

Instituto Superior Técnico
Master's Degree in Information Systems and Computer Engineering
Software for Embedded Systems
2017-2018

2nd Lab work: Sensing the Real World

| | | |
|----------------|-------|---------------------------------|
| Group: Grupo 6 | | |
| Student 1: | 79042 | Jorge Francisco Catarino Heleno |
| Student 2: | 87786 | João Carlos Santos Fernandes |

Goal:

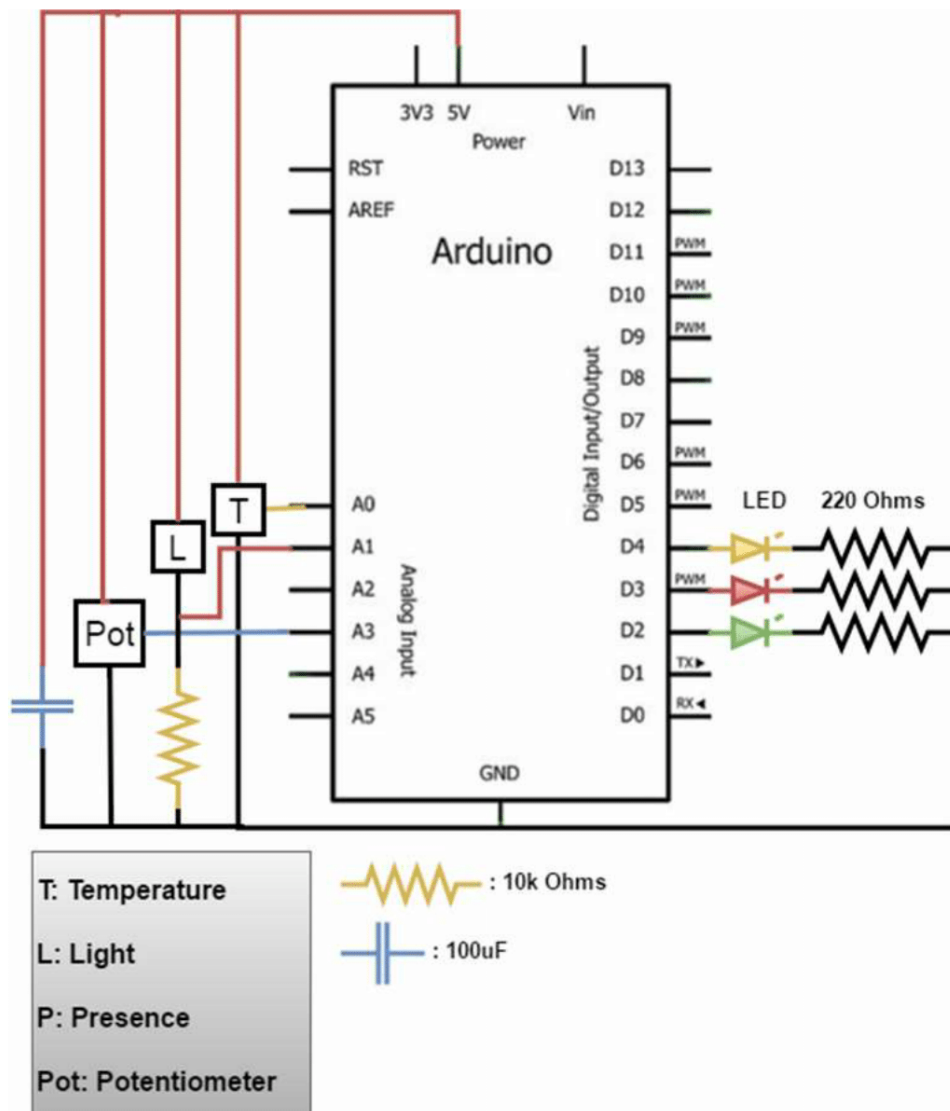
The goal of this work is to sense physical quantities and to control actuators according to the measured environment.

Description:

Build an embedded system using an Arduino UNO board to simultaneously control 3 LEDs depending on the state of 3 different sensors – temperature, rotation angle (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 continuously change its own light intensity based on the light intensity sensed in the environment from
 - Darkness → LED ON, to
 - Brightness → LED OFF(see Reference 5 about Pulse Width Modulation).

A diagram of the circuit is represented in the figure.






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>

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 circuit disconnect from the source. |  |
| Call the professor or responsible for the laboratory, before you connect the circuit to the source. |  |
| Make sure the circuit is well connected (resistors, capacitors, etc.) to prevent a short circuit, or damage the hardware. |  |

Mapping analog measurements:

Usually the digital readings retrieved from sensors do not correspond directly to the value of the physical quantity, but rather to values between 0 and a maximum binary value (say $1023 = 2^{10} - 1$ for a 10 bits word). Therefore some mapping may be required.

