FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Electric motor cooling system
Student:	Lev Paramonov
Supervisor:	Ing. Filip Bazakas
Study program:	Informatics
Branch / specialization:	Computer engineering
Department:	Department of Digital Design
Validity:	until the end of summer semester 2022/2023

Instructions

Create a cooling system for an electric motor with a temperature of the cooling liquid used in diesel motors (85-95 degrees Celsius).
Components to use:
1. Processor ATmega328P
2. Heating unit (boiler)
3. Pump
4. Switches for heating unit and pump
5. Display with current temperature and state of the pump on it
6. Temperature sensor
7. If needed other sensors (will come out during implementation)

This system will be connected into a test environment, which is simulating an electro vehicle: electromotor, drives, computer. A good addition will be to analyze behavior of electromotor in a real vehicle.
During the implementation and testing phases, the system will be connected to a PC, at the end, it should work separately as a standalone system. The purpose of the system is to operate the heating unit and the pump, to detect current temperature. During the implementation phase, additional functions can be added, for example, detection of liquid leakage.

Electronically approved by prof. Ing. Hana Kubátová, CSc. on 8 October 2021 in Prague.

Bachelor's thesis

ELECTRIC MOTOR COOLING SYSTEM

Lev Paramonov

Faculty of Information Technology
Department of Digital Design FIT
Supervisor: Ing. Filip Bazakas
May 4, 2022

Contents

List of Figures

List of Tables

List of code listings

I would like to express my sincere gratitude to my parents for the support they have given on my learning path. For their wise advices, mental and financial support, shown care. Without them, it wouldn't be possible for me to get this far.
I would like to thank EATON company, my manager Ing. Petr Novak and lead engineer Kumar Piyush, PhD. for giving me an opportunity to join the company as an intern and creating this thesis topic for me. Also, I would like to express my gratitude to my supervisor, Ing. Filip Bazakos, for accepting my thesis work for supervision and his guidance for this work.

Declaration

Abstract

This work is focused on building a motor cooling system, that is meant to be used in simulating processes. The working temperature range, 85 to 95 degrees Celsius, is picked to compensate the fact, that motors in a testing environment will not produce the same amount of heat as in real electrical vehicles, so we need to heat it up with the coolant, for proper working temperatures. Existing methods of liquid cooling and solutions of real modern electric vehicles are described, followed by properties of different cooling liquids. After that, based on gained knowledge, needed components are picked with explanation for our liquid cooling system. An implementation process is divided into two phases, HW and SW. A solution is based on programming a microcontroller Arduino Uno and peripherals connected to it, such as sensor amplifier, relays, display. The chosen coolant, water, is flowing in a closed loop, which contains reservoir, pump, motor. To make sure everything works properly, the system was tested with different temperatures of the coolant, first without and then with a motor. In the conclusion, we discuss how the system can be upgraded and modified, what it can be used for and which goal of the thesis were completed.

Keywords electric motor cooling, liquid cooling, system building, simulation of EV, Eaton Elektrotechnika, s.r.o.

Abstrakt

Tato práce je zaměřena na sestavení chladicího systému motoru, který má být použit při simulaci elektrického vozidla. Rozsah pracovních teplot 85 až 95 stupňů Celsia je zvolen tak, aby kompenzoval skutečnost, že motory v testovacím prostředí neprodukují stejné množství tepla jako ve skutečných elektrických vozidlech, takže je musíme ohřívat chladicí kapalinou, pro správné pracovní teploty. Jsou popsány stávající způsoby kapalinového chlazení a řešení skutečných moderních elektromobilů, následně vlastnosti různých chladicích kapalin. Poté jsou na základě získaných znalostí vybrány potřebné komponenty s vysvětlením pro náš systém kapalinového chlazení. Proces impelemtace je rozdělen do dvou fází, HW a SW. Řešení je založeno na naprogramování mikrokontroléru Arduino Uno a periferií k němu připojených, jako je zesilovač snímače, relé, displej. Zvolená chladicí kapalina, voda, proudí v uzavřené smyčce, která obsahuje nádrž, čerpadlo, motor. Aby vše fungovalo správně, byl systém testován s různými teplotami chladicí kapaliny nejprve bez motoru a poté s motorem. V závěru pojednáváme o tom, jak lze systém upgradovat a upravit, k čemu jej lze využít a jaké cíle práce byly splněny.

Klíčová slova chlazení elektro motoru, kapalinové chlazení, stavba systému, simulace elektrického vozidla, Eaton Elektrotechnika, s.r.o.

Chapter 1
Introduction

This thesis is based on one of long existing industrial problem, electric motor cooling. The first battery powered electric motor was constructed by an American blacksmith Thomas Davenport and his wife Emily Davenport in 1834 [1] and the first Electric Vehicle (EV) appeared at the end of XIX century, around 1881 it was tested in Paris by Gustave Trouvé [2].

Ever since then, engineers and scientists around the world came up with different solutions of removing waste heat, byproduct heat that is produced by a motor during work. But at the beginning of the XXI popularity of EVs raised due to: global climate problems, search of alternative fuels for vehicles. And so motor cooling problem inside these vehicles gain popularity too, the majority of car manufacturers nowadays have at least one EV in their catalog, and are making researches in this field.

As mentioned above, popularity of EVs is raising day by day and engineering questions around them are getting more attention, motor cooling is one of them. But it is not reasonable trying to find new solutions by testing new ideas right on the vehicle, it is not safe, will cost more and is less comfortable for adjusting. For these reasons there are laboratory versions of cooling systems, where different motors, pumps, heating and cooling elements and other components can be tested. This thesis is about creating one of that laboratory systems, so it can be used for testing and simulating purposes in a process of invention of a new motor or even a whole new EV. Even though we are creating a cooling system, the temperature range of 85 to 95 degrees Celsius was picked, because motors in the test environment will not heat up the same way as in EVs, so we will heat it up with our coolant, to have more realistic working temperatures.

Currently, there are three main methods of electric motor cooling: conductive, air, liquid. This thesis is focused on creating a system using the last method. First will come theoretical analysis of the method, description of different coolants, liquid that is used in a cooling system, also methods of transportation of those liquids in and from a motor will be mentioned. Then we will take a look at existing solutions from modern EVs. After getting theoretical knowledge, we will move to choosing right components for our own system.

Implementation process will be divided into two steps, SW creation and HW management. One after another we will connect components to our system, adding needed libraries, making them work together. When the program is finished and tested, we will get into HW placement, starting from sensors and heating element and finishing with making sure that everything is sealed properly, so water will not evaporate.

At the end of this thesis we will discuss results of theoretical and implementation processes, evaluate created system, point out successes, fails and reasons for them.

Chapter 2

Goals of the thesis

Theoretical and analytic goal of this thesis is to describe the behavior of cooling systems of electric motors in real modern cars. Look at what methods are used, how the coolant is delivered and cooled/heated. Finding solutions, that exists in real “in production” cars, is important for realization of practical part of the work, since our system will be used to simulate an EV.

Before realization of practical part will come theoretical designing, particularly choosing right components. For example, choosing between using a boiler or taking some industrial dish and heating element and creating a custom boiler. Picking right sensor. Find out which display is more suitable for purposes of the thesis.

For the physical part of implementation, placing of each component should be described and reasoned. Choose correct tubes and connections for proper water cycling in the system. Software implementation will require finding suitable libraries for each component and creating the right functions.

Chapter 3

Theory and Analysis

3.1 Liquid motor cooling

A modern electric motor has a limited power output based on the ability to feed an electric current to the stator and rotor. Increasing the current we are sending to the motor, we are increasing its power output, but with higher power comes a large amount of heat. This heat, produced by various electrical and mechanical processes, can decrease the lifetime of the motor. Therefore, any electric motor needs a proper cooling for its better performance and also for the safe and reliable operation. The question of thermal management of the motor has gained popularity over the years, as more and more powerful electric motors were produced, and remains a major challenge for engineers in that field. [3]

Motor cooling methods are divided into two groups:

- passive
  - conductive
- active
  - forced air
  - liquid

Passive method means, that it doesn't use any active mechanism like fan or pump. These methods rely on natural heat conduction to transfer waste heat from the motor. An advanced example of passive method is a fin pin method, it works on principle of increasing conductive surface. Active methods are more effective, but also more expensive, noisy and need power for active mechanisms. There are many variations of forced air methods, but all of them working on the idea, that a fan is blowing an air over or inside the motor, removing the waste heat. Active air methods are mostly used for small and medium-sized motors.[3]

In this thesis, we are interested in liquid methods, as we need to build a system feeding a coolant into the motor. Below, we will take a look at some modern liquid techniques, so we can understand how it works inside a motor. Later we will use that knowledge in choosing components process.

3.1.1 State of the art

As we mentioned above, liquid techniques are more costly than others and for that reason they are mostly used for larger high-power motors. But due to higher heat transfer and being able to remove thermal restrictions on the design of motors, liquid cooling is gaining popularity in other applications. Let's now take a look at these techniques. [3]

Liquid cooling methods are divided into two groups:

- indirect
  - cold plates
  - copper pipes
  - cooling channels
  - ...
- direct
  - cooling of bundled magnet wires
  - spray cooling
  - liquid immersion
  - liquid jet impingement

A technique being indirect or direct assumes that the coolant is in direct or indirect contact with the heated parts of the motor. Indirect cooling methods are using heat exchangers, components with coolant inside. Without direct contact with cooled parts of the motor, there should be a good path for heat to reach the heat exchangers. An example of such an exchanger is a cold plate, which are made of metal tubes bonded into, typically aluminum. These plates are placed on the surface of the motor to provide good heat exchange. Other indirect techniques are working on the same principle, the difference is in the way the heat exchanger is placed and constructed. [3]

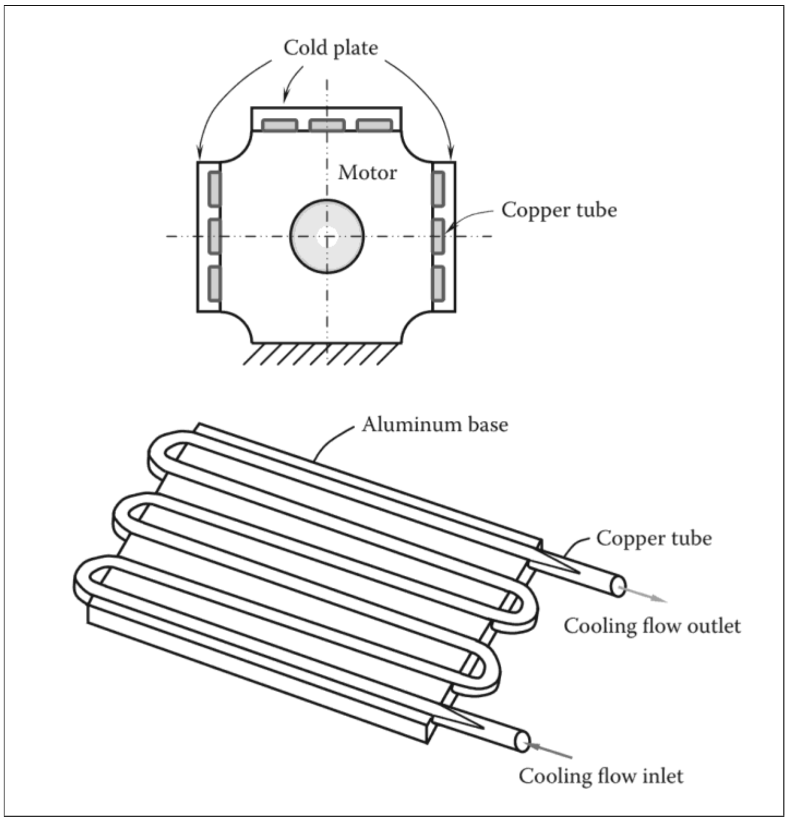


Figure 3.1 Cold plates used for motor cooling, mounted on the three sides of the motor [3]

