ENVIRONMENTAL MONITORING USING ATMEL MICROCONTROLLER

ALEXANDRA (STANIMIRESCU) ŞOICA[1]

Abstract: The purpose of this paper is to establish the maximum speed of microcontroller (from a Arduino uno board) when used with different environmental monitoring sensors, compared to the rated maximum frequency stated in the datasheet (16 MHz).

Keywords : environment, speed, microcontroller, arduino

1. INTRODUCTION

As I am living in the age of speed and everything is moving faster around us it only seem natural that I am concern if our projects are good enough and achieve maxim speed.

In other projects [1][2] I have made various environmental measurements an now I have to find out the real speed of measurement versus the rated speed. Air quality monitoring is very important in nowadays and it is an established science that began in the 1980s [2].

Due to many industrial activities and various pollutants released in the environment it is highly recommended to track the evolution of the environment quality and intervene when necessary.

The environment affects all human health and plays a major role in quality of life, years of healthy life lived, and health issues. Poor air quality can cause premature death, cancer, and other long-term damage to respiratory and cardiovascular systems.

The most efficient way to prevent environmental hazard is to permanently measure all quality indicators and based on those measurements take all necessary actions to prevent environmental degradation

[1] *Assistant, Eng. Ph.D., University of Petrosani, alexandra_valynikalay@yahoo.com*

2. METHODOLOGY AND RESULTS

The materials used in order to establish the maximum real speed of the Atmel ATMEGA328P are the following:
- Arduino uno board
- Siglent SDS 1202X-E Oscilloscope
- Arduino IDE open-source software
- EasyScopeX software

In order to measure the maximum real speed I have tried a few approaches as described in the following lines.

The first attempt was using the included "ready to use" functions of the Arduino IDE software and I have used the included "Blink" code with minimum delay:

```
void setup()
{
      pinMode(13, OUTPUT);
}
void loop()
{
      digitalWrite(13, HIGH);
      delay(1);
      digitalWrite(13, LOW);
      delay(1);
}
```

The next step was to reduce the delay even more using replacing the delay function with delay `Microseconds ()` function. I went further and removed the delay function completely.

These included functions and codes from Arduino IDE were not enough and made us go further and explore better option to achieve maximum speed. In our opinion, the best way to this is by manipulation the microcontrollers registers directly instead of digital `Write ()` function. For this purpose I wrote the following program versions :

Table 1 **The first version of the program**

Minimal delay (miliseconds)	Microseconds delay

`void setup()`
`{`
`  pinMode(13, OUTPUT);`
`}`
`void loop()`
`{`
`  PORTB = B00100000;`
`  delay(1);`
`  PORTB = B00000000;`
`  delay(1);`
`}`
	`void setup()`
	`{`
	`  pinMode(13, OUTPUT);`
	`}`
	`void loop()`
	`{`
	`  PORTB = B00100000;`
	`  delayMicroseconds(1);`
	`  PORTB = B00000000;`
	`  delayMicroseconds(1);`
	`}`

Table 2-**The secund version of the program**

No delay	Full memory

`void setup ()`
`{`
`  pinMode(13, OUTPUT);`
`}`
`  void loop() {`
`  PORTB = B00100000;`
`  PORTB = B00000000;`
`}`
	`void setup()`
	`{`
	`  pinMode(13, OUTPUT);`
	`}`
	`  void loop() {`
	`  PORTB = B00100000;`
	`  PORTB = B00000000;`
	`//.. (repeat until the ATMEGA328P memory is full)`
	`  PORTB = B00100000;`
	`  PORTB = B00000000;`
	`}`

After developing the software part of the microcontroller I connected digital pin 13 to the oscilloscope and obtained the frequency and waveform for each software version me made.

Figure no. 1 describes the characteristic of the signal generated with Blink code and minimum delay. This is the slowest frequency obtain (495,518 Hz).

