meest rechter pin)
- SDA naar SDA op Arduino (op Leonarde op een na meest rechter pin)
- Plus 3.3V
- Min

Thermometer

image.png and or type unknown

Van links naar rechts (gaatjes van blauwe blokje boven).

- naar pin 4 Arduino Leonarde
- Plus 3.3V
- Min

Zie <https://www.roc.ovh/books/arduino/page/temperatuur-en-luchtvochtigheid>

Display

image.png and or type unknown

Zie <https://www.roc.ovh/books/arduino/page/display>

Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <DS3231.h>

#define DHTPIN 4           // Pin connected to the DHT11 data pin
#define DHTTYPE DHT11      // Specify DHT11 sensor
#define SIGNAL_INTERVAL 600000 // 10 minutes
```

```
DHT dht(DHTPIN, DHTTYPE);

DS3231 myRTC;
bool century = false;
bool h12Flag;
bool pmFlag;

int count = 0;
int pirPin = 12;      // Pin for the HC-S501 sensor
int pirValue;

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);
unsigned long previousMillis = 0;
const long dhtInterval = 2000; // Interval for DHT readings

// Custom characters for the LCD arrows
byte downChar[] = {
  B00000,
  B00000,
  B00100,
  B00100,
  B00100,
  B10101,
  B01110,
  B00100
};

byte upChar[] = {
  B00100,
  B01110,
  B10101,
  B00100,
  B00100,
  B00100,
  B00100,
  B00000,
  B00000
};

byte degreeChar[] = {
```

```

B00110,
B01001,
B01001,
B00110,
B00000,
B00000,
B00000,
B00000
};

//-----
// Generic MeasurementSensor Class Template
//-----

template <typename T>
class MeasurementSensor {
private:
    T measurement;          // Current measurement value
    int arrow;              // Arrow indicator: 0 = no arrow, 1 = down, 2 = up
    unsigned long arrowMillis; // Last time the arrow was updated/reset
    const long signalInterval; // Time interval to reset the arrow

public:
    // Constructor: initializes with an initial measurement value.
    MeasurementSensor(long sigInterval, T initValue)
        : measurement(initValue), arrow(0), arrowMillis(0), signalInterval(sigInterval) {}

    // Update the measurement reading and determine the arrow indicator
    void update(T newMeasurement, unsigned long currentMillis) {
        // If the new measurement equals the previous value and the signal interval has passed, reset arrow.
        if (newMeasurement == measurement && currentMillis - arrowMillis >= signalInterval) {
            arrow = 0;
        } else {
            if (newMeasurement > measurement) {
                arrow = 2; // Up arrow
                arrowMillis = currentMillis;
            }
            if (newMeasurement < measurement) {
                arrow = 1; // Down arrow
                arrowMillis = currentMillis;
            }
        }
    }
};

```

```

    }

    // If the previous value is the initial invalid value, clear the arrow indicator.
    if (measurement == static_cast<T>(-99)) {
        arrow = 0;
    }

    measurement = newMeasurement;
}

// Getters for measurement and arrow
T getMeasurement() const { return measurement; }
int getArrow() const { return arrow; }
};

//-----
// Global Instances for Temperature and Humidity
//-----
MeasurementSensor<float> tempSensor(SIGNAL_INTERVAL, -99.0);
MeasurementSensor<int> humiditySensor(SIGNAL_INTERVAL, -99);

void setup() {
    Serial.begin(9600);
    dht.begin();
    delay(2000);

    lcd.init();
    lcd.backlight();
    lcd.clear();
    lcd.createChar(1, downChar);
    lcd.createChar(2, upChar);
    lcd.createChar(3, degreeChar);

    pinMode(pirPin, INPUT);

    // myRTC.setYear(2025);
    // myRTC.setMonth(3);
    // myRTC.setDate(20);
    // myRTC.setHour(23);
    // myRTC.setMinute(49);

```

```
// myRTC.setSecond(0);
}

void print2digits(int number) {
  if (number < 10) {
    lcd.print("0");
  }
  lcd.print(number, DEC);
}

void loop() {
  delay(20);

  // lcd.print(":");
  // print2digits(myRTC.getSecond());