When the value is relative just to an offset a simple mapping is adequate, but in general a more complex scale conversion will be needed. The `map` function, available in the Arduino IDE, simplifies these types of conversions. For example, when rotating a Servo motor with input from a potentiometer we know that when the value of the potentiometer is 0 then the servo angle must be 0° as well. Logically, if the value of the potentiometer is 1023 (its maximum reading) then the servo angle must be 180° (its maximum position):

```
map(value, 0, 1023, 0, 180)
```

Besides this some sensors require a linearization of its transfer function (physical quantity \rightarrow electrical quantity, such as voltage). For example, to convert the reading from a circuit with a temperature sensor (in our case a temperature dependent resistor) to the real temperature several transformations may be required

- to linearize the transfer function of the sensing device $R_T = f(T)$, and
- to linearize the transfer function of the circuit in which the sensor is included.

For the device and configuration to be used (based on TMP35) check the function

$$T = (((\text{sensor value} / 1024.0) * 5.0) - 0.5) * 100$$

Programming with analog sensors:

To control an external analog sensor you must attach it to an Arduino analog pin. The software allocation of a sensor to an analog pin is done using the following code:

```
int const tempSensor = A0;
```

where `A0` is the physical pin where the sensor is attached.

To read the value assigned to a specific pin use Arduino function `analogRead(PIN)`, as follows:

```
int temperatureValue = analogRead(tempSensor);
```

Debug:

In order to control some variables and *debug* your program it is possible to print them to the Serial Monitor present in Arduino IDE. This kind of tool is useful, for example, to keep track of the temperature variation or to help the calibration of the system.

To use this feature, first start a serial communication with the PC:

```
void setup() { ...; Serial.begin(9600); ...; }
```

Then, just *Serial.print* your variables and/or strings:

```
Serial.println(temperatureValue);
```

```
const int greenLedPin = 2;
const int redLedPin = 3;
const int yellowLedPin = 4;

const int tempSensor = A0;
const int lightSensor = A1;
const int angleSensor = A3;

int tempRead = 0;
int lightRead = 0;
int angleRead = 0;
int lastAngleRead = -1;

int lightReadMinValue = 1023; //Start on max
int lightReadMaxValue = 0; //Start on min

float temperature = 0;
int light = 0;

int angle = 0;
boolean isAngleOn = false;
int intervalTime = 0;
unsigned long previousTime = 0, currentTime = 0;

void setup() {
  pinMode(greenLedPin, OUTPUT);
  pinMode(redLedPin, OUTPUT);
  pinMode(yellowLedPin, OUTPUT);

  // Calibrate Light Sensor

  // Start Calibration
  digitalWrite(greenLedPin, HIGH);
  digitalWrite(redLedPin, HIGH);
  digitalWrite(yellowLedPin, HIGH);

  // Calibration During the First Five Seconds
  while (millis() < 3000) {
    lightRead = analogRead(lightSensor);

    // Record the Maximum Value Read
    if (lightRead > lightReadMaxValue) {
      lightReadMaxValue = lightRead;
    }

    // Record the Minimum Value Read
    if (lightRead < lightReadMinValue) {
      lightReadMinValue = lightRead;
    }
  }
}
```

```

}

// End Calibration
digitalWrite(greenLedPin, LOW);
digitalWrite(redLedPin, LOW);
digitalWrite(yellowLedPin, LOW);
}

void loop() {

    // Temperature
    tempRead = analogRead(tempSensor); // Read Temperature
    temperature = (((tempRead / 1024.0) * 5.0 ) - 0.5) * 100;

    if(temperature > 30) {
        digitalWrite(greenLedPin, HIGH);
    } else {
        digitalWrite(greenLedPin, LOW);
    }

    // Light
    lightRead = analogRead(lightSensor); // Read Light
    light = map(lightRead, lightReadMinValue, lightReadMaxValue, 255, 0); //
Convert Analog Value
    light = constrain(light, 0, 255);

    // Noise Reduction
    if(light < 20) {
        light = 0;
    } else if (light < 40) {
        light = 30;
    } else if (light < 60) {
        light = 50;
    } else if (light < 80) {
        light = 70;
    } else if (light < 100) {
        light = 90;
    } else if (light < 120) {
        light = 110;
    } else if (light < 140) {
        light = 130;
    } else if (light < 160) {
        light = 150;
    } else if (light < 180) {
        light = 170;
    } else if (light < 200) {
        light = 190;
    } else if (light < 220) {
        light = 210;
    }