Direct cooling methods have higher cooling efficiency in comparison with air and indirect cooling, because coolant is in contact with heat producing components of the motor. Both direct and indirect techniques use closed-loop systems for a coolant to reduce cost of the system and consumption of coolant. The difference is that we can't use non-dielectric coolants in direct methods to prevent an electrical short. Let's take a look at some examples of direct cooling. [3]

Jet impingement cooling is the method where a coolant is delivered into the motor through an array of nozzles. Dielectric liquid is in direct contact with heat producing elements, reducing the heat and stabilizing the temperature. The next example is a spray method, when the coolant is applied onto the motor through nozzles. Cooling liquid is then going through the motor and evaporate or drops intro the coolant tank and then is pumped into nozzles again. All this direct methods using the ability of dielectric liquids to not create an electrical short within the motor and applies those liquids directly on heat producing elements, which is making these techniques the most effective ones. A method that does it at the maximum is an immersion cooling, where the hole structure of the cooled subject is dipped into the coolant. [3]

3.1.2 Properties of different coolants

For any liquid cooling system, we need a coolant. There is a big variety of them on the market, their difference is in working temperatures, conductivity and other thermal parameters. Wei Tong has a good selection of different coolants with their properties in his book, which we can see on Table 3.1. Coolants are being picked by looking on working temperatures of the system and comparing them to boiling and freezing temperature points of coolant. Also, convection ability of the liquid is one of the most important parameters, since it determines how much heat can be transferred with the minimum of liquid flow. [3]

The system we are building in this thesis intended to be used for indirect techniques in room temperatures and a coolant being heated up to a maximum of 95 degrees Celsius. Therefore, we can and will use water as our coolant, since it's the cheapest and most accessible option for us, and it meets all temperature thresholds. This coolant pick will allow us to use simpler solutions for the pump, since water doesn't need any special equipment for that, but on the other hand it will block us an opportunity to use any other coolants, because it can damage common water and put it out of work.

3.2 Existing solutions

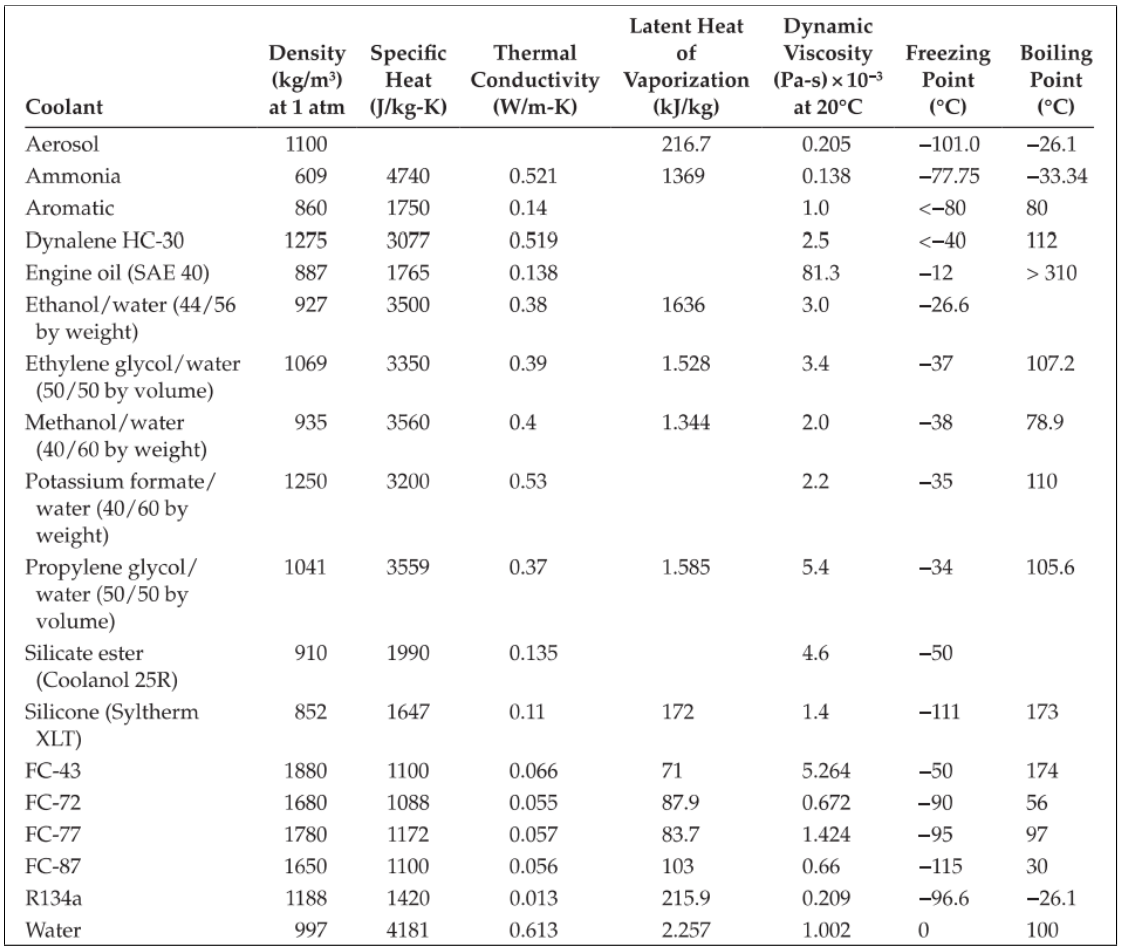
In this section, we will take a look at existing solutions through market offers, patents of car manufacturers and articles about cooling systems in cars. We will discuss how cooling systems of the batteries and motors are working along each other in EVs. For laboratory examples, we will find some pre-build systems on the market and industry solutions. We are mostly interested in closed loop liquid cooling systems, since we are trying to gain knowledge for the structure we are building.

3.2.1 EV systems examples

Three different vehicle models of modern car manufacturers were picked for EVs systems examples, particularly: Tesla Model 3, Audit e-Tron, Formula E cars. All of them are high-end modern EVs, that have its own approach to the motor cooling, that we will discuss in the following.

The first solution we will take a look at will be from Tesla with their Tesla Model 3, a company that made EVs popular all over the world. Their approach to cooling in this model is explained in a patent called "Electric motor waste heat mode to heat battery". As it's mentioned in the title of that patent, a waste produced by the motor is used to heat up the battery packs. There are two loops in the car, connected to a four-way valve, that is placed on the reservoir of coolant.

Table 3.1 Table with thermophysical properties of different coolants compared to water [3]



Coolant	Density (kg/m³) at 1 atm	Specific Heat (J/kg-K)	Thermal Conductivity (W/m-K)	Latent Heat of Vaporization (kJ/kg)	Dynamic Viscosity (Pa-s) × 10⁻³ at 20°C	Freezing Point (°C)	Boiling Point (°C)
Aerosol	1100			216.7	0.205	−101.0	−26.1
Ammonia	609	4740	0.521	1369	0.138	−77.75	−33.34
Aromatic	860	1750	0.14		1.0	<−80	80
Dynalene HC-30	1275	3077	0.519		2.5	<−40	112
Engine oil (SAE 40)	887	1765	0.138		81.3	−12	> 310
Ethanol/water (44/56 by weight)	927	3500	0.38	1636	3.0	−26.6	
Ethylene glycol/water (50/50 by volume)	1069	3350	0.39	1.528	3.4	−37	107.2
Methanol/water (40/60 by weight)	935	3560	0.4	1.344	2.0	−38	78.9
Potassium formate/ water (40/60 by weight)	1250	3200	0.53		2.2	−35	110
Propylene glycol/ water (50/50 by volume)	1041	3559	0.37	1.585	5.4	−34	105.6
Silicate ester (Coolanol 25R)	910	1990	0.135		4.6	−50	
Silicone (Syltherm XLT)	852	1647	0.11	172	1.4	−111	173
FC-43	1880	1100	0.066	71	5.264	−50	174
FC-72	1680	1088	0.055	87.9	0.672	−90	56
FC-77	1780	1172	0.057	83.7	1.424	−95	97
FC-87	1650	1100	0.056	103	0.66	−115	30
R134a	1188	1420	0.013	215.9	0.209	−96.6	−26.1
Water	997	4181	0.613	2.257	1.002	0	100

The first loop is for power electronics and the second one for the batteries. The power electronics loop is connected to the motor heat exchanger, collecting all wasted heat from it and moving it through pipes near the batteries to radiators, so that wasted heat is heating up the batteries. When it comes to the motor cooling, the drive motor fluid pump fills the hollow cylindrical body of the motor, which forces that liquid to be sprayed on the stator and then moved to the heat exchanger. So they are using direct spray method, and then with another coolant transport heat from heat exchanger. [4, 5]

Audi picked a bit more complex approach for their e-Tron, with a combination of convection and water cooling. They are cooling the motor internally and externally. For external part, a cooling jacket on the stator is used, the coolant if flowing through power electronics then to that jacket and in the end inside the rotor. With this method, elements of the motor are connected in series and the rotor at the end of that series has higher temperature than the stator. That's why additional internal cooling is needed, for that reason fan blades are placed on the rotor, cooling it by forced air. Here we can see that Audi is using indirect jacket method and forced air method combined. [6, 7]

The last solution we will discuss will be from one of the most powerful EVs in the world at the moment, Formula E cars. These vehicles are pushed to its limits in every way possible, so when it comes to working temperatures it can't be overheated or over cooled, because it will lead into performance loose or even complete shut down. Cooling method in Formula E cars uses the fact, that they are moving at very high speed, so there is a lot of air being pushed into the car. This air is being collected by the radiators on both sides of the car. One of them is cooling the coolant for motor and other power electronics of the vehicle and the second one is for battery cooling system. The most common coolant for these cars is glycol. [8, 9] Unfortunately, it's near to impossible to find information about how exactly the motor is cooled in the car, because every team is keeping it as a secret.

All of these solutions are using closed-loop liquid cooling solutions. No matter what methods are combined, there is always some reservoir for a coolant, a pump and heating/cooling unit. That is the information we were looking for, to build our own simulating system.

3.3 Components choosing

In the first two sections of that chapter, we gained theoretical knowledge about a structure we are trying to build, now we can get into the process of creation of our own electric motor liquid cooling system. And before starting the implementation process, we need to pick the right components. As the first step it's needed to think about how our system will look, so we should create a simplified scheme, that will show us what we have to acquire and how everything will be connected.

