2021 年全国大学生电子设计竞赛设计报告

竞赛选题：信号失真度测量装置（A 题）

基本信息

学校名称	西南大学		
参赛学生 1	廖子麒	Email	943973976@qq.com
参赛学生 2	陈玺	Email	2819371273@qq.com
参赛学生 3	陈宇	Email	cy1034429543@email.swu.edu.cn
指导教师 1	赵仲勇	Email	zhaozy1988@swu.edu.cn
指导教师 2		Email	
指导教师简介	赵仲勇，博士，副教授，重庆市巴渝学者青年学者(省部级人才计划)，中国电工技术学会青年工作委员会委员，重庆市电机工程学会青年和教育工作委员会委员，中国电工技术学会高级会员，IEEE Member，IEEE TII 等多个期刊的评审专家。2011、2017 年在重庆大学电气工程学院取得电气工程专业学士、博士学位，2015-2016 年在澳大利亚科廷大学进行联合培养，现在西南大学工程技术学院任教。长期从事电气设备绝缘故障状态检测与智能诊断、脉冲功率技术及其应用的研究工作。近年来主持国家自然科学基金、重庆市自然科学基金、重庆市留学回国人员创业创新支持计划等 6 项科研项目。授权中国发明专利 5 项，共计发表论文 30 余篇，其中 IEEE Trans. Industrial Electronics、IEEE Trans. Energy Conversion、IEEE Trans. Power Delivery 等 IEEE 汇刊论文 10 余篇。获 2017 年重庆市科学技术奖(技术发明类)1 项。		

信号失真度测量装置（A 题）

1. 摘要

为了保证电子信号的稳定传输，本报告设计了一种基于 FFT（Fast Fourier Transform，快速傅里叶变换）的信号失真度测量装置。此装置包括：物理采集与电源模块，采集、控制与计算模块、发送与显示模块三大模块构成。仿真结果表明：对自动生成的任意波形信号，基波频率为 1kHz-100kHz，幅值为 30mV-600mV，利用本装置中所提供的算法计算的 *THD*(total harmonic distortion，总谐波失真)与真实值 THD_o 的误差在 1%以内。实验结果：信号发生器产生的该信号，通过此装置的 THD_x 与真实值 THD_o 的误差在 3%以内，并且它的波形、频谱等指标能在显示器或者手机上成功显示。

2. 设计方案工作原理

2.1 预期实现目标定位

制作一个精确度较高的信号失真测量装置。通过物理采集和电源模块采集信号发生器产生的信号和提供稳定电压；在采集、控制与计算模块中，以 MSP432P401R 单片机作为主控芯片，通过芯片内 ADC 模块将物理采集模块收集到的信号转换为 14 位的数字型号，然后将数字信号通过数字滤波，加窗，然后利用 FFT 算法得到此信号的 THD_x 值；在显示与发送模块中，利用液晶屏屏幕显示计算值和波形图，并且本装置基于主控芯片的输出和 Echart 搭建前端网页，同时用户可以利用 WIFI 发送模块 ESP8266 使用手机浏览器访问主控芯片上的网页 IP 地址得到信号频谱图像与归一化频谱等数据。

2.2 技术方案分析比较

⑴ 失真度计算方案的选择

方案 1：采用基波抑制电路的模拟测量法。该方法的优点是无需采用任何复杂的算法，直接利用各个基本运算电路求得失真值；缺点是频率都是不固定的，二次谐波损耗大，频率特性不均匀，调谐范围窄，测量精度低和测量范围受限。

方案 2：采用频谱分析法（FFT 分析法）的数字测量法。该方法的优点能准确的测量出信号的总失真度值；缺点是实际的数据采集很难做到整周期采样，由此导致 FFT 分析发生频谱泄漏引入不可避免的方法误差。

方案 3：采用曲线拟合法。该方法的优点是不存在基于 FFT 的测量方法中的“栏栅效应”和“泄漏”问题，并且对采样序列的长度没有严格的限制；缺点是曲线拟合的好坏直接影响到实验结果，不确定性太大。

通过对比分析，方案 2 的方法精确度和稳定性最高，并且合理的采样率、采样点和窗函数的选择能够克服此方案的种种缺点。

⑵处理器的选择

方案 1：MSP432P401R 作为主控芯片。该主控芯片的优点是主频较高，内部资源丰富，运算能力强，IO 驱动能力较强，方便高效的开发环境，具有低功耗模式，并且具有较强的操控性；缺点是该芯片没有丰富的库函数。

方案 2：STM32 作为主控芯片。该主控芯片的优点是高性能、低功耗、低成本，具有丰富的库函数，对程序有较高的兼容性。

通过对比分析，方案 1 的主控芯片能保证微安级别的电流，保证电路低功耗运行，并且保证产品需要较强的操控性能。

⑶物理采集与电源的选择

方案 1：电压跟随器与二阶有源 RC 低通反相滤波电路。该方案的优点是共模电压较小；缺点是输入阻抗很小，不适合较大的放大倍数。

方案 2：电压跟随器与二有源阶 RC 低通同相滤波电路。该方案的优点是输入电阻较大；缺点是具有一定的共模电压干扰。

根据设计要求，方案 1 更适合本设计，因为电压跟随器克服了输入阻抗较小的缺点，根据本设计实际的截止频率与放大倍数要求二阶有源 RC 低通反相滤波电路也可行。此外 ADC 采样只能单极性采集和实验室原件限制，在此构建 LM317 模块电路稳压输出+2.5V 作为虚拟地。

⑷手机端显示方式的选择

根据所存在的实验器件，构建网页，之后利用手机通过 WIFI 传输模块访问主控芯片网页 IP 地址是目前存在的最优选择。

2.3 系统结构工作原理

⑴基于 FFT 的 *THD* 分析

FFT 是分析数字信号频谱一种的一种行至有效的方式。但是 FFT 常会存在“栅栏效应”和阶次截断产生的误差。通常会加窗进行运算。设定采样点位为 N，模数转换后的数字信号为 $f(N)$,窗函数为 $window(N)$,则经过窗函数处理之后的信号为：

$$f_w(N) = f(N) \times window(N) \tag{1}$$

对 $f_w(N)$进行 FFT 变换，则频谱 $H(k)$为:

$$H(k) = \sum_{n=0}^{N-1} f_W(n) \times e^{-j\frac{2\pi}{N}kn} \tag{2}$$

加窗会导致数字信号能量减少，通常经过 FFT 比较之后要乘以一个补偿系数 A,补偿后的频谱为 $H_c(k)$：

$$H_c(k) = A \times H(k) \tag{3}$$

(2) 系统结构设计实现方法

本系统分为三个模块，分别是物理采集与电源模块，采集、控制与计算模块与显示与发送模块。总模块图如图 1 所示，各部分模块的功能已在图中所示。

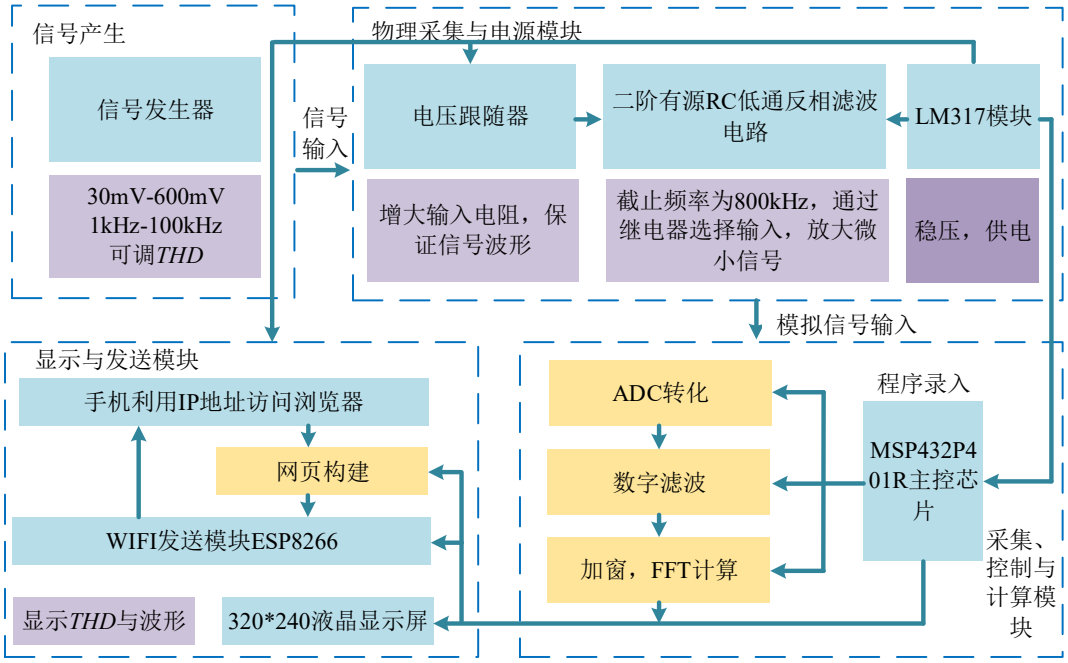


图 1. 系统模块图

2.4 功能指标的实现方法

假设信号为 $f(t)=U_{o1}\cos(wt+\varphi_1)+U_{o2}\cos(2wt+\varphi_2)+U_{o3}\cos(3wt+\varphi_3)+\ldots$，信号的真实 THD 为：

$$THD = \frac{\sqrt{U_{o2}^2 + U_{o3}^2 + \ldots}}{U_{o1}} \tag{4}$$

本装置的失真度测量采用近似方式，测量与分析输入信号的谐波成分限定只处理到 5 次谐波，THD_o 为：

$$THD_o = \frac{\sqrt{U_{o2}^2 + U_{o3}^2 + U_{o4}^2 + U_{o5}^2}}{U_{o1}} \tag{5}$$

