

Instituto Superior Técnico

Applications and Computation for the Internet of Things

1st Lab work: Building an embedded system.

Group:		
Student 1:		
Student 2:		
Student 3:		

Goal:

The aim of this work is to familiarize students with the Arduino environment, including the Arduino Starter Kit and IDE. The project involves controlling simple actuators (in this case, LEDs) and using the Node-RED user interface to manage those components from the computer.

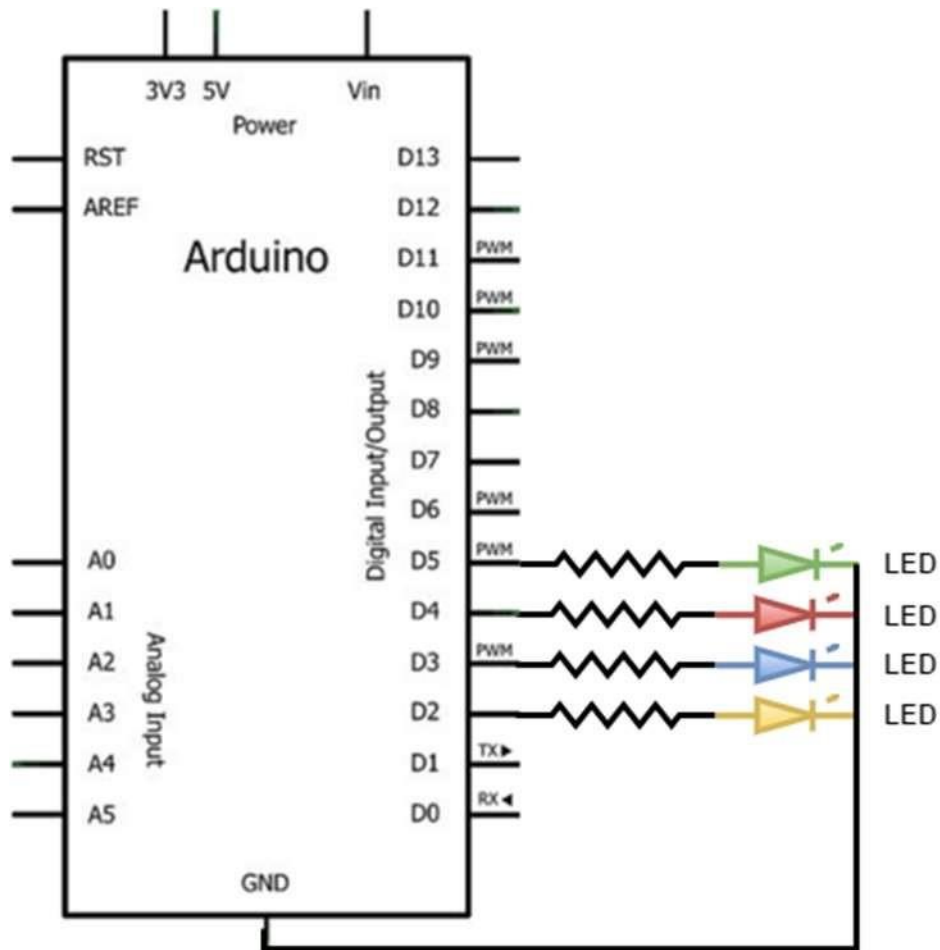
Description:

Build an embedded system using the Arduino UNO board to manage 4 differently coloured LEDs. During normal operation, the system will follow a specific activity pattern every 5 seconds, with a maximum of one LED active at a time in 1 second slots:

- 1 – Red LED ON
- 2 – Green LED ON
- 3 – Blue LED ON
- 4 – Yellow LED ON
- 5 – All LEDs OFF

The behaviour of the system repeats indefinitely.

The following figure represents the circuit to drive the LEDs to be assembled:



We will be controlling the normal operation of the circuit with a digital button on the Node-RED user interface. When the button activates, the currently active LED should remain ON. This state makes it easier to read the voltage drop across each LED. During the 5th stage, the system should stop with all LEDs turned OFF. Upon pressing the button again, the system should resume its normal operation sequence.

To be able to transmit data between the Node-RED UI and the Arduino UNO board you should create a local MQTT broker. MQTT (Message Queuing Telemetry transport) is a lightweight messaging protocol used for communication between IoT (Internet of Things) devices. It works on top of the TCP/IP protocol and allows devices to send (publish) and receive (subscribe) messages. A local MQTT broker is a server that manages MQTT communication for devices on a local network. It receives messages from publishing devices and sends them to subscribing devices through designated topics.

In the context of this project, your computer and the Node-RED UI will communicate with each other through this local MQTT broker. The Node-RED UI will publish messages (such as the state of a button) to the broker, and your computer will subscribe to these messages and send them through the serial port to your Arduino UNO board.

