Departament de Disseny i Programació
de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Using the Arduino IDE to sample the output of the LM35 Temperature Sensor

Jordi Bonet-Dalmau

September 20, 2013

Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

This document covers same issues regarding the use of the **analogRead** and **analogReference** functions included in the Arduino IDE environment. This functions are used to convert an analog input (A0 to A5) of the Arduino UNO board to a digital 10-bit value.

Depending on the type of the analog input being used the acquisition time given by the **analogRead** function is not enough (actually it is zero) to completely charge the 14 pF capacitor (in series with a 1..100 kΩ) of the sample and hold circuitry of the microcontroller. This problem has been observed when changing (via a multiplexer) the analog input connected to the ADC from one input (with a high value) to another input (with a low value) like the **LM35 temperature sensor**. The ADC starts the conversion when the capacitor is not completely charged. The problem is exacerbated because of the oscillations that appears at the output of the LM35 when the output is switched from a high impedance to the RC sample and hold circuitry with a high voltage. We can avoid all this inconveniences using a damping circuit (75 Ω in series with 1 uF) at the output of the LM35. Another interesting way to avoid the oscillations is to modify the **wiring_analog.c** file and insert (actually uncomment) a delay between the selection of the multiplexer input and the start of the conversion in order to increase the acquisition time.

Another troublesome aspect of the IDE environment is the use of the **analogReference** function. When using this function the analog reference isn't changed, just a variable is changed. This variable is used when calling **analogRead** and the change of the analog reference is made simultaneously with the selection of the multiplexer input, and immediately before the start of the conversion. So, again the voltages aren't stabilized when the conversion is started.

Contents

Chapter 1

Discovering the stabilization problem

Following we show same experiments that have been made to get insight into the problem and limit its consequences.

1.1 Measuring the temperature

The first trial was to measure temperature using two LM35 temperature sensors connected to A0 and A1 and an humidity using one HIH-4000 series humidity sensors connected to A2 at one second interval. We observed that one of the temperature sensors gives and stable lecture (A1), while the other (A0) seems to be noisy. After some (costly) efforts to identify the origin of the noise we concluded, by excluding other causes, that the reason was the order in which the readings were made: the first reading after the humidity lecture was noisy, but a temperature reading after another temperature reading wasn't noisy. We simplify the circuit to use only one temperature sensor and two stable voltages. We use the following code.

```
/*Xbee_4*/

#define VoT1 0
#define VoT2 1
#define VoT3 2

float v_T1,v_T2,v_T3;

void setup() {
  Serial.begin(9600);
  analogReference(DEFAULT);//5V
  Serial.println("Inici");
}

void loop() {
  delay(1000);
```

```
  v_T1=analogRead(VoT1);
  Serial.print("T1=");Serial.print(v_T1);

  delay(1000);
  v_T2=analogRead(VoT2);
  Serial.print(",  T2=");Serial.print(v_T2);

  delay(1000);
  v_T3=analogRead(VoT3);
  Serial.print(",  T3=");Serial.println(v_T3);
}
```

Connecting 3.3 V to A0, the LM35 to A1 and GND to A2 we read the following values:

```
Inici
T1=678.00,  T2=71.00,  T3=0.00
T1=678.00,  T2=39.00,  T3=0.00
T1=678.00,  T2=52.00,  T3=0.00
T1=677.00,  T2=39.00,  T3=0.00
T1=678.00,  T2=47.00,  T3=0.00
T1=678.00,  T2=38.00,  T3=0.00
```

But connecting GND to A0, the LM35 to A1 and 3.3 V to A2 we read the following values:

```
Inici
T1=0.00,  T2=45.00,  T3=678.00
T1=0.00,  T2=45.00,  T3=678.00
T1=0.00,  T2=45.00,  T3=678.00
T1=0.00,  T2=45.00,  T3=678.00
T1=0.00,  T2=45.00,  T3=678.00
T1=0.00,  T2=45.00,  T3=678.00
```

So, an LM35 reading after a high value 3.3 V is noisy and a reading after a low value GND isn't noisy.