  // Check for movement
  pirValue = digitalRead(pirPin);
  if (pirValue) {
    lcd.backlight();
  }

  // Do we need to update the display?
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= dhtInterval) {
    previousMillis = currentMillis;

    // Update backlight based on PIR sensor
    pirValue = digitalRead(pirPin);
    if (!pirValue) {
      lcd.noBacklight();
    }

    count++;

    // Read new values from the DHT sensor
    float newTemperature = dht.readTemperature();
    int newHumidity = dht.readHumidity();

    if (isnan(newTemperature) || isnan(newHumidity)) {
```

```
Serial.println("Failed to read from DHT sensor!");
return;
}

// Update our measurement sensors
tempSensor.update(newTemperature, currentMillis);
humiditySensor.update(newHumidity, currentMillis);

// Print sensor readings to the Serial Monitor
Serial.print(count);
Serial.print(", Temperature: ");
Serial.print(tempSensor.getMeasurement(), 1);
Serial.print("°C");
Serial.print(", Humidity: ");
Serial.print(humiditySensor.getMeasurement());
Serial.print("%");
Serial.println("");
Serial.println(myRTC.getSecond(), DEC);
Serial.println("");

// Update the LCD display for temperature
lcd.setCursor(0, 0);
if (tempSensor.getArrow()) {
    lcd.write(tempSensor.getArrow());
} else {
    lcd.print(" ");
}
lcd.print(tempSensor.getMeasurement(), 1);
lcd.write(3); // degree symbol
lcd.print("C ");

// Update the LCD display for humidity (set cursor on second row)
lcd.setCursor(12, 0);
if (humiditySensor.getArrow()) {
    lcd.write(humiditySensor.getArrow());
} else {
    lcd.print(" ");
}
lcd.print(humiditySensor.getMeasurement());
lcd.print("%");
```

```

    lcd.setCursor(1, 1);
    print2digits(myRTC.getDate());
    lcd.print("-");
    print2digits(myRTC.getMonth(century));

    lcd.setCursor(10, 1);
    print2digits(myRTC.getHour(h12Flag, pmFlag));
    lcd.print(":");
    print2digits(myRTC.getMinute());
  }
}

```

Code met highest/lowest

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <DS3231.h>

#define DHTPIN 4           // Pin connected to the DHT11 data pin
#define DHTTYPE DHT11      // Specify DHT11 sensor
#define SIGNAL_INTERVAL 600000 // 10 minutes

DHT dht(DHTPIN, DHTTYPE);

DS3231 myRTC;
bool century = false;
bool h12Flag;
bool pmFlag;

int count = 0;
int pirPin = 12; // Pin for the HC-S501 sensor
int pirValue;

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);
unsigned long previousMillis = 0;
const long dhtInterval = 2000; // Interval for DHT readings

// Custom characters for the LCD arrows

```

```
byte downChar[] = {  
    B00000,  
    B00000,  
    B00100,  
    B00100,  
    B00100,  
    B10101,  
    B01110,  
    B00100  
};
```

```
byte upChar[] = {  
    B00100,  
    B01110,  
    B10101,  
    B00100,  
    B00100,  
    B00100,  
    B00000,  
    B00000  
};
```

```
byte degreeChar[] = {  
    B00110,  
    B01001,  
    B01001,  
    B00110,  
    B00000,  
    B00000,  
    B00000,  
    B00000  
};
```

```
//-----  
// Generic MeasurementSensor Class Template  
//-----  
template<typename T>  
class MeasurementSensor {  
private:  
    T measurement; // Current measurement value
```



```

T lowestMeasurement;
T highestMeasurement;

int arrow;          // Arrow indicator: 0 = no arrow, 1 = down, 2 = up
unsigned long arrowMillis; // Last time the arrow was updated/reset
const long signalInterval; // Time interval to reset the arrow

public:
    // Constructor: initializes with an initial measurement value.
    MeasurementSensor(long sigInterval, T initValue)
        : measurement(initValue), arrow(0), arrowMillis(0), lowestMeasurement(0), highestMeasurement(0),
        signalInterval(sigInterval) {}

    // Update the measurement reading and determine the arrow indicator
    void update(T newMeasurement, unsigned long currentMillis) {

        // If this is the first valid measurement, initialize min/max
        if (lowestMeasurement == static_cast<T>(0) && highestMeasurement == static_cast<T>(0)) {
            lowestMeasurement = newMeasurement;
            highestMeasurement = newMeasurement;
        }

        // Determine highest and lowest
        if (newMeasurement < lowestMeasurement) {
            lowestMeasurement = newMeasurement;
        }
        if (newMeasurement > highestMeasurement) {
            highestMeasurement = newMeasurement;
        }

        // If the new measurement equals the previous value and the signal interval has passed, reset arrow.
        if (newMeasurement == measurement && currentMillis - arrowMillis >= signalInterval) {
            arrow = 0;
        } else {
            if (newMeasurement > measurement) {
                arrow = 2; // Up arrow
                arrowMillis = currentMillis;
            }
            if (newMeasurement < measurement) {
                arrow = 1; // Down arrow
                arrowMillis = currentMillis;
            }
        }
    }

