UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
ETSEIB
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial

Embedded and Real Time Systems (ERTS)
Laboratory 1: Introduction to Arduino

1. Introduction.

The goal of the first laboratory session is to introduce the Arduino UNO board and its programming using the Arduino IDE. First, the LED-blinking example will be used to illustrate the whole programming and running process and to illustrate the basic anatomy of a *sketch* (Arduino program). After that, the groups of students will code several sketches that will use different types of sensors and actuators.

Each group of students will have a PC running the Arduino IDE and the following hardware:

- Arduino UNO (Rev3) board.
- USB cable (to connect the Arduino UNO board to the PC).
- Ultrasonic distance sensor HC-SR04.
- Infrared distance sensor GP2Y0A21YK0F.
- Temperature sensor LM35.
- Cables (to connect the sensors to the Arduino UNO inputs).
- Computer fan QFR0812DE.
- 12VDC power supply (to provide power to the fan).

2. The Arduino UNO board.

The Arduino UNO board is the basic reference model of the Arduino family. It is based on the Atmel ATmega328 microcontroller and its main characteristics are the following:

- Powered via USB or with external power supply.
- In-System programming through the USB.
- Microcontroller: ATmega328P.
- Clock speed: 16 MHz.
- Flash memory: 32 KB (program memory; 0.5 KB used by the bootloader).
- EEPROM: 1 KB (non-volatile data memory).
- SRAM: 2 KB (volatile data memory).
- Digital I/O pins: 14 (including PWM outputs).
- PWM outputs: 6.
- Analog inputs: 6.

Figure 1 (next page) shows an Arduino UNO board and highlights its main elements.

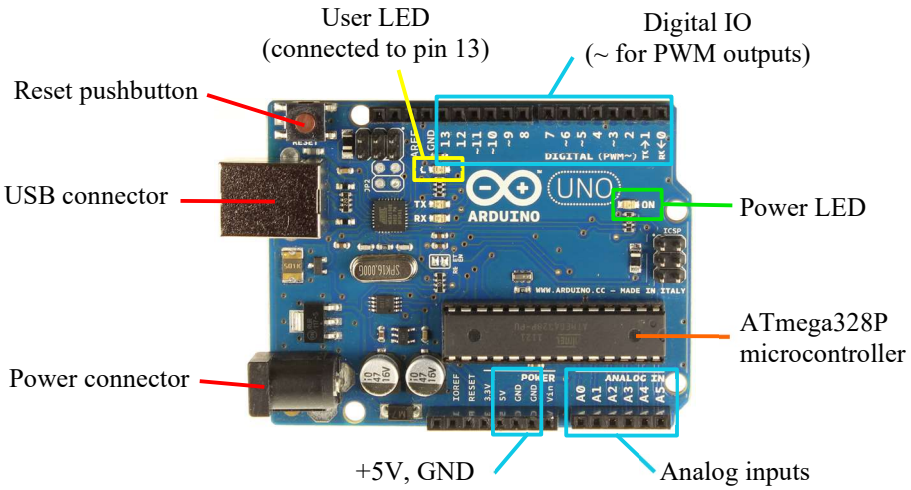

Figure 1. Arduino UNO board.

3. The Arduino IDE: the LED-blinking example.

The Arduino IDE (Integrated Development Environment) is an open-source multi-plattform (Windows, Linux, macOS) IDE that is the common environment used to program the Arduino at a beginner level. The last available version (Arduino 2.3.2 in September 2024) can be downloaded from: https://www.arduino.cc/en/software.

