ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΙΓΑΙΟΥ
UNIVERSITY OF THE
AEGEAN

Department of Information and Communication Systems Engineering (ICSD)

Microcontrollers (ICSD328P)

Project Report Fire Alarm System

Professor :
Asimakis Leros

Authors:

Chrysovalantis Pateiniotis / ICSD18174

31/05, 2023

Contents

List of Figures

1 Introduction:

The following paper serves as a report on the design process and creation of a fire alarm system using knowledge acquired from the laboratories and theory classes throughout the whole semester. The system is supposed to be a very simplistic take on a fire alarm system using basic components in order to achieve a warning-like system so that the operators know if a fire or high temperatures have been detected in an area. Following this in this paper the thinking process as well as the tools and the results of the work that was put into this project will be presented alongside images and sources for any external information found.

2 Methodology:

For the creation of the project, it was necessary to study carefully the needs and the requirements that were placed by the professor and see which was the optimal way of tackling these challenges.

Requirements set by the professor:

- The system must resemble and show some sort of basic functionality with a subroutine.
- The system must make use of at least one timer.
- The system must utilize at least one interrupt function.
- The system must use sensors and read the data provided by said sensors via the analog channels of the Arduino board.
- Ready-made methods and libraries should not be used since the point of the project is to create these functionalities with the knowledge acquired by the course.

The preferable method was to first use an emulator in order to lay down the foundation of the circuit and make sure that a good design and placement of the hardware were achieved. This is a necessary and essential procedure due to the fact that as we will see later on the breadboard circuitry can become quickly very confusing if cables of the wrong sizes are used. After the emulation was completed and the final design choices were made then the collection of all the materials started and the assembly of the system. It is important to note that some initial code was used for the simulator but it was only a placeholder and was later replaced with the actual code that will be presented in section Code.

2.1 Logic of the system:

The system has a very simple logic it is meant to be used on dark server rooms and calculate the temperature of the room as well as some basic light readings. When the system read low temperatures and low light then a LED lights up showing that everything is under control. When the temperature rises slightly above 30 Ceclius then the Second LED lights up and the first turns off. Finally, if very high temperatures are recorded and medium to high light intensity then the last LED turns on while the others are off at the same time a buzzer goes off making a distinct noise in order to notify the people around.

3 Hardware Used:

A short list of all the equipment used can be here:

- Arduino Uno board (ATMEGA328P)

- 1 Blue LED
- 1 Yellow LED
- 1 Red LED
- 1 LM335Z analog temperature sensor
- 1 photoresistor
- 3 1-KOhm resistor
- 1 buzzer
- Jumping wires
- breadboard

4 Design:

The design was made using the already mentioned simulator. It must be mentioned here that in the photographs that will be provided as well as the screenshots and the circuit designs, there is some additional hardware like a slide switch and a LED but these are not used they are leftover equipment and their removal does not affect the functionality of the system at all shall any reader wishes to replicate the system.