若失真度测量值为 THD_x, 则失真度测量误差的绝对值为：

$$\Delta = | THD_x - THD_o | \times 100\% \tag{6}$$

Δ 越小，说明此装置性能越优越。另外基波与谐波的归一化幅值为 1、

(U_{m2}/U_{m1})、(U_{m3}/U_{m1})…

3. 核心部件电路设计

3.1 关键器件性能分析

核心控制电路：MSP432P401R，频率最高达 48MHz，32 位 CPU，能够快速单独完成 FFT 计算。芯片内部 A/D 转换为 14 位，满足信号频率(1kHz-100kHz)的采样要求。此外此芯片搭载 WIFI 传输模块，使用户可在 10s 内访问

物理采集和电源模块电路：在物理采集中，用了 OPA350 运算放大器，此原件的带宽增益为 38MTyp，而本设计的滤波电路最大的放大倍数为 10 倍，3.8MHz 远远大于 500kHz；LM317 模块，输入电压的 5V，能输出 0V(虚拟-2.5V)，2.5V(虚拟地)，5V(虚拟 2.5V)。拉升电压用以提供 A/D 采样单极性。

3.2 电路结构工作机理

(1)电源电路与稳压电路

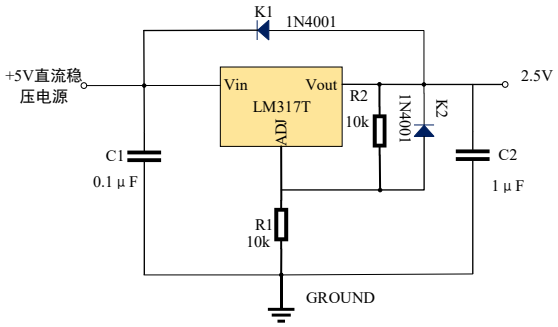


图 2. 稳压电路

外接+5V 的直流稳压电源,LM317T 构成的稳压电路为之后的所有原件提供+2.5V 的虚拟地(保证 AD 采样)。外接的+5V 直流电源，为此电路与后续原件提供电源。稳压电路如图 2 所示。

(2)物理采集电路

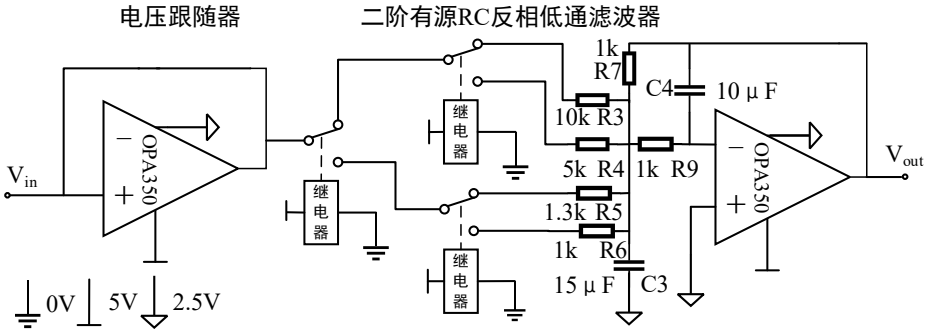


图 3. 物理采集电路

本设计的物理采集电路由电压跟随器和二阶有源 RC 反相低通滤波器构成。

电压跟随器提供无穷大的输入电阻。二阶有源 RC 反相低通滤波器，滤除信号中的高频信号，此电路的在 500kHz 的电压下降率不能超过 5%；此外，此电路利用继电器可控改变地改变了输入电阻，使放大倍数通过程序可调，为 1、2、6 和 10 倍，合理放大不同幅值下的信号。物理采集电路的电路图如图 3 所示。

3.3 电路和程序设计仿真

(1)物理采集模块的仿真

采用 Multisim 对图 3 的电路进行仿真，采用一倍的放大倍数得到其频谱仿真图如图 4 所示。可知设计的电路满足设计要求，并对高频信号进行抑制。

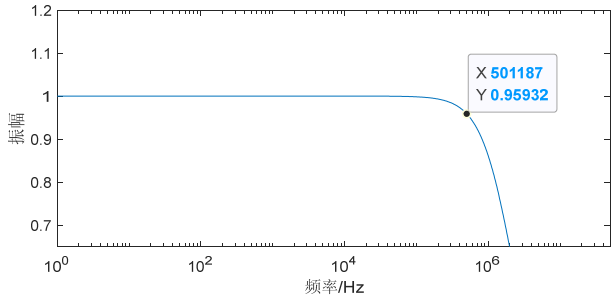


图 4. 频谱仿真图

4. 系统软件设计分析

4.1 基于 FFT 的 *THD* 测量中的窗函数、采样频率和采样点的选择与仿真

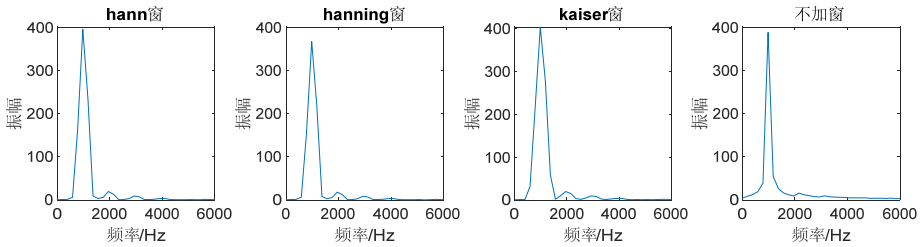


图 5. 加窗后的频谱图

在 Matalb 环境下，采取了 3 种不同的窗函数和不加窗函数的程序对已知信号 $f(t)=400\cos(1000\times2\pi t)+20\cos(2000\times2\pi t)+10\cos(3000\times2\pi t)+5\cos(4000\times2\pi t)+2.5\cos(5000\times2\pi t)+0.05\times400\text{rand}(0,1)$进行处理，在此用 5%基频幅值随机取样模拟噪声叠加。信号的 THD_0 为 5.766%，然后对此信号进行 FFT 变换，采样频率为 200kHz，采样点为 1024，频谱如图 5 所示。han 窗、hanning 窗、kaiser 窗与不加窗的平均 Δ 如表 1 所示。由表 1 可知 kaiser 窗优于其他窗函数。此窗函数的补偿系数 A 为 2.85，此函数表达式较为复杂，因此不在此赘述。之后对 kaiser 的窗函数的 b 进行选择，如表 2 所示。由表 2 可知 b 为 8 时 kaiser 窗的效果为最优。此外本设计还对采样频率和采样点给出了建议：在考虑主控芯片的资源和计算速

度的情况下，选择 1024 点，频率为 1kHz 采用 10kHz 的采样率，频率为 1kHz-5kHz 采用 50kHz 的采样率，频率为 5kHz-10kHz 采用 100kHz 的采样率，频率为 10kHz-50kHz 采用 500kHz 的采样率，频率为 50kHz-100kHz 采用 1MHz 的采样率。在这些些见一下，仿真的平均 $\Delta < 1\%$ 。

表 1. 不同窗函数下的 *THDo* 值

次数	hann	hanning	kaiser	不加窗
1	4.1184	5.233	5.3229	4.1184
2	5.7913	5.7898	5.8952	4.5265
3	5.3813	5.38	5.4435	4.2522
4	5.2139	5.2127	5.2699	4.2092
平均	5.1262	5.4038	5.4828	4.2765
Δ	0.6075	0.3298	0.2508	1.4571