```

```

    }
}

// If the previous value is the initial invalid value, clear the arrow indicator.
if (measurement == static_cast<T>(-99)) {
    arrow = 0;
}

measurement = newMeasurement;
}

// Getters for measurement and arrow
T getMeasurement() const {
    return measurement;
}
T getLowest() const {
    return lowestMeasurement;
}
T getHighest() const {
    return highestMeasurement;
}
int getArrow() const {
    return arrow;
}
void resetMinMax(T currentValue) {
    lowestMeasurement = currentValue;
    highestMeasurement = currentValue;
}
};

//-----
// Global Instances for Temperature and Humidity
//-----
MeasurementSensor<float> tempSensor(SIGNAL_INTERVAL, -99.0);
MeasurementSensor<int> humiditySensor(SIGNAL_INTERVAL, -99);

void setup() {
    Serial.begin(9600);
    dht.begin();
    delay(2000);
}

```

```
lcd.init();
lcd.backlight();
lcd.clear();
lcd.createChar(1, downChar);
lcd.createChar(2, upChar);
lcd.createChar(3, degreeChar);

pinMode(pirPin, INPUT);

// myRTC.setYear(2025);
// myRTC.setMonth(3);
// myRTC.setDate(20);
// myRTC.setHour(10);
// myRTC.setMinute(49);
// myRTC.setSecond(0);
}

void print2digits(int number) {
  if (number < 10) {
    lcd.print("0");
  }
  lcd.print(number, DEC);
}

int lastResetHour = -1; // Initialize to an invalid hour

void loop() {
  delay(20);

  // lcd.print(":");
  // print2digits(myRTC.getSecond());

  // Check for movement
  pirValue = digitalRead(pirPin);
  if (pirValue) {
    lcd.backlight();
  }

  // Do we need to update the display?
```

```
unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= dhtInterval) {
    previousMillis = currentMillis;

    // Update backlight based on PIR sensor
    pirValue = digitalRead(pirPin);
    if (!pirValue) {
        lcd.noBacklight();
    }

    count++;

    // Read new values from the DHT sensor
    float newTemperature = dht.readTemperature();
    int newHumidity = dht.readHumidity();

    if (isnan(newTemperature) || isnan(newHumidity)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    // Get current hour from RTC
    int currentHour = myRTC.getHour(h12Flag, pmFlag);
    // Check if hour has changed
    if (currentHour != lastResetHour) {
        tempSensor.resetMinMax(newTemperature);
        humiditySensor.resetMinMax(newHumidity);
        lastResetHour = currentHour;
        Serial.println("min/max values reset");
    }

    // Update our measurement sensors
    tempSensor.update(newTemperature, currentMillis);
    humiditySensor.update(newHumidity, currentMillis);

    // Update the LCD display for temperature
    lcd.setCursor(0, 0);
    if ( count % 6 ) {
        if (tempSensor.getArrow()) {
            lcd.write(tempSensor.getArrow());
        }
    }
}
```

```

    } else {
        lcd.print(" ");
    }
    lcd.print(tempSensor.getMeasurement(), 1);
    lcd.write(3); // degree symbol
    lcd.print("C  ");
} else {
    lcd.print("");
    lcd.print(tempSensor.getLowest(),1);
    lcd.write(3);
    lcd.print(" ");
    lcd.print(tempSensor.getHighest(),1);
    lcd.write(3);
    lcd.print(" ");
}

// Update the LCD display for humidity (set cursor on second row)
if (humiditySensor.getArrow()) {
    lcd.write(humiditySensor.getArrow());
} else {
    lcd.print(" ");
}
lcd.print(humiditySensor.getMeasurement());
lcd.print("%");

lcd.setCursor(1, 1);
print2digits(myRTC.getDate());
lcd.print("-");
print2digits(myRTC.getMonth(century));

lcd.setCursor(10, 1);
print2digits(myRTC.getHour(h12Flag, pmFlag));
lcd.print(":");
print2digits(myRTC.getMinute());
}
}

