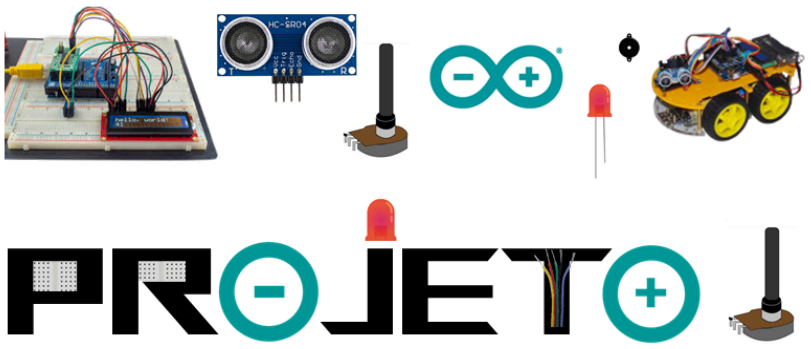

Licenciatura em Engenharia Eletrotécnica e de Computadores

Laboratorial assignment # 7

Serial Communication with external Devices

Components list:

- 2 Arduino UNOs + 2 USB cables
- 2 white breadboards
- 1 LED (any color) + 220 Ohm resistor
- 1 RGB LED
- 2 Potentiometers
- 1 Ultrasonic sensor
- 1 TFT-LCD Display
- 1 Joystick
- Software: Arduino IDE

1. Serial Communication

In this class, we will share with you a serial communication project for Arduino, which allows communication between Arduino and other serial devices.

Please copy the following Arduino code. Your circuit should be composed exclusively of one Arduino Uno. Verify that the code compiles correctly, and then carefully analyze the source code provided. Check the Serial Communication library of Arduino to understand the functions used.

```ino
// ******* Your Utility functions (Add as needed) *******
void SetBuiltInLED(){
  static boolean led_state = true;
  digitalWrite(LED_BUILTIN, led_state);

  if (led_state)
    Serial.println("[REPLY] Builtin LED state: ON");
  else
    Serial.println("[REPLY] Builtin LED state: OFF");

  led_state = !led_state;
}

void StreamData(){
  Serial.println("[REPLY] " + String(random(101)));
}

void StopStream(){
  Serial.println("[REPLY] STREAM STOPPED");
}

void UnknownReply(String req){
  Serial.println ("[REPLY] Unknown request: " + req + ".");
}

//*** NOTE: you must not use delay() in these functions ***


void setup(){
  // Serial port:
  Serial.begin(9600);

  // Max blocking time (in ms) for reading data from the serial monitor:
  Serial.setTimeout(100);

  // Initialize digital pin LED_BUILTIN as an output:
  pinMode(LED_BUILTIN, OUTPUT);

  // Activate other pins as needed...
}


void loop(){
  static boolean stream = false;
  boolean unknown_req = true;
  String request = Serial.readString();

  if(request.length()>0){
    if(request.endsWith(String('\n')))
      request.remove(request.length()-1); //remove linefeed '\n' char

    Serial.println("[REQUEST] " + request); //print req

  }else{ //empty req
    if (stream)
      request="STREAMING";  // streaming data at Serial "Timeout" speed
  }


  //Translate String Requests into Actions:
  if (request.equals("TOGGLE_BUILTIN_LED")){
    SetBuiltInLED();
    unknown_req=false;
  }

  if (request.equals("STREAMING")){
    stream = true;
    StreamData();
    unknown_req=false;
  }

  if (request.equals("STOP")){
    stream = false;
    StopStream();
    unknown_req=false;
  }

  //Add here other actions as needed:
  //if (request...("")){
```

Fig. 1. Barebone Arduino IDE code for bidirectional communication.

1.1. Initial Serial Monitor Tests

Communication works by sending **requests** to the Arduino via the serial port and receiving **replies** from the Arduino.

1. Open the Serial Monitor to test the three existing requests in the code provided and describe to your teacher what each of the requests defined does.
2. Now create TURN_LED_ON request to turn on an external LED, and the TURN_LED_OFF request to turn off the same external LED.
3. Modify the streaming behavior to toggle the external led on/off, every time a new random number is generated.
4. Test all requests in the serial monitor.

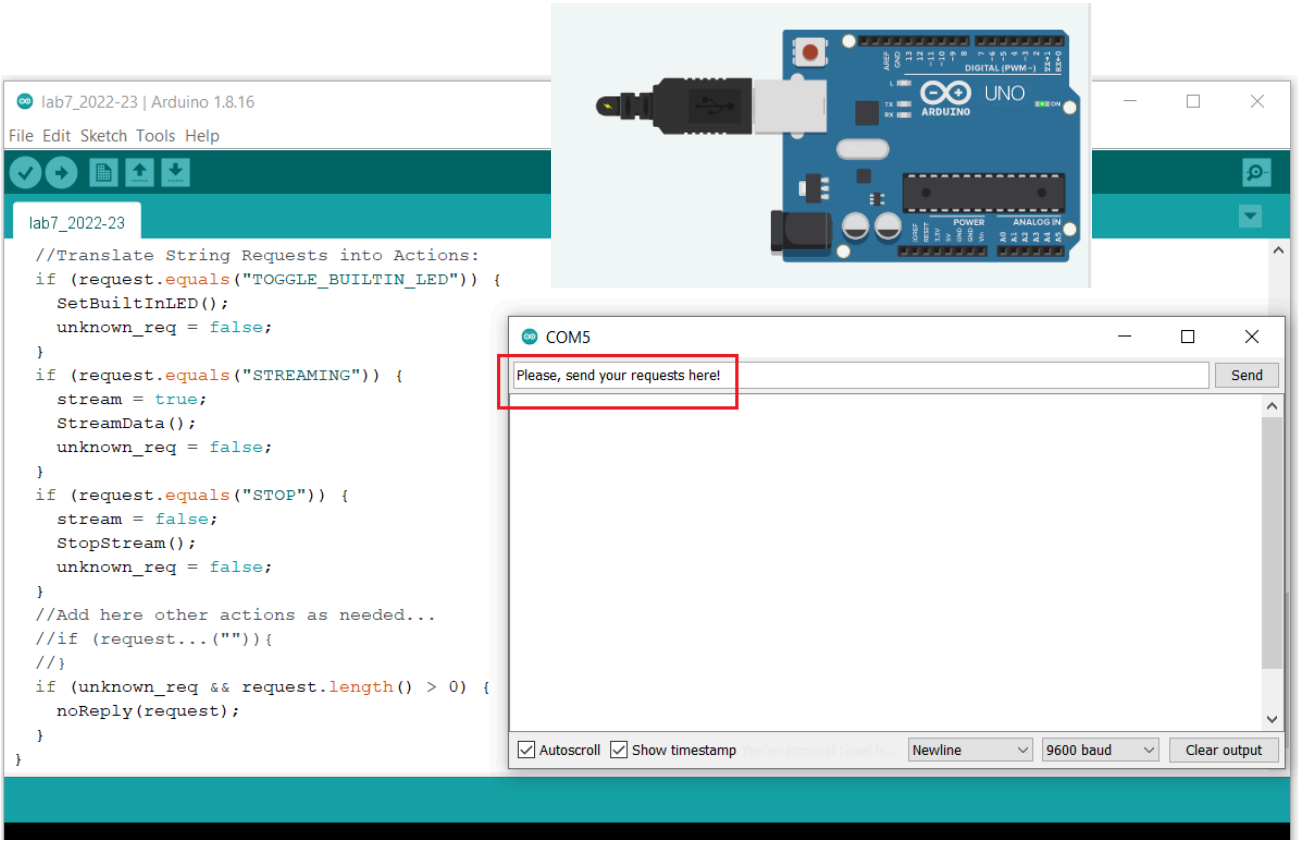

Fig. 2. Serial Monitor Interface for the Arduino Uno.

1.2. Create Additional requests with input parameters

1. Create the "RGB_LED r g b" request, where r, g and b are integers between 0 to 255, corresponding to the color scales (R, G and B). Test the request in the serial monitor with the following values: "RGB_LED 255 255 0", "RGB_LED 0 255 255" , "RGB_LED 255 0 255".

   Parse the incoming request to read each of the three values correctly. We recommend the use of function startsWith(), from the String class, together with sscanf() to extract the three integers.

The Arduino should reply with: “[REPLY] RGB LED state: r,g,b”

2. Modify the function StreamData() to continuously send the data of 1 ultrasonic sensor and 2 potentiometers. The data must be in the integer format. Test the request with the “STREAMING”. command: The reply should be a continuous stream of the following format: “[REPLY] ultrasonic_sensor potentiometer1 potentiometer2”. Stop the stream with “STOP” command.
3. Test all requests in the serial monitor.

2. Two Arduinos Communicating

2.1 Communication between two Arduinos: Sending and Receiving Data

1. Add a second Arduino to your circuit. Connect the **RX port** of the new Arduino board (on the right of Figure 3) to the **TX port** of the original one (on the left of Figure 3), and the **GND** ports, as illustrated in the figure below.

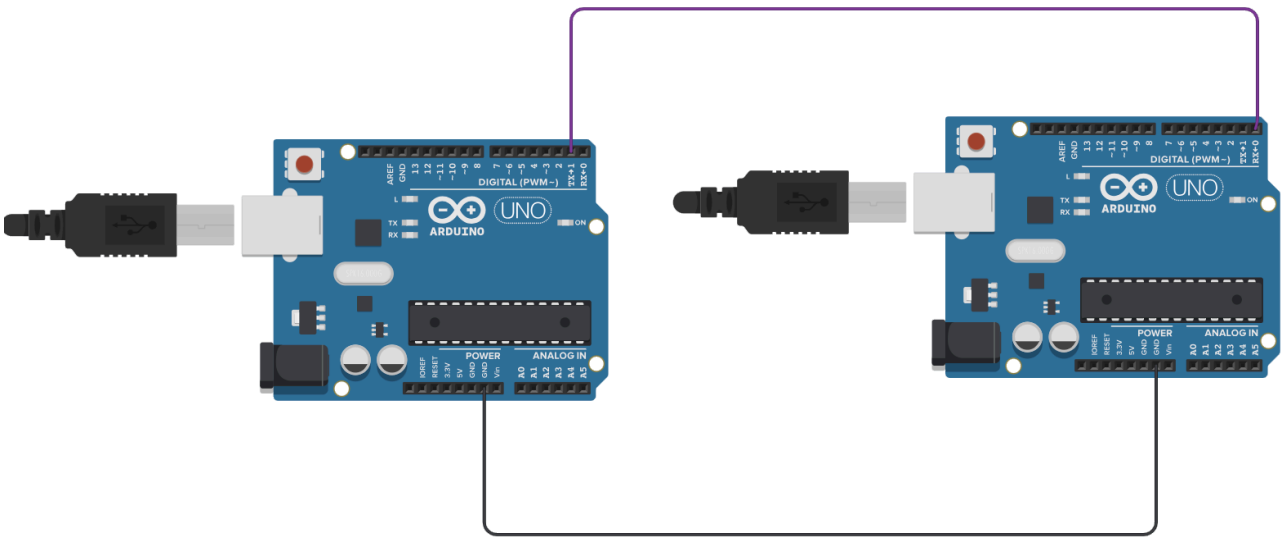

Fig. 3. Wired connection of two Arduinos Uno for serial communication.

2. Upload the following code to the new Arduino:

```
void setup(){
  // Serial port:
  Serial.begin(9600);
  Serial.setTimeout(10); //max blocking time for reading string
}

void loop(){
  String received_data = Serial.readString();
  Serial.print(received_data);
}
```

Fig. 4. Receiving data on the second Arduino.

3. Now activate the “STREAMING” function on the original Arduino through the Serial Monitor and check the Serial Monitor on the new Arduino.
4. Change the code in the new Arduino to parse the ultrasonic sensor/potentiometer received and present the data in the Serial Monitor like this:

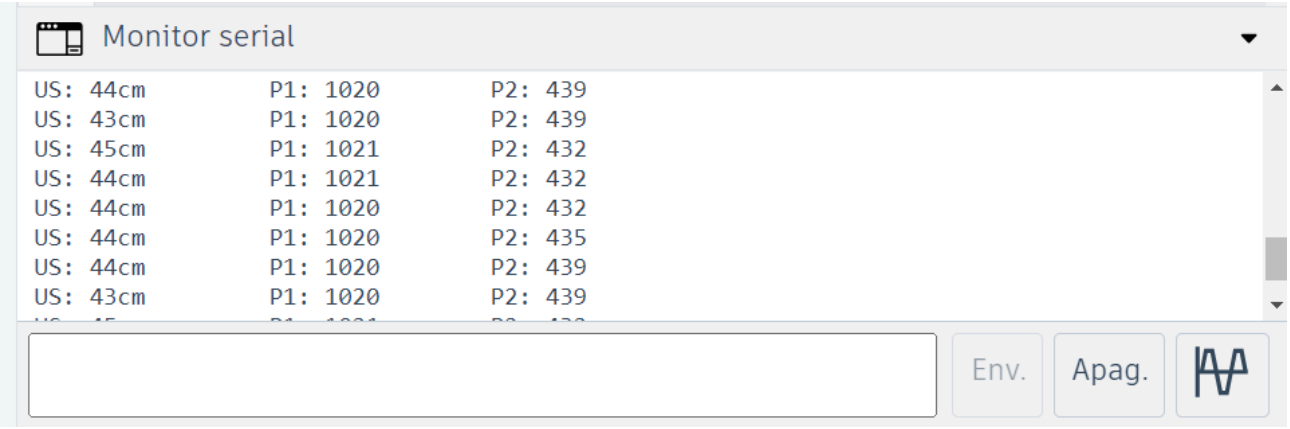

Fig. 5. Printing in the Serial Monitor the data received by the second Arduino.

2.2 Communication between two Arduinos: Sending Data to an LCD display

Add a TFT-LCD Display to the new Arduino (see last class) and create a new “LCD text” request on the original Arduino, which receives text as a string parameter through the Serial Monitor. On the new Arduino, receive the text/numbers (e.g. “LCD 2024”, “LCD 2025”, “LCD Projeto1”, etc.) requested by the original Arduino, and have the text displayed on the LCD. Creativity is a plus!

3. “Shoot the target game” with two Arduinos

Make a “shoot the target game” by:

1. Drawing a random target on an (x, y) coordinate of the TFT-LCD Display of the new Arduino.
2. Using two potentiometers on the original Arduino, control the x and y position of the shooting position. Be careful to normalize the potentiometers to the x/y axis intervals of the LCD display.
3. Stream the shooting position from the original Arduino to the new Arduino.
4. Plot the shooting position on the new Arduino.
5. When the shooting position equals the target position, generate a new target position.
6. Have fun!

Extra: Replace the two potentiometers in step 2 by a joystick.

How could you change 2.1 / 2.2 / 3 to use wireless communications between two Arduinos, instead of wired communication? Check for instance the HC05 Bluetooth module.