Introduction

Company VEIT decided to **make a coffee maker** for their customers named **VEIT One Special**.

You are a part of the VEIT team composed of HW architecture engineers, embedded systems firmware engineers and software engineers. The description bellow is common for the whole team. Decide, which parts suits you the best to help the team get the job done and show us, how would you make your part. You have a complete freedom to make any design decision you like, even to completely throw away the prescribed interface and make a better one.

Description

Write the software/make the hardware that controls a simple coffee maker VEIT One Special. Deliver not only the final code/schemes but the description of the reasons for your design and implementation decisions as well. The final solution needn't to be 100% complete, but needs to be as complete as you see fit.

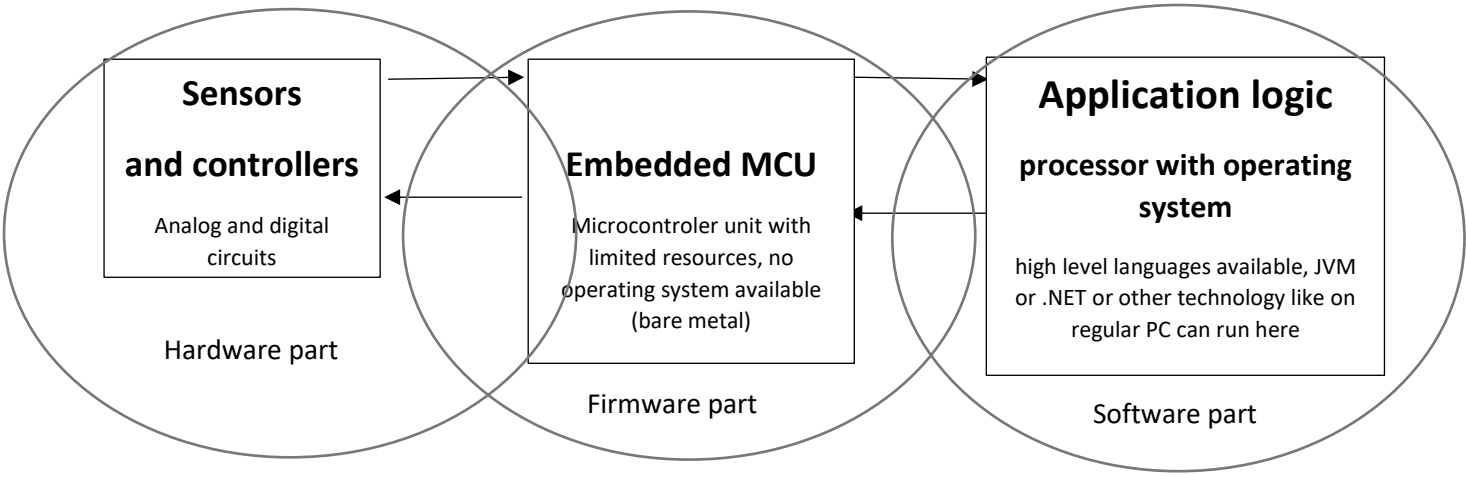
Proceed by these steps:

1. Read the detailed description of the tasks bellow. Do the parts you think you are capable of doing (the more the better, of course).
2. Estimate a time schedule and explain your estimations of how long
    a. it takes you to get the exercise done (when you'll send back the designed/implemented solution)
    b. it takes you to get the complete product done with all the parts it includes
    c. if those two estimations are different, explain why
3. Start working. For implementation/schemes select a language and technologies you see most suitable for you. Our preferables are described by each part
4. Deliver all the materials.

Detailed description

The „VEIT One Special" makes up to 12 cups of coffee at a time. The user places a filter in the filter holder, fills the filter with coffee grounds, and slides the filter holder into its receptacle. The user then pours up to 12 cups of water into the water strainer and presses the Brew button. The water is heated until boiling. The pressure of the evolving steam forces the water to be sprayed over the coffee grounds, and coffee drips through the filter into the pot. The pot is kept warm for extended periods by a heater plate, which turns on only if coffee is in the pot. If the pot is removed from the heater plate while water is being sprayed over the grounds, the flow of water is stopped so that brewed coffee does not spill on the heater plate. The following hardware needs to be monitored or controlled:

- The heating element for the boiler. It can be turned on or off.
- The heating element for the heater plate. It can be turned on or off.
- The sensor for the heater plate. It has three states: `heaterEmpty`, `potEmpty`, `potNotEmpty`.
- A sensor for the boiler, which determines whether water is present. It has two states: `boilerEmpty` or `boilerNotEmpty`.
- The Brew button. This momentary button starts the brewing cycle. It has an indicator that lights up when the brewing cycle is over and the coffee is ready.
- A pressure-relief valve that opens to reduce the pressure in the boiler. The drop in pressurestops the flow of water to the filter. The value can be opened or closed.



Obrázek 1 Basic description of the system parts

The decision to give the application logic to different machine (processor) then the MCU is unchangeable.

Hardware part

The firmware/software for the „VEIT One Special" has been designed and is currently under development. Firmware team has decided to use NXP Kinetis Cortex-M ARM microcontroler, concretely **MK60DN256VLQ10** and is allready implementing some functionality and is waiting for your design decisions to be able to continue with the development of this part of the firmware.

Your task is to select all the apropriate sensors to be used and design a scheme and PCB to connect the sensors to the MCU. You need to create the complete circuit, prototype it and test it. Create a series of tests to test fully the circuit according to conditions, which may occur in the system. Also you need to select the supplier for the parts of the final product – price of the solution does matter.

You're the one who decides, how should the programmers work with the sensors and you need to give them the necessary materials so they'll be able to program the MCU.

The programmers will listen to your advice according to changing anything in the system including the change of the selected MCU, but beware, they are allready working on the firmware, so don't do any design decisions, which may lead to throwing their job away if you don't need to do so. You have to have arguments for your decisions to argue with them.

Make the scheme and PCB preferably in Eagle or on the paper. Explain the decisions in some document, design the tests and explain how would you make them, deliver all the necesary materials including the comparison of the supliers/sensors you've considered.

Firmware part

The hardware/software for the „VEIT One Special" has been designed and is currently under development. Hardware and software teams designed their parts and they are waiting for your design decisions to be able to continue with the development.

Your task is to select an appropriate MCU (preferably some which is based on ARM Cortex-M) achitecture). Tell the posibilities to the hardware developers about where and how they can connect the sensors to the MCU and what functionality the sensors has to provide so they are able to work with your MCU.

You're the one who creates and desings the technology/protocol to communicate with the software part, you even need to create some library on the software part to be able for the software to communicate with the MCU. Tell the software guys how should they use your system. The example code of such an interface functions is shown below in the section **Common interface**, but feel free to change it as you wish.

Create the scheme on the paper or in Eagle. You don't need to proceed any further than at the edge of the MCU (pin mapping), the circuits are HW part. Design the program for the MCU and write it in ANSI C with the assuption, you may have some library tools to work with the basics of MCU (you don't need to set appropriate bits in appropriate registers to fully set the MCU). If you assume some library – you decide how this library would probably look like and you would be the one at the end who would write it.

Software part

The firmware/hardware for the „VEIT One Special" has been designed and is currently under development. The firmware/hardware engineers have even provided a low-level API, so you don't have to write any bittwiddling I/O driver code. The code for these interface functions is shown below in the section **Common interface**. If this code looks strange to you, keep in mind that it was written by hardware engineers. Hardware and firmware teams designed their parts and they are waiting for your design decisions to be able to continue with the development.

Make an aplication logic for the system.

Use a OO language for implementation, preferably C# and .NET platform, Java or C++. Other languages are also possible but if the one, who'll go through your code will not know them, then the design describtion is more valuable than the code itself.

Plus points

Show us, how would you proceed, if the VEIT One Special should be controlled via mobile device (phone, tablet, anything, let aside the value of such a solution). Base it the assumption, that the complete implementation of the machine, as described above, is done. Give us at least:

- Description, of how would you proceed
- Technological decisions
- Architectural decisions

Common interface

Description of the HW code in C#

```
namespace CoffeeMaker
{
	public enum HeaterPlateStatus
	{
		HEATER_EMPTY,
		POT_EMPTY,
		POT_NOT_EMPTY
	};
	public enum BoilerStatus
	{
		EMPTY,NOT_EMPTY
	};
	public enum BrewButtonStatus
	{
		PUSHED,NOT_PUSHED
	};
	public enum BoilerState
	{
		ON,OFF
	};
	public enum HeaterState
	{
		ON,OFF
	};
	public enum IndicatorState
	{
		ON,OFF
	};
	public enum ReliefValveState
	{
		OPEN, CLOSED
	};
	public static class CoffeeMakerAPI
	{
		/*
		* This function returns the status of the heater-plate
		* sensor. This sensor detects the presence of the pot
		* and whether it has coffee in it.
		*/
		[DllImport("lowapi.dll")]
		public static extern HeaterPlateStatus GetHeaterPlateStatus();

		/*
		* This function returns the status of the boiler switch.
		* The boiler switch is a float switch that detects if
		* there is more than 1/2 cup of water in the boiler.
		*/
		[DllImport("lowapi.dll")]
		public static extern BoilerStatus GetBoilerStatus();

		/*
		* This function returns the status of the brew button.
		* The brew button is a momentary switch that remembers
		* its state. Each call to this function returns the
		* remembered state and then resets that state to
		* NOT_PUSHED.
```

```
        *
        * Thus, even if this function is polled at a very slow
        * rate, it will still detect when the brew button is
        * pushed.
        */
        [DllImport("lowapi.dll")]
        public static extern BrewButtonStatus GetBrewButtonStatus();

        /*
        * This function turns the heating element in the boiler
        * on or off.
        */
        [DllImport("lowapi.dll")]
        public static extern void SetBoilerState(BoilerState s);

        /*
        * This function turns the heating element in the heater
        * plate on or off.
        */
        [DllImport("lowapi.dll")]
        public static extern void SetHeaterState(HeaterState s);

        /*
        * This function turns the indicator light on or off.
        * The indicator light should be turned on at the end
        * of the brewing cycle. It should be turned off when
        * the user presses the brew button.
        */
        [DllImport("lowapi.dll")]
        public static extern void SetIndicatorState(IndicatorState s);

        /*
        * This function opens and closes the pressure-relief
        * valve. When this valve is closed, steam pressure in
        * the boiler will force hot water to spray out over
        * the coffee filter. When the valve is open, the steam
        * in the boiler escapes into the environment, and the
        * water in the boiler will not spray out over the filter.
        */
        [DllImport("lowapi.dll")]
        public static extern void SetReliefValveState(ReliefValveState
        s);
    }
}
```