University of Montenegro

Faculty of Electrical Engineering

Course: **Industrial electronic**

Theme: **PH sensor**

Mentor:

Prof. dr Radovan Stojanović

Students:

Olivera Nikčević 14/19
Filip Živković 35/19
Budimir Anđelić 30/19

Date and place: 20/04/2020

Podgorica

Contents

Introduction

This is a lab project, made in MEDEL electronics lab at the University of Montenegro, with mentoring of prof. dr Radovan Stojanović. In further text, the problem is explained in details and offered an elegant solution. In addition to that, we have enclosed high level design and description of our solution, alongside with hardware and software structure. The accent is on PH sensor, using Arduino Uno board and open-source Software (IDE).

What is a timer?

A timer or to be more precise a timer/counter is a piece of hardware built in the Arduino controller (other controllers have timer hardware, too). It is like a clock, and can be used to measure time events. The timer can be programmed by some special registers. You can configure the pre-scaler for the timer, or the mode of operation and many other things. The controller of the Arduino is the Atmel AVR ATmega168 or the ATmega328. These chips are pin compatible and only differ in the size of internal memory. Both have 3 timers, called timer0, timer1 and timer2. Timer0 and timer2 are 8bit timer, unlike timer1 which is a 16bit timer. The most important difference between 8bit and 16bit timer is the timer resolution. 8bits means 256 values where 16bit means 65536 values for higher resolution. All timers depends on the system clock of your Arduino system. Normally the system clock is 16MHz. The timer hardware can be configured with some special timer registers. In the Arduino firmware all timers were configured to a 1 kHz frequency and interrupts are generally enabled.

Timer Register

You can change the Timer behavior through the timer register. The most important timer registers are:

- TCCRx - Timer/Counter Control Register. The pre-scaler can be configured here.
- TCNTx - Timer/Counter Register. The actual timer value is stored here.
- OCRx - Output Compare Register
- ICRx - Input Capture Register (only for 16bit timer)
- TIMSKx - Timer/Counter Interrupt Mask Register. To enable/disable timer interrupts.
- TIFRx - Timer/Counter Interrupt Flag Register. Indicates a pending timer interrupt.

Clock select and timer frequency

Different clock sources can be selected for each timer independently. To calculate the timer frequency (for example 2Hz using timer1) you will need:

1. CPU frequency 16Mhz for Arduino
2. Maximum timer counter value (256 for 8bit, 65536 for 16bit timer)
3. Divide CPU frequency through the chosen pre-scaler (16000000 / 256 = 62500)
4. Divide result through the desired frequency (62500 / 2Hz = 31250)
5. Verify the result against the maximum timer counter value (31250 < 65536 success) if fail, choose bigger pre-scaler.

Timer modes

Timers can be configured in different modes:

- PWM mode. Pulse width modulation mode. the OCxy outputs are used to generate PWM signals
- CTC mode. Clear timer on compare match. When the timer counter reaches the compare match register, the timer will be cleared.

What is an interrupt?

The program running on a controller is normally running sequentially instruction by instruction. An interrupt is an external event that interrupts the running program and runs a special interrupt service routine (ISR). After the ISR has been finished, the running program is continued with the next instruction. Interrupts must be generally enabled or disabled with the function interrupts () / nointerrupts ().

Timer interrupts

A timer can generate different types of interrupts. The register and bit definitions can be found in the processor data sheet and in the I/O definition header file.

