Design with Microprocessors 2021-2022

Heating control system

-

Arduino project

Chereji Iulia-Adela

Group 30434-1

CONTENTS

1.PROJECT SPECIFICATION AND NEEDED COMPONENTS

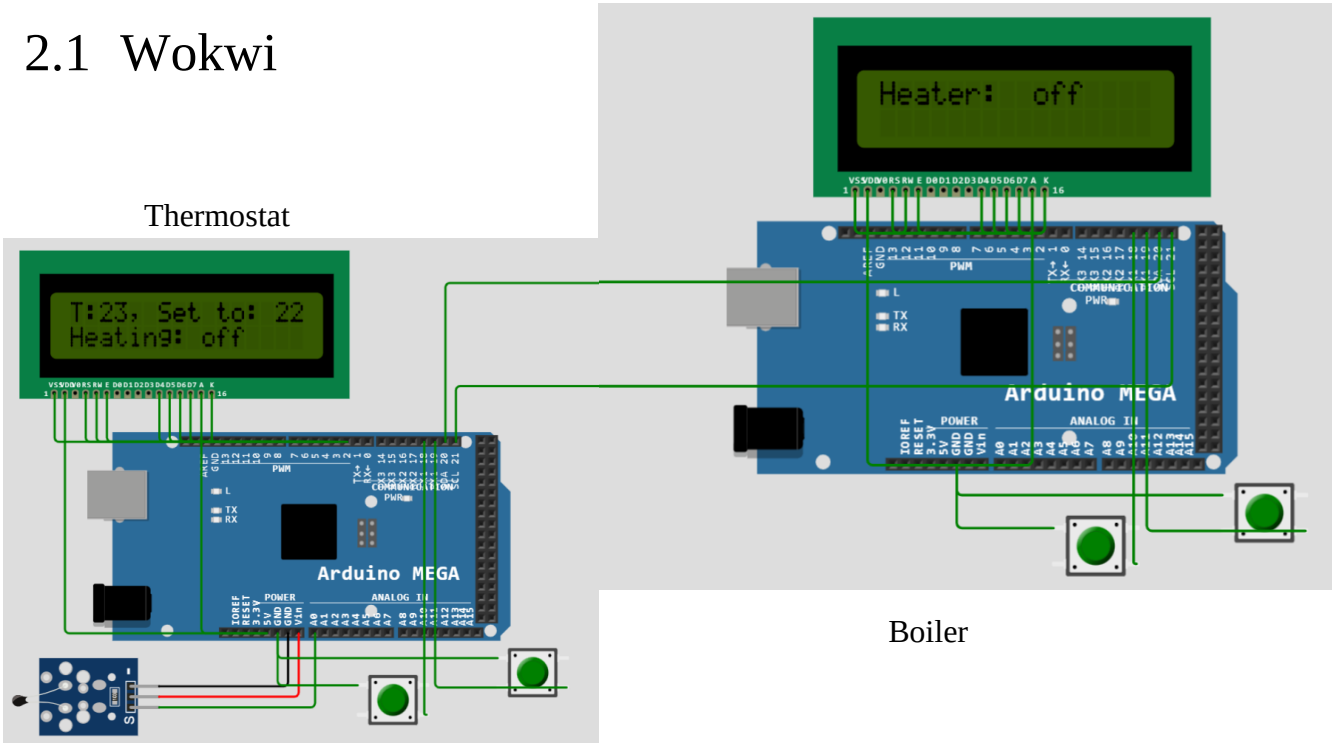
The project simulates the functionality of a home heating system using Arduino. It is composed of two parts: the thermostat and the boiler. The thermostat uses a temperature sensor to detect the current temperature of environment and it compares its value with a set temperature and if the first one is smaller it sends a signal to the boiler to start the heating, otherwise the heating should be stopped. The set temperature has the default value set to 22 degrees Celsius and it can be modified by the user from both the thermostat and the boiler using two buttons (+,-). If the set temperature is modified from the boiler, it sends the thermostat a message with the updated temperature and the same way around. The status of the system is displayed on an LCD display on both thermostat and boiler.