1.1.1 Conclusions

When we select a different input with the multiplexer, the hold and sample capacitor needs to be charged to a new value. If the observed problem were a large time constant, a new reading after a high value would be bigger than the final value. Instead, this value oscillates around a mean value (45). A quick look at the LM35 datasheet shows us that an step-up voltage in V_{IN} gives a second order response in V_{OUT}. So the output could also give a second order response when connecting this output to a different voltage.

1.2 LM35 output voltage waveform

The next step was to measure the output of the LM35 sensor in order to verify what happens when this output is switched from a high impedance to the RC sample and hold circuitry charged to a high value initial condition.

We have slightly modified the previous code to have a trigger pin. We put pin A3 at high level before executing **analogRead** and at low level after its execution. These modifications are marked with //***.

```
/*Xbee_4*/

// define
#define VoT1 0  //A0
#define VoT2 1  //A1
#define VoT3 2  //A2
#define Sync 17 //A3 //***

float v_T1,v_T2,v_T3;

// the setup routine runs once when you press reset:
void setup() {
  Serial.begin(9600);
  pinMode(Sync, OUTPUT); //***
  analogReference(DEFAULT);//5V
  Serial.println("Inici");
}

void loop() {
  delay(1000);
  v_T1=analogRead(VoT1);
  Serial.print("T1="); Serial.print(v_T1);

  delay(1000);
  digitalWrite(Sync, HIGH); //***
  v_T2=analogRead(VoT2);
  digitalWrite(Sync, LOW);    //***
  Serial.print(",  T2="); Serial.print(v_T2);

  delay(1000);
  v_T3=analogRead(VoT3);
  Serial.print(",  T3="); Serial.println(v_T3);
}
```

Connecting 3.3 V to A0, the LM35 to A1 and GND to A2 we observe the following. As it is shown in *Figure* 1.1, an almost second order response is observed when the multiplexer selects the LM35 input (step-up purple signal). This response lasts more that 20 us. We also observe that the **analogRead** function lasts 120 us instead of the 100 us claimed at http://arduino.cc/en/Reference/analogRead.

The multiplexer takes action approximately 1.5 us after starting the execution of **analogRead**, as is showed in *Figure* 1.3. This figure also shows how the output changes from 250 mV (i.e. 25 °C) to 500 mV and then starts and oscillation that at the beginning seems a charge and discharge of an RC circuit and then a damped second order response with the expected final value 250 mV that we want to read. It is clear from *Figure* 1.2 that the reading must be done at least 30 us after the multiplexer has changed its

input.

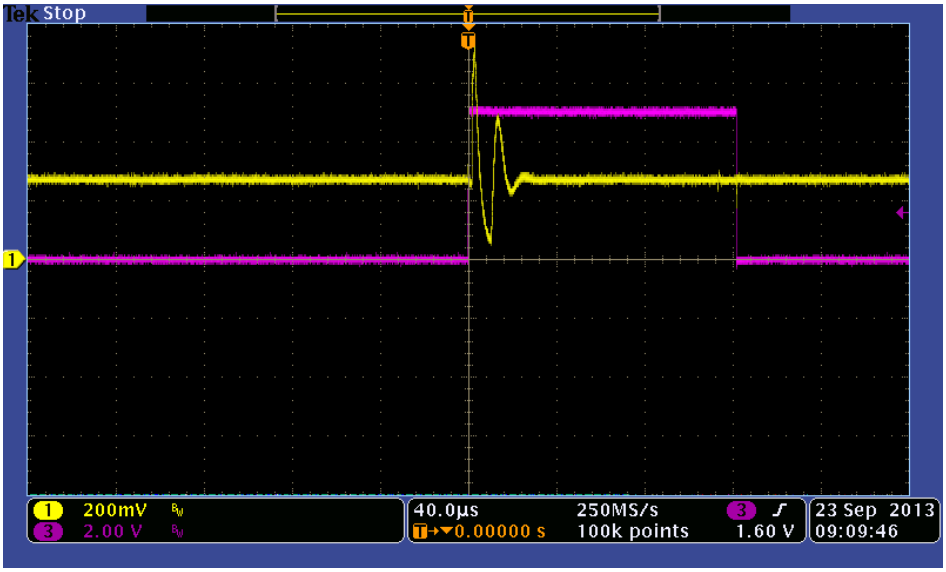

Figure 1.1: Transient response (yellow) $>$ 20 us. Execution time of the **analogRead** function (purple) $=$ 120 us.