PH Sensor

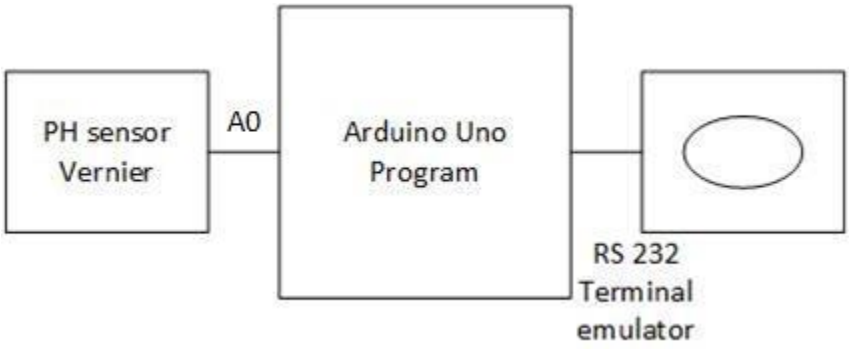
For our problem we use Vernier PH sensor. The pH Sensor can be used for any lab or demonstration that can be done with a traditional pH meter, including acid-base titrations, monitoring pH in an aquarium, and investigating the water quality of streams and lakes. The pH amplifier inside the handle is a circuit that allows the standard combination pH electrode to be monitored by a lab interface. The cable from the pH amplifier ends in a BTA plug. The PH sensor produces a voltage of approximately 1.75 volts in a pH 7 buffer. The voltage increases by about 0.25 volts for every pH number decrease. The voltage decreases by about 0.25 volts/pH number as the pH increases. This sensor is designed to be used in aqueous solutions and designed to make measurements in the PH range of 0 to 14. For many experiments, calibration of the pH sensor is not required. The calibration equation is stored on each pH sensor prior to delivery, which is used by the Vernier software as default. We recommend calibration for the most accurate measurements using this sensor. To calibrate the pH sensor or confirm that the stored pH calibration is correct, you should have a supply of pH solution buffers covering the pH range to be measured. With this sensor, in our project, we measure the pH value of three liquids: water, salt water and vinegar.

Problem description

Our job is the sampling of sensors on the timer interrupt level, with the possibility of the START/STOP and the possibility to set the sampling frequency. Results, and values of measurements will be shown on the serial monitor.

Solution

In further text, we are showing two solutions for this problem and short description for program. In the first method we use loop, and in second method we use timer interrupt.



High level design

Short description

The program starts when A is pressed. When B is pressed, the program stops working. When we enter C200 we get a period of 200ms. Also, if C and any other number are entered, the sampling period will be that number. After each change of period its values are stored in EEPROM. During the execution of commands, the device remember the last entered period, so that in the case of a power failure, the sampling period will be the last entered value.

Code

Timer interrupt method:

```
#include <avr/eeprom.h>
unsigned long int storage;
String inString = "A";
int sensorvalue = 0 ;
int milis;
float napon;
float ph;
int addr = 0;
float frekv;
boolean procitao =false;
void setup() {
  pinMode(A0,INPUT);
  Serial.begin(57600);
  noInterrupts();
// TIMER 1 for interrupt frequency
of 1 Hz:
```