The project contains 2 source files: centrala.ino for the boiler and termostat.ino for the thermostat.
Needed components:

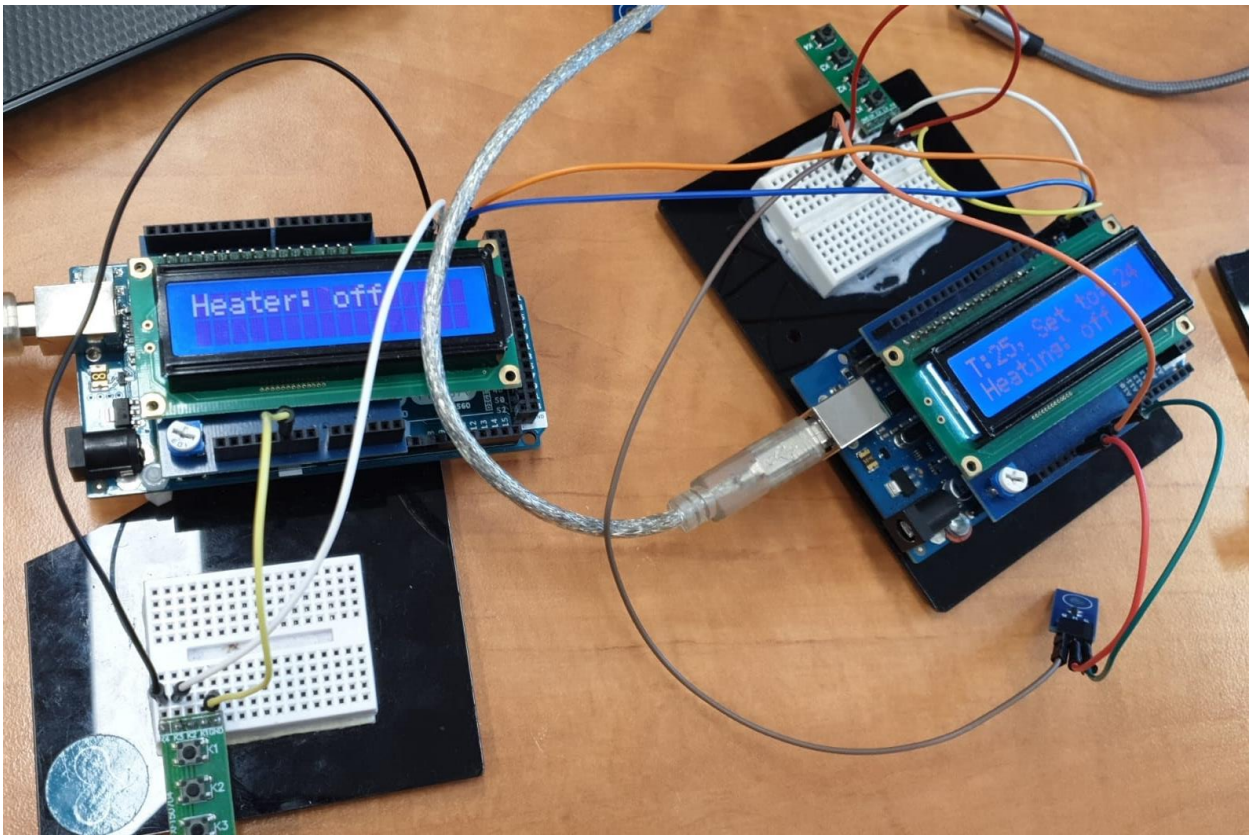
- 2 Arduino Mega 2560 boards
- 2 LCD1602
- A temperature sensor
- 2 sets of 2 buttons
- 2 USB cables
- 12 male-male wires