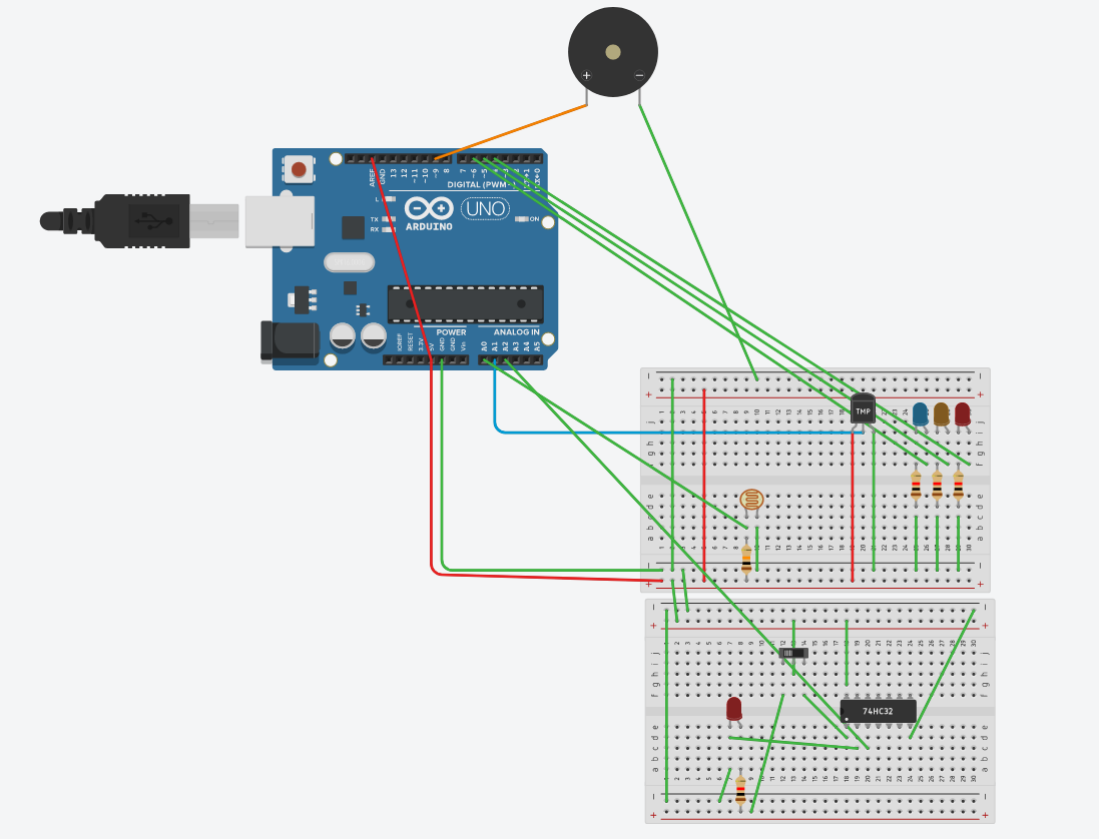

Figure 1: Here we can see the design of the fire alarm when in the simulator. As already mentioned the slide switch and the green LED can be completely removed from the system.