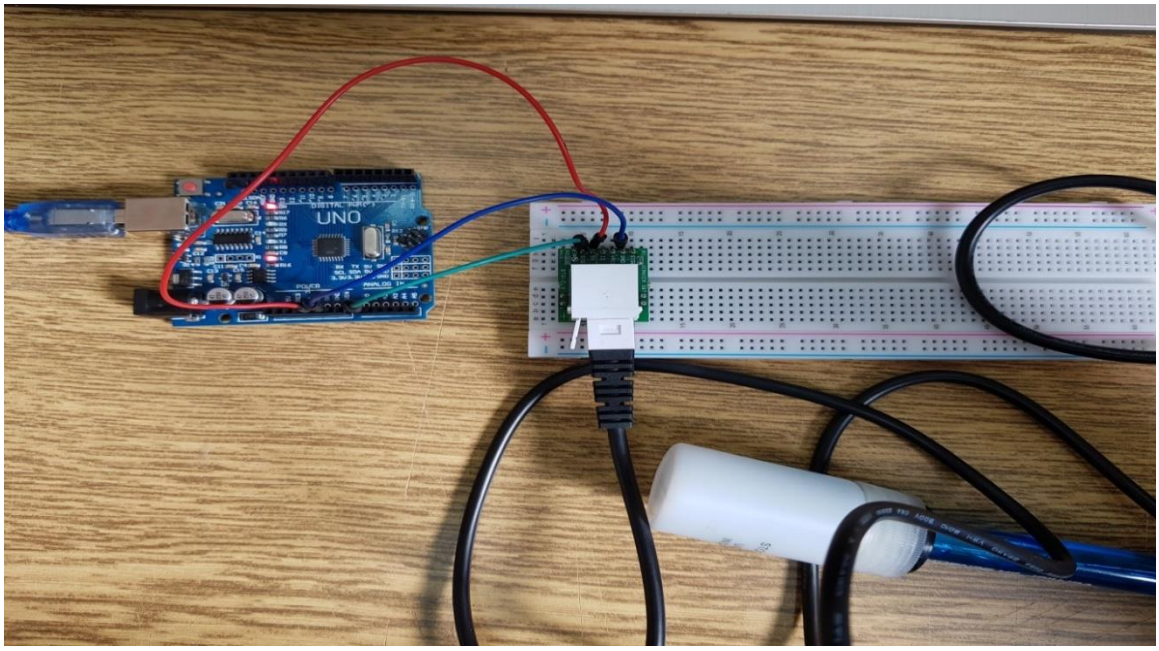
```
TCCR1A = 0; // set up TCCR1A
register to 0
TCCR1B = 0; // set up TCCR1B
register to 0
TCNT1  = 0; // initialize counter
to 0
// set up compare match register
for 1 Hz inkremente
storage = eeprom_read_word (0x00);
if(storage!=0){
OCR1A = storage;}
else{
  OCR1A=62499;
  Serial.println("Frekvencija je
1HZ");
}
// turn up CTC mode
TCCR1B |= (1 << WGM12);
// set up CS12, CS11 i CS10 bits
for 256 pre-scaler
TCCR1B |= (1 << CS12) | (0 << CS11)
| (0 << CS10);
// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);
interrupts();
}
void loop() {
  if(procitao && (inString=="A" ||
inString[0]=='C')){  //  if enter A
or Cxxx.. where x are some numbers,
we start
    Serial.print("PH: ");
    Serial.println(ph);
    procitao=false;
  }
  if(procitao==false &&
inString=="B"){ //  if enter B then
we stop
    procitao=true;
  }
}
void serialEvent(){
  if(Serial.available()>0){
    inString=Serial.readString();
  }
  if(inString[0]=='C'){ //  we
extract xxx from the string Cxxx ie
the sampling period
    milis=0;
    for (int
i=1;i<inString.length();i++){
      if(isDigit(inString[i])){

milis=(milis*10+(inString[i]-'0'));
        }
        }
        frekv=(1000/(float)milis);
        OCR1A = (int)(16000000.0 /
(256.0 * frekv) - 1); //  change
the value with which we compare the
counter ie. we change the timer
frequency
        eeprom_write_word (0x00,
OCR1A);
        procitao=true;
        }
}
ISR(TIMER1_COMPA_vect){
  sensorvalue=analogRead(A0); //
read the sensor value from 0 to
1023
  napon = 5*(sensorvalue/1023.0);
// get a voltage value from 0 to 5V
  ph=14-napon/0.25; // get the ph
value over voltage
 procitao=true; }
```

Polling method:

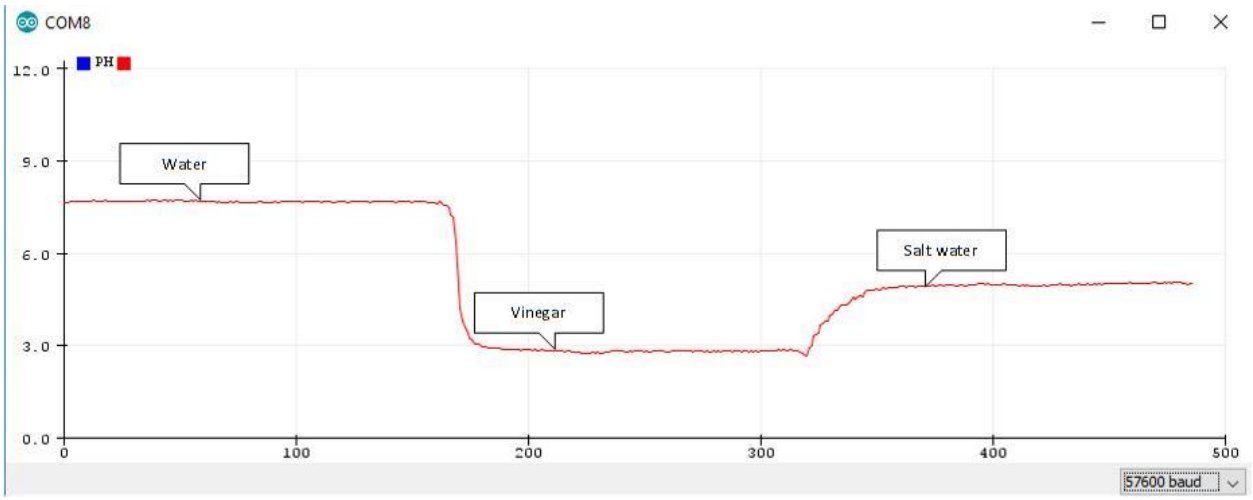
```
#define sensorPin A0
int frekvencija=500;
int f=0;
String START ="A";
String STOP ="B";
String str="B";
int i;
float napon;
float PH;
int vrijednost=0;
void setup() {
  pinMode(sensorPin,INPUT);
  Serial.begin(57600);
}
void loop() {
  if(Serial.available()>0){
    str=Serial.readString();
    if(str==START){
      Serial.println("START");
    }
    if(str==STOP){
      Serial.println("STOP");
```

```
    }
  }
    if(str[0]=='C'){
      f=0;
      for (i=1;i<str.length();i++){
        if(isDigit(str[i])){
          f=(f*10+(str[i]-'0'));
        }
    }
    frekvencija=f;
    Serial.print("Frekvencija
odabiranja je ");
    Serial.print(frekvencija);
    Serial.println("ms");
    str="A";
    }

  if(str==START and str!=STOP){
      vrijednost =
analogRead(sensorPin);
      napon =
5*(vrijednost/1023.0);
      PH=14-napon/0.25;
      Serial.println(PH);
 delay(frekvencija);

  }
  if(str==STOP){
  }
```

