Heater Simulation

Project Name: Heater Simulation with Atmega328P
Prepared by: Minh Nguyen
Issue Date: 28.04.2021
Version: 0.2
Distribution: Public

Version history:

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
0.1	Minh Nguyen	28.4.2021			Initial version
0.2	Minh Nguyen	29.04.2021		29.04.2021	Finalize report

Introduction

Heater Simulation is intended for a simple demonstration of how we could utilize multiple modules of the Atmega328P, in constructing a heating control system. The basic functionality of the heater is to allow users to either remotely (via USART Communication) or manually (with potentiometer) adjust the heating element output, in addition with the ON/OFF push button and External/internal mode toggle switch. The current version of the program has currently satisfied all of the desired functionalities, which was simulated with Proteus 8.

Hardware Schematic design

The Hardware design of Heater Simulation board includes sketching a general schematic and the drawing PCB layout. Since the purpose of this design is also to simulate with Proteus, there are two versions of the schematic: one with actual models of real components for PCB Layout design while the other with Proteus's simulation supported components for testing.

In general, the logical schematic design, as shown in Appendix A, is divided into multiple blocks, each has its own purpose. The actuators blocks "Toggle switch", "Tactile switch" and "USART pins" on the right comprise of the user interface of the system.

The "Toggle switch" is used to select the operating mode. If the switch is at position 2-1, the control of the Heating element is by reading the analogue value from ADC channel 2 (pin PC2), which is called "internal" mode. Otherwise, if the switch connect pin 2 and 3, the control of the Heating is performed via USART communication, which is called "external"

mode. The user could establish serial communication either from the 2-pin USART RX/TX or the 6-pin ICSP for SPI protocol.

Pushing the “Tactile switch”, which is designed with low-pass filter debounce, can toggle the sleep mode of the Atmega328P microprocessor. During sleep mode, all timers and interrupts are disabled except for external interrupt, which can trigger the system to wake up.

The Heating element (RC-circuit), located on the top right of the schematic, was designed based on the suggested model from the project’s instruction. connects with the OCA1 (PWM external hardware pin PB1) for setting the output voltage and ADC Channel 1 for reading its value.

The blocks on the left side are related to the AVR chip. The design is based on Schematic of Arduino UNO, which also use Atmeaga328P microprocessor. However, the additional block “Analog noise cancelling circuit” is added for the accuracy enhancement of ADC conversion. Lastly, the suggested design of the power circuit is included, although it wouldn’t be used in the Simulation of Proteus.

PCB Layout

The PCB, as shown in Appendix B, is created from Eurocard 2-layer template (red traces are on top copper while blue traces on the bottom copper) with the layout thickness, as demonstrated in the following figure.

ID	Name	Type	Material	Thickness	Dielectric	Power Plane
TR	Top Resist	Surface	Resist	10um	3.50	
TOP	Top Copper	Signal	Copper	18um		
		Core	FR4	1.55mm	4.80	
BOT	Bottom Copper	Signal	Copper	18um		
BR	Bottom Resist	Surface	Resist	10um	3.50	

Figure 1. Eurocard Layout description

Most of components (resistors, capacitors, inductors) are surface mount (SMD) which could be soldered on the PCB with the sketched traces and vias. Atmega328P, 1K Potentiometer, SPST switch, diode, 7805 power supply regulator, and pin headers, on the other hand, are through-hole components.

Components that are logically grouped in blocks of the schematic were placed close to each other on the Layout. In addition, most are labelled by their functions according to the schematic design, as illustrated in the Layout figure in Appendix B.

Software design

The program of the heating system was implemented using Microchip Studio, and its functions are based on the required functionalities of the project. The flow of main() function is summarized in Figure 2. The codes comprise of a C-source file and a header file, which follows with the programming C constructs, and are thoroughly commented.

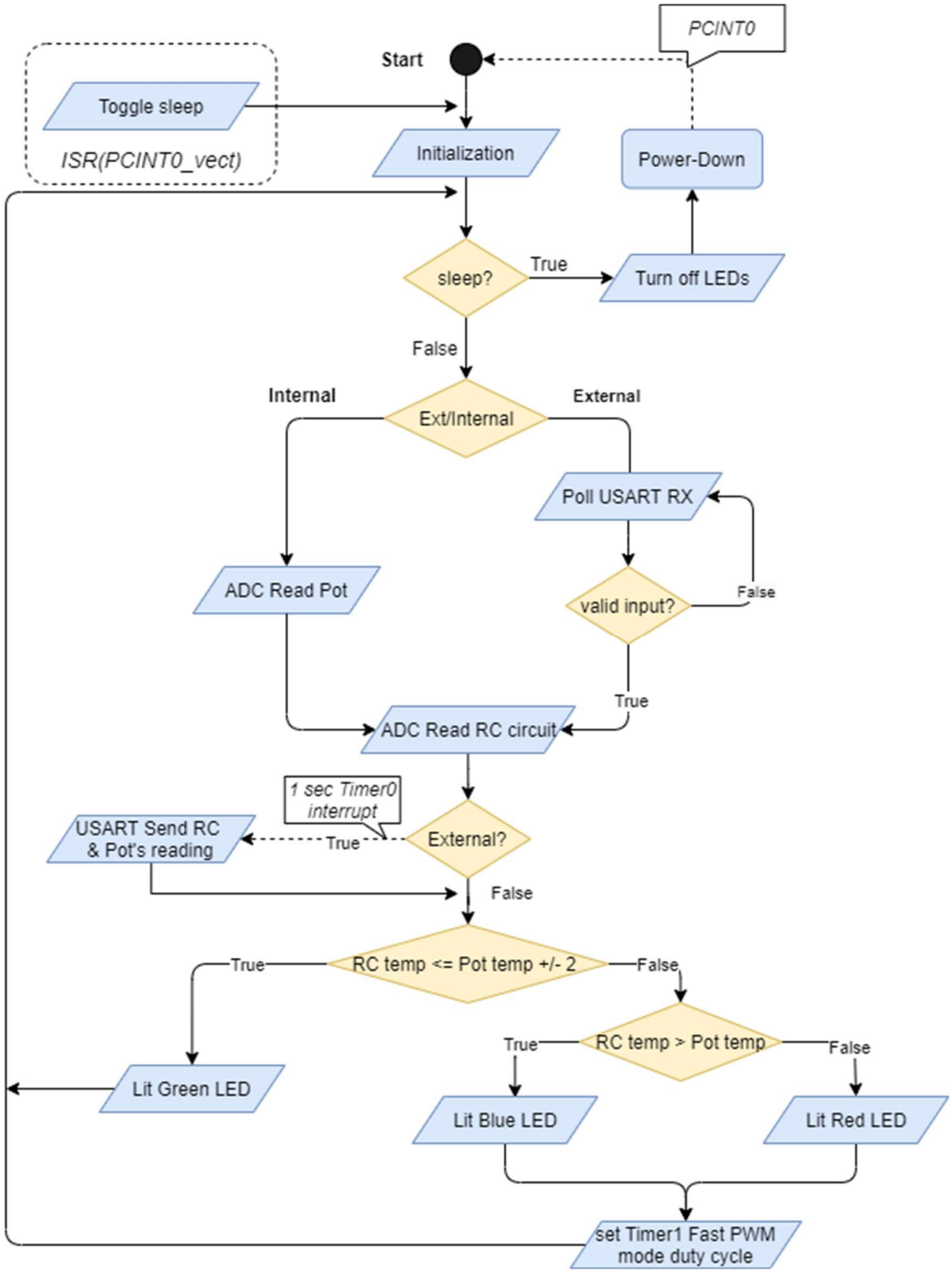
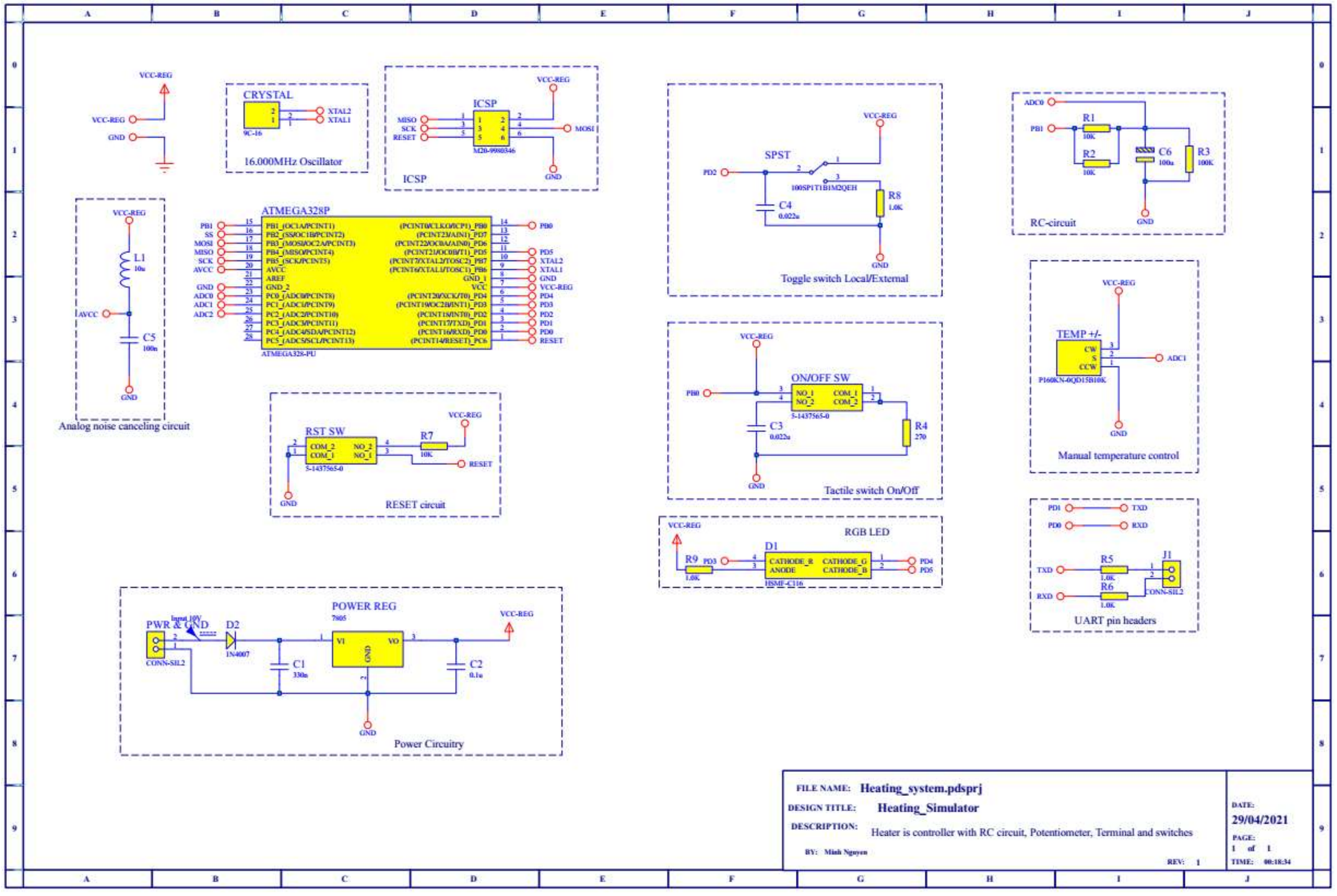


Figure 2. the main program loop

Appendix A: Schematic



Appendix B: Layout