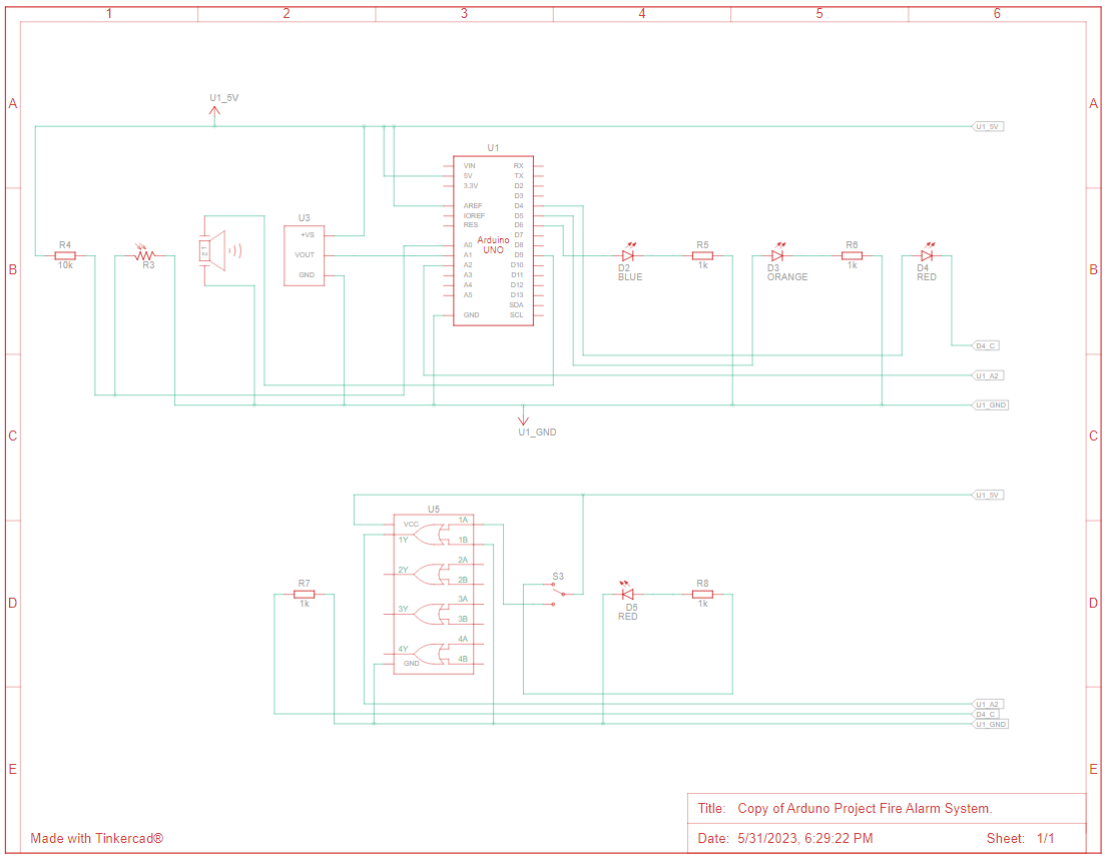

Figure 2: Here is a schematic of the system showing more clearly the connections and the ports used by the system this can help understand the system in depth and it will also be used as a guide in the section of the code where ports will be used for referencing certain functions of the systems.

5 Code:

```
/*
 * main.c
 * Created: 5/30/2023 12:39:42 PM
 *  Author: icsd18174
 */
#include <xc.h>
#include <avr/interrupt.h>
#include <avr/io.h>
float temp;
float vout;
int is_on = 0;
volatile uint16_t adc_result_A0;
volatile uint16_t adc_result_A1;

void setup(){
cli(); // stop interrupts
/* Initialization */
DDRD = 0b11111100; /* Set pins 2--7 as outputs */
DDRB = 0b11111111;
ADMUX = 0b00100000; /* bits 7,6: select AREF as reference */
/* bit 5: left adjust A/D result */
/* bits 3-0: select ADC0 as input */
```

```
ADCSRA = 0b11001111; /* bit 7: A/D enable */
/* bit 6: start conversion */
/* bit 5: auto trigger enable */
/* bit 3: interrupt enable */
/* bits 2-0: prescaler clock/128 */
ADCSRB = 0;

sei(); // allow interrupts
}
void buzzer_ON(){
TCCR1A = 0b01000000; // set entire TCCR1A register to 0
TCCR1B = 0; // same for TCCR1B
TCNT1  = 0; // initialize counter value to 0
// set compare match register for 5000 Hz increments
// turn on CTC mode
// Set CS12, CS11 and CS10 bits for 1 prescaler 64
TCCR1B = 0b00001011;
OCR1A = 500 ; // = 16000000 / (1 * 5000) - 1 (must be <65536)
}
void buzzer_OFF(){
TCCR1A = 0; // set entire TCCR1A register to 0
TCCR1B = 0; // same for TCCR1B
TCNT1  = 0; // initialize counter value to 0
// set compare match register for 5000 Hz increments
OCR1A = 500 ; // = 16000000 / (1 * 5000) - 1 (must be <65536)
// turn on CTC mode
TCCR1B |= (1 << WGM12);
// Set CS12, CS11 and CS10 bits for 1 prescaler 64
TCCR1B |= (0 << CS12) | (0 << CS11) | (0 << CS10);
}
ISR(ADC_vect){
if (ADMUX == 0b00100001){
adc_result_A1 = ADCH;
ADMUX =  0b00100000; // Switch to channel ADC0 (A0) for the next conversion
}
else{
adc_result_A0 = ADCH;
ADMUX = 0b00100001; // Switch to channel ADC1 (A1) for the next conversion
}
ADCSRA =  0b11001111; /* start new conversion */
}
void loop(){
vout = adc_result_A1;
temp = vout * 5000 /1024;
temp = (temp -500) / 10; //lab reading was 158 translating to 27 celsius

if (temp>10 && adc_result_A0 < 100){
PORTD |= (1 << PORTD4);
PORTD &= ~(1 << PORTD5);
PORTD &= ~(1 << PORTD6);
if (is_on == 0){is_on =1;
buzzer_ON();}
}
else if ( temp>30 && temp<=80 ){
PORTD |= (1 << PORTD5);
PORTD &= ~(1 << PORTD6);
PORTD &= ~(1 << PORTD4);
if(is_on == 1){
```

```
is_on = 0;
buzzer_OFF();
}}
else if (temp>10 && temp <=30 && adc_result_A0 > 150){
PORTD |= (1 << PORTD6); //set PD6 to logic 1
PORTD &= ~(1 << PORTD5); //set PD5 to logic 0
PORTD &= ~(1 << PORTD4);
if(is_on == 1){
is_on = 0;
buzzer_OFF();
}}}
int main(void){
setup();
while(1){
    loop();}}
```