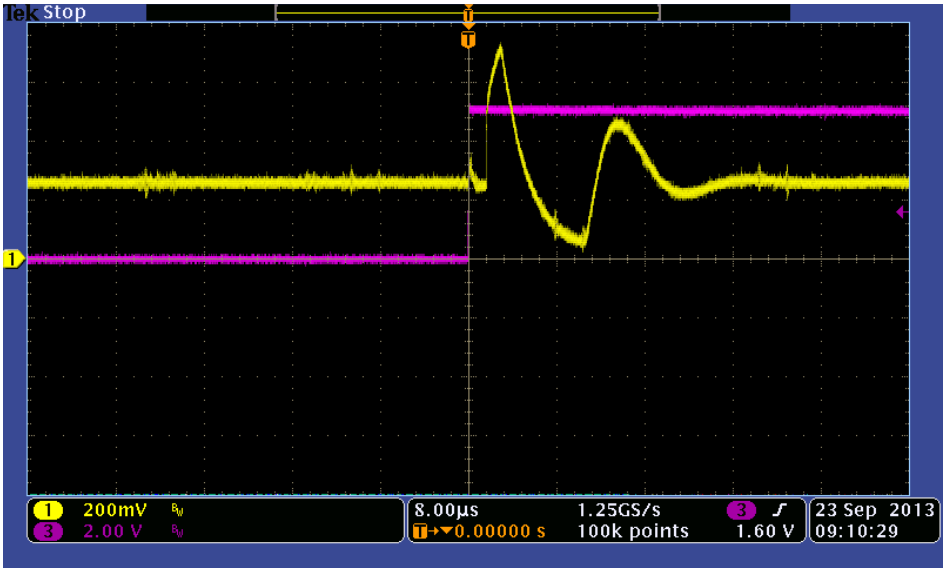

Figure 1.2: The damped oscillation lasts about 30 us to reach the final value of 250 mV (i.e. 25 °C).

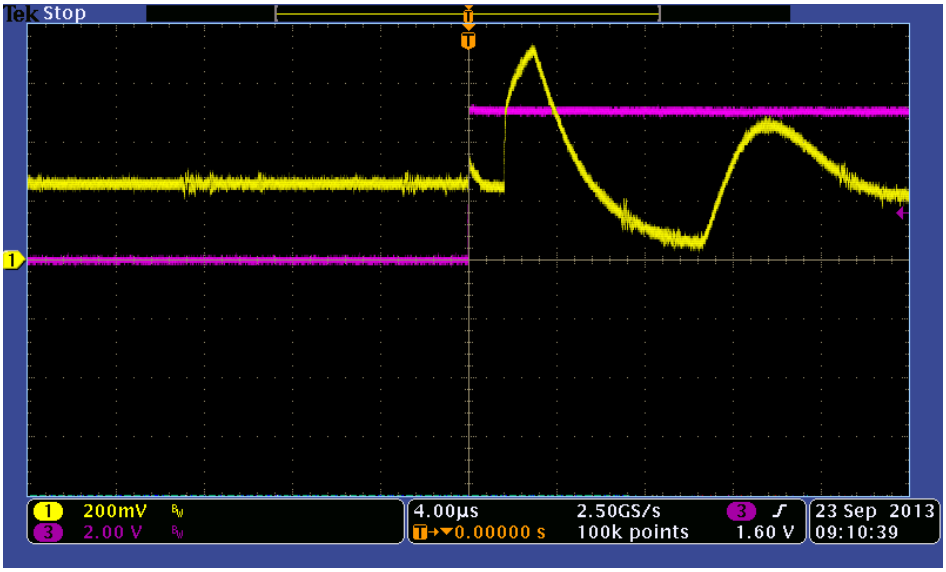

Figure 1.3: Multiplexer takes action 1.5 us after starting **analogRead**.

Connecting GND to A0, the LM35 to A1 and 3.3 V to A2 we observe the following. No response like the one observed previously has been found.