The following figures show the characteristics of each signal generated by the different software codes presented earlier**.**

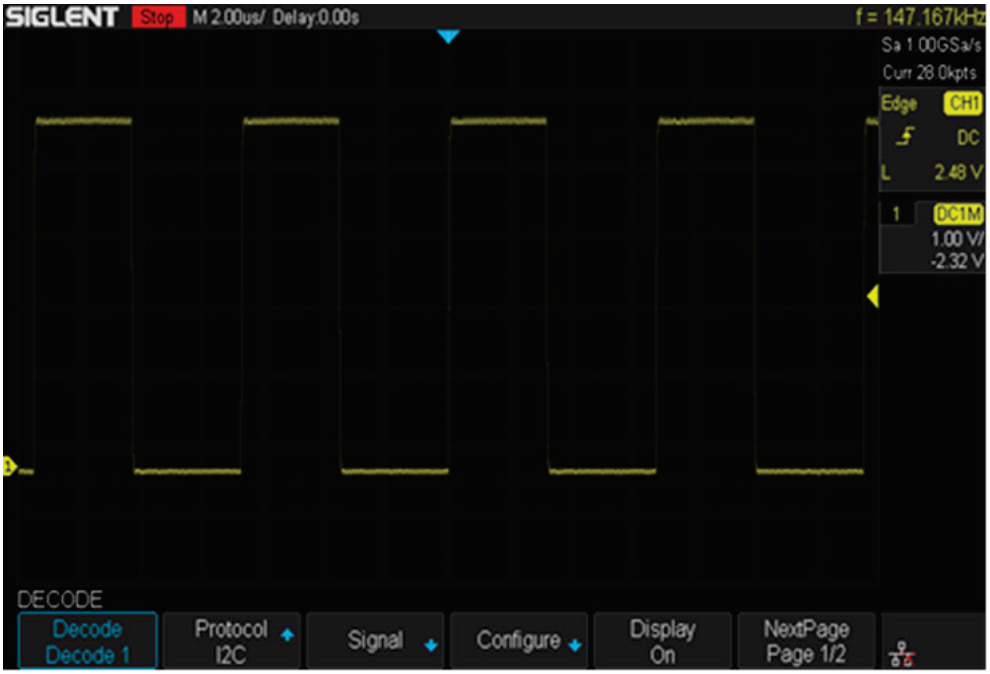

Fig. 1 Blink code with minimum delay

Fig. 2 Blink code with delayMicroseconds() function

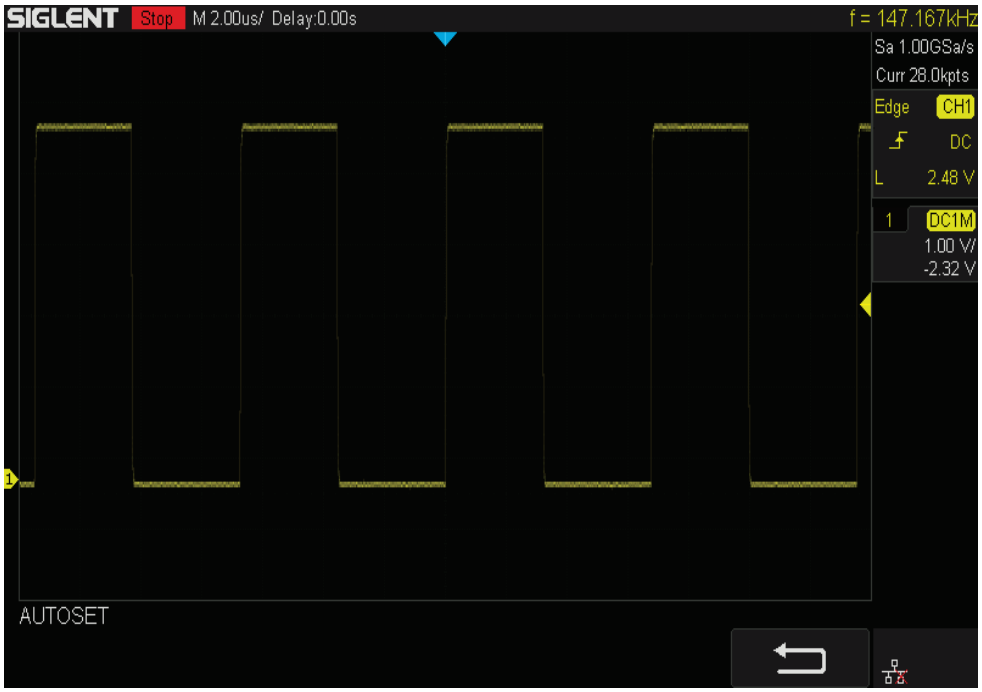

Fig. 3 Blink code without delay

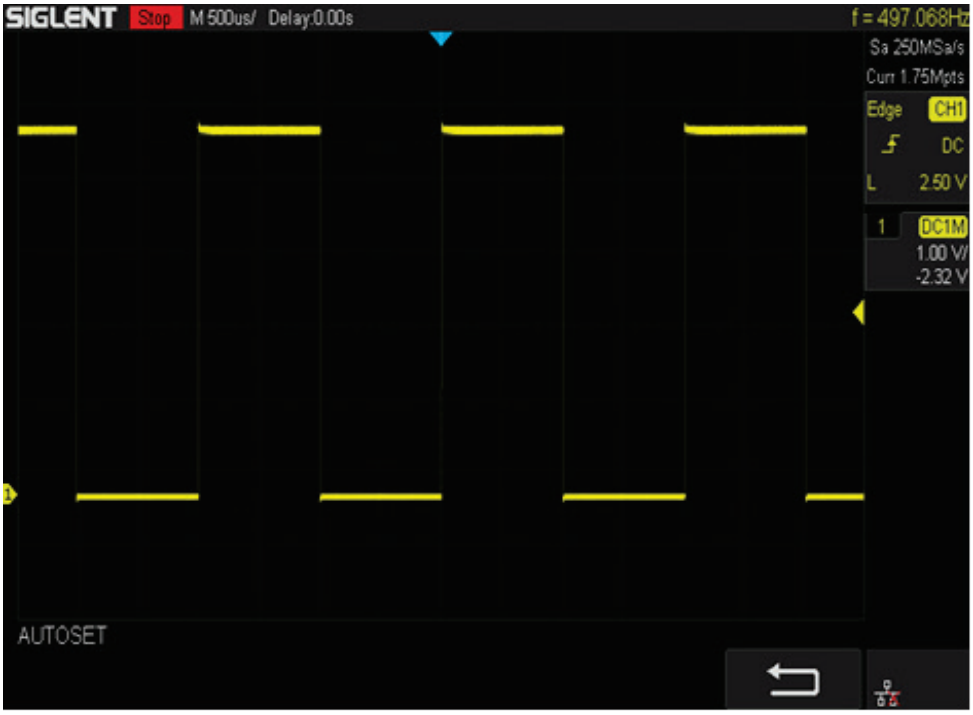

Fig. 4 Port manipulation with minimal delay

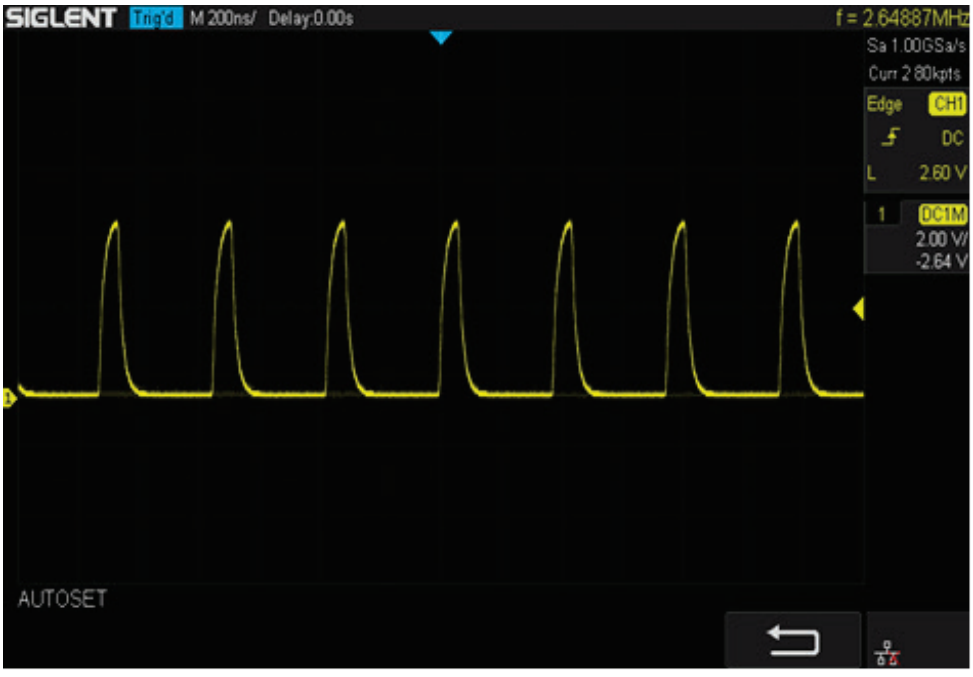

Fig. 5 Port manipulation with delayMicroseconds() function

Fig. 6 Port manipulation without delay

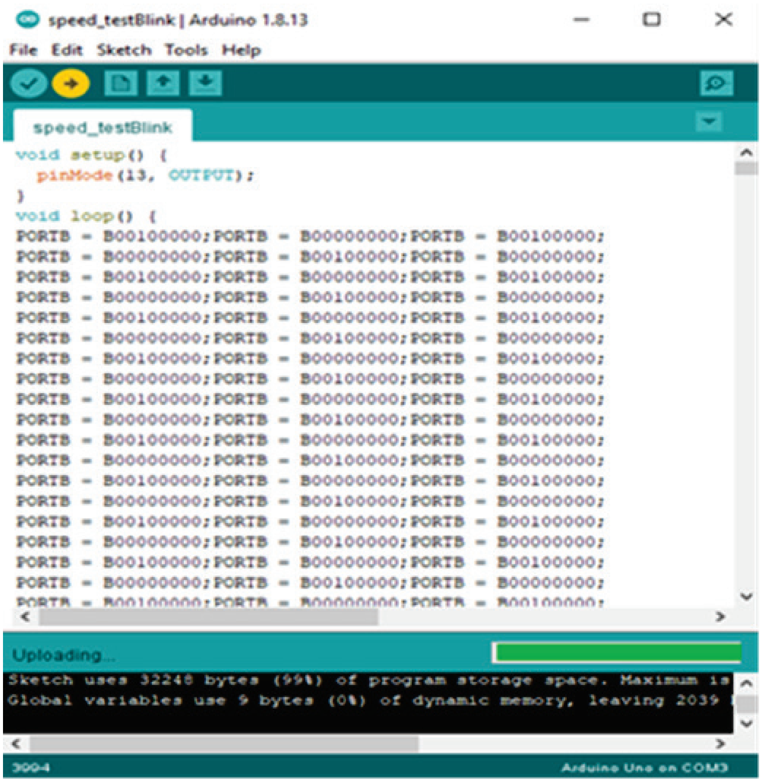

Fig.7 Full memory write with Arduino IDE

Fig. 8 Full memory write results

I also did the same measurements on a Atmel Atmega2560 microcontroller an obtained similar results. The Atmega 2560 is also a 16Mhz rated microcontroller but offers more programing space, larger SRAM memory and many more programmable I/O lines

3. CONCLUSIONS