5.1 Code Explanation:

While the code is pretty self-explanatory there are some key points that need further explanation since they reveal the logic behind the main functionality of the system.
ADC result variables: These are used for storing the values from the LM335Z temperature sensor and the photoresistor. These two are essential since the whole logic of the system is based upon them from proper operation.
ADMUX: This register is mainly used to select between two channels one that the photoresistor is connected and one that the LM335Z is connected this can be better seen in the interrupt method where the switching between the two channels occurs. To be more specific when the ADMUX is set to 0b00100000 we read the ADC0 (A0) while when it set to 0b00100001 we read the ADC1 (A1).
ADCSRA: We set this register to 0b11001111 since the first two bits (bit 7/6 from right to left) are for enabling the ADC convertor and beginning a new conversion while the fourth (bit 3) is enabling the system to use interrupting function and the last 3 bits (2-0 bits) are used for the prescaler setting the division factor to 128.
ADCSRB: This register is not used however is it initiated at 0 to make sure that nothing interferes with the proper operation of the code.
TCCR1A: This is set to 0b01000000 to just toggle the OC1A on compare match.
TCCR1B: Here we set 0b00001011 when we need the buzzer to turn on by setting the 3rd bit to 1 so that the WGM12 is activated and the rest of the bits that are responsible for the prescaler clock frequency are set to 0-1-1 so that we can get a clkI/O/64 output. When we turn off the buzzer we set the prescaler to 0-0-0 this has the effect of disabling the clock as a source so no wavelength is produced and thus the buzzer has no input.
TCNT1: This helps to initialise the counter to 0.
OCR1A: This sets the frequency that we need for our wavelength.
Analog Value reading: We read the analog values by using interrupts and each time the interrupt is called A0/A1 are read thus giving us a voltage value that we then transform to Celsius with a simple mathematical equation. This is only done for the temperature readings since the photoresistor voltage values are more than enough to help us make a decision on how our if statements will work.

6 Operation Testing:

Link to video of operation safe and fire positions only:
https://www.youtube.com/shorts/opmId7xLtqw

In this section, the operation and the testing environment will be shortly presented alongside images that were taken at the laboratory. **Testing Environment :** In order to showcase the

functionality of the system during testing the parameters for the if cases were changed to reflect room temperature (25 Celcius). The room lights were also turned off so that the light could be controlled more effectively with the flashlight.