2.SKETCH

2.1 Wokwi



2.2 Real life



3.IMPLEMENTATION

3.1 Design decisions

- To display the current status of the system I used an LCD display and the library LiquidCrystal.h
- For the communication between the 2 boards I used I2C protocol and I connected the wires between boards on the pins 20 (SDA) and 21 (SCL). The thermostat is the master that gives the clock signal and the boiler is the slave. I used the Wire.h library and the address 9. Both the thermostat and the boiler have a function “void receiveEvent(int bytes)” that processes the message from the other one. The thermostat transmits to the boiler if the heating should be on or off and the boiler transmits to the thermostat if the temperature was set from there (a button was pressed).
- The thermostat also communicates with the computer it is plugged into through the Serial interface. It sends a message when an event happens followed by the time of the event got with the millis() function: the detected temperature changes, the set temperature is modified, heating starts/stops.
- The temperature sensor is connected to the board on pin A0 (analog input), and the reading is done using AVR’s ADC registers.
- The buttons use interrupts and are connected to the pins 18 and 19. Each of them has a function associated that modifies the set temperature when the button is pressed (and debounced): functieButonMinus() and functieButonPlus() .

3.2 Source code

3.2.1 termostat.ino

```
#include <LiquidCrystal.h>

#include <Wire.h>

//master

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

unsigned long time;

volatile int setTemperature;

const float BETA = 3950; // should match the Beta Coefficient of the thermistor

int celsius, previousTemperature;

bool heatingOn=false;

int receivedButtonPressed=0; //0 not pressed, 1 plus, 2 minus

float resolutionADC = .0049 ; // default ADC resolution for the 5V

//reference = 0.049 [V] / unit

float resolutionSensor = .01 ;
```

```
void setup() {
  Wire.begin(9);
  Wire.onReceive(receiveEvent);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("T:  ,");
  lcd.setCursor(6, 0);
  lcd.print("Set to:");
  lcd.setCursor(0, 1);
  lcd.print("Heating:");
  Serial.begin(9600);
  setTemperature=22;

  pinMode(18, INPUT);
  digitalWrite(18, HIGH);
  attachInterrupt(digitalPinToInterrupt(18), functieButonPlus, CHANGE);
  pinMode(19, INPUT);
  digitalWrite(19, HIGH);
  attachInterrupt(digitalPinToInterrupt(19), functieButonMinus, CHANGE);
  previousTemperature=-100;
  time=millis();
  Serial.print("Temperature set to: ");
  Serial.print(setTemperature);
  Serial.print(" at time ");
  Serial.println(time);

  // set the ADC clock to 16MHz/128 = 125kHz
 ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));
 ADMUX |= (1<<REFS0); //Set the ref. voltage to Vcc (5v)
 ADCSRA |= (1<<ADEN); //Activate the ADC
 ADCSRA |= (1<<ADSC);
```

```
}

void loop() {
  time = millis();
  lcd.setCursor(14, 0);
  lcd.print(setTemperature);
  //int analogValue = analogRead(A0);
  //celsius = 1 / (log(1 / (1023. / analogValue - 1)) / BETA + 1.0 / 298.15) - 273.15;
  celsius = read_adc(0);

  lcd.setCursor(2, 0);
  lcd.print(celsius);

  delay(1000);
  lcd.setCursor(9, 1);
  if(celsius < setTemperature){
    lcd.print("on ");
    Wire.beginTransmission(9); // transmit to device #9
    Wire.write(1); //heating on
    Wire.endTransmission(); // stop transmission
    if(heatingOn==false){
      heatingOn=true;
      Serial.print("Heating turned on at time ");
      Serial.println(time);
    }
  }
  else{
    lcd.print("off");
    Wire.beginTransmission(9); // transmit to device #9
    Wire.write(0);
```

```
    Wire.endTransmission(); // stop transmission
    if(heatingOn==true){
      heatingOn=false;
      Serial.print("Heating turned off at time ");
      Serial.println(time);
    }
  }

  if(celsius != previousTemperature){
    Serial.print("Current temperature: ");
    Serial.println(celsius);
    previousTemperature=celsius;
  }
}

void functieButonPlus()
{
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  // If interrupts come faster than 200ms, assume it's a bounce and ignore
  if (interrupt_time - last_interrupt_time > 200)
  {
    setTemperature++;
    Serial.print("Temperature set to: ");
    Serial.print(setTemperature);
    Serial.print(" at time ");
    Serial.println(time);
  }
  last_interrupt_time = interrupt_time;
}
```

```
void functieButonMinus()
{
 static unsigned long last_interrupt_time = 0;
 unsigned long interrupt_time = millis();
 // If interrupts come faster than 200ms, assume it's a bounce and ignore
 if (interrupt_time - last_interrupt_time > 200)
 {
  setTemperature--;
  Serial.print("Temperature set to: ");
  Serial.print(setTemperature);
  Serial.print(" at time ");
  Serial.println(time);
 }
 last_interrupt_time = interrupt_time;
}

uint16_t read_adc(uint8_t channel)
{
 ADMUX &= 0xE0; // delete MUX0-4 bits
 ADMUX |= channel&0x07; //Sets in MUX0-2 the value of the
 //new channel to be read
 ADCSRB = channel&(1<<3); // Set MUX5 value
 ADCSRA |= (1<<ADSC); // start conversion
 while(ADCSRA & (1<<ADSC)); //Wait for the conversion to
 //finish
 int reading=ADCW;
 float voltage = reading * resolutionADC;
 float tempCelsius = (voltage - 0.5) / resolutionSensor ;
 return tempCelsius;
 //return ADCW;
}
```

```
void receiveEvent(int bytes) {
 receivedButtonPressed = Wire.read(); //read I2C received character
 if(receivedButtonPressed==1) setTemperature++;
 else if(receivedButtonPressed==2) setTemperature--;
}
```