Mosquitto is an open-source message broker that implements the MQTT protocol. To install and run Mosquitto on Windows, follow these steps:

1. Download the Mosquitto installer from the official website: <https://mosquitto.org/download/>.
2. Run the installer and follow the on-screen instructions.
3. Save the path to the mosquitto installation folder on the Windows Environmental Variables.
4. Add the following lines to the mosquitto.conf file:

```
listener 1883 0.0.0.0
```

```
allow_anonymous true
```

5. Open Windows PowerShell as Administrator and run the following commands:

```
cd 'C:\Program Files\mosquitto'
```

```
mosquitto -v -c mosquitto.conf
```

Note: to stop the mosquitto broker run - *net stop mosquitto*

Administrator: Command Prompt - mosquitto -v -c mosquitto.conf

```
1 Microsoft Windows [Version 10.0.22631.3593]
2 (c) Microsoft Corporation. All rights reserved.
3
4 C:\Windows\System32>cd "C:\Program Files\mosquitto"
5
6 C:\Program Files\mosquitto>mosquitto -v -c mosquitto.conf
7 1716494778: mosquitto version 2.0.18 starting
8 1716494778: Config loaded from mosquitto.conf.
9 1716494778: Opening ipv4 listen socket on port 1883.
10 1716494778: mosquitto version 2.0.18 running
```

6. Run the following command to get the IPv4 address which will be used to connect to the broker:

```
ipconfig
```

To install the Node-RED UI on Windows follow these steps:

1. Download the Node.js installer from the official website: <https://nodejs.org/en/download/>.
2. Run the installer and follow the on-screen instructions.
3. Open Windows PowerShell as Administrator and run the following commands:

```
npm install node-red
```

```
node-red
```

4. Open <http://127.0.0.1:1880/>.

After successfully installing the Node-RED we will create a new flow that will contain a “switch node” that will function as the button and an “MQTT out node” that will send the information from the button to an MQTT topic.

To be able to add the switch node we will need to install a library called node-red-dashboard (<https://flows.nodered.org/node/node-red-dashboard>). It is installed by going to Manage palette > Palette > Install > node-red-dashboard. After installing this library, you will be able to create a new switch node (ui_switch) and connect it to a mqtt out node.

The node-red flow configuration should be something like the following. Change the MQTT topic to your student number (istxxxxx) to ensure a reliable transmission of data:

The image consists of two screenshots from the Node-RED web interface. The top screenshot shows the 'Edit switch node' configuration panel. On the left, a flow named 'Flow 1' is visible with an 'Activate' switch node connected to a 'Lab 1' MQTT node, which is then connected to a 'Debugging' node. The 'Edit switch node' panel on the right has the following settings: 'Size' is 'auto', 'Label' is 'Activate', 'Tooltip' is 'optional tooltip', 'Icon' is 'Default', 'Pass through msg if payload matches valid state' is checked, 'When clicked, send:' is checked, 'On Payload' is '1', 'Off Payload' is '0', and 'Topic' is 'msg. topic'. The bottom screenshot shows the 'Edit mqtt out node' configuration panel. The flow 'Flow 1' on the left shows the 'Lab 1' MQTT node connected to the 'Debugging' node. The 'Edit mqtt out node' panel on the right has the following settings: 'Server' is 'localhost', 'Topic' is 'Lab 1', 'QoS' is '0', 'Retain' is 'false', and 'Name' is 'Name'. A yellow tip box at the bottom states: 'Tip: Leave topic, qos or retain blank if you want to set them via msg properties.'

Since the Arduino UNO boards do not have Wi-Fi capability, we need to create an external script to be run on the computer and send the MQTT information via serial port.

script.py

```
1 import serial
2 import paho.mqtt.client as mqtt
3
4 ser = serial.Serial('COM3', 115200) # Change COM3 to the port your device is connected to
5
6 def on_connect(client, userdata, flags, rc):
7     print(f"Connected with result code {rc}")
8     client.subscribe("Lab 1") # Change "Lab 1" to the topic you want to subscribe to
9
10 def on_message(client, userdata, msg):
11     print(f"{msg.topic} {msg.payload}")
12     ser.write(msg.payload)
13
14 client = mqtt.Client()
15 client.on_connect = on_connect
16 client.on_message = on_message
17
18 client.connect("192.168.1.1", 1883, 60) # Change the IP address to the IP address of your MQTT
    broker
19 client.loop_start()
20
21 while True:
22     pass
```

References:

1. <https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/>
2. <https://www.arduino.cc/reference/en/language/functions/time/delay/> (or <https://www.arduino.cc/reference/en/language/functions/time/millis/>)
3. <https://www.arduino.cc/reference/tr/language/functions/communication/serial/readstring/>

Recommendations:

For a safe working environment and to prevent any damage to the hardware, please adhere to the following recommendations. As you progress with your work, check off each box to ensure all safety measures are followed.

Always ensure that the circuit is disconnected from the power source (either the power supply or the PC) when you are working on it. This includes inserting, connecting, or disconnecting sensors and actuators.	
Before connecting the circuit to the power source and turning it on, please consult with the teacher or the person in charge of the laboratory.	
Verify that all components of the circuit (such as resistors, capacitors) are properly connected. This is crucial to prevent short circuits or hardware damage. For instance, never connect an LED directly to a terminal of the controller and GND, or VCC.	
Try to minimize bending the terminals of the components. If bending is necessary (for example, with resistors), ensure that the bend is made approximately 5 mm from the body of the component.	

Design the interface:

Determine the values of the resistors connected to the LEDs. Please note, resistors with values less than 200Ω should be avoided. Lower resistance values result in higher power consumption and increased stress on the components.

R_red

R_green

R_blue

R_yellow

Measure the voltage drops on the LEDs. Use a multimeter or read an analogue input on the Arduino and transfer the value to the PC. Describe the method used.

V_red

V_green

V_blue

V_yellow

Estimate the power consumption of the interface (the circuit with resistors and LEDs in the figure) in normal operation.

Program the application:

Add your program listing (adequately structured and commented). Organise your program in “basic blocks,” each block controlling an actuator, or processing a sensor. A basic block is a sequence of instructions with no embedded branches (except at end), and no branch targets (except at the beginning).

sketch_may23a | Arduino IDE 2.3.2

```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }  
10
```