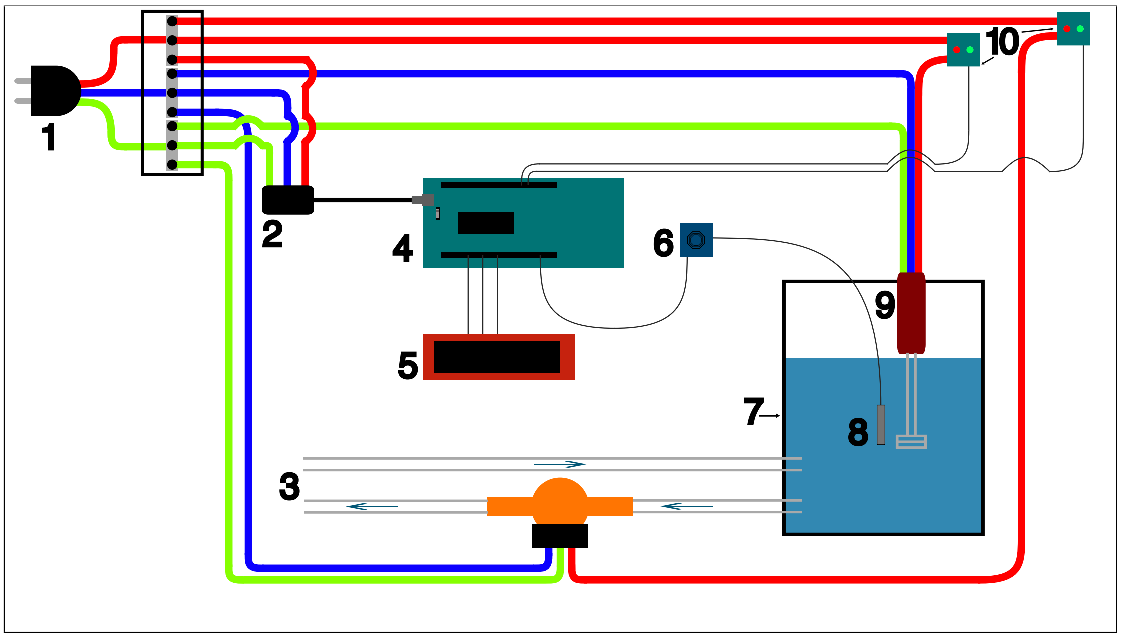


Figure 3.2 Simplified scheme of the system

1: Plug; **2**: 230V AC to 12V DC converter; **3**: Water tubes end for motor connection;
4: Microcontroller; **5**: Display; **6**: Amplifier for sensor; **7**: Water tank; **8**: Temperature sensor;
9: Heating element; **10**: Relay;
Wires: red-phase, blue-neutral, green-ground, thin black-data cables, thick black-power cable for microcontroller

There are water tubes and connectors for them on the scheme, the description of connection process and material picking for them will be in Chapter 4: "Implementation, Water tubes and their connections, pump". In sections below we will talk about electric components, that were picked for the system.

3.3.1 Microcontroller

The microcontroller is the center of our system, it will control the operation of all other components, when we power everything up. Some components like relays, display, amplifier will be connected directly to it, so we need to think about wiring and place saving when thinking about how to use our microcontroller.

In our task, we were requested to use ATmega328P, which is a major component in the Arduino board products. So we can take advantage of it and get ourselves one of those boards, particularly "Arduino Uno rev3", which has 28-pin version of ATmega328P [10]. Using this board will give us better access to all features of the ATmega328P controller, simplify wiring and power supplying. Also, Arduino has its own IDE with serial link, which we need for testing purposes, and give us an option to comfortably add libraries made for components we are using, so we will not have to use any other third-party software for implementation.

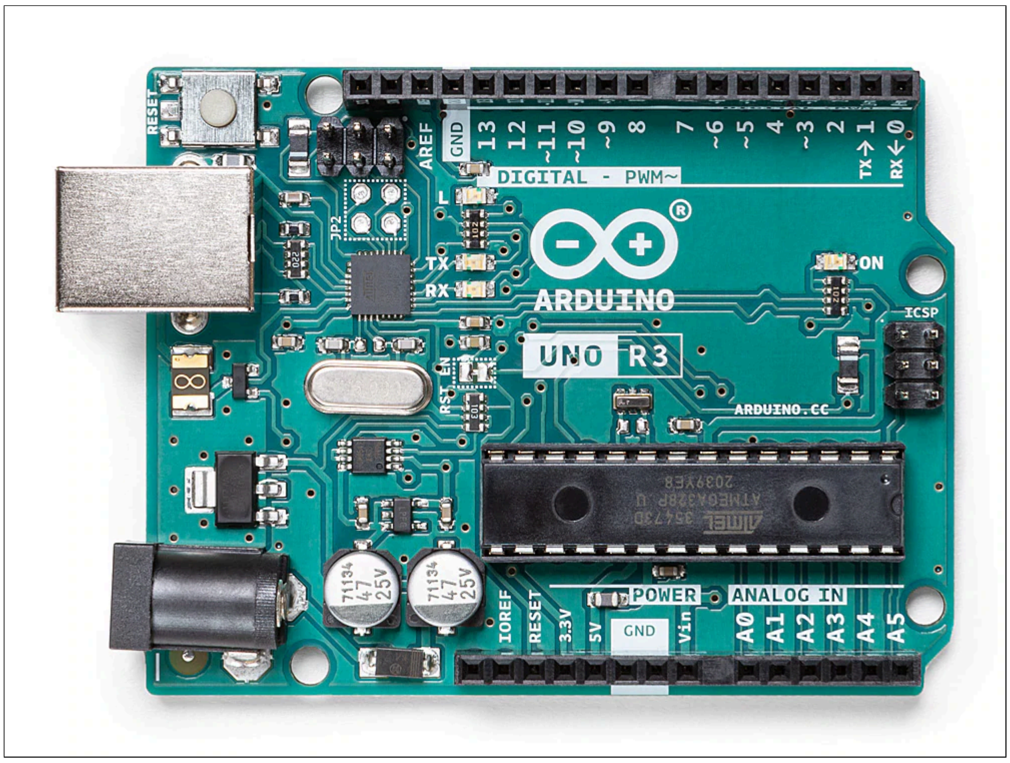


Figure 3.3 Arduino UNO R3 board [10]

3.3.2 Boiler

Since we are creating **water** cooling system, we need a tank to store our coolant. Then we need to heat the liquid to at least 85 degrees Celsius, so there should be a heating element in the water. To control that element, we need to read the temperature of the water, so we know when to turn it off or on. It means, that we need a temperature sensor.

These three components can be found in a usual boiler for your home needs. These boilers are protected from leaking, have its own automated system and some of them even can show the temperature of the water inside. So, it looks logical to use one of these ready to go commercial boilers, but aren't there some hidden problems? Yes, there are, and we will talk about them.

The first problem that goes with store-bought boilers surprisingly comes from their advantage, they have their own regulated system. For our task we need to implement an automated system using ATmega328P microcontroller, so to be sure about what's going on in our system we need to have full access to each component we are operating. We could disassemble the boiler's insides and connect to them, but it wouldn't be safe and create problems during testing phase.

The second problem comes from physical parameters of these boilers. For our task we need from 5 to 10 liters of water, depending on the size of connected motor, and modern market can't offer a lot of solutions with such volumes. Also, low volume boilers can't heat the water to the temperatures we need, because of their small size they have to use weaker heating elements. So if we get one of them and use, we will break restrictions made by manufacturer. Next problem is that there is no easy access to the inside space of store-bought boilers. So, if we would need to add some sensors for example, we wouldn't be able to so.

Because of all these problems, we will make our own custom solution. For that we need to acquire a 10l water dish, a heating unit and a temperature sensor. In the next three subsections, we will pick components for our handmade boiler.

3.3.3 Sensor

The sensor's precision is the key quality we are looking for in it, since the data we are getting from it will make the heating unit turn on and off. Therefore, if we don't want to overheat our motor, we should properly pick and then test the sensor. For our sensor, we are looking for a working temperature range between 0 and 150 degrees Celsius, as we will use our system indoor and heat our water to a maximum of 95 degrees Celsius. Currently, there are three main types of temperature sensors widely used in industry: resistance temperature detector (RTD), thermistors and thermocouples [11], we will look at each of them and find which suites us the best.

The first type we will talk about is a thermistor, using semiconductor material whose resistance changes with temperature. Their resistance increases as the temperature goes up, meaning they have negative temperature coefficient. Their operating range is limited to -50 to 300 degrees Celsius. [11, 12]

The next candidate is RTD sensor, which is a temperature measuring device working on the principle, that all metals produce a known amount of resistance under the change of the temperature. Most common metals used are platinum, copper, nickel. For our project we will consider platinum RTD sensors, which are also marked as PTX, where X is the resistance at 0 degrees Celsius (for example PT100 if it produces 100 Ohms under 0 degrees Celsius). Platinum RTDs have an operating range of -200 to 500 degrees Celsius, and their resistance increases linearly. To operate, these devices need voltage or current source, which are fed to the sensor through one of the wires. There are three solutions on the market: 2-wire, 3-wire and 4-wire. [13, 14, 15]

Two-wire solution is the simplest, to get a measured value we need to add up all the resistances values in the circuit, $R_1 + R_2 + R_3 = R_t$. This method is depending on current source accuracy and should be recalibrated each time the system gets changed. Also, this solution has the biggest measurement error, so it is not used in industrial projects. Two other methods are removing this error by using additional wires connected to sensor terminals. [12, 16]

Three-wire method has two wires connected to one terminal and one wire to another terminal. For this solution, we need 2 constant current sources. This method measures the resistance between $R_1 \& R_2$ and subtracting $R_2 \& R_3$. At the end we have $(R_1 + R_2 + R_b) - (R_2 + R_3) = R_b$, which leave us only with R_b resistance, removing measurement error. Restriction for this method is that all wire should have same resistance and length. [12, 16]

The four-wire solution is the most accurate one, it is using true bridge solution removing all unwanted resistance and also unlike three-wire solution does not rely on length of wire and their resistance equality. It has a small advantage in measuring precision for a much higher production cost. [11, 12, 16]

