Instituto Superior Técnico

Master's Degree in Information Systems and Computer Engineering

Software for Embedded Systems

3rd Lab work: Wired Communications between embedded systems

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

Results

Fill the following fields or provide the corresponding printed listings.

1. Describe the changes in the program developed in lab 2 (Sensing the Real World) to port it to the new configuration with two controllers.
    a. Describe any changes in the design pattern (such as changing tasks in the round-robin loop).

For this lab we need to split the read from sensors and mapping code to one arduino and the control of the leds code to the other. The design pattern still being roud-robin but in the sensors read and mapping its normal round-robin and in the led controller its round robin with interruptions. That interruptions are made by an event handler when the arduino receives a message from the other arduino and sets the variables that will be use in the code to turn on/off the leds, very similar to a system pattern with interruptions.

   b. Describe other implementation issues changed (implementation of tasks).

In terms of tasks implementation their are split in the following way:

- Temperature:
  Reader: Reads the sensor and maps the read value to the temperature.
  Led Controller: Turn on the led if temperature over x value our turn off otherwise.
- Light:
  Reader: Reads the sensor and maps the read value to a value between 0 and 250.
  Led Controller: Uses the value received from reader to control the intensety of a led.
- Potentiometer:
  Reader: Reads the sensor and maps the read value to the angle (0 - 180).
  Led Controller: Maps the angle received to a value in milisseconds (200-2000) and control the led blinking with that value.

```
// Lab03-Master.ino
// I2C-MASTER

#include <Wire.h>

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

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

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

float temperature = 0;
int light = 0;
int angle = 0;

void setup() {

  // Start Wire Connection
  Wire.begin();
  // Calibrate Light Sensor

  // Calibration During the First Three 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;
    }
  }
}

void loop() {

  // Temperature
  tempRead = analogRead(tempSensor); // Read Temperature
  if (lastTempRead != tempRead) {
    temperature =  (((tempRead / 1024.0) * 5.0 ) - 0.5) * 100;
    Wire.beginTransmission(8);
    Wire.write("T");
    Wire.write((int)temperature);
    Wire.endTransmission();
    lastTempRead = tempRead;
  }

  // Light
  lightRead = analogRead(lightSensor); // Read Light
  if(lastLightRead != lightRead) {
    light = map(lightRead, lightReadMinValue, lightReadMaxValue, 255, 0); // Convert Analog Value
    light = constrain(light, 0, 255);
    Wire.beginTransmission(8);
    Wire.write("L");
    Wire.write(light);
    Wire.endTransmission();
    lastLightRead = lightRead;
  }

  // Potentiometer
  angleRead = analogRead(angleSensor); // Read Angle
  if (lastAngleRead != angleRead) {
    angle = map(angleRead, 0, 1023, 0, 180); // Map Read Value to Angle
    Wire.beginTransmission(8);
    Wire.write("A");
    Wire.write(angle);
    Wire.endTransmission();
    lastAngleRead = angleRead;
  }
}

```

```
// Lab03-Slave.ino
// I2C SLAVE

#include <Wire.h>

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

int 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);

  Wire.begin(8);
  Wire.onReceive(receiveEvent);

  while(millis() < 3000);
}

void loop() {

  if(temperature > 27) {
    digitalWrite(greenLedPin, HIGH);
  } else {
    digitalWrite(greenLedPin, LOW);
  }

  // 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;
```

```cpp
  } else if (light < 220) {
    light = 210;
  } else if (light < 240) {
    light = 230;
  } else {
    light = 255;
  }
  analogWrite(redLedPin, light);

  // Potentiometer
  currentTime = millis();
  intervalTime = map(angle, 0, 180, 200, 2000); // Map Angle to Miliseconds

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

void receiveEvent() {
  char type;
  while (1 < Wire.available()) {
    type = Wire.read(); // Read Type
  }
  if(type == 'T') {
    temperature = Wire.read();
  } else if(type == 'L') {
    light = Wire.read();
  } else if(type == 'A') {
    angle = Wire.read();
  }
}

```

4. Evaluate the performance of the I2C bus in your system (data rate, latency).

For the evaluation of the I2C bus performance we developed 2 programs that allow's to calculate the values of the data rate and latency.

Data Rate:

For the data rate we made a program were the master sends 1 byte constantly and the slave counts how many bytes it received in a time period of 5 seconds and then print the counted value divide by 5 (to get the count per second) trohw the serial port. The value that we got from the program was: 4341 bytes / second which means that the bit rate is: 34,728 Kb/s (Kbits per second). We think that this is a very good value for a simple bus like that.

Latency:

For the latency we made a program were the master sends a 1 byte message to the slave and when the slave receives it, it turns on a output pin that its connected to a input pin in the master. The master after sending the message registers the time of the sent and waits for the input pin to change its value to high when it changes he registers the time and make the diference between the sent time and the receive time calculating the latency. The value we got was 12 µs (microseconds). We think that is also a very good latency for this bus, but we already expected a very low latency cause the bus is really small.

The codes of the programs used to evaluate the performance are in attachment.

```
// DataRate-Master.ino
// DATA RATE I2C MASTER

#include <Wire.h>

void setup() {
  // Start Wire Connection
  Wire.begin();
}

void loop() {
  Wire.beginTransmission(8);
  Wire.write("A");
  Wire.endTransmission();
}

```

```
// DataRate-Slave.ino
// DATA RATE I2C SLAVE

#include <Wire.h>

int count = 0;

void setup() {
  Wire.begin(8);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
}

void loop() {
  long startTime = millis();
  count = 0;
  while(millis() < startTime + 5000);
  count = count / 5;
  Serial.println(count);
}

void receiveEvent() {
  Wire.read();
  count++;
}

```

```cpp
// Latency-Master.ino
// LATENCY I2C MASTER

#include <Wire.h>

const int receivePin = 8;

void setup() {
  pinMode(receivePin, INPUT);
  Serial.begin(9600);

  Wire.begin();
}

void loop() {
  Wire.beginTransmission(8);
  Wire.write("A");
  Wire.endTransmission();

  long sendTime = micros();

  while(digitalRead(receivePin) != HIGH);

  long latency = micros() - sendTime;

  Serial.println(latency);

  delay(5000);
}

```

```
// Latency-Slave.ino
// LATENCY I2C SLAVE

#include <Wire.h>

const int sendPin = 8;

void setup() {
  pinMode(sendPin, OUTPUT);
  Wire.begin(8);
  Wire.onReceive(receiveEvent);

  digitalWrite(sendPin, LOW);
}

void loop() {
  digitalWrite(sendPin, LOW);
  delay(2000);
}

void receiveEvent() {
  Wire.read();
  digitalWrite(sendPin, HIGH);
}

```