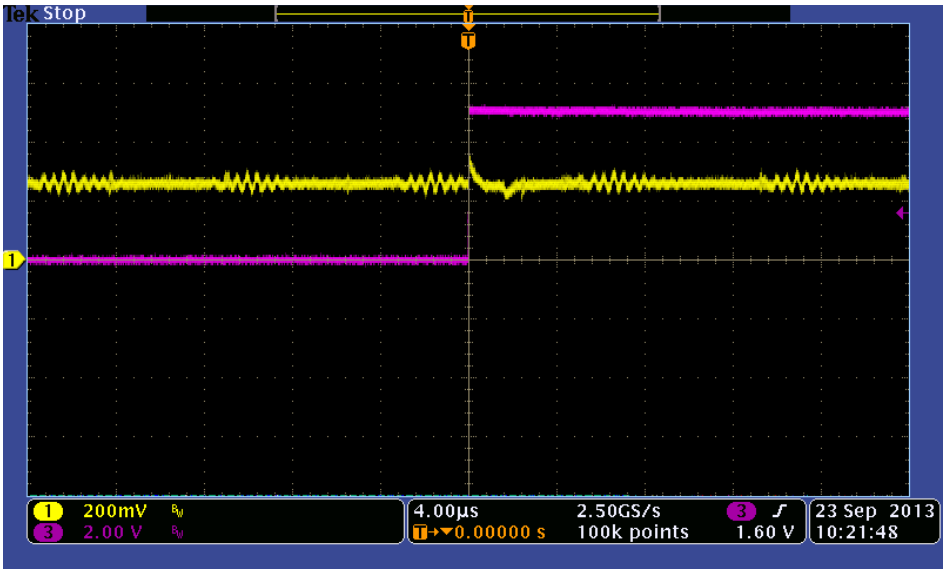

Figure 1.4: Transient response when the previous reading was from GND.

1.2.1 Conclusions

The LM35 output voltage at 25 °C is 250 mV. This value changes (as a damped oscillation) when it is connected to the RC sample and hold circuitry

that has been previously connected to a high voltage (i.e. 3.3 V). It takes about 30 us to reach again its expected value: 250 mV.

When the RC sample and hold circuitry has been previously connected to a low voltage (i.e. GND) the 250 mV doesn't change.

Chapter 2

Solving the stabilization problem

There is the hardware and the software solution.

2.1 Hardware solution

The LM35 datasheet suggests an RC damper circuit from output to GND to improve its limited ability to drive heavy capacitive loads. Using the suggested values, (75 Ω in series with 1 uF), the problem is solved.

2.2 Software solution

If we want to avoid the use of any extra hardware, we have two options.

2.2.1 Using an extra call to analogRead

The first one consist on adding a line to Xbee_4 and instead of

```
/*Xbee_4*/
  ...
  delay(1000);
  v_T2=analogRead(VoT2);
  Serial.print(",  T2=");Serial.print(v_T2);
 ...
}
```

use

```
/*Xbee_4*/
  ...
  v_T2=analogRead(VoT2);//to use if wiring_analog.c is not modified
  delay(1000);
  v_T2=analogRead(VoT2);
  Serial.print(",  T2=");Serial.print(v_T2);
```

```
  ...
}
```

This way, in the first call to **analogRead** the new input is selected and 120 us are consumed, next a delay (if more necessary) is used as an acquisition time and finally, during the second call, the result of the conversion is used.

2.2.2 Modification of analogRead

A second way consist on modifying the file where **analogRead** is defined, /home/user/arduino-1.0.5/hardware/arduino/cores/arduino/wiring_analog.c, and insert (actually uncomment) a delay between the selection of the multiplexer input, and the start of the conversion in order to increase the acquisition time. The comments and the inserted line are marked with //***, while the other are original lines

```
...
#if defined(ADMUX)
  ADMUX = (analog_reference << 6) | (pin & 0x07);
#endif

  // without a delay, we seem to read from the wrong channel
        // delay(1);

  //added to avoid noisy lecturers in high output impedances //***
  //sensors when coming from a channel with a high value //***
  delayMicroseconds(20); //***

#if defined(ADCSRA) && defined(ADCL)
  // start the conversion
  sbi(ADCSRA, ADSC);
...
```