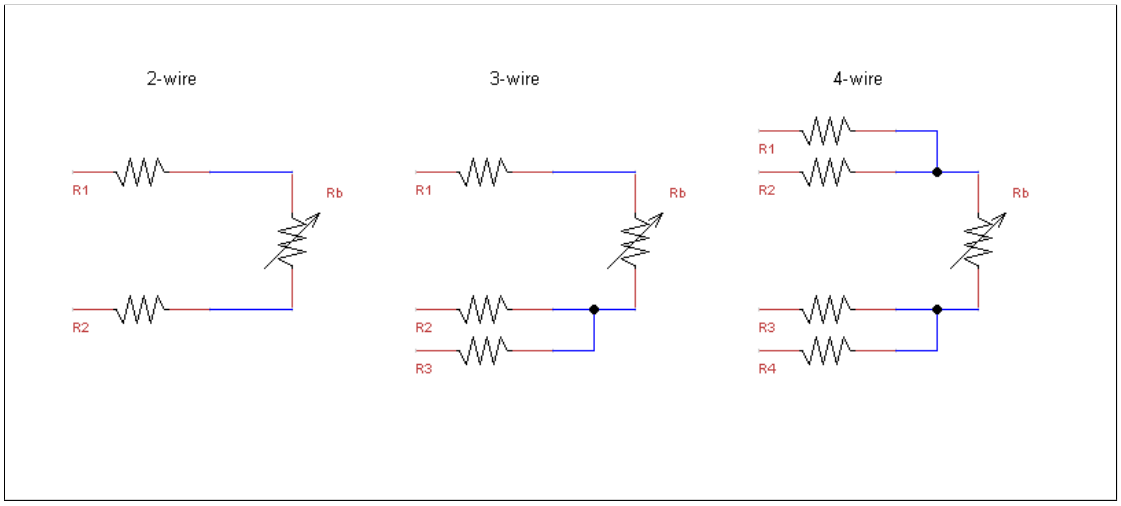


Figure 3.4 RTD sensors types

The last type of temperature sensors we will take a look at is thermocouple. They are using a fact, that if two different metals are connected, a difference in electrical potential appears at the point of connection. In thermocouple sensors, there are two junctions, reference and measured. By applying temperature changes on measured junction, due to thermoelectric effect, we are able to read a difference of voltage on these two connected types of conductors. This difference in voltage is then converted to a temperature. Depending on the type, thermocouple sensors can have an operating range from -200 to 1600 or even 3000 degrees Celsius, that's why they are the most used type in industry. [11, 12]

Considering all the facts and parameters, we can conclude that platinum three-wire RTD is the most suitable type for our project. There are a couple of reasons for that. Thermistors are basically working on the same principle, but have a smaller operating range. Thermocouples on the other hand are an overkill for such a small system, we definitely will not work with temperatures over 200 degrees Celsius, so there is no reason to spend more project founds on that type of sensors. As for four-wire version of platinum RTD, it will be an insignificant upgrade from three-wire version for more funds, and two-wire version is just worse and create us additional problems.

A three-wire PT100, which we are going to use, is a very popular model of platinum RTD, widely used in smaller industrial projects. To improve the precision of measured data, we will use MAX31865 RTD-to-Digital converter from Adafruit [17]. This device is using SPI communication interface and has its own library for Arduino board, which will help us in implementation.

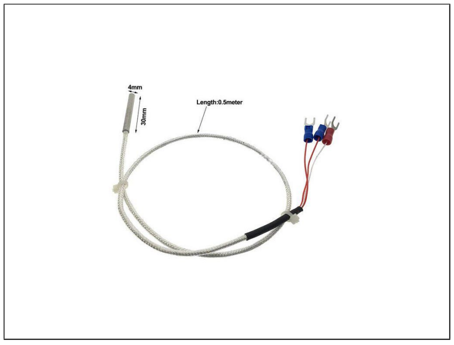


Figure 3.5 PT100 sensor [18]

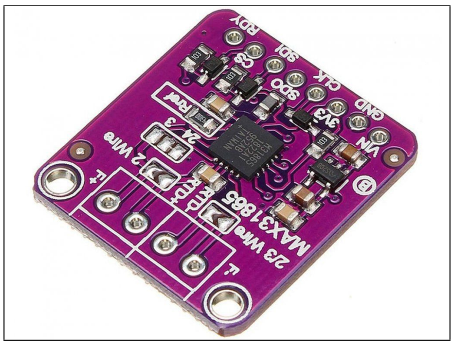


Figure 3.6 MAX31865 converter [19]

3.3.4 Water tank

Talking about a water tank for our system, we need easy access inside, 10l volume, comfortable transportation even when filled and tightness. Also, if the selected tank will not have its own input and output for water, holes for sensor and heating unit wire, we need to drill them. We will pick between four options: cooking pot, pressure vessel, plastic container, milk-churn.

Let's look at them one by one, starting from a cooking pot. The first two parameters we are looking for are presented in that option. Every cooking pot has a removable lid, so we can have access inside, and we can find a pot of almost any volume. Next quality is comfortable transportation, which can be questionable for a standard filled cooking pot, since the lid is not attached to the body in any way, it is just laying on top of it. Also, a regular cooking pot doesn't have any holes in it, we can put water tubes and all wires through the wtop, but this solution is not safe and stable, because water will flow under pressure from a pump and can move tubes around. Summarizing, we can say a cooking pot is a mediocre solution in terms of stability and safety.

The next solution we can pick from is a pressure vessel, a container designed to hold water under a pressure different from outside [20]. First, let's point out their advantages. Same as for pots, there are pressure vessels of almost any volume on the market, but the difference is that you don't have easy access inside in smaller versions of vessels. It could be a minor problem, but drilling holes in pressure vessels is a tough task requiring special power tools, because they have very thick or multiple layered walls plus rubber lining or membrane inside [20]. Therefore, we can't use such a tank for our system, it's not meeting more than half of our parameters.

A plastic container discussion will be very short, because of one reason, which wasn't mentioned in the first paragraph of that subsection, it's temperature of working heating unit. It's possible to find plastic tanks of any volume and shape, but it will not change a fact, that it will melt from temperatures produced by heating unit. So, this option can't be considered for our system too.

Our last option is milk-churn, another food related dish. It has all positive sides of a cooking pot plus some upgrades. Foremost, its lid is attached to the body and can be pressed down and fixed, so we can transport it filled and not be afraid of leakage. Also, in terms of transporting, milk-churns always come with a strong handle on top. The only customization we need to do is to drill a few holes for wires and water input, output. So, considering all facts above, we will use a milk-churn as our water tank, all customization works will be described in Chapter 4: "Implementation, Water tank and heating unit".

3.3.5 Heating unit

In our task, we were asked to have our coolant at 85-95 degrees Celsius, so we need to heat it up in a water tank. For that purpose, we will use a usual dump load water heater, because it's easy to operate and change, if needed. These devices can use a different amount of electric power, from hundreds to thousands of Watts, which determine how fast they can heat the same amount of water. For our project we will use 1000W one, because we can have maximum of 10l of water in our tank, which is a relatively small amount. During testing phase, we will see if that will be enough, or we should use a more powerful device.

3.3.6 Pump

The water in our system should be constantly moved, so it doesn't overheat or cool down too much. For that reason, we need a pump, which can handle our operating temperature of 85-95 degrees Celsius and will move our coolant fast enough. The most suitable option is a circulating pump for heating systems. It is perfect in terms of operating temperature and will create a good flow in our small system, because they are made to move water across buildings, so in our case their power will be more than enough. We will use AQUART 25/4/180, a simple version of circular pump without any in-build regulated systems, with a maximum flow speed of 48l/min.

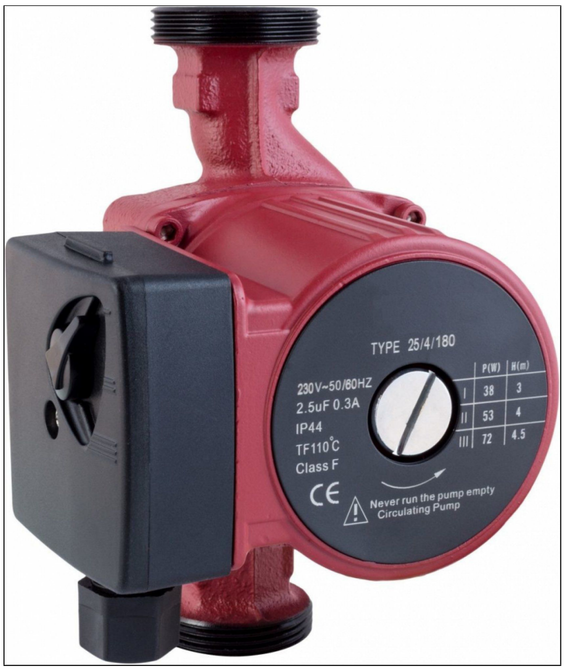


Figure 3.7 Circular pump AQUART 25/4/180 [21]

3.3.7 Display

On display, we need to show the status of the pump, heating unit and current temperature, that we read with the sensor. There are no specific parameters we are looking for in it, but we can take advantage of our converter using SPI communication and use a display with SPI interface. These two devices will share three pins on the Arduino board for serial clock, master out slave in, master in slave out. Only the fourth pin, slave select, will be different for them [22]. That fact will save us more pins, so we can connect more devices into our system, if we need. In that project, we will use 128x160 TFT display [23], since it has a library for the Arduino board.

3.3.8 Relays and power supply connections

The last electric component we need for our system is a relay, in fact two of them. We will use them to operate the pump and the heating unit. Since both devices operate at 230V, this is the voltage we need relays to handle. For our project, we will use a module with 4 relays [24] on it, so we can expand our system with new devices if needed.

As for power supply connections, we will use a single input, a plug, to power up the system, and then split it to connect all needed devices. Also, we will need a power adapter from 230V AC to 12V DC, to supply our Arduino board [10]. All devices will be grounded as it's shown on the scheme for safety reasons. The method we are using to connect devices through relays will be described later in implementation phase.

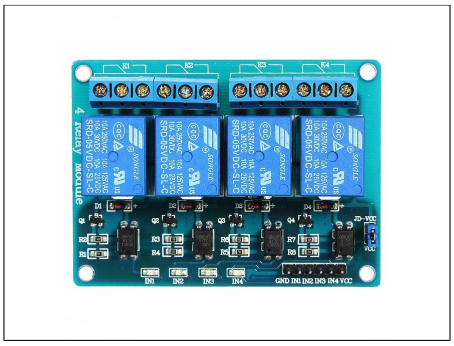


Figure 3.8 Module with 4 relays [25]

Figure 3.9 MAX31865 converter [26]

Chapter 4

Implementation

In this chapter, we will step-by-step build our system. We will start with SW section, where sensor, display and relays will be connected to microcontroller board and tested for proper functioning. Then we will get to power electronics and water loop, all hand-crafting works will be described along with how the system is power upped.

4.1 Software implementation

In this section, we will talk about SW implementation and wiring of the components to the Arduino board. We will focus on base constructors calls, setups and describing the logic of the program, rather than going into the details, since a whole code can be founded in appendix "Arduino sketch, code", where the code is properly commented and explained.

4.1.1 Needed libraries

As mentioned in Chapter 3: "Components choosing" we will use libraries for display and sensor convertor. Both of them are made by Adafruit and can be found in Arduino IDE following these steps:

$$Sketch \rightarrow Include\ library \rightarrow Manage\ libraries$$

Then type MAX31865 for converter library, ST7735 and GFX for the display libraries. ST7735 is a core HW library for the display, specifying which display we are using and what pins are on it. GFX is a graphics library, so we can draw on the display. Also, we need to install "SPI.h" Arduino default library because we are using that communication protocol on both convertor and display.

Code listing 4.1 Libraries added to the program

```
#include <SPI.h>
#include <Adafruit_GFX.h>       // Core graphics library for display
#include <Adafruit_ST7735.h>    // Hardware-specific library for display
#include <Adafruit_MAX31865.h>  // Core library for MAX31865 converter
```

4.1.2 Sensor

Our PT100 sensor comes with a U-type connectors on all three wires, the first step will be to fold them vertically in a half, so we will get J-type connectors. That will later allow us to insert those wires into the convertor terminal. Now we will put the sensor aside and prepare the convertor before we get to SW implementation, because it needs some soldering done.

