**4. Work with Wokwi simulator**

**Laboratory work No 2**

1. **Software environment Simulator** [**https://wokwi.com/**](https://wokwi.com/)

***Wokwi*** is an online Electronics simulator. You can use it to simulate Arduino, ESP32, and many other popular boards, parts and sensors.

For these programming examples, you can use one of Arduino:

* Arduino Uno
* Arduino Nano
* Arduino Mega
* ESP 32

**DHT 22 sensor – Digital Humidity and Temperature sensor (Fig. 1)**



Figure 1. **DHT 22 and DHT 11 sensor**

**Pin names:**

VCC – Positive voltage

SDA – Digital data pin (input/output)

NC – Not connected

GND - Ground

1. **Arduino Uno**

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. (<https://docs.arduino.cc/hardware/uno-rev3>)

**Pin names:**

Pins 0 to 13 are digital GPIO pins. Pins A0 to A5 double as analog input pins, in addition to being digital GPIO pins.

There are three ground pins: GND.1, which is on top of the board, next to pin 13, and GND.2/GND.3, which are on the bottom.

Pins VIN / 5V are connected to the positive power supply.

Pins 3.3V / IOREF / AREF / RESET are not available in the simulation.

Digital pins 3, 5, 6, 9, 10, and 11 have hardware PWM support.

Some of the digital pins also have additional functions:

|  |  |  |
| --- | --- | --- |
| **Pin** | **Function** | **Signal** |
| 0 | Serial (USART) | RX |
| 1 | Serial (USART) | TX |
| 2 | External interrupt | INT0 |
| 3 | External interrupt | INT1 |
| 10 | SPI | SS (Chip select) |
| 11 | SPI | MOSI |
| 12 | SPI | MISO |
| 13 | SPI | SCLK (Clock) |
| A4 | I2C | SDA (Data) |
| A5 | I2C | SCL (Clock) |

Led and Arduino UNO connection schematic (Fig. 2). **Please, connect in the simulator all components!**

Figure 2. **LED and Arduino UNO connection schematic**

* In the***Setup***function (*void setup ()*), add a code that works with variables.

  // initialize digital pin LED\_BUILTIN as an output.

  pinMode(LED\_BUILTIN, OUTPUT);

* *Loop* function (*void loop ()*), add the line:

  digitalWrite(LED\_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)

  delay(1000);                       // wait for a second

  digitalWrite(LED\_BUILTIN, LOW);    // turn the LED off by making the voltage LOW

  delay(1000);                       // wait for a second

**Start the simulation and look at the schematic.**

**Is the LED blinking?**

**Please take a screenshot to make sure the task is done!**

1. **Arduino Nano**

The Arduino Nano is very similar to the Arduino Uno, but in a smaller form factor. It carries the same ATmega328p chip, which has 32K bytes of Flash program memory, 2k bytes of SRAM and 1K bytes of EEPROM. Arduino Nano, button and LED connection schematic (Fig. 3)



Figure 3. **Arduino Nano, button and LED connection schematic**

* **Declaring Variables**

const int BUTTON = 4; // Naming switch button pin

const int LED = 3;   // Namin LED pin

int BUTTONstate = 0; // A variable to store Button Status / Input

* In the***Setup***function (*void setup ()*), add a code that works with variables.

  pinMode(LED, OUTPUT);

  pinMode (BUTTON, INPUT);

* *Loop* function (*void loop ()*), add the line:

  BUTTONstate = digitalRead(BUTTON);  // Reading button status / input

  if (BUTTONstate == HIGH)  // Condition to check button input

    {

      digitalWrite(LED, HIGH);

    }

    else

    {

      digitalWrite(LED, LOW);

    }

**Start the simulation and look at the schematic. Try to push the button!**

**Please take a screenshot to make sure the task is done!**

1. **Arduino Mega**

Arduino Mega 2560 is a **development** electronic board based on the Atmega2560 microcontroller. This board is a good match for projects that require more GPIO pins and memory space because it carries 16 analog pins and 54 digital I/O pins out of which 15 pins are used for PWM output.