表 2. 不同 b 下 kaise 窗的 *THDo* 值

次数	6	7	8	9	10	100
1	5.4323	5.1295	5.2057	5.3263	6.006	5.321
2	5.1682	5.3948	5.5726	5.4039	5.5178	5.5054
3	5.1568	5.1645	5.6363	5.7912	5.2001	5.4799
4	5.0476	5.5254	5.2452	5.8363	5.3895	5.7324
平均	5.2012	5.3035	5.4149	5.5894	5.5283	5.5096
Δ	0.5324	0.4301	0.3187	0.1442	0.2053	0.2240

4.2 主要模块程序设计

系统的总工作流程图，如图 1 所示。在此不再赘述。系统的程序模块流程图如图 6 所示。系统 HTML 通讯流程图如图 7 所示。

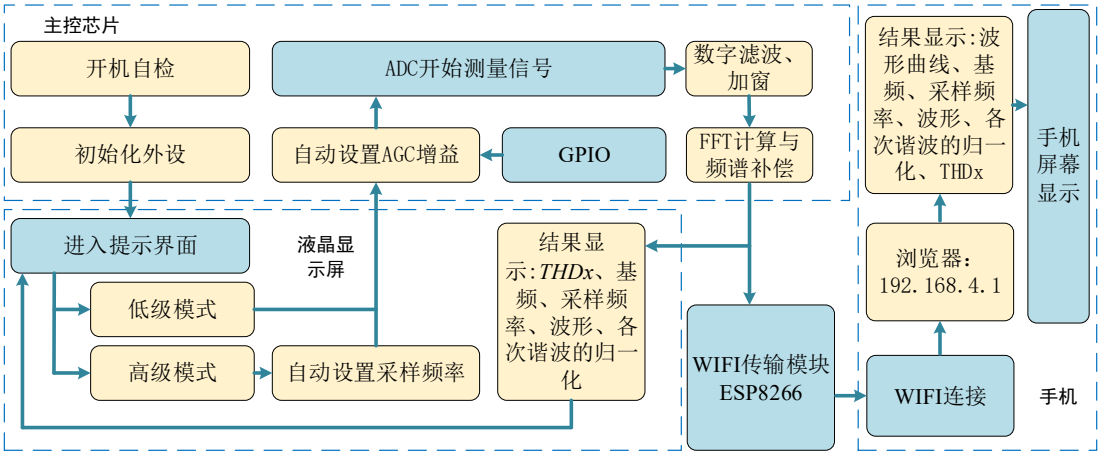


图 6. 系统程序模块设计图

4.3 关键模块程序清单

见附录。

5. 竞赛工作环境条件

(1) 设计分析软件环境：Matalb、Keil、Multisim、Visual Studio。

(2) 仪器设备硬件平台：Tektronix MDO4104C 示波器、Tektronix AFG3102C 信号发生器。

6. 作品成效总结分析

6.1 系统测试性能指标

(1) 基本要求

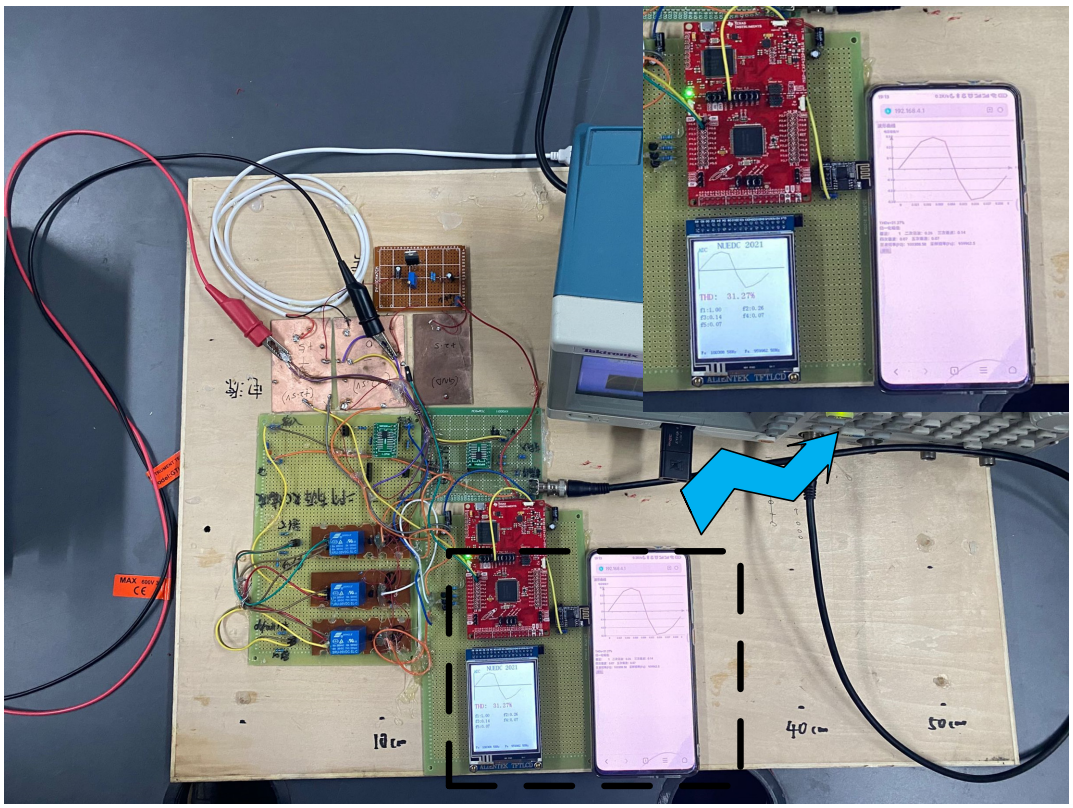


图 7. 现场实验图

实验现场图，如图 7 所示。输入信号的峰峰值电压范围：**电压为 300mV~600mV；波形为任意波形；输入信号基频：1kHz，*THD*：5%~50%**。本产品的测量参数如表 3 所示，**从表中数据可知** $\Delta < 5\%$ **。本产品在基本要求的计算时间在 1s 以内且可显示 THD_x。综上，满足所有基本要求。**

表 3. 不同峰峰值范围和 THD_x 下的 Δ 值

	300mV	400mV	500mV	600mV
5%	1.23%	0.41%	0.87%	0.24%
10%	2.32%	0.24%	0.47%	2.47%
20%	1.23%	1.57%	1.87%	1.78%
30%	2.57%	2.31%	0.25%	0.24%
40%	2.14%	2.87%	2.17%	2.22%
50%	2.34%	3.25%	4.21%	2.57%

(2) 发挥部分

表 3. 不同频率下和峰峰值下的Δ值

	30mV	80mV	100mV	300mV	600mV
1kHz	0.25%	0.58%	0.11%	0.21%	1.11%
10kHz	1.52%	1.25%	2.11%	2.01%	2.78%
25kHz	1.09%	2.58%	2.45%	2.08%	2.04%
50kHz	1.78%	2.78%	2.47%	2.17%	2.01%
75kHz	1.58%	2.22%	2.18%	2.07%	1.25%
100kHz	2.41%	2.57%	2.57%	2.01%	1.58%

输入信号的峰峰值电压范围：**30mV~600mV，输入信号基频：1kHz~100kHz。为对比实验方便，设置固定 THD_o 为 10%，**不同频率下和峰峰值下的Δ值如表 4 所示。**本产品在发挥要求的计算时间皆在 2s 以内。其中显示屏和手机显示如图 7 所示，都完成了显示输入信号的一个周期波形；显示输入信号基波与谐波的归一化幅值，只显示到 5 次谐波；在手机上显示测量装置测得并显示的输入信号 THD_x 值、一个周期波形、 基波与谐波的归一化幅值。**综上，**满足除其他外所有发挥部分要求。**

6.2 创新特色总结展望

(1)精确度较高，本装置基本可满足 30mV~600mV，基频为 1kHz~100kHz 的任意波形周期信号Δ＜3%。

(2)本装置自动选择放大倍数对电压较小信号进行放大，并对高频噪声进行抑制。

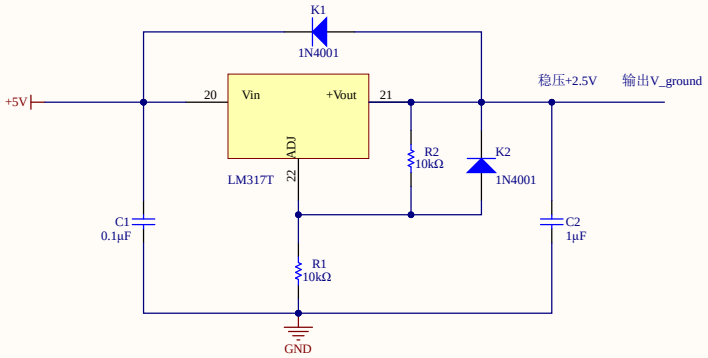
(3)拥有较优秀的可视化界面，手机屏幕和液晶显示屏可对信号的基本数据如波形、采样频率、5 次以下谐波归一化值、THD_x 等数据进行显示。S’s