A pin strip and two terminals should be solder to the convertor board. We need to put the strip long pins down and insert short-pin side into the holes on the board, this pins will be used for SPI communication protocol. We solder each pin separately, there should not be any connection between them. Now we can get to terminals, there are two of them. We put them pins down into the holes on the other side of the board. Again, we solder each pin separately. The last sing we have to do is to solder closed blobs two jumpers on the board. The first one is named “2/3 wire”, the other one is “24 3”, solder together two jumpers above number 3. Now insert two same color wires into the right terminal, and the third one into the left terminal, which hole doesn’t matter. A more detailed quid can be found on Adafruit website. [27]

Now we have installed all needed libraries and prepared the convertor, we will connect it to the microcontroller. How it should be done is shown in Table 4.1.

Table 4.1 Wiring of the convertor to the Arduino board.

Convertor SPI pins	Arduino pins
VIN	5V
GND	GND
CLK	D13
SDO	D12
SDI	D11
CS	D10

After we have wired the convertor to the microcontroller, we can start implementing needed functions. First, we need to create a variable of “Adafruit_MAX31865” type and save a constructor call result into it. This constructor parameters are the pins we piked on Arduino in the order: CD, SDI, SDO, CLK. Also, we need to define reference resistance of our RTD, in our case it’s 430, and a nominal 0-degrees-Celsius resistance, we have PT100 model, that means our nominal value is 100. In the setup section, we need to call a “begin” function with a parameter, which defines what type of RTD we have, in our case it’s “MAX31865_3WIRE”. The last thing we need to do, to read the temperature from the sensor, is to call a “temperature” function with the nominal and the reference resistance as parameters. We will save it into the “float” type variable and later use it to control the heating unit and print the temperature on the display.

Code listing 4.2 Sensor and converter setup code

```
//software SPI for MAX31865 converter: CS, DI, DO, CLK
Adafruit_MAX31865 thermo = Adafruit_MAX31865(10, 11, 12, 13);
// The value of the Rref resistor. 430 for PT100
#define RREF      430.0
// The 'nominal' 0-degrees-C resistance of the sensor 100 for PT100
#define RNOMINAL  100.0
...
// set converter to 3WIRE
thermo.begin(MAX31865_3WIRE);
...
// variable to store the temperature read from the sensor
float temp = thermo.temperature(RNOMINAL, RREF);
```

4.1.3 Display

Same as with sensor convertor, we need to connect the display to the microcontroller and set it up. Wiring is shown in Table 4.2. Then we have to call a constructor and store it to the "Adafruit_ST7735" variable type. Parameters of the constructor are again pins we picked in the order: CS, RS, SDA, CLK, RST. In the setup section, we call initiation function, fill the screen with black color, set the rotation and text wrap.

Table 4.2 Wiring of the display to the Arduino board.

Display pins	Arduino pins
VIN	5V
GND	GND
GND	GND
CLK	D13
SDA	D11
CS	D9
RST	D8
RS	D7

Code listing 4.3 Display setup code

```
//  A setup function call for ST7735 display with
//  pin numbers: CS, RS, SDA, CLK, RST
Adafruit_ST7735 tft = Adafruit_ST7735(9, 7, 11, 13, 8);
...
// init ST7735S chip, black tab
tft.initR(INITR_BLACKTAB);
// set display background to black collor
tft.fillScreen(ST7735_BLACK);
// set whether text that is too long for the screen width should
// automatically wrap around to the next line (else clip right).
tft.setTextWrap(true);
// set the display rotation on 90 degrees, so it will print text
// paralell to the longer side
tft.setRotation(1);
```

With a prepared display, we can think about how the information will be printed on it. The top half of the screen will be a place for the temperature in Celsius. Also, we want to mark if the temperature is below or above working range, because if it stays out of working range for too long, it means something is wrong and the system must be checked and repaired. For that matter, we will print the temperature below working range in a blue color, and with a red color when it's above. While it's in the range, we will use the white color. This top half space will also be used for displaying an error, that may appear on the sensor. In that case, a temperature will be replaced with a purple code of the error and the system should be checked.

The other, bottom half will be used for printing the status of the heating unit. We will write "Heating unit status" and below that "ON" or "OFF" depending on the status. This part will always be printed with a white color. When we will print any text, a background color is used, meaning we are calling "setTextColor()" function with two parameters, the first one for the text color and the other one for the background, which is the same as default background, black. It's made to completely remove the text, that was printed on that place before. However, when we are printing "ON" after "OFF", we will have to draw a rectangle on "OFF" first, because "ON" is shorter and even if we type it with a background color, a part of "OFF" will stay on the screen.

4.1.4 Relays

The relay is the simplest component that we will connect to the Arduino board, as shown in Table 4.3. Here we don't need any libraries, the only thing we have to set up is to define pins we are using on the microcontroller board. Then we will set the pin for the pump on "LOW", because it should be working all the time the system is running, so we will use normally open mode, when the relay is opened while it's on "LOW". The other pin, for the heating unit, will be initiated as "LOW" also, but only in the setup section, then it will change depending on the temperature. We will set this pin on "HIGH" when the temperature will fall below the minimum we defined, and on "LOW" when it will get over the maximum we defined.

Table 4.3 Wiring of the relay to the Arduino board.

Relay pins	Arduino pins
VIN	5V
GND	GND
IN1	D6
IN2	D5

4.1.5 Main loop

We have prepared all components, connected to the microcontroller, now we will describe how the overall logic of the program works. After the information was acquired from the sensor, we will call an error checking function before doing anything else. If there is an error, an error code will be saved into the variable. The next step is to call a temperature printing function, where if the error code is not zero, an error message will be printed, else the temperature will be displayed in a proper color. Now we can turn off or on our heating unit through a relay if needed with a heating unit controlling function. If there was an error, it will be turned off no matter what, for the safety reasons. And the last step of the loop will be printing the status of the heating unit. This logic will be completed every 500 ms.

4.2 Hardware management

In this section we will describe all HW works, that were done, such as power wiring, drilling, creation of water loop. These actions should be done very carefully as the safety of our system depends on it. We need to make sure, that water will not find its way to electrified parts of the system. To prevent any possibility of power shortage, make sure there is a fast way to shut down the system, such as switch or button. And last but not least, make the system look good.

4.2.1 Water tank and heating unit

Our water tank, milk-churn, needs some customization to be done before we connect it to the system. First, we will drill two holes 50 mm above the bottom and 100 mm apart each other. We will use 1/2" barrel connectors, so the holes should be 213 mm wide in diameter. Now we need to drill two holes in the lid of the milk-churn, one 6 mm wide for the sensor and the other one will be oval for the heating unit handle 30 mm long and 20 mm wide. For a better precision, we will always start to drill with a 2 mm bit, then 4 mm, 6 mm and at the end we will use a step bit, all of them are for metal drilling. This approach will make a drilling process more stable and lowers the possibility of a mistake.

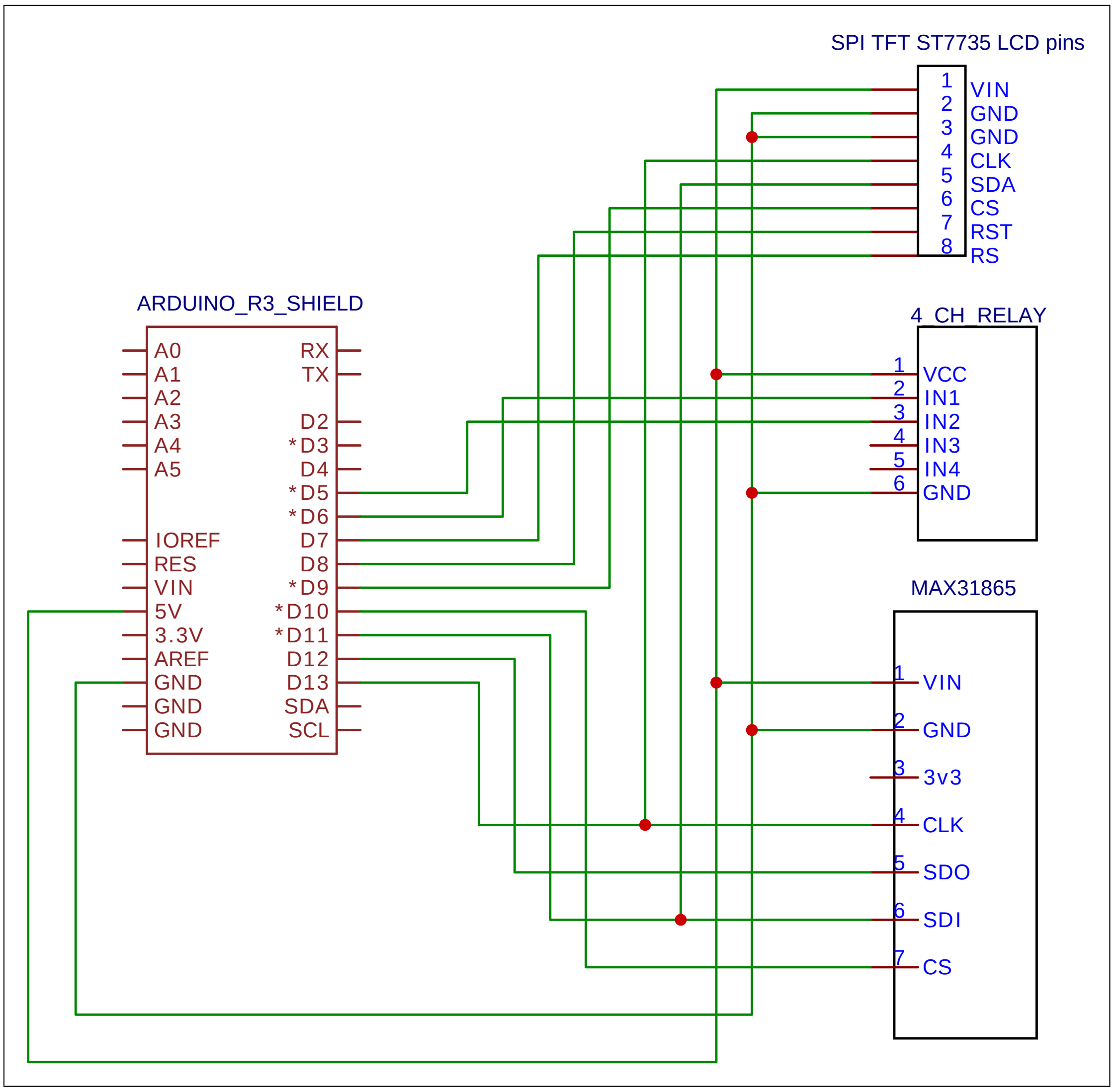


Figure 4.1 Wiring of peripherals to the Arduino board

Now we will place our barrel connectors and tight them up on the wall of our water tank. These connectors have a nut with a washer, rubber seal and a pipe with a washer on one end. We should put the seal on the pipe and push it to the washer on the end, then put it through the hole on the milk-churn, so that the washer will stay inside, and a seal will be between it and the wall. Now from the outside we will put a nut on the pipe and screw it until we touch the wall. We should tighten it, so the rubber seal spreads around the hole inside. These connections will be our input and output for the water cycle, on the reservoir side.

