Onboard 65W Type-C Charger Report

Ke-Cheng Wang, Hong-Rui Ran, Qing-Yang Ye, Group 1

INTRODUCTION

An Onboard charger is an item that we often see in our daily life and is closely related to our life. Onboard charger refers to a charger that is powered by a car battery (12V) and realizes 5/12/15/20V output through conversion. We chose this project at the time because it was significantly related to our life. For example, compared with the scientific data analysis of the python project, we thought it was more meaningful to choose an Onboard charger r project that had more to do with our daily life. While completing this project, although we encountered languages, foreign operations (e.g., welding), and hardware problems that other students did not engage in (the issued microcontroller was damaged initially), we all tried our best to overcome all difficulties and completed all the essential functions of the onboard charger. Below we introduce how to complete the project, the setbacks, and our diligence.

I. PART ONE: SOLDERING

Soldering is a wonderful experience, but since the quality of our welding torches is not particularly good, there are many twists and turns in the soldering process. In accordance with TA's requirements, we carry out the solder task step by step, and our solder technology is constantly improving. However, in the process of soldering, we also encountered many difficulties like soldering the component back, and tin oxidation caused the device to be unable to solder firmly. Even during the welding process, the smoke splashed during the cooling burned my hand due to the high temperature of the welding torch. However, with our unremitting efforts, we finally overcame many difficulties and successfully completed all the soldering tasks. Throughout the soldering process, we recognize many components such as diodes, resistors, capacitors, etc. Not only that, but we also learned about the connection of these components on the circuit board and the functions they embody in the soldering process. We are filled with a sense of accomplishment when we see the delicate boards that have been welded. In the whole process of welding, we have learned a lot of knowledge, and also understood the hard work of welders, and have a preliminary understanding and awe of the industry. Unfortunately, when we soldered the auxiliary circuit to complete the bonus, for some reason, the microcontroller did not work properly, so we removed the auxiliary circuit and did not complete the bond part of the soldering.

Main contributor: Ke-Cheng Wang, Qing-Yang Ye

II. PART TWO: ENVIRONMENT BUILDING

A. *Program Installation*

The MCU pins, clocks and ports of this project are configured in the STM32CubeMX software. The program must be installed before formal programming, and the chip STM32F103C8Tx used in this project is selected simultaneously.

Main contributor: Ke-Cheng Wang, Qing-Yang Ye，Hong-Rui Ran

B. *Construction of Environment Programming*

The programming environment of this project is Keil, and the language is C language. Before formal code writing and function implementation, we first configured the programming environment. We learned some basic knowledge of the C language in advance, such as variable types and declarations, loop structures, conditions statements, etc.

Main contributor: Ke-Cheng Wang, Qing-Yang Ye，Hong-Rui Ran

C. *MCU Programming and LED Lighting*

All functions of this project are implemented based on MCU burning, and each piece of written code needs to be imported into the MCU through the burning process. After the MCU is successfully burned, we try to write the most straightforward program to light up the LED lights and use this as the preparation for MCU programming.

During this process, we also encountered a hardware problem for the first time - the microcontroller could not be programmed. After the teaching assistant checked it, it was determined that the microcontroller was faulty, so we replaced the microcontroller. Finally, the programming and the lighting of the LED lights were successfully realized.

Code :

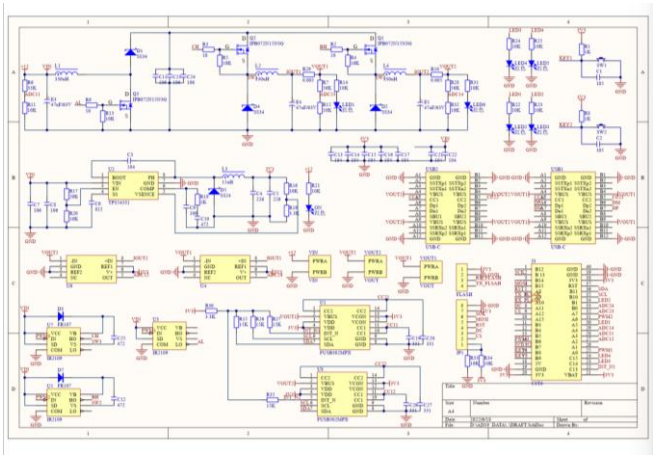
```
HAL_GPIO_WritePin(GPIOB,GPIO_PIN_12,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB,GPIO_PIN_12,GPIO_PIN_SET);
```

Main contributor: Ke-Cheng Wang, Qing-Yang Ye，Hong-Rui Ran

III. PART THREE: ACCOMPLISHMENT OF FUNCTIONS

Before formal code writing, programming, and function implementation, we need to have a basic understanding of the circuit diagram design of the project, as shown below;

FIGURE I. CIRCUIT DIAGRAM



The application of the circuit diagram is mainly reflected in the microcontroller's code programming and the pin parameters' configuration. The function corresponding to each pin is clearly marked on the diagram. The relevant parameter

configuration of the LED and switch should also be selected according to the design of the circuit diagram. You can also read the function of each part from the circuit diagram, another wiring method for programming, and the access method of the LCD screen.

A. PWM Output

Pulse-width modulation (PWM) is a very effective technique for controlling analog circuits using the digital output of a microprocessor. Widely used in many fields, from measurement and communication to power control and conversion. By changing the frequency and amplitude of the modulating wave, the frequency and amplitude of the output voltage of the inverter circuit can be adjusted. The basic constant current source and constant-voltage source need to use the duty cycle and frequency of PWM to achieve the function of numerical control. PWM output is actually an application of a timer. We need to configure the timer clock and select the clock source first.

The square wave output of PWM is one of the prerequisites for the success of this project. We need to detect the square wave output of 3.3V before doing the following work. The final open-loop control is also to control the output voltage by adjusting the duty cycle of the PWM.

We also encountered hardware problems in the single-chip microcomputer and circuit board in the process of modulating the square wave. Only one of the three-pin sets can reach 3.3V output, and the other two only have 1.7V output. The two teaching assistants used a multimeter to judge after testing. Because of the hardware failure, we replaced the microcontroller for the second time. Our square wave output had problems when the ADC was ready for the open-loop test. Since the microcontroller was removed from the circuit board, the square wave could be output normally, and the teaching assistant. After checking and repairing the faulty components of the circuit board, the correct voltage square wave output could not be obtained, and we were forced to replace the circuit board twice.

Code：

FIGURE II

```
HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1);
```

Main contributor: Ke-Cheng Wang, Qing-Yang Ye，Hong-Rui Ran

B. ADC Converter Accomplishment of 3.3V

The realization of the 3.3V ADC is to prepare for the open-loop control later and to check whether the LCD screen can display the output voltage generally in the process. In the implementation of ADC, we started with the simplest polling method to learn and try and finally realized the multi-channel acquisition of DMA.

Quality issues with the LCD screens slowed down our team during this process, and our testing time was limited due to the small number of LCD screens available, but we managed to achieve the correct display of the 3.3V ADC input voltage in the end.

Code:

FIGURE III & FIGURE IV

```
HAL_ADC_Start_DMA(&hadc1,(uint32_t*)&Value,99);

for(i=0,ad=0,ad1=0,ad2=0;i<60;)
  {
    ad=ad+Value[i++];
    ad1=ad1+Value[i++];
    ad2=ad2+Value[i++];
  }

  ad=ad*3.3/20/4096;
  ad1=ad1*3.3/20/4096;
  ad2=ad2*3.3/20/4096;
  sprintf(chFloat1_1,"%.3f",ad);
  sprintf(chFloat1_2,"%.3f",ad1);
  sprintf(chFloat1_3,"%.3f",ad2);
  ST7789_WriteString(50, 0, "Vin:", Font_11x18, RED, WHITE);
  ST7789_WriteString(50, 50, chFloat1_1, Font_11x18, RED, WHITE);
  ST7789_WriteString(50, 100, "Vout1:", Font_11x18, RED, WHITE);
  ST7789_WriteString(50, 150, chFloat1_2, Font_11x18, RED, WHITE);
  ST7789_WriteString(120, 0, "Vout2:", Font_11x18, RED, WHITE);
  ST7789_WriteString(120, 50, chFloat1_3, Font_11x18, RED, WHITE);
  ST7789_WriteString(120, 100, "DutyCycle:", Font_11x18, RED, WHITE);
  ST7789_WriteString(120, 150, DutyCycle, Font_11x18, RED, WHITE);
```

Main contributor: Ke-Cheng Wang, Qing-Yang Ye

C. Button Control of LEDs and Open-loop Control of Voltage(5V,12V,15V,20V)

We control the 4 LEDs using different modes of single-button control of the duty cycle to achieve the purpose of indicating the four output voltages (5V, 12V, 15V, 20V) with LEDs. In terms of implementation, we first choose One key press, which increases the duty cycle by 3/50 and starts from 1/50 to find the fitting curve of the load voltage and the voltage displayed by the ADC liquid crystal and the corresponding duty cycle. After fitting, we finally got four correct output voltages and the correct fit of the LCD.

The idea of writing the code is to use the five states of the button lights (four alternately on and all off) to program, starting from all off like the starting point, the four buttons represent four lights, respectively, and the fifth button restores all off. To prevent unstable lighting control caused by uneven button strength, we added a short delay to debounce and finally achieved the expected effect.

Code:

FIGURE VI & VII

```
key = 0;
pwmVal = 4;
while (1)
{
  if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_8)==0){
  HAL_Delay(50);
    if(key == 0){
      voltage = 3.0;
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,0);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1,0);
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_14,0);
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_15,0);
      key = 1;
      pwmVal = pwmVal + 3;
      __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, pwmVal);
    }
    else if(key == 1){
      voltage = 5.0;
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,1);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1,0);
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_14,0);
      HAL_GPIO_WritePin(GPIOC, GPIO_PIN_15,0);
      key = 2;
      pwmVal = pwmVal + 3;
      pwmVal1 = 4;
      __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, pwmVal1);
    }
```

```
else if(key == 2){
  voltage = 12.0;
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,0);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1,1);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_14,0);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_15,0);
  key = 3;
  pwmVal = pwmVal + 3;
  pwmVal2 = 28;
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, pwmVal2);
}
else if(key == 3){
  voltage = 15.0;
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,0);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1,0);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_14,1);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_15,0);
  key = 4;
  pwmVal = pwmVal + 3;
  pwmVal3 = 34;
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, pwmVal3);
}
else if(key == 4){
  voltage = 20.0;
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,0);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1,0);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_14,0);
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_15,1);
  key = 0;
  pwmVal = pwmVal + 3;
  pwmVal4 = 46;
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, pwmVal4);
}
```

Fit image:

FIGURE VIII & IX

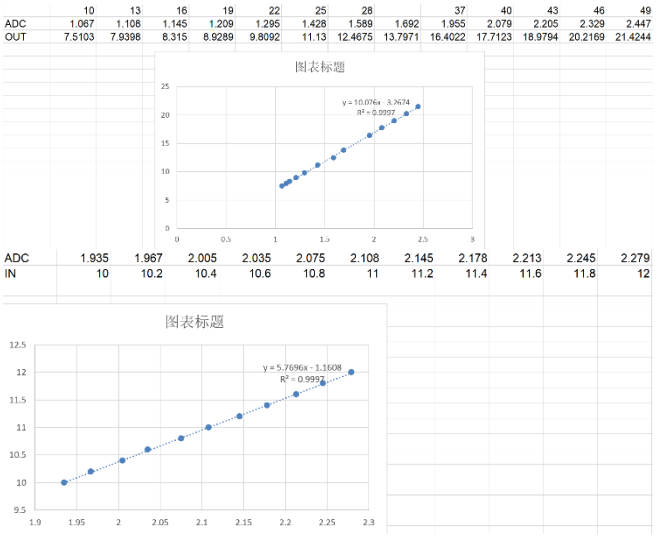


FIGURE X. DUTY CYCLE SELECTION

5	12	15	20
4	28	34	46

Main contributors: Qing-Yang Ye， Hong-Rui Ran

D. Voltage Protection

In order to ensure that the charger can work typically and will not cause safety problems because the input voltage is too large or too small, we have added a voltage protection design. When the actual input voltage is less than 8V and greater than 15V, the duty cycle of the buck converter is set to zero to ensure that the charger is working correctly and the circuit board is in good condition.

Code:

```
if ((ad*3.3/20/4096)*5.7696-1.1608 < 8.0){
    pwmVal1 = 0;
    pwmVal2 = 0;
    pwmVal3 = 0;
    pwmVal4 = 0;
  if (pwmVal<25){
    pwmVal = 0;
  }
}
if ((ad*3.3/20/4096)*5.7696-1.1608 > 15.0){
    pwmVal1 = 0;
    pwmVal2 = 0;
    pwmVal3 = 0;
    pwmVal4 = 0;
  if (pwmVal<25){
    pwmVal = 0;
  }
```

Principal contributors: Ke-Cheng Wang, Hong-Rui Ran

IV. PART FOUR: SUMMARY

In summary, we present all the essential functions of the onboard charger via PWM output, ADC converter, and open-loop voltage control with button control. During the project process, we went from the frustration caused by not being able to go home at the beginning to the enthusiasm for the project, a transformation we could not have imagined before. During the course of this project, we also encountered unimaginable setbacks. For example, welder mistakes and the inability of team members to gather outside lab hours to test performance due to the pandemic.

V. ACKNOWLEDGEMENT

The most profound and sincerest gratitude goes to Professor Hao-Yu Wang, TA Fan-Li Hu, and TA Qiu-Yu Li for their continuous and invaluable guidance throughout our project. It is a great honor and privilege that we were given the opportunity to work under their supervision. Without their help, we would not have been able to get through the entire project. Without their help, we would not have been able to get through the entire project.

Thank them again for all their incredible work, and we wish them the best of luck in their future endeavors.