Executing /*Xbee_4*/ and connecting 3.3 V to A0, the LM35 to A1 and GND to A2 we test different delays. With a delay of 10 us we read the following values:

```
T1=678.00,  T2=51.00,  T3=0.00
T1=679.00,  T2=53.00,  T3=0.00
T1=678.00,  T2=52.00,  T3=0.00
T1=678.00,  T2=52.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=50.00,  T3=0.00
T1=679.00,  T2=52.00,  T3=0.00
```

With a delay of 20 us or higher we read the following values:

```
Inici
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=50.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
```

```
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
T1=678.00,  T2=50.00,  T3=0.00
T1=678.00,  T2=51.00,  T3=0.00
```

So it seems that increasing the acquisition time 20 us is enough to read an stabilized voltage.

2.3 Conclusions

One hardware and two software solutions have been tested. Both have solved the problem, reading an stabilized voltage.

The hardware solution needs extra components, while no modification of the code is needed. The software solutions needs no extra components. The first software solution use an extra call to **analogRead** to include the necessary delay between calls. The cost of this solution is a minimum of 120 us, the execution time of the function. The main inconvenient of the second software solution is that we modify a library. This way, we increase the time needed to execute the **analogRead** function from 120 us to 140 us. **This penalty appears every time we execute analogRead, even if we are reading from an input where the stabilization problem isn't an issue.**

Chapter 3

The reference problem

Another troublesome aspect of the IDE environment is the use of the **analogReference** function. When using this function the analog reference of the ADC isn't changed, just a variable is changed. This variable is used when calling **analogRead** and the change of the analog reference is made simultaneously with the selection of the multiplexer input, and immediately before the start of the conversion. So, again the voltages aren't stabilized when the conversion is started.

If we always use the same reference, as in the previous code /*Xbee_4*/ there could be a problem, if there is, during the first readings.

But in our initial objective, we are reading from an humidity sensor whose output that can reach 5 V and from two temperature sensors whose output will go no further than 1 V (i.e. 100 °C). So choosing the DEFAULT reference of 5 V for the humidity sensor and the INTERNAL reference of 1.1 V for at least one of the temperature sensors will give better resolution.

3.1 Alternating the reference voltage

As is said in http://arduino.cc/en/Reference/AnalogReference: *After changing the analog reference, the first few readings from **analogRead()** may not be accurate.*

We have executed this code to test the effect of this change. First we repeat the reading two times without changing the analog reference.

```
/*Xbee_3*/

// define
#define VoT2 1   //A1

float v_T2,T2;

void setup() {
  Serial.begin(9600);
  Serial.println("Inici");
```

```
}

void loop() {
  analogReference(DEFAULT);
  delay(1000);
  v_T2=analogRead(VoT2);
  T2=v_T2*5.01*100/1024;//Vref=5.01V
  Serial.print("T2=");Serial.print(T2);Serial.print("C ");

  analogReference(DEFAULT);
  delay(1000);
  v_T2=analogRead(VoT2);
  T2=v_T2*5.01*100/1024;//Vref=5.01V
  Serial.print("T2="); Serial.print(T2);Serial.print("C ");

  Serial.println("");
}
```

We read the following temperatures with a resolution of 0.49 °C.

```
Inici
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=24.95C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=25.44C T2=25.44C
T2=24.95C T2=25.44C
```

Changing DEFAULT by INTERNAL and 5 by 1.1 gives stable readings with a resolution of 0.11 °C as is shown below. **Observe that the first reading (22.57C) is wrong. Also observe that the temperature is about one degree higher.**

```
Inici
T2=22.57C T2=26.40C
T2=26.61C T2=26.50C
T2=26.50C T2=26.40C
T2=26.50C T2=26.50C
T2=26.61C T2=26.50C
T2=26.61C T2=26.50C
T2=26.72C T2=26.61C
```

Next, we alternate from DEFAULT to INTERNAL and vice versa.

```
/*Xbee_3*/

// define
```

```
#define VoT2 1  //A1

float v_T2,T2;

void setup() {
  Serial.begin(9600);
  Serial.println("Inici");
}

void loop() {
  analogReference(DEFAULT);
  delay(1000);
  v_T2=analogRead(VoT2);
  T2=v_T2*5.01*100/1024;//Vref=5.01V
  Serial.print("T2="); Serial.print(T2);Serial.print("C ");

  analogReference(INTERNAL);
  delay(1000);/
  v_T2=analogRead(VoT2);
  T2=v_T2*1.09*100/1024;//Vref=1.09V
  Serial.print("T2="); Serial.print(T2);Serial.print("C ");

  Serial.println("");
}
```

We read the following temperatures. The first column with a resolution of 0.49 °C and the second column with a resolution of 0.11 °C.

```
Inici
T2=25.44C T2=4.26C
T2=25.44C T2=4.26C
T2=25.44C T2=4.26C
T2=25.44C T2=4.26C
T2=25.44C T2=4.26C
T2=25.93C T2=4.26C
```

Clearly, the reading with an INTERNAL reference after a DEFAULT reference is wrong, while the reading with a DEFAULT reference after an INTERNAL reference is right.

3.2 Using an extra call to analogRead

The function **analogReference** doesn't change the reference. It is made executing **analogRead**. It seems that the only solution to actually change the reference and read an stabilized reading is using an strategy similar to the one used in *Subsection* 2.2.1.

First execute **analogRead** to change the reference, second wait a time, and third execute again **analogRead**, this time to read a stable voltage.

The following code implements this idea. Observe that when using the DEFAULT reference we don't need to use the first call to **analogRead**.

```
/*Xbee_3*/
```

```
// define
#define VoT2 1  //A1

float v_T2,T2;

void setup() {
  Serial.begin(9600);
  Serial.println("Inici");
}

void loop() {
  analogReference(DEFAULT);
  //v_T2=analogRead(VoT2); //not necessary
  delay(1000);
  v_T2=analogRead(VoT2);
  T2=v_T2*5.01*100/1024;//Vref=5.01V
  Serial.print("T2="); Serial.print(T2);Serial.print("C ");

  analogReference(INTERNAL);
  v_T2=analogRead(VoT2);
  delay(1000);
  v_T2=analogRead(VoT2);
  T2=v_T2*1.09*100/1024;//Vref=1.09V
  Serial.print("T2="); Serial.print(T2);Serial.print("C ");

  Serial.println("");
}
```

As we can see below, the readings are stable, although there is still about one degree of difference between the two readings. This difference is inside the 2 LSB error given by the microcontroller: 0.98 °C.

```
Inici
T2=25.93C T2=26.61C
T2=25.44C T2=26.72C
T2=25.93C T2=26.50C
T2=25.93C T2=26.61C
T2=25.44C T2=26.61C
T2=25.44C T2=26.50C
T2=25.44C T2=26.72C
T2=25.44C T2=26.72C
T2=25.44C T2=26.50C
```

3.3 Conclusions

When using more than one reference we have to keep in mind a couple of things. First, changing from INTERNAL to DEFAULT is not a problem. Second, changing from DEFAULT to INTERNAL needs an extra delay between the selection of the reference and the conversion. This is made with an extra call to **analogRead** followed by a delay (10 ms seems to be enough).

Finally, it must be said that the use of an EXTERNAL reference (i.e. 3.3 V) in combination with a DEFAULT reference needs a higher delay, while in combination with an INTERNAL reference it simple doesn't work: only the EXTERNAL reference is well read.

Chapter 4

Final code

Using the previous knowledge, we have developed the following codes to read temperature (from two sensors) and humidity. We have chosen the software option that doesn't require the modification of **analogRead**.

