Ministerul Educatiei al Republicii Moldova
Universitatea Tehnica a Moldovei
Filiera Anglofona

Report

Embedded Systems

Laboratory Work #3

Performed by: Vlad Croitoru

Verified by: Andrei Bragarenco

Chisinau 2017

Topic:

Analog Digital Conversion of AVR Microcontroller. Temperature measurement using LM20 Sensor.

Task:

Retrieve the data from a temperature sensor(analog value that has to be converted). The default value to be displayed will be in **°C**. There will be two buttons, used to switch from **°C** to **°K or °F.** The temperature has to be displayed on an LCD interface or virtual terminal.

Overview:

Microcontrollers are capable of detecting binary signals: is the button pressed or not? These are digital signals. When a microcontroller is powered from five volts, it understands zero volts (0V) as a binary 0 and a five volts (5V) as a binary 1. The world however is not so simple and likes to use shades of gray. What if the signal is 2.72V? Is that a zero or a one? We often need to measure signals that vary; these are called analog signals. A 5V analog sensor may output 0.01V or 4.99V or anything inbetween. Luckily, nearly all microcontrollers have a device built into them that allows us to convert these voltages into values that we can use in a program to make a decision.

What is ADC?

An Analog to Digital Converter (ADC) is a very useful feature that converts an analog voltage on a pin to a digital number. By converting from the analog world to the digital world, we can begin to use electronics to interface to the analog world around us.

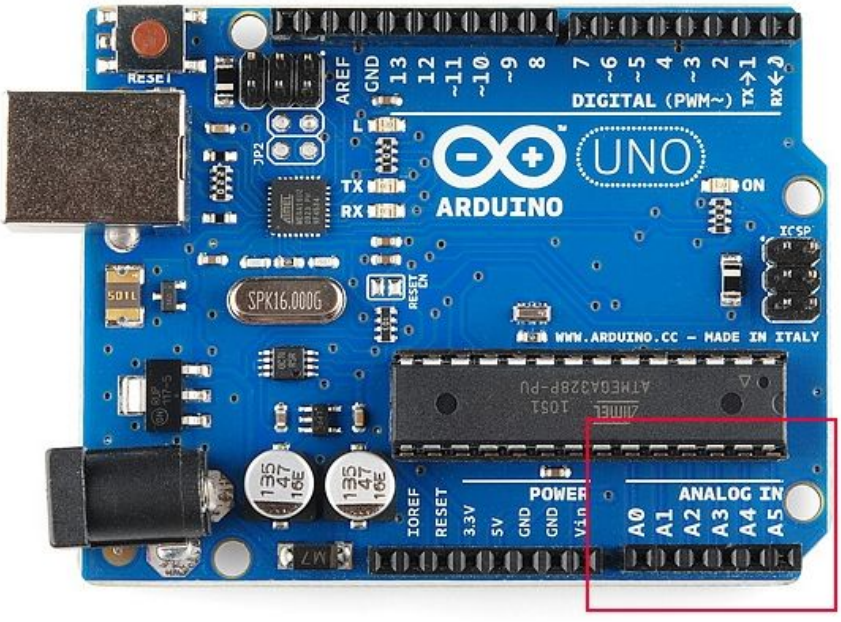


Figure 1: Physical representation of ADC pins on Arduino

Not every pin on a microcontroller has the ability to do analog to digital conversions. On the Arduino board, these pins have an 'A' in front of their label (A0 through A5) to indicate these pins can read analog voltages.

ADCs can vary greatly between microcontroller. The ADC on the Arduino is a 10-bit ADC meaning it has the ability to detect 1,024 (210) discrete analog levels. Some microcontrollers have 8-bit ADCs (28 = 256 discrete levels) and some have 16-bit ADCs (216 = 65,535 discrete levels).

The way an ADC works is fairly complex. There are a few different ways to achieve this feat (see Wikipedia for a list), but one of the most common technique uses the analog voltage to charge up an internal capacitor and then measure the time it takes to discharge across an internal resistor. The microcontroller monitors the number of clock cycles that pass before the capacitor is discharged. This number of cycles is the number that is returned once the ADC is complete.

Inbuilt ADC of AVR

The ADC is multiplexed with PORTA that means the ADC channels are shared with PORTA. The ADC can be operated in single conversion and free running more. In single conversion mode the ADC does the conversion and then stop. While in free it is continuously converting. It does a conversion and then start next conversion immediately after that.

