

Instituto Superior Técnico
Master's Degree in Information Systems and Computer Engineering
Software for Embedded Systems

3rd Lab work: Communications between embedded systems

Group:		
Student 1:		
Student 2:		
Student 3:		

Goal:

The goal of this work is the implementation and test of a wired communications link between two embedded controllers. Communications will be based on the I2C protocol.

Description:

This work is based on the assembly of the previous lab work, but with one difference

- Sensors will be read by an Arduino controller;
- Actuators will be driven by another Arduino controller.

Therefore the overall functionality of this system is identical to that of the system implemented in the 2nd lab work:

“Build an embedded system (...) to control 3 LEDs depending on the state of 3 different sensors (temperature, potentiometer and light intensity).

- *The LED associated with the temperature sensor must be turned on when the temperature read is greater than 26 °C (to be eventually redefined at the laboratory).*
- *The LED controlled by the potentiometer must blink with a period of time between 0.2 and 2 seconds, depending of the rotation applied to the potentiometer.*
- *The LED for the light intensity function must change its own light intensity based on the light intensity sensed in the environment.”*

To implement the required functionality now it is necessary to connect two controllers through a wired communications link.

A diagram of the circuit is given for students to accelerate the process of assembly.

References:

1. <https://www.arduino.cc/en/Reference/digitalWrite>
2. <https://www.arduino.cc/en/Reference/AnalogRead>
3. <https://www.arduino.cc/en/Reference/Serial>
4. <https://www.arduino.cc/en/Tutorial/Calibration>
5. <https://www.arduino.cc/en/Tutorial/PWM>
6. <https://www.arduino.cc/en/Reference/Delay>
7. <https://www.arduino.cc/en/Reference/Wire>

Recommendations:

In order to fulfill your work with security and not damaging the hardware involved, remember to carry out the recommendations below. As you are working fill the boxes to be certain that you fulfill all security measures.

Always work with the circuits disconnect from its power sources.	
Call the teacher, or responsible for the laboratory, before you connect the circuits to its power sources.	
Make sure the circuit is well connected (resistors, capacitors, etc.) to prevent a short circuit, or damage to the hardware.	

This lab work shall be implemented in two lab sessions. In the lab will be available some additional Arduino UNO boards for development and test of the communications channel. Outside the lab, groups shall cooperate to time-share their Arduino kits to develop inter-controller communications. Anyway each group must develop its own work.

To ease the connection of each controller to its specific interface circuits – sensors on one controller, actuators on the other controller – follow the guidelines for breadboard layout explained at the beginning of the first lab by the teacher.

To interconnect both controllers, avoiding electrical hazards and noise, consider the following requirements:

- supply power (VCC) to each Arduino through its USB cable;
- do not connect in any other way the VCCs of the two Arduino boards; VCCs must not be directly connected;
- connect both USB cables to the same computer running the IDE;
- connect the grounds (GND = 0 V) of the two Arduino boards (the single green connection on the diagram).

Inter-Integrated Circuit (I2C):

To put the system to work it is necessary to implement the communication link connecting the two controllers. For this we will use I2C communication. I2C works in a Master-Slave protocol

where one or more Slaves can be connected to one Master on the same bus. There are no fault tolerance or recovery mechanisms included in the protocol. If necessary they must be provided at the application level.

There is already an Arduino Library to handle the application level of the communication over an I2C bus ([read Reference 4 about the Wire library](#)).

To start communication between boards both Arduinos must start the Wire communication:

```
void setup() { Wire.begin(8); } begin only receives an argument if the board is a Slave.  
Masters do not need an address.
```

Subsequent communication transfers may be performed as Master-Writer:

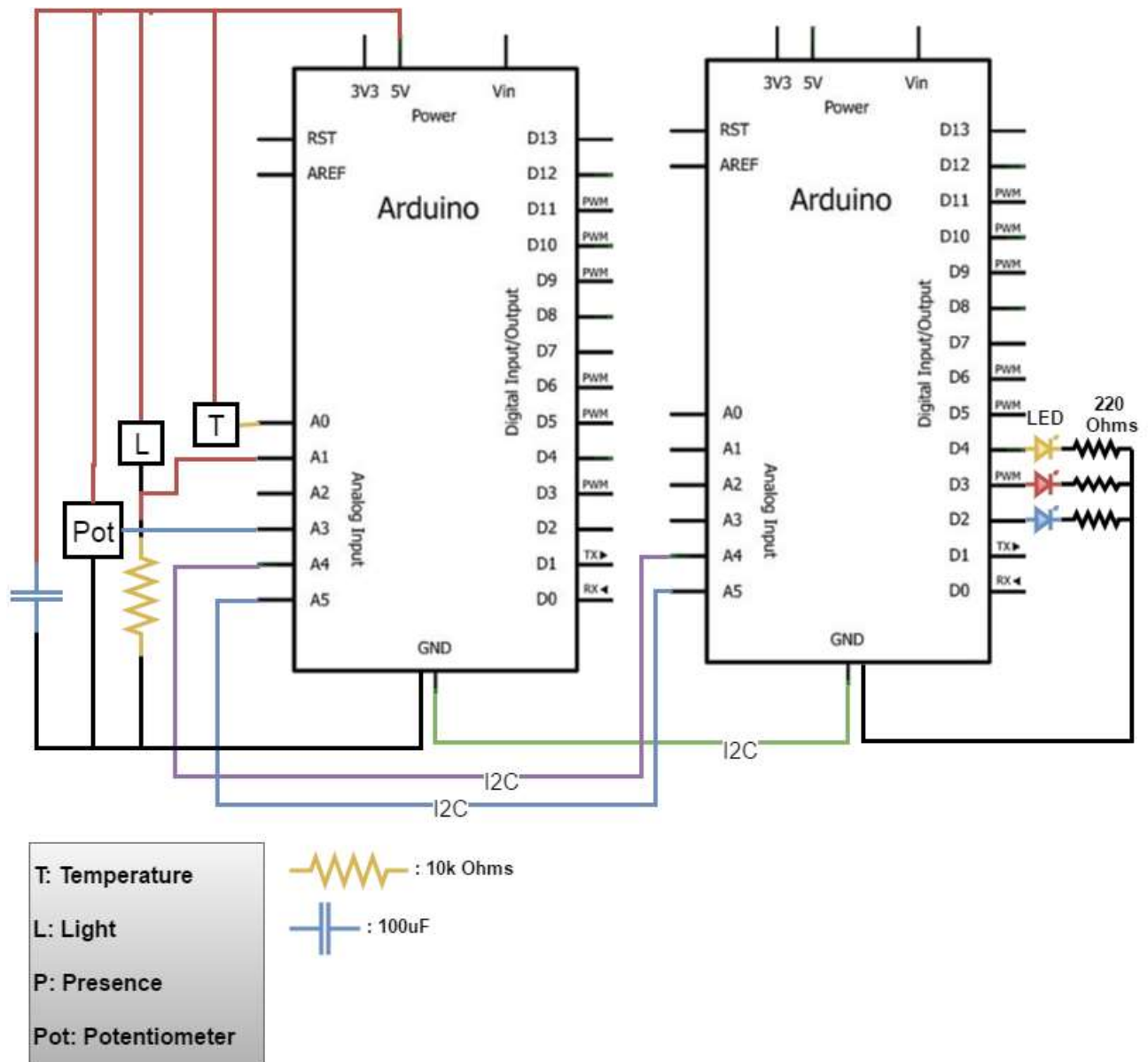
```
void loop() {  
  //Master writes for a Slave to read  
  Wire.beginTransaction(8);  Transmission for port 8 of the I2C bus  
  Wire.write("value:");      Wire.write(string) reads every char as a byte  
  Wire.write(80);            Wire.write(int) reads the int as byte  
  Wire.endTransmission();  
  ...  
}
```

On the Slave side, when some message is detected, the I2C port triggers an interrupt. The Wire Library uses the onReceive function to associate a callback function to the interrupt:

```
void setup() {  
  ...  
  Wire.onReceive(callbackFunction);  
}  
  
void loop() {  
  ...  
}
```

For the previous example, on the Master side, this function will consume and print the string and let the integer be printed separately afterwards

```
void callbackFunction(int i) {  
  while (1 < Wire.available()) {      make sure there is something to read  
    char c = Wire.read();              read the next byte as a char  
    Serial.print(c);                  print the char  
  }  
  int x = Wire.read();                read the next byte as an int  
  Serial.println(x);                  print the int in a new line  
}
```



Results:

Fill the following fields or provide the corresponding printed listings.

- Code - Arduino 1 (I2C master):

```
void setup() {
```

```
}
```

```
void loop() {
```

```
}
```

- Code - Arduino 2 (I2C slave):

```
void setup() {
```

```
}
```

```
void loop() {
```

```
}
```