The next step is to put the heating unit inside, push its wire through the hall in the lid we made, put the lid back on the milk-churn and fixate it. The sensor will be inserted through the other smaller hole later, when we connect and place all electronics. And our water tank is complete.

4.2.2 Water tubes and their connections, pump

With a prepared water tanker, we can start building our water loop. For tubes, we will use flexible hose pipes for hot water, then we also need a few pipe fittings for the pump, as it has 6/4” outputs, but we need to reduce it to 1/2”. We will use a 6/4” to 1” reduction and then 1” to 1/2”, put them on both ends of the pump and put a rubber tightening seal inside every fitting, so the water will not leak through them, if it will still leak, we can put a thread seal tape on threads of the fittings and the pump. We need to tighten every fitting to prevent leakage, first we will put 6/4” to 1” reductions on the pump and properly tighten it, we should not try to twist the fitting until the dead end, since it can cut the rubber seal inside, it’s better to tighten something up during the first system tests.

Now we can get to pipes, the same rule is applied here, don’t screw the ends of the pipes to connections too hard, so it will not cut the seals inside. The bath hose pipes have a seal in its female connection, we will use a female/female version, so we don’t need to put any additional seals. We also need to pay attention when acquiring those pipes, because there are versions of them for only cold water, we can distinguish them by colored stripes on them, only blue stripes for only cold water, blue and red stripes for cold and hot water. The first pipe will be short and will connect the pump with the water tank, 0.3 m length will be enough. Then we will need longer ones to connect the water tank to the motor output and the pump to the motor input. From the pump, we will put a 1.2 m long pipe and from the water tank a 0.6 m plus 1.2 m long pipes. For the testing purposes, before connecting it to the motor, we will loop the system with a corner type fitting.

4.2.3 Power supplying connections, relays

In our system we have three components, that need external power supply:

- pump
- heating unit
- microcontroller

The first two of them need 230V AC and an Arduino board needs 12V DC supply, so we have two options, have two different voltage sources one for the pump and the heating unit and the other for the board, or have one source but place a converter to power up the Arduino. We will use the second method and will use LYONZG S-25-12 voltage module converter.

The power will come through the plug connected to the 230V socket. Then we will have three terminals for ground, null, phase wires. From these terminals, one of each wire type will go to the voltage converter, but the output of it will not have a ground wire, only null and phase connected to the 2.1 mm connector. This type of the connector is required to power up the Arduino board.

The pump and heating unit should be connected through the relay module. The null and ground wires will be connected to them directly, and the phase will go through the relay channel. The end that comes from the phase terminal will be connected to the common port of the relay for both devices. As we mentioned before, the pump will be running in normally open mode, meaning the relay will let the voltage trough, when the input of that channel from the microcontroller is on “LOW”. The heating unit on the other hand will use normally closed mode, that’s the opposite of normally open mode. We will use this method to reduce the power drain from the board, as the pump will be running all the time, so we will never set it’s input channel on “HIGH”. And the heating unit will be set on “HIGH” less often than on “LOW”, since it will take more time for water to lose the temperature, than to heat up.

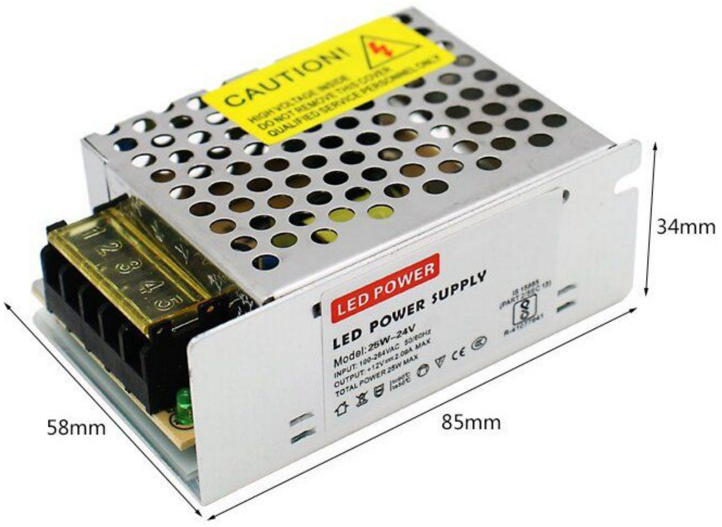


Figure 4.2 LYONZG S-25-12 [28]

4.2.4 Final placing and adjustments

The final step of the implementation is the placing of all components, to make our system more comfortable to transport and move around the test bench. We will arrange components, except the pump and the heating unit, in three groups. To separate them and keep together, we will use plastic boxes for electricity wiring.

The first group will be breadboard, Arduino board, relay module, display. The first three components will be placed in the box and the display will be on the lid of it. The boxes we will use have premade covered holes on them, that can be opened just by hand. Overall, we need to open 4 holes, one for power cables entrance, one for their exit, two others will be for the display and the sensor amplifier data wires. The next box will be a power supplying point of the system. A voltage convertor and terminals for power wires will be placed here, so we need one entrance hole for the plug cable, one exit for the pump and the heating unit cables and the last one will be the exit for the Arduino supplying cable. The last box will be placed on the lid of the milk-churn, a sensor amplifier will be hidden here. We need this cover for the amplifier, to protect if from the water and possible damaged from the outside.

At the end, we will have the pump, two boxes with electric components and the water tank. We don't count heating unit and the third box, because the first one is inside the water tank and the second one is fixed on its lid. Also, for better user experience, we will put an adapter with a switch on the plug, so we can easily turn and of the system, without touching the plug.

Chapter 5

System testing

During and after the implementation, we made test to make sure every component is working properly. We will mention them in this chapter with an explanation. We will start with SW tests, then proceed to HW tests and at the end describe overall system tests.

5.1 Software testing

Below, we will talk about tests, that were made for software testing. Here will come to use a serial link with PC, set up in the code. Software testing is needed to make sure our power electronics will be operated in the right way, because wrong data from the sensor can make the heating unit improperly, overheating or not heating enough the coolant. The relay can cause the same problems, also 230 V will go through it, so it's also a safety matter to test it properly.

5.1.1 Sensor

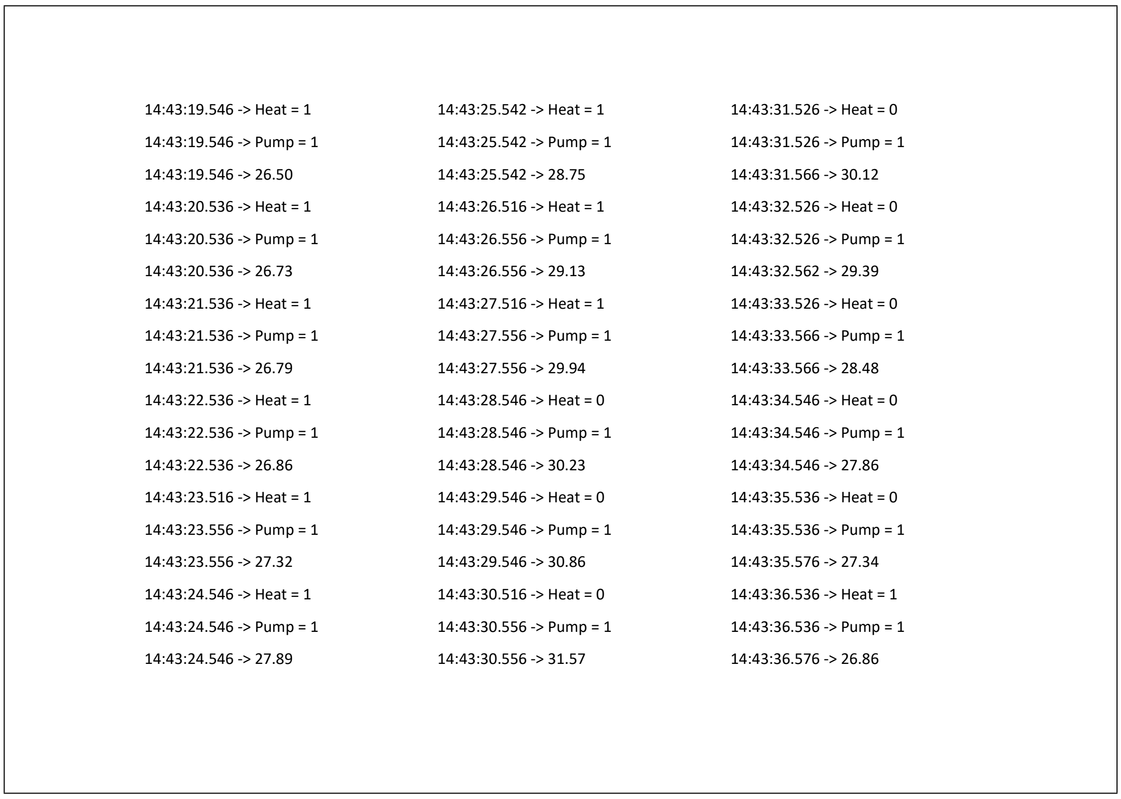
After we set up the sensor and its amplifier, wired it to the microcontroller, we added a code that prints the temperature we get from the sensor to the serial link terminal. The next step was to check the accuracy of the sensor, for that purpose we acquired cooking manual and digital thermometers, alcohol thermometer. We put all of them together with our sensor to close to boiling, room temperature, cooled water and compared the results we got, which we can see on Table 5.1. According to the test results, we can say, that our sensor is working properly, and we can rely on the information we will read from it, which allows us to move to the relay testing.