The basic use of the Arduino IDE can be illustrated through the LED-blinking example. This example is integrated in the Arduino IDE and it can be opened through: *File->Examples->01.Basics->Blink*. If everything is OK, the following window showing the code for this example should appear:

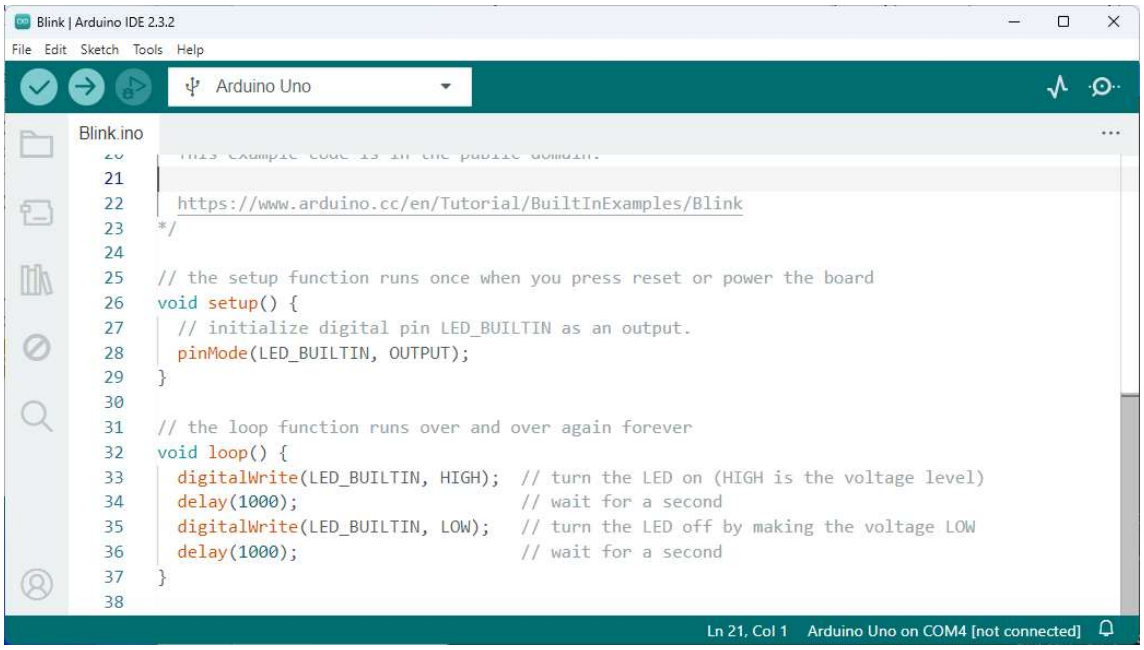

Figure 2. Code for the LED-blinking example.

The previous code shows the two main functions that have to be programmed in an sketch: the `setup()` function and the `loop()` function. The setup function is executed one time at the beginning of the sketch execution and it typically contains initialization code. In the LED-blinking example, the `pinMode()` function is used to configure the digital pin 13, where the board orange LED is connected, as output. On the other hand, the loop function is executed again and again (it is called inside a hidden forever loop). In the example, the code inside the loop function uses the functions `digitalWrite()` and `delay()` to make the LED blink at a frequency of 0.5Hz (one second on, one second off).

To test this example, three steps have to be followed (after connecting the Arduino UNO board to the computer):

1. Select the Arduino UNO board:

2. Compile the source code: *Sketch->Verify/Compile*.

3. Upload and execute the program in the Arduino UNO board: *Sketch->Upload*

4. List of useful functions

A list of the main predefined functions that can be used to program sketches is provided, organized in different categories. A complete list and a detailed description of each function can be found in the *Learning->Reference* section of the Arduino web.

Digital I/O

`pinMode()` – stablishes the directionality (input or output) of a digital pin.
`digitalWrite()` – writes to a digital output.
`digitalRead()` – reads a digital input.
`pulseIn()` – measures the duration of a pulse applied to a digital input.

Analog I/O

`analogRead()` – reads an analog input.
`analogWrite()` – writes a PWM output.

Timing

`delay()` – delay in milliseconds.
`delayMicroseconds()` – delay in microseconds.

Serial communications

`Serial.begin()` – initializes the serial port.
`Serial.print()` – writes to the serial port.
`Serial.read()` – reads from the serial port.

5. Work to be done

Once the LED-blinking example has been used to illustrate the programming of the Arduino UNO board with the Arduino IDE, the students have to solve several programming exercises based on the use of different sensors and one actuator (the computer fan).

The basic information needed to use the sensors and the fan is provided in this document. In addition, the complete datasheets can be found in Atenea.

Some of the proposed exercises have optional extensions. Try to solve these extensions only if you still have time after completing all the exercises.

Exercise 1. Reading an ultrasonic distance sensor (digital output, pulse of variable width).

For the first exercise, the students have to code and test a program able to read the data that is provided by an ultrasonic distance sensor HC-SR04 (see Figure 3).

Figure 3. Ultrasonic distance sensor HC-SR04.

The main characteristics of the HC-SR04 are the following:

- Power supply: +5V DC.
- Range: 2cm – 400cm.
- Resolution: 0.3 cm.
- Output: Digital, pulse of variable width.

The operation and use of the HC-SR04 is quite simple (see Figure 4, in next page). To start a measurement, a trigger pulse (5V, 10 microseconds min.) has to be applied to the TRIG pin (see the pinout in Figure 3). In response, the sensor generates an 8 cycle of ultrasonic burst at 40kHz, it sets the ECHO output to '1' and it waits for a reflected burst (obtained if there is an object in front of the sensor). Once the reflected burst is received, the ECHO output returns to '0'. Hence, the ECHO presents a pulse whose duration is directly related with the measured distance according to the sound speed (a simplified formula that you can use is given in Figure 4).

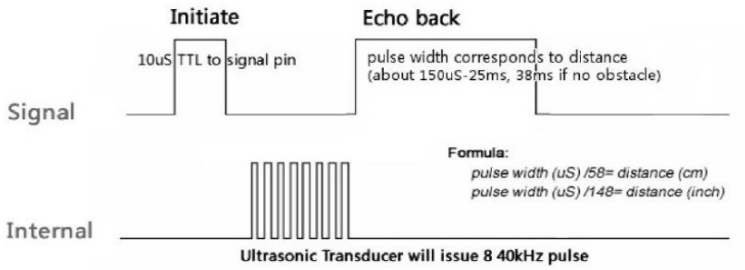

Figure 4. Timing diagram for HC-SR04 signals.

Connect the HC-SR04 to the Arduino UNO board using 4 wires: 5V from Arduino to sensor Vcc; Arduino GND to sensor GND; Arduino digital input/output (0-13) to sensor TRIG pin; Arduino digital input/output (0-13) to sensor ECHO pin.

Code a program that reads the sensor and sends the obtained measurements through the serial port (speed: 9600 bits per second). You can use the serial monitor or the serial plotter integrated in Arduino IDE to visualize the obtained values.

Functions – In order to code the program, the following functions can be useful:

- `pinMode()`.
- `digitalWrite()`.
- `delayMicroseconds()`, `delay()`.
- `pulseIn()`.
- `Serial.begin()`, `Serial.write()`.

Optional – The program can be easily coded using the `pulseIn()` function, try to provide a second solution that does not use this function.

Exercise 2. Reading an analog temperature sensor (analog output, linear).

The goal of this exercise is to code and test a program able to read the data that is provided by an analog temperature sensor LM35 (see Figure 5).

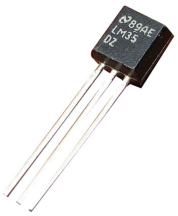

Figure 5. Temperature sensor LM35.

The main characteristics of the LM35 are the following:

- Power supply: 4V – 20V.
- Range: -55 to +155ºC.
- Accuracy: 0.5º (at 25ºC).
- Output: Analog, linear.

The LM35 has only three pins (see Figure 6): +Vs, GND and the output Vout.

Figure 6. LM35 pinout.

The main advantage of the LM35 is that its output is linear. The formula that relates the sensor output voltage and the temperature (in Celsius) is the following: Vout (mV) = 10*T(ºC).

Connect the LM35 to the Arduino UNO board using 3 wires: 5V from Arduino to sensor +Vs; Arduino GND to sensor GND; Vout sensor output to Arduino UNO analog input (A0-A5).

Code a program that reads the sensor and sends the obtained measurements through the serial port (speed: 9600 bits per second). You can use the serial monitor or the serial plotter integrated in Arduino IDE to visualize the obtained values.

Functions – The main function to be used in the program (to read the sensor output) is:

- `analogRead()`.

It is important to notice that the `analogRead()` function provides the output value of the 10-bit ADC integrated in the Arduino UNO board, not the voltage. The voltage at the ADC input can be computed according to the following formula:

$$V_{in} = \frac{V_{ref} * D_{out}}{1023},$$

where: D_{out} is the value at the ADC output; V_{in} is the voltage at the ADC input; V_{ref} is the ADC voltage reference, whose default value is +5V in the Arduino UNO.

Optional – Once you have finish a basic program that reads the LM35, you can combine the solutions of the first two exercises. In Figure 4, a formula that relates the HR-SC04 output pulse width with the distance is proposed. This formula assumes that the sound speed is fixed, but this is not true and the sound speed depends on the temperature. Search for a formula that provides the sound speed as function of the temperature and code a program that, by using the LM35 together with the HR-SC04, provides a more precise distance measurement.

Exercise 3. Reading an infrared distance sensor (analog output, non-linear).

The goal of this exercise is to code and test a program able to read the data that is provided by an infrared distance sensor GP2Y0A21YK0F (see Figure 7).

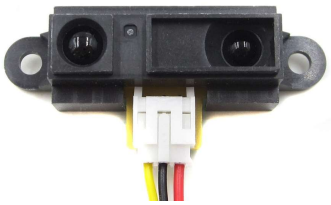

Figure 7. Infrared distance sensor GP2Y0A21YK0F.

The GP2Y0A21YK0F has three pins: +Vcc (with red cable connected in Figure 7), GND (black cable) and Vo (analog output, yellow cable).

The main characteristics of the GP2Y0A21YK0F are the following:

- Power supply: +5V DC.
- Range: 10cm – 80cm.
- Output: analog.

The GP2Y0A21YK0F provides an analog voltage that depends on the distance according to the nonlinear curve presented in Figure 8.

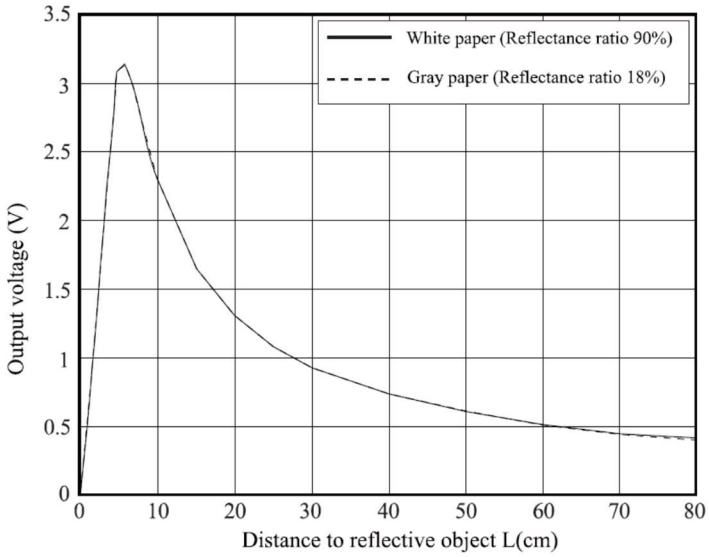

Figure 8. GP2Y0A21YK0F output curve.

Connect the GP2Y0A21YK0F to the Arduino UNO board using 3 wires: 5V from Arduino to sensor +Vcc; Arduino GND to sensor GND; Vo sensor output to Arduino UNO analog input (A0-A5).

Code a program that reads the sensor and sends the obtained measurements through the serial port. The main difference with the previous exercise (reading the LM35) is that the non-linear relation between the sensor voltage output and the measured variable has to be introduced in the program. One possible strategy to do this is based on dividing the measurement range (10-80 cm) in different intervals and assume a local linear behaviour (for instance, it can be observed that the behaviour inside the interval 40-50 cms is quite linear). Other strategy can be based on using data interpolation (using

Matlab, or Excel) to obtain a global analytical expression that approximates the nonlinear response and that can be programmed in Arduino IDE using the available mathematical functions.

Exercise 4. Controlling a computer fan (using a PWM signal).

The goal of this exercise is to implement a program able to control the speed of a computer fan. In particular, the fans that are available at laboratory are manufactured by Delta Electronics under the reference number QFR0812DE-F00 (see Figure 9).

Figure 9. The QFR0812DE-F00 fan.

The main characteristics of the QFR0812DE-F00 fan are the following:

- Dimensions: 80*80*38mm.
- Voltage supply: +12V DC.
- Input current: 1.15A nominal, 1.70A max.
- Speed: 9000 RPM max.
- Max. air flow: 105 CFM (*cubic feet per minute*).
- PWM input; 25 KHz nominal frequency.
- Integrated speed sensor.

From a practical point of view, the main characteristic of the QFR0812DE-F00 fan is that it has a PWM input that can be directly connected to a microcontroller PWM output (an H-bridge module is not needed).

The QFR0812DE-F00 has four wires of different colours with the following usage:

- Red: +12V DC.
- Black: GND.
- Yellow: PWM input.
- Blue: F00 - Speed sensor output.

Connect the fan to the power supply using the associated connectors. And connect the fan to the Arduino UNO: connect the fan PWM input (yellow cable) to one Arduino digital input/output with PWM capabilities (these are the pins with a ~ symbol: 3, 5, 6, 9, 10, 11); connect Arduino GND with fan GND via the available additional cable in the fan connector.

Code a program that applies a sequence of increasing PWM values to the fan input and observe the response of the fan.

Functions – The main function to be used in the program (to generate PWM values) is:

- `analogWrite()`.

Optional – The main goal of this exercise is to implement a program that runs the fan at different speeds by applying different values through its PWM input. Additionally, you can extend the program to monitor the real fan speed by reading its internal sensor. To do this, the timing diagram presented in Figure 10, that shows the behaviour of the F00 signal, has to be taken into account.

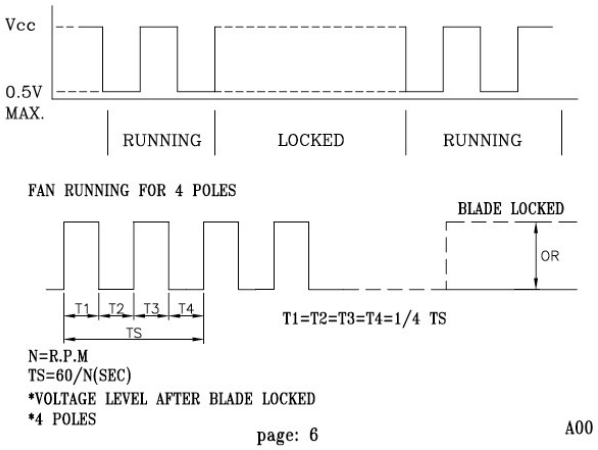

Figure 10. Timing diagram for the F00 (speed sensor) output.

It is important to highlight that reading the speed sensor output requires (datasheet, page 6) a pull-up resistor (resistor connected to Vcc). Since the digital inputs/outputs of the Arduino UNO integrate internal pull-ups, external resistors are not needed, only the internal pull-ups have to be activated using the `pinMode()` function.

6. Report

Please write a report describing your work. For each exercise, you should consider to include in the report:

- Electrical diagram showing the connection of the different components (Arduino UNO, sensors, fan).
- Explanation of the program, in particular with a justification of the used formulas.
- Commented source code.
- Results (evidences) of the program execution: screenshoots and/or links to videos.