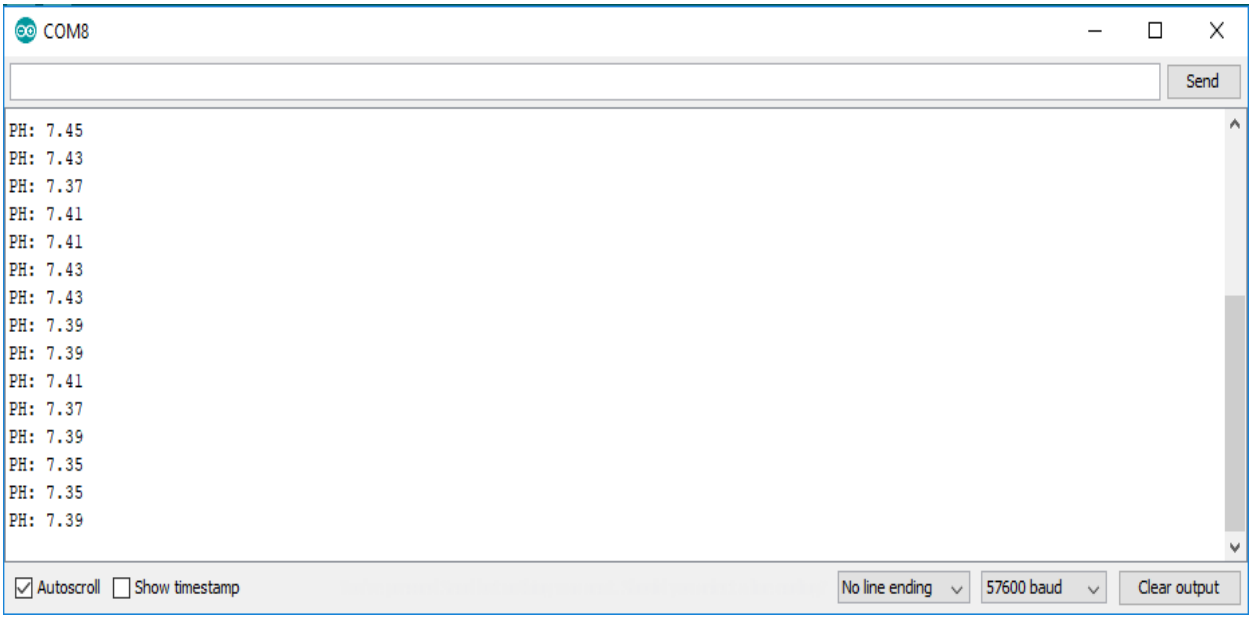
Hardware Solution



Red wire is connected to an Arduino board on 5V, blue on ground, while the green wire is on analogue port A0.



Diagram



Serial monitor

Conclusion

In this simulation and explanation, we were using two different methods of programming this problem (polling and timer interrupt method). Timer interrupt method is better solution, and the programme is working as follows:

- The programme does not read values of PH sensor very often, only in case we need it. For example if the period is 1s, then we will read value of sensor measurements every second.
- On the other hand, using pooling method programme reads values of measurements every moment and we have to use delay function. This causes problems with other things we have to do, until delay is done.