7.参考资料及文献

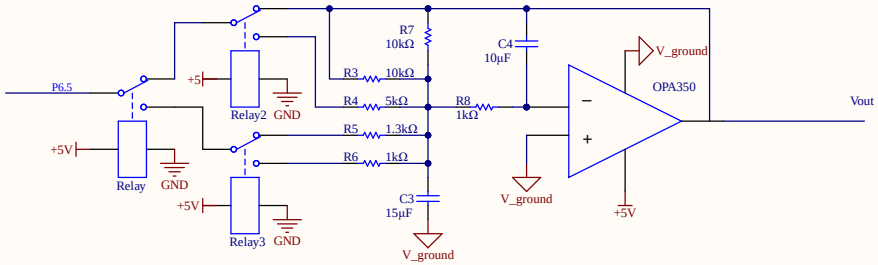
[1]吴礼福,吴佳伟,田朋溢.基于 STM32 和 LoRa 技术的噪声采集分析系统[J].现代电子技术,2021,44(21):30-34.

[2]杜太行,梁倩伟,孙曙光,王景芹.基于加窗插值压缩感知的谐波/间谐波检测方法[J/OL].电测与仪表:1-6[2021-11-07].

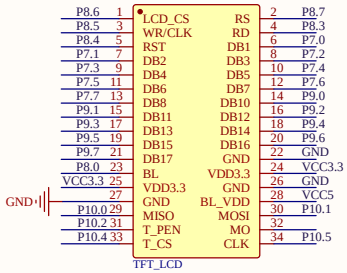
[3]张文宇,谭保华,黄程旭,何嘉奇,郑焙天,杨桂梅,黄新蕊,朱硕.基于全相位 FFT 三谱线校正的电网谐波与间谐波检测算法[J/OL].华中师范大学学报(自然科学版):1-8.



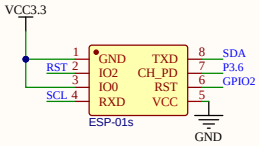
稳压供电模块



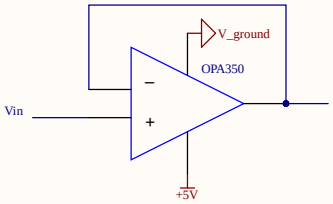
增益档位选择　　二阶有源RC低通滤波器



LCD模块



WIFI模块

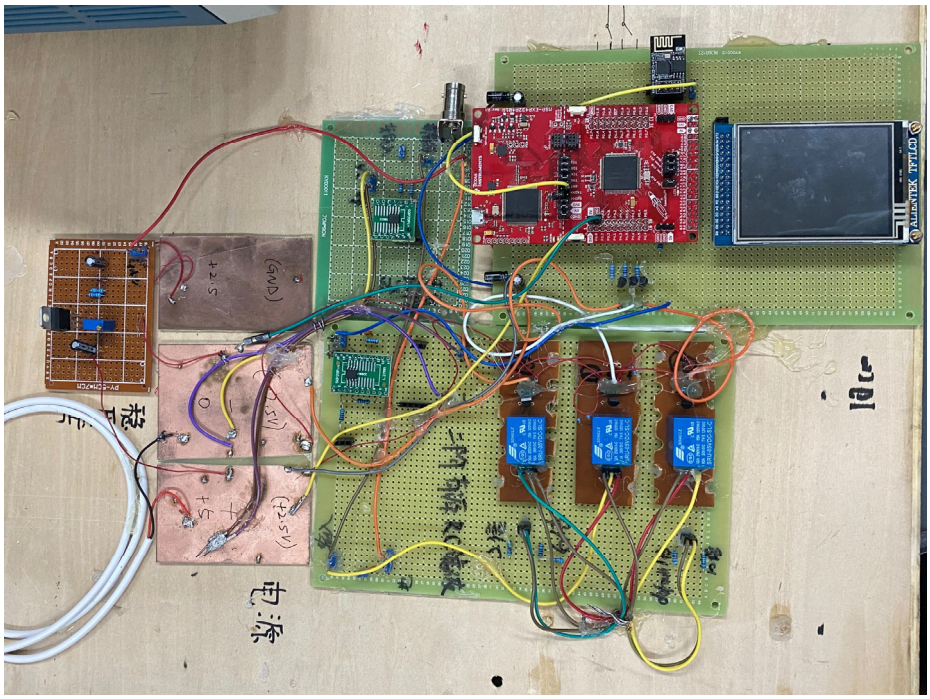


电压跟随器

照片


2021
TI杯全国大学生
电子设计竞赛
(重庆赛区)
TI Cup National Undergraduate
Electronic Design Contest

作品



附录（三） 主要程序清单

adc.c

```
#include "adc.h"
#include "tim.h"
#include <stdio.h>
#include "mygpio.h"

extern uint8_t main_adc_mode;
extern uint8_t main_page;

uint8_t adc_flag=0;
uint32_t adc_count=0;
volatile float resultsBuffer[1024];

static uint16_t sample_N=1024;
static uint32_t sample_T = 0;
static float sample_Rate = 0.0;

void adc_set_N(uint16_t __N)
{
    sample_N = __N;
}

float adc_get_fs(void)
{
    sample_T = (0xffffffffful - Timer32_getValue(TIMER32_0_BASE));

    sample_Rate = (double)sample_N * 48000000.0 / sample_T;

    tim_32_Config();

    return sample_Rate;
}
```

