Assignee: Mehmet Hekimoğlu	**Kuartis HWE Team** Embedded Software Assignment	**Due Date**: 15.10.2021

Board Description:

You are given a hood appliance controller board whose microcontroller is atmega808 (32 pin TQFP package). The hood controller board drives a 4 speed motor via the relay elements which are controlled by the following pins of atmega808:

- Motor relay 1: PORTD6
- Motor relay 2: PORTD5
- Motor relay 3: PORTD4
- Motor relay 4: PORTD2

4 leds are available on the panel of the hood which are used as user interface elements. The led behavior is decided according to the motor speed, the presence of error within the device and a special (developer) mode that is initiated by a command sent by the USART. The led control pins are hooked to the following pins of atmega808:

- LED 1: PORTA0
- LED 2: PORTA2
- LED 3: PORTA1
- LED 4: PORTA5

The leds are connected to the MCU pins in a pull-up configuration. That is, you need to drive low to light on the leds. Furthermore, the panel of the hood has an IR receiver module that decodes the incoming 38kHz modulated IR data into a stream of 1's and 0's associated with the NEC protocol. This IR receive module is tied to PORTA3 of the atmega808 controller. The panel board is hooked to the hood controller board over the 10 pin IDC connector J8 as shown in Figure 1:

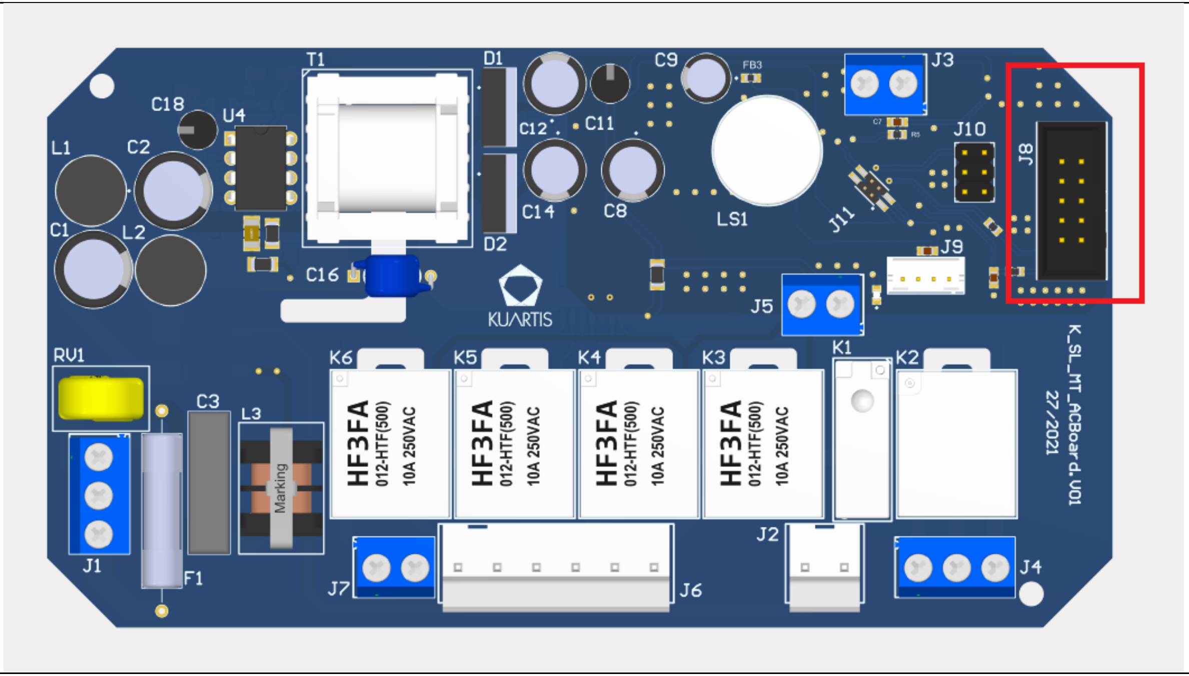

Figure 1: Hood controller top view: UI panel interface connection J8

Next, the hood controller board is required to be interfaced to an IoT controller over USART, where PORTC1 is used as the receive pin (IoT_TX_hood_RX) and PORTC0 is used as the transmit pin (IoT_RX_hood_TX). You can access the UART port via J9 (see Figure 2). The pin order is given with respect to the hood controller board.