4.1 Reference: DEFAULT

```
/*Xbee_2*/

// define
#define dela 500
#define led 13
#define llindar_T2 29

#define pin_T1 0//A0
#define pin_T2 1//A1
#define pin_H  2//A2

#define VsT1 17//A3
#define VsT2 18//A4
#define VsH  19//A5

float n_T1, n_T2, n_H, T1, T2, H, HT;

void setup() {
  Serial.begin(9600);
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
  // put digital pins at high level to use as a 5V source
  pinMode(VsT1, OUTPUT);
  pinMode(VsT2, OUTPUT);
  pinMode(VsH, OUTPUT);
  digitalWrite(VsT1, HIGH);
  digitalWrite(VsT2, HIGH);
  digitalWrite(VsH, HIGH);
```

```
  Serial.println("");Serial.println("");
  Serial.println("Inici");
}

void loop() {
  // Temperature sensor T1
  analogReference(DEFAULT);
  // ************************************************************
  n_T1=analogRead(pin_T1);//It is necessary after a pin of high value
  // ************************************************************
  delay(dela);
  n_T1=analogRead(pin_T1);
  T1=n_T1*5.01*100/1024;//Vref=5.01V
  Serial.print("T1=");Serial.print(T1);Serial.print("C ");

  // Temperature sensor T2
  analogReference(DEFAULT);
  delay(dela);
  n_T2=analogRead(pin_T2);
  T2=n_T2*5.01*100/1024;//Vref=5.01V
  //T2=n_T2*1.09*100/1024;//Vref=1.09V
  Serial.print("T2=");Serial.print(T2);Serial.print("C ");

  // The LED turns ON if T2>Ilindar_T2
    if (T2>Ilindar_T2) {
    digitalWrite(led, HIGH);}
    else{
    digitalWrite(led, LOW);}

  // Humidity sensor H
  analogReference(DEFAULT);
  delay(dela);
  n_H=analogRead(pin_H);
  H=(n_H/1024-0.16)/0.0062;
  Serial.print("H(@25C)=");Serial.print(H);Serial.print("% ");

  // temperature correction of the Humidity sensor H
  HT=((n_H/1024-0.16)/0.0062)/(1.0546-0.00216*T2);
  Serial.print("H(@T2)=");Serial.print(HT);Serial.print("%");

  Serial.println("");
}
```

In this first code, the readings are stabilized but we observe that T1 is 1 °C higher than T2.

```
Inici
T1=25.93C T2=24.95C H(@25C)=55.63% H(@T2)=55.59%
T1=25.93C T2=24.95C H(@25C)=55.15% H(@T2)=55.11%
T1=25.93C T2=24.95C H(@25C)=55.00% H(@T2)=54.96%
```

Changing

```
...
#define pin_T1 0//A0
```

```
#define pin_T2 1//A1
...
```

by

```
...
#define pin_T1 1
#define pin_T2 0
...
```

we read

```
Inici
T1=24.95C T2=25.93C H(@25C)=55.15% H(@T2)=55.23%
T1=24.95C T2=25.93C H(@25C)=55.31% H(@T2)=55.39%
T1=24.95C T2=25.93C H(@25C)=55.15% H(@T2)=55.23%
```

So, it seems that the reason of this temperature difference is in the sensor an not in the way this sensors are read.

4.2 Reference: DEFAULT and INTERNAL

We use again

```
...
#define pin_T1 0//A0
#define pin_T2 1//A1
...
```

and in order to read with the INTERNAL reference voltage T2 we change

```
...
  // Temperature sensor T2
  analogReference(DEFAULT);
  delay(dela);
  n_T2=analogRead(pin_T2);
  T2=n_T2*5.01*100/1024;//Vref=5.01V
...
```

by

```
...
  // Temperature sensor T2
  analogReference(INTERNAL);
  // ************************************************************
  n_T2=analogRead(pin_T2);//It is necessary when changing to INTERNAL
  // ************************************************************
  delay(dela);
  n_T2=analogRead(pin_T2);
  T2=n_T2*1.09*100/1024;//Vref=1.09V
...
```

These are the readings

```
Inici
T1=25.93C T2=26.08C H(@25C)=55.63% H(@T2)=55.72%
T1=25.93C T2=25.97C H(@25C)=55.94% H(@T2)=56.03%
T1=25.93C T2=26.08C H(@25C)=55.78% H(@T2)=55.88%
```

Now, when using the INTERNAL reference T2 is 1 °C higher than when the DEFAULT reference was used. As said before, this difference is inside the 2 LSB error given by the microcontroller: 0.98 °C when using DEFAULT reference.

Chapter 5

Conclusions

The use of the Analog to Digital Converter in the Arduino IDE environment can give some problems because we cannot control the delays between critical operations such as the selection of the reference or the input and the acquisition time before starting the conversion.

http://garretlab.web.fc2.com/en/arduino/inside/arduino/wiring_analog.c/analogReference.html

http://garretlab.web.fc2.com/en/arduino/inside/arduino/wiring_analog.c/analogRead.html