Table 5.1 Sensor temperature accuracy test results in degrees Celsius

Thermometer / Tested water	Cooking manual	Cooking digital	Alcohol	Sensor
Near the boiling point	96.7	96.73	97.7	96.72
Room temperature	23.1	23.08	23.0	23.12
Cooled	9.95	10.01	9.98	10.01

5.1.2 Relay

The relay related software was tested for the opening and closing the electric channel depending on the temperature we read from the sensor. As a setup for this test, we prepared a room temperature water and put the working temperature range on 27 to 30 degrees Celsius, so we can warm the sensor with our hands and then cool it in the water. Before connecting electrical devices to the relay channels, we used a serial link again, printing the status of the channels and the temperature on it. On Figure 5.1 we can see a part of the test result, printed in the serial link terminal.



```
14:43:19.546 -> Heat = 1      14:43:25.542 -> Heat = 1      14:43:31.526 -> Heat = 0
14:43:19.546 -> Pump = 1      14:43:25.542 -> Pump = 1      14:43:31.526 -> Pump = 1
14:43:19.546 -> 26.50         14:43:25.542 -> 28.75         14:43:31.566 -> 30.12
14:43:20.536 -> Heat = 1      14:43:26.516 -> Heat = 1      14:43:32.526 -> Heat = 0
14:43:20.536 -> Pump = 1      14:43:26.556 -> Pump = 1      14:43:32.526 -> Pump = 1
14:43:20.536 -> 26.73         14:43:26.556 -> 29.13         14:43:32.562 -> 29.39
14:43:21.536 -> Heat = 1      14:43:27.516 -> Heat = 1      14:43:33.526 -> Heat = 0
14:43:21.536 -> Pump = 1      14:43:27.556 -> Pump = 1      14:43:33.566 -> Pump = 1
14:43:21.536 -> 26.79         14:43:27.556 -> 29.94         14:43:33.566 -> 28.48
14:43:22.536 -> Heat = 1      14:43:28.546 -> Heat = 0      14:43:34.546 -> Heat = 0
14:43:22.536 -> Pump = 1      14:43:28.546 -> Pump = 1      14:43:34.546 -> Pump = 1
14:43:22.536 -> 26.86         14:43:28.546 -> 30.23         14:43:34.546 -> 27.86
14:43:23.516 -> Heat = 1      14:43:29.546 -> Heat = 0      14:43:35.536 -> Heat = 0
14:43:23.556 -> Pump = 1      14:43:29.546 -> Pump = 1      14:43:35.536 -> Pump = 1
14:43:23.556 -> 27.32         14:43:29.546 -> 30.86         14:43:35.576 -> 27.34
14:43:24.546 -> Heat = 1      14:43:30.516 -> Heat = 0      14:43:36.536 -> Heat = 1
14:43:24.546 -> Pump = 1      14:43:30.556 -> Pump = 1      14:43:36.536 -> Pump = 1
14:43:24.546 -> 27.89         14:43:30.556 -> 31.57         14:43:36.576 -> 26.86
```

Figure 5.1 Serial link terminal data for the relay test

Now we can test the relay with light bulbs connected to the relay channels, so we would not damage the heating unit or a pump during the test, if something would not work properly. The bulb connected to the pump channel should be on all the time during the test, and the one connected to the heating unit channel should be turned on when the temperature falls below 27 degrees Celsius, and turn off when the temperature gets above. This test was also successful, allowing us to connect the heating unit to the relay, the pump will be connected later, after we test the water loop.

5.2 Hardware testing

Before fully assembling our system, we need to run checks on hardware. This tests will check the water loop for leakage, electricity connections for the correct voltage and recheck the wiring. We need to make these inspections for the safety reasons, since a mistake in these fields can lead to the fire. Also, wrong voltage fed to the Arduino board can make the board shut down or peripherals work incorrectly.

5.2.1 Pump and the water loop

For this test, we connected all the tubes, the pump and our water tank into the loop. The pump was connected directly to the plug with a switch. The first thing we are looking for is the leakage of the coolant from the reservoir. We filled the water tank up to the barrel connectors installed on it and observed the outside of these connectors, no leakage appeared. The next step is to fill the water tank to the half, fully covering the barrel connectors, start the pump and observe the whole loop for coolant leakage. During the first run, a leakage appeared on the pump, a rubber tightening between 6/4" to 1" and 1" to 1/2" reductions was too thin and was replaced with a thicker one. The second and later runs were successful, no leakage was found.

5.2.2 Power supplying connections

A multimeter will be required for these inspections. With it, we want to check the output voltage on the voltage converter, input from the plug, relay channels and current on the Arduino board with all running peripherals. This information will allow us to start the final assembling of the system, as we will be sure that the system will function properly and will not cause any danger. The results of these inspections are shown on Table 5.2, these results are satisfying for us to assemble the system, since all the voltage values are on needed level and the current on 5V pin of the Arduino is below 0.8A, which is the possible maximum for it [10].

Table 5.2 Measured values from tested points

Tested point	Measured value with units
Voltage converter output	11.9V
Input from the plug	230V
Relay pump channel	230V
Relay heating unit channel	230V
Current drain on 5V pin of the Arduino	0.07 A

5.3 Full system tests

Now we have tested all the components, software, water loop and assembled the system, placed smaller components into the boxes as we described in Chapter 4: "Final placing and adjustments", we can get to the final full system tests. Again, for safety reasons, we will test the system without the motor and only when we are sure there is no danger causing any damage to it, we will connect the motor, we were provided.

The idea of these tests is to see how fast we will heat up the coolant to the needed temperature, observe the power supplying connections and make sure that the program we made works properly over a long period of time. We will start with heating up the coolant to different temperatures and measure the time it will take, starting temperature is 22 degrees Celsius. The result of these tests are shown on Table 5.3.

Table 5.3 Time spent to heat the coolant to the target temperature

Target temperature in degrees Celsius	Time in minutes
40	10
60	19
70	25
85	32

During these tests, we have observed how the microcontroller is working over a long period of time, no overheating or failure in the program appeared, meaning the program is working properly, and the circuit is build with no mistakes. Also, we have measured the same points as in the previous subsection, and got no power spices or drops, all measured values were stable and in the working range.

The last, final step is to connect our system to the running motor and observe the same properties, mentioned above. Three testing sessions, one hour each, were made. The first session was stable with no failures, electricity was stable, the microcontroller operated everything properly. But, the second session brought an unexpected problem, an Arduino board was periodically restarting itself, it didn't cause any problems in the system functionality, but it still indicated that there is some error in the circuit. After everything was measured and debugged, a power spices on the voltage converter output were found, a new one was installed. The third testing session was stable, everything worked properly, therefore we can claim that the system is done and can be used in simulation projects it was made for.

Chapter 6
Conclusion

The goal of the thesis was to build a liquid cooling system for an electric motor, with a coolant temperature at 85 to 95 degrees Celsius. To acquire correct components for our system, we had to analyze existing methods of liquid cooling, possible coolants and solutions in in-production electrical vehicles.

We have analyzed direct and indirect liquid cooling systems. For direct methods, we took a look at the cold plate method, which is used on the motor we were provided to test our system. For indirect methods, we have mentioned the jet impingement cooling, spray cooling and immersion cooling methods. We have provided a table with the thermophysical properties for the selection of coolants, based on which we have picked water as our coolant. A research we made about existing solutions in modern EVs, brought us the information that it's always a combination of different methods, but in all mentioned vehicles it's a closed circulating loop of coolant.

Based on gained knowledge about liquid cooling, we have picked components for our system. For the sensor, we have chosen a platinum RTD with 100 Ohm resistance at 0 degrees Celsius. That decision was made based on the comparison to other types of temperature sensors, thermocouples and thermistors. We have programmed an Arduino Uno board, which is based on ATmega328P microcontroller, mentioned it the thesis task, to operate display, relays and sensor amplifier. An amplifier was added to increase precision of the sensor. Also, we have picked components for our water cycle, such us pipes, pump and water tank, based on working temperatures of our coolant, and described the handcrafts made on the water tank. The implemented system was properly tested and put into the work with an actual motor, after we have fixed failures, that appeared, the system worked as it was meant. Therefore, the main task of building the liquid cooling system was completed. Our system is capable of feeding water at a temperature up to 90 degrees Celsius to motors, that use an indirect cooling method.

Changes were made in the system in comparison with our task. Instead of separate switches for the heating unit and the relay, mentioned in the task, we have automized the turning on and off process by adding relays, operated by our program. Due to the pump running all the time, we made a decision to put the heating unit status, instead of the pump state. Additional sensors were not added, but a proper wiring and components placement allows us to add them if needed. Also, as we have used a four channel relay and put to work only two channels, we can add two more power electric devices, additional heating for example.

To develop and upgrade our system for a more complex simulations and tests, we can replace a pump with another one, that can control the flow speed. As we mentioned above, we have a possibility to add more sensors to the systems, such as leakage sensor, water level sensor, flow speed sensor. We can craft or found more proper boxes for our smaller electrical components. Change water pipes to once with a higher working temperature range.

Appendix A

Arduino sketch, code

Code listing A.1 Sensor and converter setup code

```
#include <SPI.h>
#include <Adafruit_GFX.h>       // Core graphics library for display
#include <Adafruit_ST7735.h>    // Hardware-specific library for display
#include <Adafruit_MAX31865.h>  // Core library for MAX31865 converter

//software SPI for MAX31865 converter: CS, DI, DO, CLK
Adafruit_MAX31865 thermo = Adafruit_MAX31865(10, 11, 12, 13);
// The value of the Rref resistor. 430.0 for PT100
#define RREF      430.0
// The 'nominal' 0-degrees-C resistance of
// the sensor 100.0 for PT100
#define RNOMINAL  100.0

//  A setup function call for ST7735 display
// with pin numbers: CS, RS, SDA, CLK, RST
Adafruit_ST7735 tft = Adafruit_ST7735(9, 7, 11, 13, 8);

//define relay pins
#define PUMP_IN         5  // pump relay IN pin
#define HEATER_IN       6  // heating unit IN pin

//define working temperatures of the system
#define MAX_TEMP      29.0
#define MIN_TEMP      28.0

// variable to store the heating unit state 0-off 1-on
short heatState  = 0;
// variable to store the heating unit previous state
// needed for printing on display
short prevHState = 0;
// varible to store an fault check for the sensor
short errCheck   = 0;
```

```
void setup() {

  // set converter to 3WIRE
  thermo.begin(MAX31865_3WIRE);
  // init ST7735S chip, black tab
  tft.initR(INITR_BLACKTAB);
  // set display background to black collor
  tft.fillScreen(ST7735_BLACK);
  // set whether text that is too long for the screen width should
  // automatically wrap around to the next line (else clip right).
  tft.setTextWrap(true);
  // set the display rotation on 90 degrees, so it will
  // print text paralell to the longer side
  tft.setRotation(1);
  // set relay pins into output mode
  pinMode(PUMP_IN, OUTPUT);
  pinMode(HEATER_IN, OUTPUT);
  // set both relay pins on LOW, so the pump will start running
  // and the heating unit will wait until we set it on high
  digitalWrite(PUMP_IN, LOW);
  digitalWrite(HEATER_IN, LOW);

  // serial link setup at 9600 bps
  Serial.begin(9600);

}

// function to print the temperature read ftom the sensor on the display
// takes a pointer to the variable that stores the temperature as
// a parameter sets cursor on the needed position on the display and
// prints the temperature if there was a fault on the sensor
// print error code message
void printTemp(float* temp) {

  // reset text color to white
  tft.setTextColor(ST7735_WHITE, ST7735_BLACK);
  // set text size for display on 4
  tft.setTextSize(4);

  if (errCheck) {
    tft.setCursor(10, 15);
    tft.setTextColor(ST7735_MAGENTA, ST7735_BLACK);
    switch (errCheck) {
      case 1:
        tft.print("ERTMAX");
        break;
      case 2:
        tft.print("ERTLOW");
        break;
      case 3:
        tft.print("ERREFL");
        break;
      case 4:
        tft.print("ERREFH");
        break;
      case 5:
        tft.print("ERRTDL");
        break;
      case 6:
        tft.print("ERRVOL");
        break;
      default:
        tft.print("ERRUNK");
        break;
    }
  }
  else {
    if (*temp < MIN_TEMP) {
      tft.setTextColor(ST7735_BLUE, ST7735_BLACK);
    }
    else if (*temp > MAX_TEMP) {
      tft.setTextColor(ST7735_RED, ST7735_BLACK);
    }
    tft.setCursor(25, 15);
    tft.print(*temp);
  }
  Serial.print(*temp);
}

// function to check the sensor fault status, used for testing or repairing
// purposes if the sensor works incorrecctly it will be detected and
// the proper message will be sent to the serial link if its setted up
void sensorErrorCheck() {

  uint8_t fault = thermo.readFault();
  if (fault) {
    Serial.print("Fault 0x"); Serial.println(fault, HEX);
    if (fault & MAX31865_FAULT_HIGHTHRESH) {
      Serial.println("RTD High Threshold");
      errCheck = 1;
    }
    if (fault & MAX31865_FAULT_LOWTHRESH) {
      Serial.println("RTD Low Threshold");
      errCheck = 2;
    }
    if (fault & MAX31865_FAULT_REFINLOW) {
      Serial.println("REFIN- > 0.85 x Bias");
      errCheck = 3;
    }
    if (fault & MAX31865_FAULT_REFINHIGH) {
      Serial.println("REFIN- < 0.85 x Bias - FORCE- open");
      errCheck = 4;
    }
    if (fault & MAX31865_FAULT_RTDINLOW) {
      Serial.println("RTDIN- < 0.85 x Bias - FORCE- open");
      errCheck = 5;
    }
    if (fault & MAX31865_FAULT_OVUV) {
      Serial.println("Under/Over voltage");
      errCheck = 6;
    }
    thermo.clearFault();
  }
}
```

```
// function to control the heating unit
// as parameter takes a pointer to the temperature variable
// comparing temperature to the working tresholds, and depending on
// the result turns the pump on or off
// if there is a fault on the sensor, the heating unit will be turned off
void heatingControl( float* temp) {

  prevHState = heatState;

  if (errCheck) {
    heatState = 0;
    prevHState = 1;
    digitalWrite(HEATER_IN, LOW);
  }
  else {
    if (*temp < MIN_TEMP && heatState == 0) {
      heatState = 1;
      digitalWrite(HEATER_IN, HIGH);
    }
    else if (*temp > MAX_TEMP && heatState == 1) {
      heatState = 0;
      digitalWrite(HEATER_IN, LOW);
    }
  }

}

// function for printing the heating unit status
void printHeatState() {

  // reset text color to white
  tft.setTextColor(ST7735_WHITE, ST7735_BLACK);

  // set text size for display on 2
  tft.setTextSize(2);
  tft.setCursor(10, 60);
  // status word is shifted so it will be displayed
  // in the middle of the next row
  tft.print("Heating unit    status");

  // set text size for display on 3
  tft.setTextSize(3);
  if (heatState) {
    if (prevHState == 0) {
      tft.fillRect(49, 99, 90, 120, ST7735_BLACK);
    }
    tft.setCursor(65, 100);
    tft.print("ON");
  }
  else {
    tft.setCursor(60, 100);
    tft.print("OFF");
  }

}
```

```
// main function calling operating functions ever 0.5 s
void loop() {

  // variable to store the temperature read from the sensor
  float temp = thermo.temperature(RNOMINAL, RREF);

  sensorErrorCheck();
  printTemp(&temp);
  heatingControl(&temp);
  printHeatState ();

  delay(500);

}
```