dma.c

```
/* DriverLib Includes */
#include  <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdbool.h>
#include "tim.h"
#include "mygpio.h"
#include "stdio.h"
#include "delay.h"

#define ARRAY_LENGTH 1024
extern uint8_t main_adc_mode;
/* DMA Control Table */
#if defined(__TI_COMPILER_VERSION__)
#pragma DATA_ALIGN(MSP_EXP432P401RLP_DMAControlTable, 1024)
#elif defined(__IAR_SYSTEMS_ICC__)
#pragma data_alignment=1024
#elif defined(__GNUC__)
__attribute__ ((aligned (1024)))
#elif defined(__CC_ARM)
__align(1024)
#endif
static DMA_ControlTable MSP_EXP432P401RLP_DMAControlTable[16];
extern uint8_t adc_flag;
extern volatile uint16_t InputADC[1024];
volatile uint16_t dummy[1024];

void dma_init(void)
```

```
{
    /* Initializing ADC (SMCLK/1/1) */
    MAP_ADC14_enableModule();
    MAP_ADC14_initModule(ADC_CLOCKSOURCE_ADCOSC, ADC_PREDIVIDER_1, ADC_DIVIDER_1,0);
    /*
     * Configuring GPIOs (5.5 A0)
     */
    MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P5, GPIO_PIN5,
    GPIO_TERTIARY_MODULE_FUNCTION);
    /*
     * Configuring ADC Memory, repeat-single-channel, A0
     */
    MAP_ADC14_configureSingleSampleMode(ADC_MEM0, true);
        ADC14_setPowerMode(ADC_UNRESTRICTED_POWER_MODE);
        Interrupt_enableInterrupt(INT_ADC14); //ADC 模块的中断
    /*
     * Configuring ADC Memory, reference, and single ended conversion
     * A0 goes to mem0, AVcc is the reference, and the conversion is
     * single-ended
     */
    MAP_ADC14_configureConversionMemory(ADC_MEM0, ADC_VREFPOS_AVCC_VREFNEG_VSS,
    ADC_INPUT_A0, false);
    /*
     * Configuring the sample trigger to be sourced from Timer_A0 CCR1 and on the
     * rising edge, default samplemode is extended (SHP=0)
     */
    MAP_ADC14_setSampleHoldTrigger(ADC_TRIGGER_SOURCE1, false);
    /* Enabling the interrupt when a conversion on channel 1 is complete and
     * enabling conversions */
    MAP_ADC14_enableConversion();
    /* Configuring DMA module */
    MAP_DMA_enableModule();
    MAP_DMA_setControlBase(MSP_EXP432P401RLP_DMAControlTable);
    /*
     * Setup the DMA + ADC14 interface
     */
    MAP_DMA_disableChannelAttribute(DMA_CH7_ADC14,
                                    UDMA_ATTR_ALTSELECT | UDMA_ATTR_USEBURST |
                                    UDMA_ATTR_HIGH_PRIORITY |
                                    UDMA_ATTR_REQMASK);
    /*
     * Setting Control Indexes. In this case we will set the source of the
     * DMA transfer to ADC14 Memory 0 and the destination to the destination
     * data array.
     */
    MAP_DMA_setChannelControl(UDMA_PRI_SELECT | DMA_CH7_ADC14,
        UDMA_SIZE_16 | UDMA_SRC_INC_NONE | UDMA_DST_INC_16 | UDMA_ARB_1);

    MAP_DMA_setChannelTransfer(UDMA_PRI_SELECT | DMA_CH7_ADC14,
        UDMA_MODE_PINGPONG, (void*) &ADC14->MEM[0],
            (void*)InputADC, ARRAY_LENGTH);

    MAP_DMA_setChannelControl(UDMA_ALT_SELECT | DMA_CH7_ADC14,
        UDMA_SIZE_16 | UDMA_SRC_INC_NONE | UDMA_DST_INC_16 | UDMA_ARB_1);

    MAP_DMA_setChannelTransfer(UDMA_ALT_SELECT | DMA_CH7_ADC14,
        UDMA_MODE_PINGPONG, (void*) &ADC14->MEM[0],
            (void*)dummy, ARRAY_LENGTH);
    /* Assigning/Enabling Interrupts */
    MAP_DMA_assignInterrupt(DMA_INT1, 7);
    MAP_DMA_assignChannel(DMA_CH7_ADC14);
    MAP_DMA_clearInterruptFlag(7);

    /* Enabling Interrupts */
    MAP_Interrupt_enableInterrupt(INT_DMA_INT1);
    Interrupt_enableMaster();
}
```

```
/* Completion interrupt for ADC14 MEM0 */
void DMA_INT1_IRQHandler(void)
{
        //MAP_Timer_A_stopTimer(TIMER_A0_BASE);
        uint16_t i;
    uint32_t status = DMA_getInterruptStatus();
    DMA_clearInterruptFlag(status);

        ADC_TIMER_STOP;
    MAP_DMA_disableChannel(7);
    /*
     * Switch between primary and alternate bufferes with DMA's PingPong mode
     */
    if (MAP_DMA_getChannelAttribute(7) & UDMA_ATTR_ALTSELECT)
    {
    MSP_EXP432P401RLP_DMAControlTable[7].control =
            (MSP_EXP432P401RLP_DMAControlTable[7].control & 0xff000000 ) |
                (((ARRAY_LENGTH)-1)<<4) | 0x03;
    }
    else
    {
    MSP_EXP432P401RLP_DMAControlTable[15].control =
            (MSP_EXP432P401RLP_DMAControlTable[15].control & 0xff000000 ) |
                (((ARRAY_LENGTH)-1)<<4) | 0x03;
    }
    adc_flag = 1;
    led_b(0);
}

void dma_start(void)
{
    MAP_DMA_enableChannel(7);
    ADC_TIMER_START;
    delay_ms(100);
}
```

fft.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include "fft.h"
#include "mygpio.h"
#include "tim.h"

#define PI 3.1415926535

extern float main_thd;
extern float main_base;
extern float main_f1;
extern float main_f2;
extern float main_f3;
extern float main_f4;
extern float main_f5;
extern uint8_t main_adc_mode;
extern uint8_t main_page;

extern uint8_t main_point_N;
extern float main_point_data[40];
extern float main_point_dt;
extern volatile float resultsBuffer[1024];
extern volatile uint16_t InputADC[1024];
//复数 FFT
void FFT(float dataR[],float dataI[],float dataA[],int N,int M)
{
```

```
    int i,j,k,r;
    int p,L,B;
    unsigned int I,J;
    float Tr,Ti,temp;
    for(I=0;I< N;I++)
    {
        /*反序*/
        J = 0;
        for(j = 0; j < M; j++)
        {
            J |= ((I & (1 << j)) << (M - 1 - j)) >> j;
        }

        if(I<J)
        {
            temp = dataR[I];
            dataR[I] = dataR[J];
            dataR[J] = temp;
            temp = dataI[I];
            dataI[I] = dataI[J];
            dataI[J] = temp;
        }
    }

//FFT
    for(L=1; L<=M;L++)   //FFT 蝶形级数 L 从 1--M
    {
        /*第 L 级的运算*/
        B = (int)(1<<(L-1));
        for(j=0;j<=B-1;j++)
        {   /*同种蝶形运算*/
            k = (int)(1<<(M-L));
            //计算旋转指数 p
            p=j*k;
            for(i=0; i<=k-1;i++)
                {
                    //数组下标定为 r
                    r=j+2*B*i;
                    Tr=dataR[r+B]*cos(2.0*PI*p/N) + dataI[r+B]*sin(2.0*PI*p/N);
                    Ti=dataI[r+B]*cos(2.0*PI*p/N) - dataR[r+B]*sin(2.0*PI*p/N);
                    dataR[r+B]=dataR[r]-Tr;
                    dataI[r+B]=dataI[r]-Ti;
                    dataR[r]=dataR[r]+Tr;
                    dataI[r]=dataI[r]+Ti;
                }
            }
    }
    //幅值
    for ( i=0;i<N;i++ )
    {
        dataA[i]=sqrt(dataR[i]*dataR[i]+dataI[i]*dataI[i]);
    }
}

//实数 FFT
void FFTR(float dataR[],float dataI[],float dataA[],int N,int M)
{
    int i;
    float Tr,Ti;
    float x1R[N/2],x2R[N/2],x1I[N/2],x2I[N/2],xR[N],xI[N];
    for(i=0;i<N/2;i++)
    {
        dataI[i]=dataR[2*i+1];
        dataR[i]=dataR[2*i];
    }

    FFT(dataR,dataI,dataA,N/2,M-1);
    //求 X1(k)和 X2(k)
```

```
    for(i=0;i<N/2;i++)
    {
        if(i==0)
        {
            x1R[i]=dataR[i];
            x1I[i]=dataI[i];
            x2R[i]=dataI[i];
            x2I[i]=-dataR[i];
        }
        else
        {
            x1R[i]=(dataR[i]+dataR[N/2-i])/2;
            x1I[i]=(dataI[i]-dataI[N/2-i])/2;
            x2R[i]=(dataI[i]+dataI[N/2-i])/2;
            x2I[i]=(dataR[N/2-i]-dataR[i])/2;
        }
    }

    //第 M 级的蝶形运算
    for(i=0;i<=N/2-1;i++)
    {
        Tr=x2R[i]*cos(2.0*PI*i/N) + x2I[i]*sin(2.0*PI*i/N);
        Ti=x2I[i]*cos(2.0*PI*i/N) - x2R[i]*sin(2.0*PI*i/N);
        xR[i]=x1R[i]+Tr;
        xI[i]=x1I[i]+Ti;

        if(i==0)
        {
            xR[N/2]=x1R[0]-x2R[0];
            xI[N/2]=x1I[0]-x2I[0];
        }
        else
        {
            xR[N-i]=xR[i];
            xI[N-i]=-xI[i];
        }
    }

    //复数输出
    for(i=0;i<N;i++)
    {
        dataR[i]=xR[i];
        dataI[i]=xI[i];
    }
    // 幅值
    for ( i=0;i<N;i++ )
    {
        dataA[i]=sqrt(dataR[i]*dataR[i]+dataI[i]*dataI[i]);
    }
}

//fs = 20kHz
//f = 20000/2048*N
float SinInI[1024],SinInA[1024];


int fft_window(float *adc_f, float fs)
{
    uint16_t i,j,k;
    float sum, avrg;

    /*find DC part*/
    k=1;
    for (i=0;i<1024;i+=32)
    {
        sum = 0;
        for (j=0;j<32;j++)
        {
```

```
            sum += adc_f[i+j];
        }
        sum /= 32;
        avrg += sum;
        k++;
    }
    avrg /= k;
    /*print adc*/
    for (i=0;i<1024;i++)
    {
        adc_f[i] = (adc_f[i]-avrg);
    }
    /*kaiser windows & DC  filt*/
    for (i=0;i<1024;i++)
    {
        adc_f[i] = cazer_win_1024[i]*(adc_f[i]);
    }
    return 0;
}

uint8_t fft_test_alx(float fs)
{
    uint16_t i,j;
    float thd, peak, f0;
    unsigned short base=0, hmny2=0, hmny3=0, hmny4=0, hmny5=0, buf;
    float sum,avrg;
    /*abs -> amptitude*/
    for (i=0;i<1024;i++)
    {
        SinInA[i] = SinInA[i] * 2.48 * 2 / 1024;
    }
    SinInA[0] = SinInA[0] / 2;

    /*find base frequent*/
    peak = SinInA[10];
    for (i=20;i<512;i++)
    {
        if (peak<SinInA[i])
        {
            peak = SinInA[i];
            base = i;
        }
    }
    f0 = base/1024.0*fs;

    switch(main_adc_mode)
    {
        case ADC_10K:
        {
            break;
        }
        case ADC_50K:
        {
            if (f0 <= 1000)
            {
                main_adc_mode = ADC_10K;
                main_page = 7; //restart
                return 1;
            }
            break;
        }        case ADC_100K:
        {
            if (f0 <= 5000)
            {
                main_adc_mode = ADC_50K;
                main_page = 7; //restart
                return 2;
            }
```

```
            break;
        }
        case ADC_500K:
        {
            if (f0 <= 10000)
            {
                main_adc_mode = ADC_100K;
                main_page = 7; //restart
                return 3;
            }
            break;
        }
        case ADC_1000K:
        {
            if (f0 <= 50000)
            {
                main_adc_mode = ADC_500K;
                main_page = 7; //restart
                return 4;
            }
            break;
        }
        default:
            break;
    }

    hmny2 = fft_find_max(SinInA, base*2);
    hmny3 = fft_find_max(SinInA, base*3);
    hmny4 = fft_find_max(SinInA, base*4);
    hmny5 = fft_find_max(SinInA, base*5);

    /*calculate thd*/
    thd = sqrt(  SinInA[hmny2]*SinInA[hmny2]+
                               SinInA[hmny3]*SinInA[hmny3]+
                               SinInA[hmny4]*SinInA[hmny4]+
                               SinInA[hmny5]*SinInA[hmny5]
                         )/SinInA[base];

    main_thd = thd;
    main_base = base/1024.0*fs;
    main_f1 = 1;
    main_f2 = SinInA[hmny2]/SinInA[base];
    main_f3 = SinInA[hmny3]/SinInA[base];
    main_f4 = SinInA[hmny4]/SinInA[base];
    main_f5 = SinInA[hmny5]/SinInA[base];

    main_point_N = (uint8_t)(fs/f0)+1;
    for (i=0;i<1024;i++)
    {
        resultsBuffer[i] = (float)InputADC[i] * 3.3/16384;
    }

    /*find DC part*/
    avrg=0;
    for (i=0;i<1024;i+=32)
    {
        sum = 0;
        for (j=0;j<32;j++)
        {
            sum += resultsBuffer[i+j];
        }
        sum /= 32;
        avrg += sum;
    }
    avrg /= 32;

    for (i=0;i<1024;i++)
    {
```

```
        resultsBuffer[i] -= avrg;
    }

    peak = fabs(resultsBuffer[0]);
    for (i=0;i<1024-main_point_N-1;i++)
    {
        if (peak>fabs(resultsBuffer[i]))
        {
            buf = i;
            peak = fabs(resultsBuffer[i]);
        }
    }
    for (i=0;i<main_point_N;i++)
    {
        main_point_data[i] = resultsBuffer[i+buf];
    }
    main_point_dt = 1000/fs;

    return 0;
}

void fft_out_raw(float *adc_f, uint16_t len)
{
    uint16_t i;

    printf("num of %d data as below:", len);
    for (i=0;i<len;i++)
    {
        printf("%0.3f,", adc_f[i]);
    }
}

#define __MIN(a,b) ((a)<(b)?(a):(b))
#define __MAX(a,b) ((a)>(b)?(a):(b))

static uint16_t fft_find_max(float *adc_f, uint16_t center)
{
    uint16_t i, ret, start,end;
    float buf;
    start = __MAX(center-10, 10);
    end = __MIN(center+10, 512);

    buf = adc_f[start];
    for(i=start;i<end;i++)
    {
        if (buf < adc_f[i])
        {
            buf = adc_f[i];
            ret = i;
        }
    }
    RETURN RET;
}
```

main.c

```
#INCLUDE "SYSINIT.H"
#INCLUDE "USART.H"
#INCLUDE "DELAY.H"
#INCLUDE "ADC.H"
#INCLUDE "TIM.H"
#INCLUDE "MYGPIO.H"
#INCLUDE "FFT.H"
#INCLUDE "LCD.H"
#INCLUDE "STRING.H"
#INCLUDE "DMA.H"
#INCLUDE "MATH.H"
```

```
VOID MAIN_FFT_OUT(VOID);
VOID MAIN_WELCOME(VOID);

VOLATILE UINT16_T INPUTADC[1024];
EXTERN UINT8_T ADC_FLAG;
EXTERN VOLATILE FLOAT RESULTSBUFFER[1024];
EXTERN FLOAT SININA[1024];
//EXTERN FLOAT OUTPUTADC[1024];
EXTERN FLOAT SININI[1024];

UINT16_T KK=0;
EXTERN UINT16_T KK_CHANGE;

FLOAT MAIN_THD = 0.0;
FLOAT MAIN_BASE = 0.0;
FLOAT MAIN_F1 = 0.0;
FLOAT MAIN_F2 = 0.0;
FLOAT MAIN_F3 = 0.0;
FLOAT MAIN_F4 = 0.0;
FLOAT MAIN_F5 = 0.0;
UINT8_T MAIN_ADC_MODE = 0;
UINT8_T MAIN_PAGE = 0;

UINT8_T MAIN_AGC = 1;

/*JSON SEND*/
UINT8_T MAIN_POINT_N = 40;
FLOAT MAIN_POINT_DATA[50] =
{
    0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,
    1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,2.0,
    2.1,2.2,2.3,2.4,2.5,2.4,2.3,2.2,2.1,2.0,
    1.9,1.8,1.7,1.6,1.5,1.4,1.3,1.2,1.1,1.0
};
FLOAT MAIN_POINT_DT = 1.0;

FLOAT SAMPLE_FS = 0.0;

VOID MAIN_SEND_JSON()
{
    UINT8_T I;
    UINT16_T LEN;
    CHAR JSON[800];


    SPRINTF(JSON, "{\"NUM\":%D,\"DATA\":[", MAIN_POINT_N);

    FOR (I=0;I<MAIN_POINT_N-1;I++)
    {
        LEN = STRLEN(JSON);
        SPRINTF(JSON+LEN, "%0.2F,", MAIN_POINT_DATA[I]);
    }
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "%0.2F],\"TIME\":[", MAIN_POINT_DATA[MAIN_POINT_N-1]);

    FOR (I=0;I<MAIN_POINT_N-1;I++)
    {
        LEN = STRLEN(JSON);
        SPRINTF(JSON+LEN, "%0.3F,", MAIN_POINT_DT*I);
    }
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "%D],\"UNIT\":\"%S\",", (MAIN_POINT_N-1), "MS");

    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"THDX\":%0.2F,", MAIN_THD*100);
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"F1\":%0.2F,", MAIN_F1);
```

```
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"F2\":%0.2F,", MAIN_F2);
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"F3\":%0.2F,", MAIN_F3);
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"F4\":%0.2F,", MAIN_F4);
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"F5\":%0.2F,", MAIN_F5);

    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"FS\":%0.2F,", SAMPLE_FS);
    LEN = STRLEN(JSON);
    SPRINTF(JSON+LEN, "\"F0\":%0.2F}", MAIN_BASE);

    PRINTF("CLEAR\R");
    PRINTF("%S\R", JSON);
}

VOID MAIN_LCD_DRAW()
{
    UINT8_T DX = 200/MAIN_POINT_N;
    FLOAT Y_MAX,DY;
    UINT8_T I,J;

    Y_MAX = FABS(MAIN_POINT_DATA[0]);
    FOR(I=1;I<MAIN_POINT_N;I++)
    {
        IF (Y_MAX<FABS(MAIN_POINT_DATA[I]))
        {
            Y_MAX = FABS(MAIN_POINT_DATA[I]);
        }
    }
    DY = 51.0/Y_MAX;

    POINT_COLOR = RED;
    FOR (I=0;I<MAIN_POINT_N-1;I++)
    {

    LCD_DRAWLINE((I*DX)+8,79-(INT16_T)(1.0*MAIN_POINT_DATA[I]*DY),((I+1)*DX)+8,79-
(INT16_T)(1.0*MAIN_POINT_DATA[I+1]*DY));
    }
}

INT MAIN(VOID)
{
    UINT16_T I;
    UINT8_T RET=0;
    U8 X=0;
    FLOAT __MAX,__MIN;