```

Arduino projects ideas

Arduino projects ideas

1. LED Blinking Traffic Light

- **Description:** Build a simple traffic light simulation with red, yellow, and green LEDs that cycle in sequence.
- **Learning Objectives:** Digital output, basic timing with `delay()`, and code structure (`setup/loop`).
- **Components:** 3 LEDs, resistors, breadboard, jumper wires.
- **Suggested Board:** Arduino UNO R4 Minima.
- **Code Concept:** Use `digitalWrite()` to turn LEDs on/off with timed delays.
- **Challenge:** Build a second traffic light that interacts with the first and emulates a real crossing.

2. Button-Controlled Buzzer

- **Description:** Press a button to make a buzzer play a tone; press again to stop it.
- **Optional:** add a led that shows when the buzzer is playing.
- **Learning Objectives:** Digital input, conditional statements (`if`), basic sound generation.
- **Components:** Pushbutton, buzzer, resistor, breadboard.
- **Suggested Board:** Arduino Nano Every.
- **Code Concept:** Read button state with `digitalRead()` and use `tone()` for sound.

3. Temperature Monitor with LCD

- **Description:** Display room temperature on an LCD screen using a temperature sensor.

- **Learning Objectives:** Analog input, sensor interfacing, LCD output.
- **Components:** TMP36 or DHT11 sensor, 16x2 LCD, breadboard.
- **Suggested Board:** Arduino UNO R4 Minima.
- **Challenge:** add a led and/or buzzer that switches on when the temperature is outside of a preset range.

4. Auto lock PC/Laptop

- **Description:** Build a system that detects if a user is present at a computer using a motion or distance sensor. If no one is detected for a set time (e.g., 5 minutes), it sends a signal to shut down the PC.
- **Learning Objectives:** Sensor interfacing (analog/digital), timing logic, serial communication, basic PC interaction.
- **omponents:** HC-SR04 ultrasonic sensor (distance) or PIR sensor (motion),
- **Suggested Board:** Arduino UNO R4 Minima (simple) or Arduino Nano 33 IoT (for added connectivity).
- **Code Concept:**
 - Use pulseIn() (for HC-SR04) or digitalRead() (for PIR) to detect presence.
 - Implement a timer with millis() to track inactivity.
 - Send a lock command via Serial (software) or toggle a relay (hardware simulation).

PageESP32-WROOM-32S 32d is

The Dual Core 30pin ESP32-WROOM-32S 32d is compatible with the Arduino IDE. To get started, you'll need to install the ESP32 board definitions through the Arduino Board Manager. Here's how:

1. **Install the Board Package:**

In the Arduino IDE, open *File > Preferences* and add the following URL to the "Additional Boards Manager URLs" field:

```
https://dl.espressif.com/dl/package_esp32_index.json
```

2. **Use the Board Manager:**

Go to *Tools > Board > Boards Manager*, search for "ESP32," and install the package provided by Espressif.

3. **Select Your Board:**

Once installed, select the appropriate ESP32 board from *Tools > Board*. Your Dual Core ESP32-WROOM-32S should appear in the list, allowing you to program it using the familiar Arduino environment.

With these steps, you can leverage the Arduino IDE to develop and upload code to your ESP32 board.