Appendix B

Components list

Table B.1 Microcontroller and peripherals

Name	Amount
Arduino Uno rev3, original	1
1.8” 128x160 TFT display, ST7735, SPI	1
PT100 platinum temperature sensor 0.5 m, 3-wired	1
Amplifier for PT100 sensor, MAX31865, SPI	1
4-channel relay module	1
Breadboard 400 pins	1
Male to Male Solderless Flexible Breadboard Jumper Cable Wires	28
Female to Female Solderless Flexible Breadboard Jumper Cable Wires 4-pin	8

Table B.2 Power electrical devices and wiring

Name	Amount
AQUART 25/4/180 circular pump	1
Immersion heater 1000W	1
LYONZG S-25-12 supply module 230V AC-DC 12V/2A 25W	1
Cable 3x1	3 m
Cable 3x1 with a plug	2 m
5-pin terminals	6

Table B.3 Water loop components

Name	Amount
Aluminum milk-churn, 10l	1
Hose pipe for hot and cold water, Female to Female, 0.3 m	1
Hose pipe for hot and cold water, Male to Female, 0.6 m	1
Hose pipe for hot and cold water, Female to Female, 1.2 m	2
Barrel connectors 1/2”	2
Reduction 6/4” to 1”	2
Reduction 1” to 1/2”	2
Rubber tightening 1”	2
Rubber tightening 1/2”	2

Bibliography

1. VARE, Ethlie Ann; PTACEK, Greg. *Patently Female*. Hoboken, NJ: John Wiley I& Sons, 2001. ISBN 978-0-471-02334-0.
2. WAKEFIELD, Ernest Henry. *History of the Electric Automobile*. Warrendale, PA: Society of Automotive Engineers, 1994. ISBN 978-1-56091-299-6.
3. TONG, W. *Mechanical Design of Electric Motors*. CRC Press, 2014. ISBN 9781420091441. Available also from: `https://books.google.cz/books?id=DizNBQAAQBAJ`.
4. TESLA, Inc. *Electric motor waste heat mode to heat battery*. US 20180083509A1. 2018-03. Available also from: `https://patents.google.com/patent/US20180083509A1/en`.
5. FOX, Eva. *Tesla New Patent To Use Electric Motor Waste Heat Mode To Heat Battery* [online]. 2020 [visited on 2022-04-20]. Available from: `https://www.tesmanian.com/blogs/tesmanian-blog/electric-motor-waste-heat-mode-to-heat-battery`.
6. AG, Audi. *Audi e-tron cooling concept e-motor (animation)* [online]. 2019 [visited on 2022-04-21]. Available from: `https://www.audi-mediacenter.com/en/audimediatv/video/audi-e-tron-cooling-concept-e-motor-animation-4847`.
7. THOMAS, Robin; HUSSON, Hugo; GARBUIO, Lauric; GERBAUD, Laurent. *Comparative study of the Tesla Model S and Audi e-Tron Induction Motors* [online]. 2021 [visited on 2022-04-21]. Available from: `https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/326-1_4-kanaly-rele-modul-5vdc-250vac-10a.jpg?61d95c9e`.
8. CONGRESS, Green Car. *BMW using flax fiber cooling shaft in iFE.20 Formula E racer* [online]. 2019 [visited on 2022-04-21]. Available from: `https://www.greencarcongress.com/2019/12/20191229-bmw.html`.
9. ABB FORMULA E, YouTube chanel. *How Do Formula E Cars Reduce Overheating?* [Online]. 2017 [visited on 2022-04-21]. Available from: `https://www.youtube.com/watch?v=fREvd4dLY2s`.
10. ARDUINO. *Arduino UNO R3* [online]. 2022 [visited on 2022-03-22]. No. SKU: A000066. Available from: `https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf`.
11. BENTLEY, J P. Temperature sensor characteristics and measurement system design. *Journal of Physics E: Scientific Instruments*. 1984, vol. 17, no. 6, pp. 430–439. Available from DOI: `10.1088/0022-3735/17/6/002`.
12. MATHIVANAN, N. *PC-BASED INSTRUMENTATION: CONCEPTS AND PRACTICE*. PHI Learning, 2007. ISBN 9788120330764. Available also from: `https://books.google.cz/books?id=OB65rMNQDo8C`.

13. JONES, C.T. *Programmable Logic Controllers: The Complete Guide to the Technology.* Patrick-Turner, 1998. ISBN 9781889101002. Available also from: https://books.google.cz/books?id=AuMzoz90j10C.

14. JUNSING, Tipparat. Interface Circuit for Three-Wire Resistance Temperature Detector with Lead Wire Resistance Compensation. In: *2019 Research, Invention, and Innovation Congress (RI2C).* 2019, pp. 1–4. Available from DOI: 10.1109/RI2C48728.2019.8999909.

15. KIM, Jikwang; KIM+, Jongsung; SHIN, Younghwa; YOON, Youngsoo. *A Study on the Fabrication of an RTD (Resistance Temperature Detector) by Using Pt Thin Film.* 2001. Available from DOI: 10.1007/BF02707199.

16. HORDESKI, M.F. *HVAC Control in the New Millennium.* Fairmont Press, 2001. ISBN 9780881733990. Available also from: https://books.google.cz/books?id=8m5IofzofNwC.

17. LASKAKIT. *MAX31865 RTD-to-Digital Converter* [online]. 2015 [visited on 2022-04-09]. Available from: https://www.laskakit.cz/user/related_files/max31865.pdf.

18. LASKAKIT. *PT100 temperature sensor, three-wire* [online]. 2022 [visited on 2022-04-09]. Available from: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/4916_pt100.jpg?6137b46c.

19. LASKAKIT. *MAX31865 converter* [online]. 2022 [visited on 2022-04-09]. Available from: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/4919-1_pt100-max.jpg?6137b46c.

20. MOSS, D.R. *Pressure Vessel Design Manual.* Elsevier Science, 2004. ISBN 9780080524122. Available also from: https://books.google.cz/books?id=1EvxhJDf3JgC.

21. HEUREKA. *AQUART 25/4/180* [online]. 2022 [visited on 2022-04-09]. Available from: https://im9.cz/iR/importprodukt-orig/583/583149b2251d17f8ab4e796186f4543b--mm2000x2000.jpg.

22. LEENS, Frederic. An introduction to I2C and SPI protocols. *IEEE Instrumentation Measurement Magazine.* 2009, vol. 12, no. 1, pp. 8–13. Available from DOI: 10.1109/MIM.2009.4762946.

23. SITRONIX. *ST7735* [online]. 2010 [visited on 2022-04-09]. Available from: https://www.laskakit.cz/user/related_files/st7735.pdf.

24. RELAY, Songle. *SUBMINATURE HIGH POWER RELAY* [online] [visited on 2022-04-09]. Available from: https://www.laskakit.cz/user/related_files/songle_relay_srd.pdf.

25. LASKAKIT. *4-chanel relay module 5VDC 250VAC 10A* [online]. 2022 [visited on 2022-04-09]. Available from: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/326-1_4-kanaly-rele-modul-5vdc-250vac-10a.jpg?61d95c9e.

26. LASKAKIT. *1.8" 128x160 TFT displej, ST7735, SPI* [online]. 2022 [visited on 2022-04-09]. Available from: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/1832_disp.jpg?6137b46c.

27. ADA, Lady. *Adafruit MAX31865 RTD PT100 or PT1000 Amplifier* [online]. 2016 [visited on 2022-04-24]. Available from: https://learn.adafruit.com/adafruit-max31865-rtd-pt100-amplifier/overview.

28. LASKAKIT. *LYONZG S-25-12 supply module 230V AC-DC 12V/2A 25W* [online]. 2022 [visited on 2022-05-01]. Available from: https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/5742-2_5742-2-lyonzg-s-25-12-modulovy-napajeci-230v-ac-dc-zdroj-12v-2a-25w.jpg?6137b46c.

Content of the attached medium

`readme.txt`	brief desription of the medium content
`src`	
`└── impl`	source code of the microcontroller porgram
`└── thesis`	source form of the thesis in LaTeX form
`text`	thesis text
`└── BP_Paramonov_Lev_2022.pdf`	text of the thesis in PDF