**Pin names:**

Pins 0 to 53 are digital GPIO pins. Pins A0 to A15 double as analog input pins, in addition to being digital GPIO pins.

There are five ground pins: GND.1 (next to pin 13), GND.2/GND.3 (next to the Vin pin), and GND.4/GND.5 (at the bottom of the dual-row female header connector)

Pins VIN / 5V are connected to the positive power supply. There are also two additional power supply pins, 5V.1/5V.2, at the top of the dual-row female header connector.

Pins 3.3V / IOREF / AREF / RESET are not available in the simulation.

Digital pins 2 … 13, 44, 45, and 46 have hardware PWM support (total of 15 PWM channels).

Some of the digital pins also have additional functions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin** | **Function** | **Signal** | **External interrupt** |
| 0 | Serial | RX |  |
| 1 | Serial | TX |  |
| 2 |  |  | INT4 |
| 3 |  |  | INT5 |
| 19 | Serial1 | RX | INT2 |
| 18 | Serial1 | TX | INT3 |
| 17 | Serial2 | RX |  |
| 16 | Serial2 | TX |  |
| 15 | Serial3 | RX |  |
| 14 | Serial3 | TX |  |
| 20 | I2C | SDA (Data) | INT1 |
| 21 | I2C | SCL (Clock) | INT0 |
| 50 | SPI | MISO |  |
| 51 | SPI | MOSI |  |
| 52 | SPI | SCK (Clock) |  |
| 53 | SPI | SS (Chip select) |  |

Arduino Mega and LCD connection schematic (Fig. 4)



Figure 4. **Arduino Mega and LCD** **connection schematic**

* **Add library**

#include <LiquidCrystal\_I2C.h>

* **Set the LCD address**

LiquidCrystal\_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display

* In the***Setup***function (*void setup ()*), add a code that works with variables.

  lcd.init();                      // initialize the lcd

  // Print a message to the LCD.

  lcd.backlight();

  lcd.setCursor(3,0);

  lcd.print("Hello, world!");

  lcd.setCursor(1,1);

  lcd.print("Time for Arduino!");

  lcd.setCursor(4,2);

  lcd.print("Poland 2022");

  lcd.setCursor(3,3);

  lcd.print("Power By Anda!"); //Write Your name

**Start the simulation and look at the schematic.**

**Please take a screenshot to make sure the task is done!**

1. **Esp32**

ESP32 is a series of low-cost, low-power system on chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth.

Esp32 and DHT22 connection schematic (Fig.5)



Figure 5. **Esp32 and DHT22 connection schematic**

* **Add library to the code. Also add library in the Library Manager (Fig.  6).**

#include "DHTesp.h"



Figure 6. **Add the Library**

* **Declaring Variables**

const int DHT\_PIN = 15;

* **Define the sensor.**

DHTesp dhtSensor;

* In the***Setup***function (*void setup ()*), add a code that works with variables.

  **Serial**.begin(115200);

  dhtSensor.setup(DHT\_PIN, DHTesp::DHT22);

* *Loop* function (*void loop ()*), add the lines:

  TempAndHumidity  data = dhtSensor.getTempAndHumidity();

  **Serial**.println("Temp: " + String(data.temperature, 2) + "°C");

  **Serial**.println("Humidity: " + String(data.humidity, 1) + "%");

  **Serial**.println("---");

  delay(1000);

**By default, DHT22 values are 24.00° C Temperature and 40.0% Humidity. Start the simulation and look at the output** (Fig. 7)**.**



Figure 7. **Output**

**Please take a screenshot to make sure the task is done!**

# 100 FastLED Projects-Wokwi Embedded Systems Simulator 2022 <https://create.arduino.cc/projecthub/ldir-arcostasi-urish-sutaburosu-stevesigma-stepko-on-wokwi-discord/100-fastled-projects-wokwi-embedded-systems-simulator-2022-39fcc0>