```

```
} else if (light < 240) {
    light = 230;
} else {
    light = 255;
}
analogWrite(redLedPin, light);

// Potentiometer
angleRead = analogRead(angleSensor); // Read Angle
if (lastAngleRead != angleRead) {
    angle = map(angleRead, 0, 1023, 0, 180); // Map Read Value to Angle
    intervalTime = map(angle, 0, 180, 200, 2000); // Map Angle to Milliseconds
    lastAngleRead = angleRead;
}

currentTime = millis();

if (currentTime - previousTime >= intervalTime) {
    isAngleOn = !isAngleOn;
    if(isAngleOn) {
        digitalWrite(yellowLedPin, HIGH);
    } else {
        digitalWrite(yellowLedPin, LOW);
    }
    previousTime = currentTime;
}
}
```

- Questions:

Question:

For each of the three pairs sensor-actuator describe:

1. the mapping process implemented;
2. the calibration setup;
3. the process, or technique, used to modulate the behavior of the actuator;
4. the setup prepared to demonstrate the functionality of the system.

Answer:

Light

For the light we implemented a mapping that when the photoresistor output its high (light is on) the intensity of the led is low, so our mapping is inverse, high values on photoresistor read = low values on the led output. The calibration its done in the first 3 seconds of working of the arduino, we read the values on the sensor and regist the max and min value readed, then we save it and use it for the mapping, we also make some interval for the light values to reduce the variations produced by the noise in the photoresistor read. The technique used to modulate the behaviour was the PWM that allow us to control the voltage sent through the output by changing in a high frequency the output from 0-1 (0V to 5V). That allow us to control the intensity of the led light, producing the behaviour wanted. To test and demonstrate the functionality we cover the photoresistor with our hands and uncover to see the variations of intensity on the led light.

Temperature

For the temperature we used a mapping expression that was provided in the lab sheet that allow us to calculate the temperature with the value read from the temperature sensor. We didn't need to calibrate it but if we need it we could use a thermometer to see if the values that we are get from the mapping expression was the right values of temperature and if wasn't we needed to correct the expression. We just needed to change the temperature limit for the led to turn on because the temperature in the room was over 26°C so we need to change it to be a little bit bigger to can test the functionality. The technique used to modulate was a simple if that when the temperature read was over a defined value we turn the led on if it was not we turned the led off. To test and demonstrate it we just used our fingers to warm the temperature sensor and see if the led turned on and then we release the temperature sensor, waited for it to become colder and then the led was expected to go off.

Potentiometer

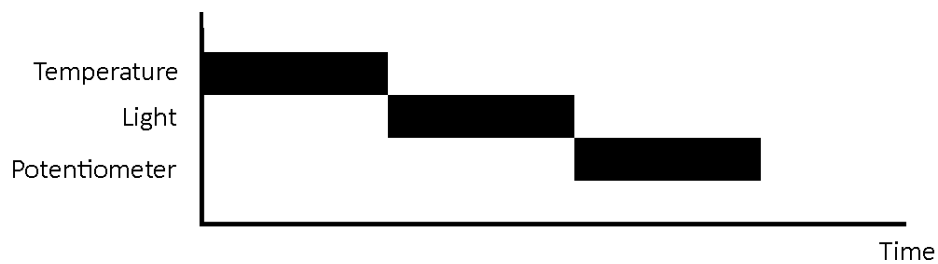
For the potentiometer the mapping used was easy, we just mapped the values read to an angle 0°-180° and then the angle to milliseconds 200-2000. There was no calibration needed we just needed to see if the values were accurate. To modulate the behaviour we used variables to count time and readjust the blinking interval based on the value of the potentiometer. So we turned on and off the led based on the interval gived by the mapping of the potentiometer read value. To test and demonstrate the working of our system we just changed the potentiometer value and watched the led blinking in different intervals as expected, we counted the time with a cronometer to see if it was more a less the right time values.

Question:

What is the system software pattern of the application?
What are the timing constraints of the system?

Answer:

Our application follows a Round Robbin Software Pattern. We handle with the sensor reads, mapping and output actions of each “device” in sequence per device. As can be seen in the code we make the tasks of each sensor-actuator separately but in the same loop, making the execution as the illustrated in the following image:



The time constraints are just the time that each task takes to be done, but with a exception. The Potentiometer complete Task only is done from some to some time so it will make the system have different time executions per loop. But it will be a very small difference between those loops.