    SYSINIT();          // 第 3 讲 时钟配置
    UART_INIT(115200); // 第 7 讲 串口配置
    DELAY_INIT();       // 第 4 讲 滴答延时

    /* 启用浮点运算的 FPU */
    FPU_ENABLEMODULE();
    FPU_ENABLELAZYSTACKING();
    TIM_A0_CONFIG(ADC_500K);
    TIM_A1_CONFIG();
    TIM_32_CONFIG();
    GPIO_INIT();
    LCD_INIT();

    DMA_START();
    ADC_FLAG=2;

    DMA_INIT();
```

```
DELAY_MS(1000);
GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P3, GPIO_PIN6);
DELAY_MS(200);
GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P3, GPIO_PIN6);
DELAY_MS(3000);

MAIN_WELCOME();

WHILE (1)
{
    IF (ADC_FLAG==2)
    {
        ADC_FLAG=0;
        SAMPLE_FS = ADC_GET_FS();
        MAIN_PAGE = 0;
        CONTINUE;
    }
    ELSE IF (ADC_FLAG==1)
    {
        ADC_FLAG=0;
        SAMPLE_FS = ADC_GET_FS();

        FOR (I=0;I<1024;I++)
        {
            RESULTSBUFFER[I] = (FLOAT)INPUTADC[I] * 3.3/16384;
        }

        /*AGC 增益选择*/
        __MAX = RESULTSBUFFER[0];
        __MIN = RESULTSBUFFER[0];
        FOR (I=0;I<1024;I++)
        {
            IF (__MAX<RESULTSBUFFER[I])
                __MAX = RESULTSBUFFER[I];
            IF (__MIN>RESULTSBUFFER[I])
                __MIN = RESULTSBUFFER[I];
        }
        __MAX = __MAX - __MIN;
        SWITCH (MAIN_AGC)
        {
            CASE 1:
            {
                IF (__MAX<0.3)
                {
                    GPIO_SET_AGC(2);
                    MAIN_PAGE = 7;CONTINUE;
                }
                BREAK;
            }
            CASE 2:
            {
                IF (__MAX<0.1)
                {
                    GPIO_SET_AGC(6);
                    MAIN_PAGE = 7;CONTINUE;
                }
                BREAK;
            }
            CASE 6:
            {
                IF (__MAX<0.05)
                {
                    GPIO_SET_AGC(10);
                    MAIN_PAGE = 7;CONTINUE;
                }
                BREAK;
            }
            CASE 10:BREAK;
```

```
                DEFAULT:
                {
                    GPIO_SET_AGC(1);
                    MAIN_AGC = 1;
                    CONTINUE;
                }
            }

            IF (SAMPLE_FS <800)
            {
                MAIN_PAGE = 7;
                CONTINUE;
            }

            LED_R(1);
            FFT_WINDOW((FLOAT*)RESULTSBUFFER, SAMPLE_FS);
            FFTR((FLOAT*)RESULTSBUFFER, SININI, SININA,1024,10);//调用 FFTR 计算 DFT
            RET = FFT_TEST_ALX(SAMPLE_FS);
            IF (RET!=0)
                CONTINUE;
            MAIN_SEND_JSON();
            MAIN_LCD_DRAW();
            LED_R(0);



            LCD_CLEAR(WHITE);
            MAIN_PAGE = 2;
        }

        SWITCH(MAIN_PAGE)
        {
            CASE 0://欢迎页面
            {
                SWITCH(MAIN_ADC_MODE)
                {
                    CASE ADC_10K:    LCD_SHOWSTRING(56,245,200,24,16, "FUNDAMENTAL
MODE"); BREAK;
                    CASE ADC_1000K: LCD_SHOWSTRING(52,245,200,24,16, " 1MHZ AGC MODE
");BREAK;
                    DEFAULT:BREAK;
                };
                BREAK;
            }
            CASE 1:
            {
                MAIN_FFT_OUT();
                BREAK;
            }
            CASE 2:
            {
                MAIN_FFT_OUT();
                MAIN_LCD_DRAW();
                BREAK;
            }

            CASE 5: //BACK
            {
                LCD_CLEAR(WHITE);
                MAIN_WELCOME();
                MAIN_PAGE = 0;
                BREAK;
            }
            CASE 6: //START
            {
                MAIN_PAGE = 255;

                LCD_SHOWSTRING(52,288,200,24,16, "  CALCULATING.. ");
```

```
                    DMA_START();
                    BREAK;
                }
                CASE 7: //RESTART
                {
                    //LCD_CLEAR(WHITE);
                    MAIN_PAGE = 255;

                    LCD_SHOWSTRING(52,288,200,24,16, "  CALCULATING.. ");
                    DMA_START();
                    BREAK;
                }
                DEFAULT:
                {
                    BREAK;
                }
            }

        }
}

VOID MAIN_WELCOME(VOID)
{
    POINT_COLOR=RED;
    LCD_SHOWSTRING(60,0,200,24,24,"NUEDC 2021");

    //BOARD
    LCD_DRAWRECTANGLE(40,70,200,270);

    //BUTTON
    LCD_DRAWRECTANGLE(40,140,45,150);
    LCD_DRAWRECTANGLE(200,140,195,150);

    POINT_COLOR=BLUE;
    LCD_SHOWSTRING(50,135,200,24,16, "START");
    LCD_SHOWSTRING(155,135,200,24,16, "MODE");

}

VOID MAIN_FFT_OUT(VOID)
{
    UINT8_T SBUF[50];
    UINT16_T I;



    POINT_COLOR=BLACK;
    LCD_SHOWSTRING(60,0,200,24,24,"NUEDC 2021");

    LCD_DRAWLINE(8,28,8,130);//单周期信号复现
    LCD_DRAWLINE(8,79,230,79);


    POINT_COLOR=BLUE;
    LCD_SHOWSTRING(10,14,200,24,16,"ADC");
//  LCD_SHOWSTRING(10,137,200,24,16,"FFT");

    POINT_COLOR=RED;
    SPRINTF(SBUF, "THD:  %0.2F%%", MAIN_THD*100);
    LCD_SHOWSTRING(8,140,200,24,24,SBUF);

    POINT_COLOR=BLACK;
    SPRINTF(SBUF, "F1:%0.2F", MAIN_F1);
    LCD_SHOWSTRING(8,180,200,24,16,SBUF);
    SPRINTF(SBUF, "F2:%0.2F", MAIN_F2);
    LCD_SHOWSTRING(8+110,180,200,24,16,SBUF);
    SPRINTF(SBUF, "F3:%0.2F", MAIN_F3);
    LCD_SHOWSTRING(8,200,190,24,16,SBUF);
```

```
    SPRINTF(SBUF, "F4:%0.2F", MAIN_F4);
    LCD_SHOWSTRING(8+110,200,200,24,16,SBUF);
    SPRINTF(SBUF, "F5:%0.2F", MAIN_F5);
    LCD_SHOWSTRING(8,220,200,24,16,SBUF);

    SPRINTF(SBUF, "FS: %0.2FHZ", SAMPLE_FS);
    LCD_SHOWSTRING(120,300,200,24,12, SBUF);

    SPRINTF(SBUF, "FO: %0.2FHZ", MAIN_BASE);
    LCD_SHOWSTRING(8,300,200,24,12, SBUF);
}
```

mygpio.c

```
#INCLUDE "MYGPIO.H"
#INCLUDE "LCD.H"
#INCLUDE "DELAY.H"
STATIC CONST UINT32_T GPIO_PORT_TO_BASE[] =
{   0X00,
    (UINT32_T)P1,
    (UINT32_T)P1+1,
    (UINT32_T)P3,
    (UINT32_T)P3+1,
    (UINT32_T)P5,
    (UINT32_T)P5+1,
    (UINT32_T)P7,
    (UINT32_T)P7+1,
    (UINT32_T)P9,
    (UINT32_T)P9+1,
    (UINT32_T)PJ
};

EXTERN UINT8_T MAIN_AGC;

VOID GPIO_SET_AGC(UINT8_T AGC)
{
    MAIN_AGC = AGC;
    SWITCH(AGC)
    {
        CASE 1:
        {
            GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P6, GPIO_PIN5);
            GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P1, GPIO_PIN5);
            BREAK;
        }
        CASE 2:
        {
            GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P4, GPIO_PIN1);
            GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P1, GPIO_PIN5);
            BREAK;
        }
        CASE 6:
        {
            GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P4, GPIO_PIN1);
            GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P1, GPIO_PIN5);
            BREAK;
        }
        CASE 10:
        {
            GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P1, GPIO_PIN5);
            GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P6, GPIO_PIN5);
            BREAK;
        }
        DEFAULT:
        {
            GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P6, GPIO_PIN5);
            GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P1, GPIO_PIN5);
```