ADC Channels

The ADC in ATmega32 has 8 channels that means you can take samples from eight different terminal. You can connect up to 8 different sensors and get their values separately.

ADC Registers.

As you know the registers related to any particular peripheral module(like ADC, Timer, USART etc.) provides the communication link between the CPU and that peripheral. You configure the ADC according to need using these registers and you also get the conversion result also using appropriate registers. The ADC has only four registers. ADC Multiplexer Selection Register – ADMUX : For selecting the reference voltage and the input channel. ADC Control and Status Register A – ADCSRA : As the name says it has the status of ADC and is also use for controlling it.
The ADC Data Register – ADCL and ADCH : The final result of conversion is here.

LM20

The LM20 is a precision analog output CMOS integrated-circuit temperature sensor that operates over −55°C to 130°C. The power supply operating range is 2.4 V to 5.5 V. The transfer function of LM20 is predominately linear, yet has a slight predictable parabolic curvature. The accuracy of the LM20 when specified to a parabolic transfer function is ±1.5°C at an ambient temperature of 30°C. The temperature error increases linearly and reaches a maximum of ±2.5°C at the temperature range extremes. The temperature range is affected by the power supply voltage. At a power supply voltage of 2.7 V to 5.5 V, the temperature range extremes are 130°C and −55°C. Decreasing the power supply voltage to 2.4 V changes the negative extreme to −30°C, while the positive extreme remains at 130°C.

The LM20 quiescent current is less than 10 µA. Therefore, self-heating is less than 0.02°C in still air. Shutdown capability for the LM20 is intrinsic because its inherent low power consumption allows it to be powered directly from the output of many logic gates or does not necessitate shutdown.

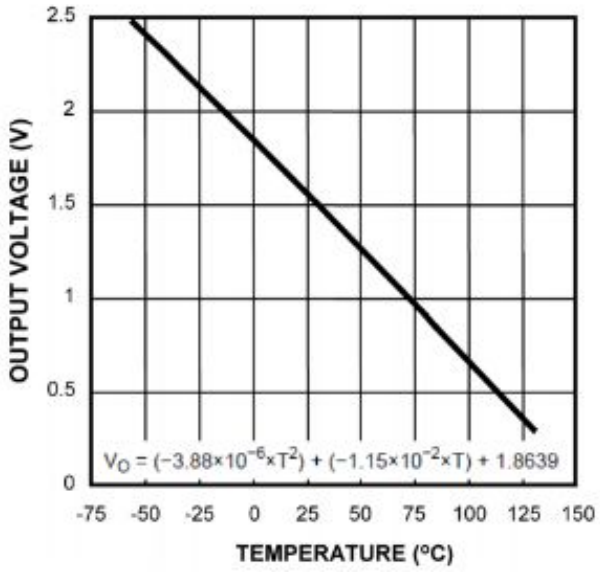


Chart 1: *Output voltage vs. Temperature*

Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Celsius to Fahrenheit Conversion

The temperature *T* in degrees Fahrenheit (°F) is equal to the temperature *T* in degrees Celsius (°C) times 9/5 plus 32:

$$T_{(°F)} = T_{(°C)} \times 9/5 + 32$$

Celsius to Kelvin Conversion

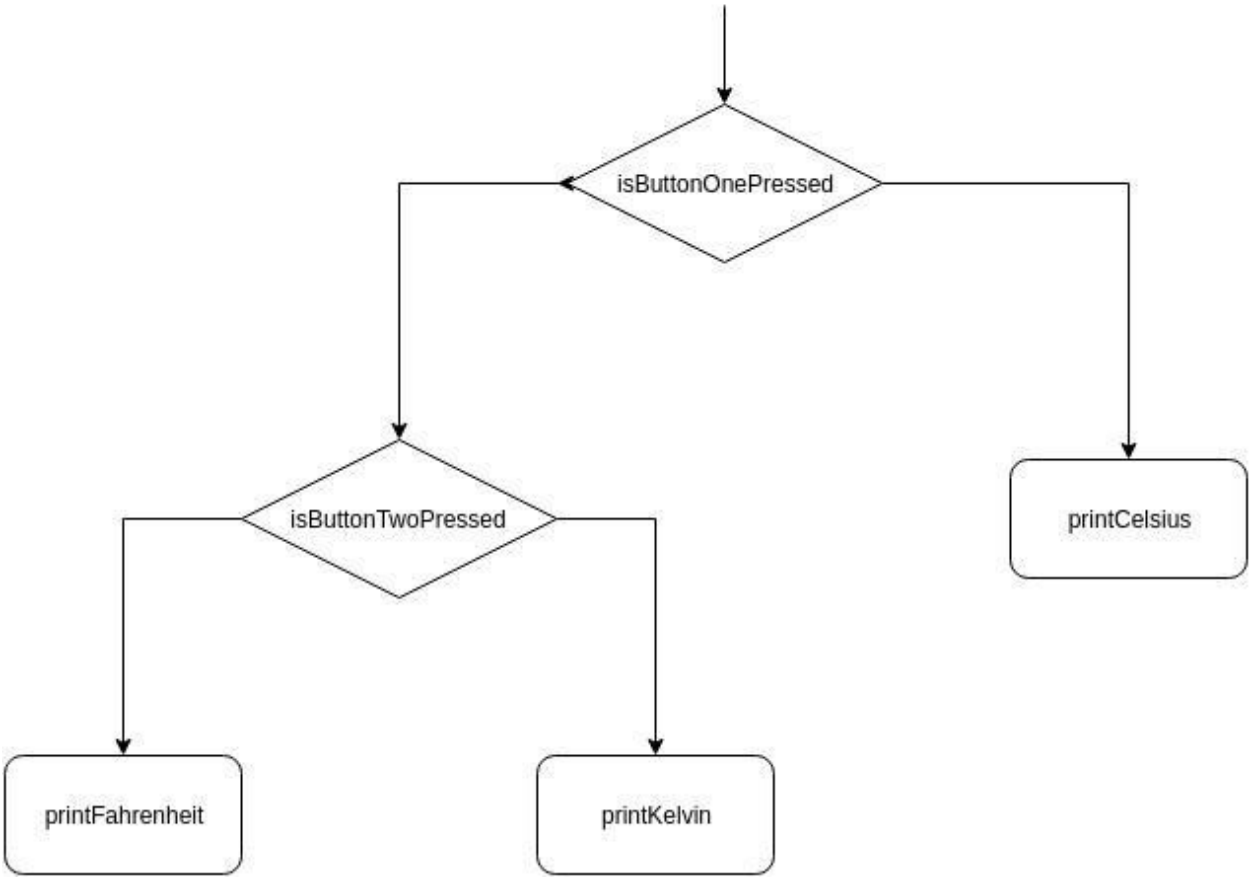
The temperature *T* in Kelvin (K) is equal to the temperature *T* in degrees Celsius (°C) plus 273.15:

$$T_{(K)} = T_{(°C)} + 273.15$$

Solution:

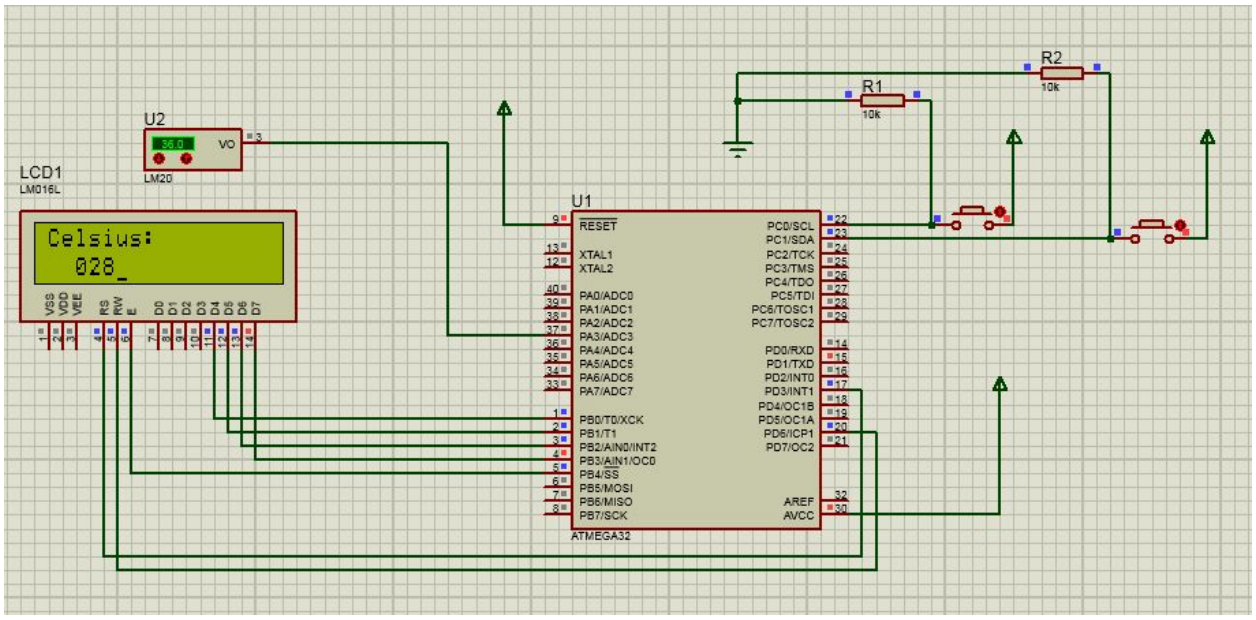
The main point of this laboratory work is Analog Digital Conversion and temperature measurement using LM20 Sensor(voltage to temperature conversion).