Operation:

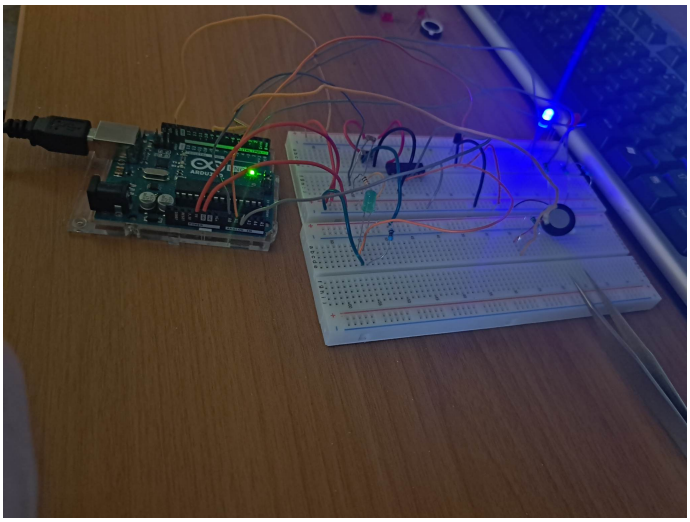

Figure 3: With low light and low temperature, the system indicates that everything is safe indicated by the blue LED.

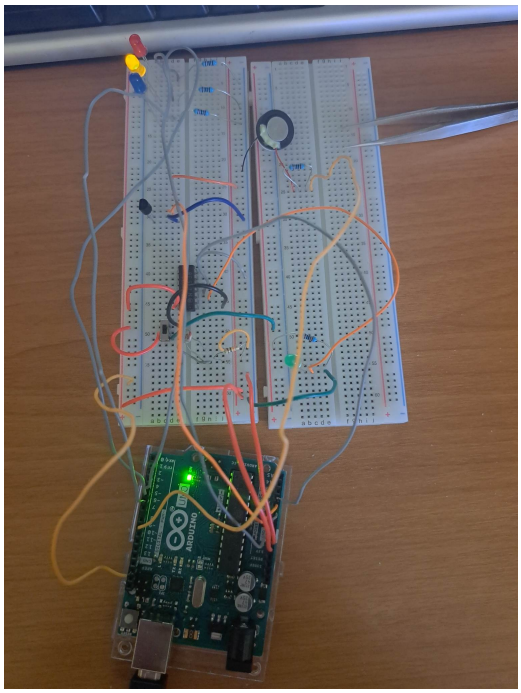

Figure 4: When the temperature rises to dangerous temperatures for computers to operate (80 - 100 Celcius) the yellow LED lights up.

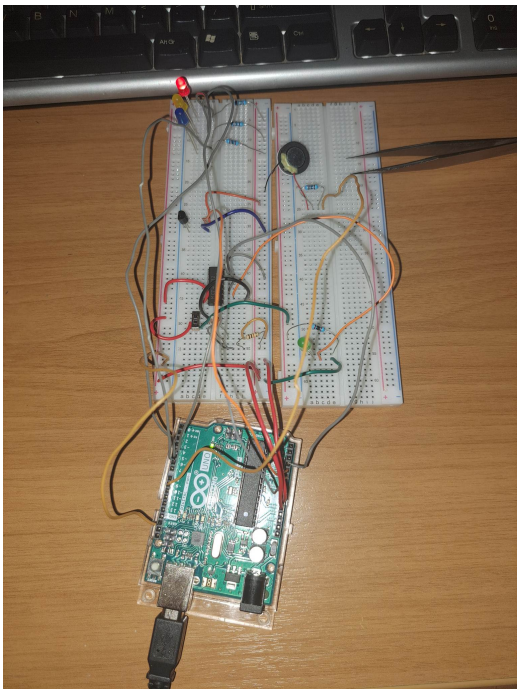

Figure 5: When a lot of light and high temperatures are recorded the red LED light turns on indicating that something is wrong in the room and the buzzer goes off to let anyone nearby know that there might be a fire.