```
            GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P4, GPIO_PIN1);
            BREAK;
        }
    }
}

VOID GPIO_INIT(VOID)
{
    GPIO_SETASOUTPUTPIN(GPIO_PORT_P1, GPIO_PIN0);
    GPIO_SETASOUTPUTPIN(GPIO_PORT_P2, GPIO_PIN0);
    GPIO_SETASOUTPUTPIN(GPIO_PORT_P2, GPIO_PIN1);
    GPIO_SETASOUTPUTPIN(GPIO_PORT_P2, GPIO_PIN2);

    GPIO_SETASOUTPUTPIN(GPIO_PORT_P6, GPIO_PIN5);
    GPIO_SETASOUTPUTPIN(GPIO_PORT_P1, GPIO_PIN5);
    GPIO_SETASOUTPUTPIN(GPIO_PORT_P4, GPIO_PIN1);

    GPIO_SETASOUTPUTPIN(GPIO_PORT_P3, GPIO_PIN6);


    GPIO_SETOUTPUTLOWONPIN(GPIO_PORT_P6, GPIO_PIN5);
    GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P1, GPIO_PIN5);
    GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P4, GPIO_PIN1);

    GPIO_SETOUTPUTHIGHONPIN(GPIO_PORT_P3, GPIO_PIN6);

    GPIO_SETASINPUTPINWITHPULLUPRESISTOR(GPIO_PORT_P1, GPIO_PIN1);
    GPIO_SETASINPUTPINWITHPULLUPRESISTOR(GPIO_PORT_P1, GPIO_PIN4);

    LED_R(0);
    LED_G(0);
    LED_B(0);
}


VOID GPIO_WRITE(UINT_FAST8_T SELECTEDPORT, UINT_FAST8_T VAL)
{
  HWREG8(GPIO_PORT_TO_BASE[SELECTEDPORT] + OFS_LIB_PAOUT) = VAL;
}

UINT8_T GPIO_READ(UINT_FAST8_T SELECTEDPORT)
{
    VOLATILE UINT_FAST16_T INPUTPINVALUE=0;
    INPUTPINVALUE = HWREG8(GPIO_PORT_TO_BASE[SELECTEDPORT] + OFS_LIB_PAIN);
    RETURN INPUTPINVALUE;
}
```

tim.c

```
#include "tim.h"
#include "mygpio.h"
#include "stdio.h"

extern uint8_t main_adc_mode;
extern uint8_t main_page;

/* Timer_A Continuous Mode Configuration Parameter */
static uint16_t t0 = 0, t1 = 0, t2 = 0;
static uint16_t key1_t0 = 0, key1_t1 = 0, key1_t2 = 0;
static uint16_t key2_t0 = 0, key2_t1 = 0, key2_t2 = 0;

uint16_t kk_change = 0;

//void tim_dma_Config(void)
//{
//
//}
```

```
void tim_a0_Config(uint8_t speed)
{
        Timer_A_UpModeConfig upModeConfig =
        {
                        TIMER_A_CLOCKSOURCE_SMCLK,              // SMCLK Clock Source
                        TIMER_A_CLOCKSOURCE_DIVIDER_1,        // SMCLK/1 = 48000kHz
                        2400,
                    //48000kHz/4800 = 10kHz
                        TIMER_A_TAIE_INTERRUPT_DISABLE,       // Disable Timer ISR
                        TIMER_A_CCIE_CCR0_INTERRUPT_DISABLE, // Disable CCR0
                        TIMER_A_DO_CLEAR                      // Clear Counter
        };

        /* Timer_A Compare Configuration Parameter */
        Timer_A_CompareModeConfig compareConfig =
        {
                        TIMER_A_CAPTURECOMPARE_REGISTER_1,          // Use CCR1
                        TIMER_A_CAPTURECOMPARE_INTERRUPT_DISABLE,   // Disable CCR
interrupt
                        TIMER_A_OUTPUTMODE_SET_RESET,                    // Toggle
output but
                        20                                          // 16000 Period
        };

        switch (speed)
        {
            case ADC_10K:
            {
                upModeConfig.timerPeriod = 2400;//10kHz
                break;
            }
            case ADC_50K:
            {
                upModeConfig.timerPeriod = 1200;//50kHz
                break;
            }
            case ADC_100K:
            {
                upModeConfig.timerPeriod = 240;//100kHz
                break;
            }
            case ADC_500K:
            {
                upModeConfig.timerPeriod = 48;//500kHz
                break;
            }
            case ADC_1000K:
            {
                upModeConfig.timerPeriod = 24;//1MHz
                break;
            }
            default:
                break;
        }

    /* Configuring Timer_A in continuous mode and sourced from ACLK */
    Timer_A_configureUpMode(TIMER_A0_BASE, &upModeConfig);

    /* Configuring Timer_A0 in CCR1 to trigger at 16000 (0.5s) */
    Timer_A_initCompare(TIMER_A0_BASE, &compareConfig);
}

void tim_a1_Config()
{
    Timer_A_UpModeConfig upModeConfig =
    {
        TIMER_A_CLOCKSOURCE_SMCLK,            // SMCLK Clock Source
```

```
        TIMER_A_CLOCKSOURCE_DIVIDER_64,        // SMCLK/64 = 750kHz
        750,
    //750kHz/4800 = 1kHz
        TIMER_A_TAIE_INTERRUPT_DISABLE,       // Disable Timer ISR
        TIMER_A_CCIE_CCR0_INTERRUPT_ENABLE, // Disable CCR0
        TIMER_A_DO_CLEAR                       // Clear Counter
    };

    Timer_A_configureUpMode(TIMER_A1_BASE, &upModeConfig);

    Timer_A_startCounter(TIMER_A1_BASE, TIMER_A_UP_MODE);

    Timer_A_clearCaptureCompareInterrupt(TIMER_A1_BASE,
TIMER_A_CAPTURECOMPARE_REGISTER_0);

    Interrupt_enableInterrupt(INT_TA1_0);
}

void tim_32_Config(void)
{
    Timer32_initModule(TIMER32_0_BASE,      TIMER32_PRESCALER_1,      TIMER32_32BIT,
TIMER32_PERIODIC_MODE);
    Timer32_setCount(TIMER32_0_BASE, 0xffffffff);
}

void TA1_0_IRQHandler(void)
{
//static uint16_t t0 = 0, t1 = 0, t2 = 0;
//static uint16_t key1_t0 = 0, key1_t1 = 0, key1_t2 = 0;
//static uint16_t key2_t0 = 0, key2_t1 = 0, key2_t2 = 0;
    Timer_A_clearCaptureCompareInterrupt(TIMER_A1_BASE,
TIMER_A_CAPTURECOMPARE_REGISTER_0);

    if (t0++ > 500)
    {
        GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0);
        t0 = 0;
    }

    /*左按钮*/
    if (getKey1 == 0)//key 1 down
    {
        if ((key1_t0 > 1000)&&(key1_t1==0))
        {
            /*key1 long push*/
            key1_t1=1;
            kk_change=2;
        }
        else
        {
            key1_t0++;
        }
    }
    else//click or long click end
    {
        if (key1_t0 > 1000)
        {
            key1_t0=0;//long push complete
            key1_t1=0;

            led_r(0);
            led_g(0);
            led_b(0);
        }
        else if (key1_t0 > 20)//click complete
        {
            /*key1 click*/
            key1_t0=0;key1_t1=0;
```

```
            kk_change = 1;

            if (main_page == 0)
                main_page = 6; //start
            else
                main_page = 7; //restart
        }
    }

    /*右按钮*/
    if (getKey2 == 0)//key 2 down
    {
        if ((key2_t0 > 1000)&&(key2_t1==0))
        {
            /*key1 long push*/
            key2_t1=1;
            kk_change=4;

            if (main_page==0)
            {
                main_adc_mode = main_adc_mode==ADC_10K?ADC_1000K:ADC_10K;
                gpio_set_agc(1);
            }
        }
        else
        {
            key2_t0++;
        }
    }
    else//click or long click end
    {
        if (key2_t0 > 1000)
        {
            key2_t0=0;//long push complete
            key2_t1=0;
        }
        else if (key2_t0 > 20)//click complete
        {
            /*key1 click*/
            key2_t0=0;key1_t1=0;
            kk_change = 3;

            if ((main_page == 1)||(main_page==2))
                main_page = 5;//back
        }
    }
}
```