The main function goes following the next flowchart:

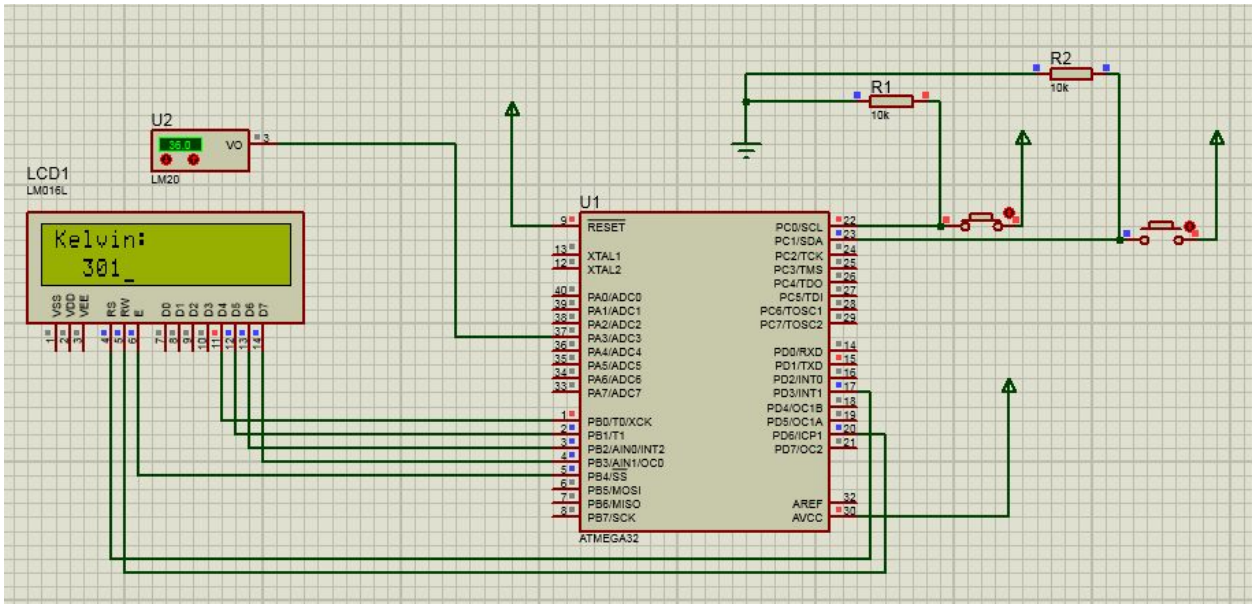


The result:

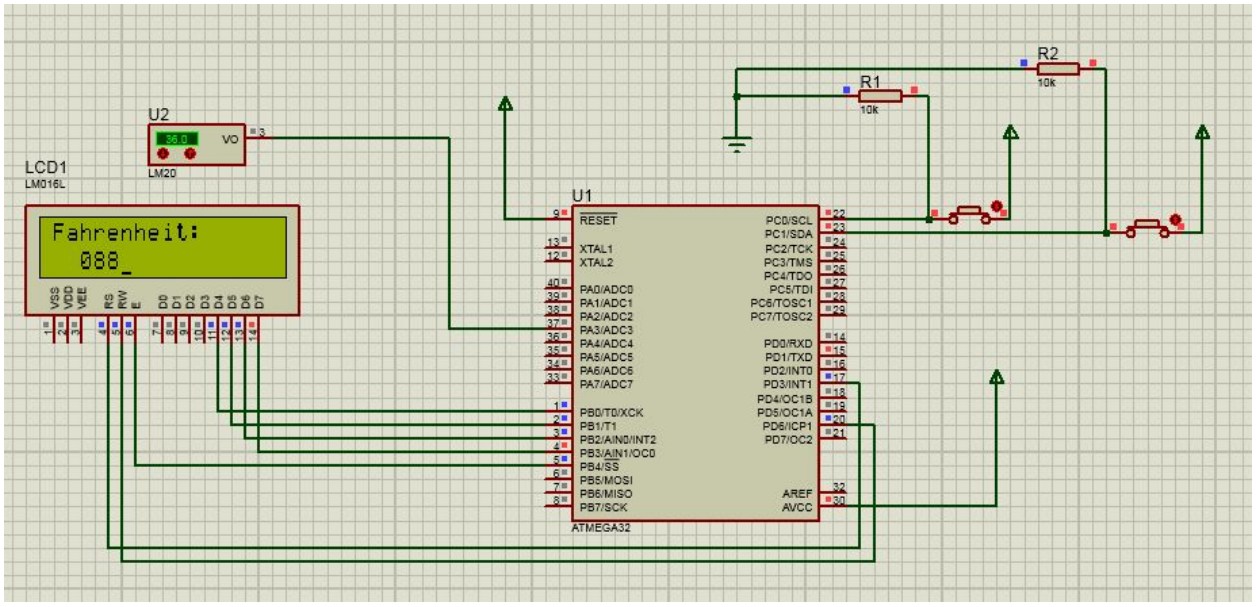
Celsius (no buttons pressed)



Kelvin (one button pressed)



Fahrenheit (both buttons pressed)



Conclusion:

The main thing we got from this laboratory work is that we learned how to get analog values from the environment and convert them into digital values by ADC. How to connect an LCD screen to an Atmega32 microcontroller and display the required information on it, which in itself opens a whole new domain of utility.

Appendix:

main

```
#include <avr/io.h>
#include "button.h"
#include "lm20.h"
#include "lcd.h"
#include <avr/delay.h>


int main(void) {

    initButtonOne();
    initButtonTwo();
    initLM();
    uart_stdio_Init();

    //Initialize LCD module
   LCDInit(LS_BLINK|LS_ULINE);

   //Clear the screen
   LCDClear();
```

```
   while(1) {
       _delay_ms(1000);

       if(isButtonOnePressed()) {
           if(isButtonTwoPressed()) {
               LCDClear();
               LCDWriteString("Fahrenheit:");
               LCDWriteIntXY(1, 1, convertCelsiusToFahrenheit(getTemp()),3);
               printf("Fahrenheit: %d\n", convertCelsiusToFahrenheit(getTemp()));
           }else {
               LCDClear();
               LCDWriteString("Kelvin:");
               LCDWriteIntXY(1, 1, convertCelsiusToKelvin(getTemp()),3);
               printf("Kelvin: %d\n", convertCelsiusToKelvin(getTemp()));
           }

       } else {
           LCDClear();
           LCDWriteString("Celsius:");
           LCDWriteIntXY(1, 1, getTemp(),3);
           printf("Celsius : %d\n", getTemp());
       }
   }
}
```

lm20.h

```c
#ifndef LM20_H_
#define LM20_H_

#include "adc.h"

void initLM();
int getTemp();
int convertCelsiusToKelvin(int temp);



#endif /* LM20_H_ */
```

lm20.c

```c
#include "lm20.h"

int temp = 0;


void initLM() {
    initADC();

}

int getTemp() {
    temp = (382 - getData()) / 3;
    return temp;
}

int convertCelsiusToKelvin(int temp) {
    return temp + 273;
}

int convertCelsiusToFahrenheit(int temp) {
    return temp * 2 + 32;
}
```

adc.h

```
#ifndef ADC_H_
#define ADC_H_

void initADC();
int getData();
void toVoltage(int t);



#endif /* ADC_H_ */
```

adc.c

```
#include "adc.h"
#include <avr/io.h>
int data;

void initADC() {

    ADMUX = (1 << REFS0);
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
}

int getData() {

    int adcData = 0;
    int port = 3;
    while(ADCSRA & 1 << ADSC);
    port &= 0x07;
    ADMUX = (ADMUX & ~(0x07)) | port;
    ADCSRA |= (1<<ADSC);
    while (ADCSRA & (1<<ADSC));
    adcData = ADC;
    return adcData;

}
```