As shown previously the lowest speed obtained was 495,518 Hz and the fastest speed was 7,94 MHz, while the microcontroller is rated 16 MHz. This means that using the built-in functions of Arduino IDE will result in a very slow sensor readings, due the low frequency that can be achieved. A frequency of 495 Hz can be translated as one sensor reading at each 2 ms period, but this reading, also, need to be processed thus, adding additional delay to the period between each sensor reading.

The rated speed of 16Mhz is basically impossible to obtain since each instruction can last from 1 clock to 4 clocks. The rated speed will actually give us the maximum clock frequency and the maximum speed for sensor readings, impulse sending, or other purposes.

In conclusion if we need maximum frequency for our reads or writes we'll have integrate port and registry manipulation in our code or use assembly language to optimize the code and eliminate any unwanted delay in the program. For those whom speed is not an issue the Arduino Ide is a sufficient and effective platform.

REFERENCES

[1.] Stanimirescu, A., Egri, A, Soica, F.F., Radu, S.M. *Measuring the change of air temperature with 8 LM75A sensors in mining area* (MATEC Web of Conferences, 2020)

[2.] Stanimirescu, A., Egri, A. *The quality of the environment in the city of petrosani in the opinion of citizens.* Annals of the University of Petrosani Mechanical Engineering. 2019,. Vol. 21, pp. 83-87.

[3.] Stanimirescu, A., Radu, S.M. *Measurement of air quality in the Jiu Valley with the help of RNMCA stations.* Annals of the University of Petrosani Mechanical Engineering. 2019,. Vol. 21, pp. 89-94.

[4.] Stanimirescu, A., Radu, S.M., Soica F.F., *Measurements made using dsm501a sensors versus measurements made by national air quality monitoring network.* Annals of the University of Petrosani Mechanical Engineering. 2020,. Vol. 22, pp. 49-54.

[5.] ATmega328P User Guide. Available on https://alldatasheet.com