3.2.2 centrala.ino

```
#include <LiquidCrystal.h>
//slave
// include I2C library
#include <Wire.h>
int heatingOn = 0;

volatile int setTemperature=0, previousSetTemperature=0;
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
void setup() {
 // Start i2C slave at address 9
 Wire.begin(9);
 // attach a function to be called when we receive
 //something on the I2C bus
 Wire.onReceive(receiveEvent);
 lcd.begin(16,2);
 lcd.print("Heater:");


  pinMode(18, INPUT);
  digitalWrite(18, HIGH);
  attachInterrupt(digitalPinToInterrupt(18), functieButonPlus, CHANGE);
  pinMode(19, INPUT);
  digitalWrite(19, HIGH);
```

```
  attachInterrupt(digitalPinToInterrupt(19), functieButonMinus, CHANGE);

}
void receiveEvent(int bytes) {
 heatingOn = Wire.read(); //read I2C received character
}
void loop() {
 lcd.setCursor(8, 0); // display received character
 if(heatingOn==0) lcd.print("off");
 else lcd.print("on ");
 if(setTemperature>previousSetTemperature){
  Wire.beginTransmission(9); // transmit to device #9
    Wire.write(1); //buton plus
    Wire.endTransmission(); // stop transmission
    previousSetTemperature=setTemperature;
 }
 else{
  if(setTemperature<previousSetTemperature){
  Wire.beginTransmission(9); // transmit to device #9
    Wire.write(2); //buton minus
    Wire.endTransmission(); // stop transmission
    previousSetTemperature=setTemperature;
 }
 else {
  Wire.beginTransmission(9); // transmit to device #9
    Wire.write(0); //not pressed
    Wire.endTransmission(); // stop transmission
 }
 }
}
void functieButonPlus()
```

```
{
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  // If interrupts come faster than 200ms, assume it's a bounce and ignore
  if (interrupt_time - last_interrupt_time > 200)
  {
    setTemperature++;
  }
  last_interrupt_time = interrupt_time;
}

void functieButonMinus()
{
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  // If interrupts come faster than 200ms, assume it's a bounce and ignore
  if (interrupt_time - last_interrupt_time > 200)
  {
    setTemperature--;
  }
  last_interrupt_time = interrupt_time;
}
```

4.BIBLIOGRAPHY

https://wokwi.com/

https://docs.wokwi.com/parts/wokwi-lcd1602

https://create.arduino.cc/projecthub/Hack-star-Arduino/how-to-use-wokwi-arduino-simulator-in-2022-what-is-wokwi-304e6b

https://forum.arduino.cc/t/use-i2c-for-communication-between-arduinos/653958/3

https://forum.arduino.cc/t/debouncing-an-interrupt-trigger/45110/2

https://arduino.stackexchange.com/questions/16413/how-to-change-variable-when-a-button-connected-to-pin-2-is-pressed

https://forum.arduino.cc/t/using-button-to-change-variable-value-on-the-go/179323

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce