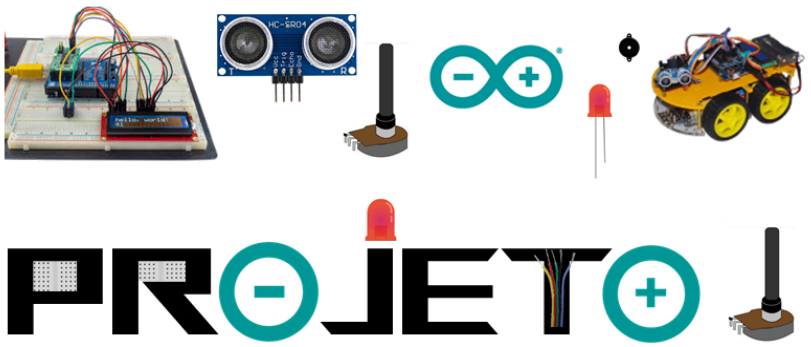

Licenciatura em Engenharia Eletrotécnica e de Computadores

Laboratorial assignment # 4

Ultrasonic distance sensors and DC motors

Components list:

- Arduino UNO
- 1 USB cable
- 1 white breadboard
- 1 Ultrasonic sensor
- 1 RGB LEDs + 3 220 Ohm resistors
- 1 push button + 10K Ohm resistor
- 1 potentiometer
- Wires
- 9V power supply
- 1 DC motor
- 1 TIP-120 transistor + 1K Ohm resistor
- 1 1N4001 diode
- Software: Arduino IDE

Measuring distances using ultrasounds

The HC-SR04 ultrasonic sensor (also called proximity sensor) measures the time interval between the transmission of a sound pulse and its echo. Since the velocity of sound is known, one can convert this time interval into a distance. The sensor has four pins that have the following functions:

• GND (0V)	• Echo – pin to receive the echo signal
• Vcc (5V)	• Trigger – pin to send a pulse

When the Arduino sends a pulse of 10 µs to the trigger pin, the ultrasonic sensor produces a pulse and then waits for its echo. The echo is received at the echo pin. The distance in centimeters is computed by dividing the interval time (measured by the function **`pulseIn()`**) by 58 as explained in the OT class. The Arduino should wait at least 60 ms until the next distance measurement. The code depicted below illustrates an example function to measure the distance using an ultrasonic sensor. Analyze the datasheet available at Inforestudante for more information on the ultrasound sensor.

```
#define trigPin 13
#define echoPin 12

long measureDistance() {
  long duration, distance;

  //Send pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  //Wait for echo and measure time until it happens
  duration = ...;

  //Compute distance
  distance = duration/...;

  return distance;
}
void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  ...
}

...
```

Serial communications

Please recall from the previous assignment that the serial port of the Arduino is able to communicate with the computer to send and receive data. As seen before, the operation of this serial communication is made using the following functions:

- **`Serial.begin(baud_rate)`**—indicates to the Arduino the data rate speed used to communicate. It should be set in the initial **`setup()`** function;
- **`Serial.print(value)`**—send the **`value`** through the serial port. It is able to send variables in several formats, including characters or strings;
- **`Serial.println(value)`**—send the **`value`** through the serial port and change line (introduces a line feed + carriage return character '\n' at the end).

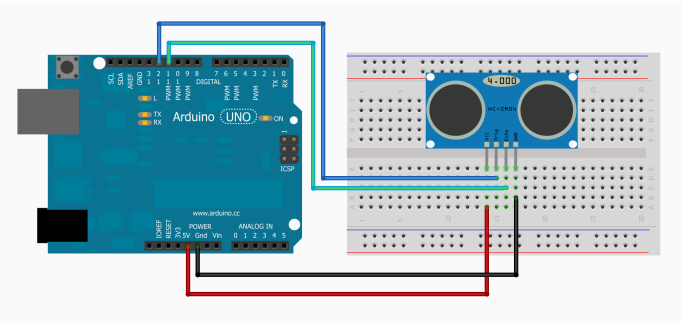

Figure 1 – Circuit with an ultrasonic sensor.

1. Using the ultrasonic sensor HC-SR04

1.1 Distance measurement

Mount the ultrasonic sensor as explained above, calculate and send the distance value to the computer through the serial port. Use the Serial Monitor (Tools ⇒ Serial Monitor) to observe the values sent through the port. In order to automatically calculate the distance between the ultrasound sensor and the object being detected, consider that the speed of sound at 20ºC is 344m/s. Also, note that the distance measured represents the double of the distance to the object. Why?

You must explain the implemented calculations to obtain the distance to the professor during class.

1.2 Measuring the ceiling and table height

Use the circuit to measure the ceiling height of the room and the table height.

1.3 Parking sensor with RGB LED

Implement a system similar to a parking sensor. Use the ultrasonic sensor and mount an RGB LED to be lit depending on the distance to the objects. If the distance is lower than 20 cm the LED should be turned on red. If the detected distance lies between 20 cm and 40 cm, the LED should turn yellow. If the distance lies between 40 cm and 60 cm, it should turn green. For distances between 60 cm and 80 cm, the LED must progressively switch from green to white depending on the distance. Above 80 cm, the LED should always be white.

1.4 Alert message

For distances under 20cm, the device must send a warning message and the measured distance through the serial port. Additionally, when the distance is under 20cm the LED should blink red, with a blinking frequency that varies with the distance (blinks faster as the distance decreases).