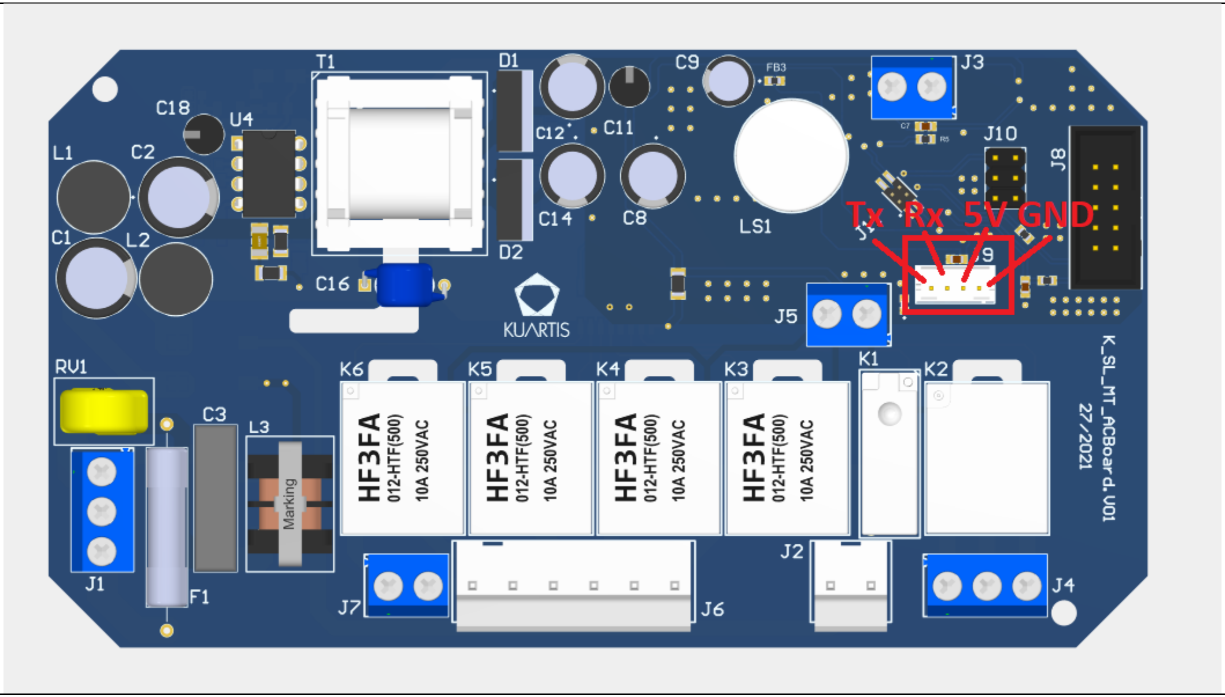

Figure 2: Hood controller board top layer. UART Port J9.

In addition, there is a buzzer and a led on the controller board. The buzzer is designated by LS1 and both can be seen in Figure 3. The led is used for debug purposes. But the buzzer is used as a UI element to signal various messages.

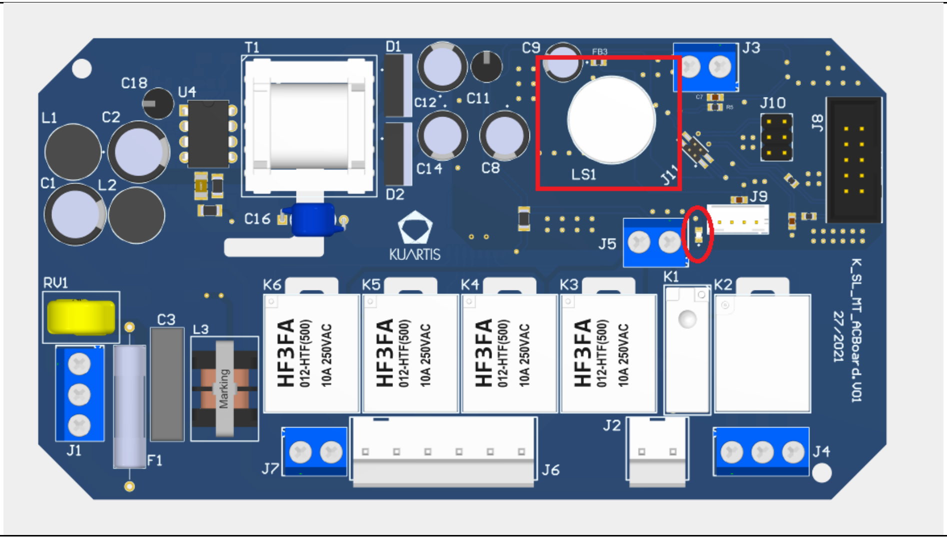

Figure 3: Hood controller top view. Buzzer and debug led.

Finally, there is a special light controller unit of this board that is controlled by two outputs: PORTD1, for powering on/off the light and PORTF4 for setting its brightness (by PWM) with 10 levels.

What you are required to do is:

1) Control the speed of the hood motor such that;

When speed 1 is selected, activate relay 1 only and turn on LED 1 only. When speed 2 is selected, activate relays 1 and 2 only and turn on Led 1&2. When speed 3 is selected, activate the relays 1, 2 and 3 and turn on leds 1,2 and 3 (I think you got the point :). But when speed 4 is selected activate all the relays and turn on leds 1, 2 and 3 and blink led 4 with period of 1 seconds. **This mode is named as the Boost mode** and there is a special behavior defined for this mode. When the Boost mode is activated, it runs for 15 minutes and after 15 minutes the speed is set to 3 automatically (also modifying the led behavior).

2) Implement an algorithm that decodes the messages of the remote IR transmitter. There are 4 codes+repeat code transmitted by this device:

Power: 129 (decimal)

+: 133 (decimal)

- : 138 (decimal)

Light: 153 (decimal)

These codes and their ones complements are transmitted after 2 bytes of address codes 129 and 102. You may need to use an oscilloscope to understand the packet structure for these commands. Furthermore, your algorithm is required to understand the repeat codes and how long a repeat code (for what command) is received.

You may find the following reference useful:
https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol

3) UI flow: Implement the following behaviour on the device.

When power button is pressed on the remote controller device:

If the hood was in off state, then the hood is turned on at speed 1. If the hood was already turned on, turn off the hood by turning off all the leds and motor speed relays.

When the + button is pressed and the hood was already turned on, increment the speed of the motor. Note that you can not go further than speed 4.

When the – button is pressed and the hood was already turned on, decrement the speed of the motor. Note that you can not go below speed 1 (you should not turn off the device when – is pressed at speed 1).

Light: Turns on the light if light was turned off. Turns off the light if the light was already turned on.

This part is a bit tricky:

+/- button behavior after light is turned on: If the lights are already turned on and after then if light button is held for 4 seconds at this mode, the device should enter the brightness adjustment mode. In brightness adjustment mode + button increments the brightness and – button decrements the brightness. The selected brightness level should be stored to the device by pressing the light button for 4 seconds. Otherwise, the previously set brightness level is restored by the device after 15 seconds.

Whenever a message is received by the remote controller a beep signal is generated by the buzzer. Work on for different tones to generate different sounds. The buzzer has a resonance frequency of 4kHz.

3) Special command sequence: If the power button is held for 4 seconds and after that the light button is held for 2 seconds the device should enter a special mode called the developer mode. In this mode the motor is disabled and the following led behaviour is observed on the panel: two consecutive leds will be active for 1 seconds and will switch to the next two consecutive leds and this will be swept for the entire 4 leds as shown below:

○●●○	1 sec	○○●●	1 sec	●○○●	1 sec	●●○○	1 sec

The developer mode is exited with the same command sequence on the remote controller.

4) Optional but recommended: USART Driver and component: As said before, the appliance controller MCU communicates with the IoT board over USART. You don't have the IoT controller board but you can simulate it by a computer software by hooking an FTDI USART-USB bridge board to the hood controller board (Note that you don't need to connect the 5V when you want to interface the FTDI bridge). The messaging structure sent by the IoT board (or your software) is as follows:

Packet description:

Byte Number	**Description**	**Value**
Byte 0	Header	0xAA
Byte 1	Command High byte	
Byte 2	Command Low byte	
Byte 3	Command argument length	Arg Length
Byte 4	Arg[0]	
Byte 3 + Arg Length	Arg**[**Arg Length-1]	
Byte 4 + Arg Length	Bitwise Checksum	

Bitwise checksum is the bitwise XOR of all the bytes transmitted in a packet.

You are asked to implement a USART driver that encodes and decodes the packed to transmit and packet received according to this communication scheme. If the received packet is not valid, then the driver should discard the packet. A command parser is also required as a component that signals the MCU that a specific command is received.

Assignee: Mehmet Hekimoğlu	**Kuartis HWE Team** Embedded Software Assignment	**Due Date**: 15.10.2021

The transmission of the packet should be done delayless without interrupting the transmission of any byte. If a significant delay is encountered between bytes, the driver should be aware of it and immediately ignore the incoming data knowing that the incoming packet is corrupt.

Some of the commands sent by the IoT board and associated arguments are listed below:

Command	**Command Value in Hex**	**Command Arguments**	**Command Argument Description**
Set Motor Speed	0x4332	0x01 – 0x04	Speed encoded in 1 to 4
Enable Motor	0xA3E2	0x00 or 0x01	Enable: 1, Disable 0
Enter Developer Mode	0xE310	0x55	Enters dev mode
Exit Developer Mode	0xA12E	0xAA	Exits dev mode
Get Motor speed	0x198C		
Get Motor status	0x28CB		

Having such commands received the appliance controller responds to the IoT board as follows:

Command Response	**Command Value in Hex**	**Command Arguments**	**Command Argument Description**
Set Motor Speed Response	0x4333	0x01 – 0x04	Speed encoded in 1 to 4
Enable Motor Response	0xA3E3	0x00 or 0x01	Enable: 1, Disable 0
Enter Developer Mode Response	0xE311	0x55	Entered dev mode
Exit Developer Mode Response	0xA12F	0xAA	Exited dev mode
Get Motor speed Response	0x198D	0x01 – 0x04	Speed encoded in 1 to 4
Get Motor status response	0x28CC	0x00 or 0x01	Enable: 1, Disable 0