1.5 Alternate mode

Introduce a button to activate a second operating mode. In this mode, the LED should blink, cycling through the following colors: RED → MAGENTA → YELLOW → GREEN → CYAN → BLUE → WHITE and back to RED.

If the button is pressed again, the previous mode should be resumed. You must include debounce functionality on the button.

2. The DC Motor

Let's now focus on a different piece of hardware—a low-voltage DC Motor. In this project, you must power the Arduino using the 9v DC power supply and **NOT** the USB cable.

The transistor in Figure 2 controls the power delivered to the motor. The digital pins of the Arduino output a maximum of 40mA. The DC Motor requires around 500mA to operate at full speed and this is obviously too much for the Arduino. If we try to drive the motor directly from a pin on the Arduino, **we can cause permanent damage to the board**!

We must take power directly from the power pins of the Arduino, which draw power from the 9V power supply. A DC regulator reduces the input voltage, ensuring a maximum 800mA output current - more than enough for the DC motor. Using a transistor and PWM, we can control the average voltage applied to the motor, consequently controlling its speed.

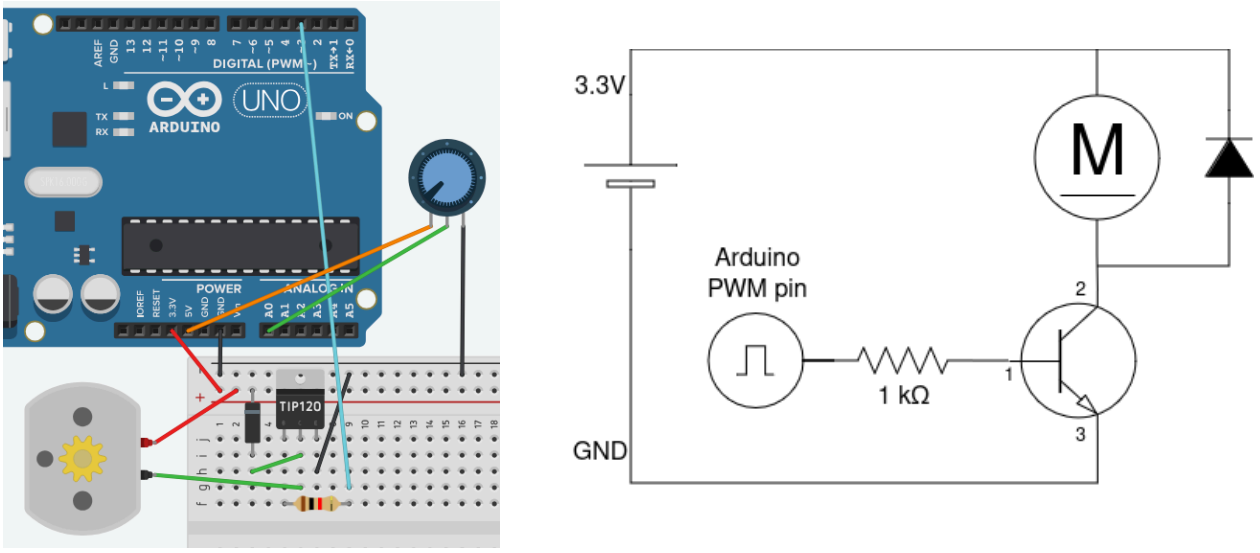

Figure 2 – Circuit to control a DC motor's speed using a PWM signal and a potentiometer.

In the circuit of Figure 2, the 3.3V is connected to the positive motor terminal and the transistor connects the negative terminal to the ground. The base of the transistor is connected via a 1KΩ resistor to digital pin 11. When the Arduino pin outputs a HIGH signal, the transistor turns on and closes the circuit, applying the 3.3V to the motor. By applying a PWM signal to the transistor base, we can control its duty cycle.

Consequently, we can control the average voltage applied to the motor (between 0 and 3.3V). This way, we can effectively control the speed of the motor.

Since the motor is an inductive circuit, when power is removed, a reverse voltage is generated, which can damage the Arduino. Therefore, **we need to install a diode to protect the Arduino**. Make sure the diode is installed correctly, with the white stripe pointing to the positive voltage, as indicated in the figure. This has been put on the circuit to protect the Arduino.

Note: the circuit presented in Figure 2 is meant to be used with a 3V DC motor. If you use a 5V DC motor, you should use the 5V pin from the Arduino to supply the motor circuit, instead of the 3.3V pin.

Controlling the DC Motor's speed

2.1. Mount the DC motor as indicated in Figure 2 and program the Arduino to vary the speed of the DC motor depending on the value read from the potentiometer. Please note that the Arduino's analog inputs convert voltages from 0-5V into numbers in the range 0-1023 and the duty cycle of the PWM analog outputs are controlled from a range 0-255.

2.2. Now, make the motor speed change according to the distances of the ultrasound sensor. The motor's speed should decrease when the ultrasound sensor reports a low distance (stopping when the distance is 5cm or lower), and progressively increase when the distance increases (max speed when distance is 50cm or higher). Print the current distance and duty cycle to the serial port.

2.3. Add a button and an RGB LED to the previous exercise. Make the LED change color continuously (progressively) with the speed of the motor (you can choose any colors you prefer, except RED). The button should be considered as an emergency button to stop the system. When you press the button, the system turns off and the LED turns RED.

Important note: creativity is a plus, so please surprise us!