电赛2023B题

任务

设计并制作一个同轴电缆长度与终端负载检测装置（以下简称“装置”），如图1所示。待测电缆始端通过电缆连接头与装置连接，电缆终端可开路或接入电阻、电容负载。设置“长度检测”和“负载检测”两个按键，用以选择和启动相应功能。负载电阻值范围：10Ω~30Ω，电容值范围：100pF~300pF。装置由不大于6V的单电源供电。

图 1　同轴电缆长度及终端负载检测装置示意图

要求

1. 基本要求

（1）装置能够显示工作状态、电缆长度、负载类型、负载参数，显示格式见表 1。

表1 装置显示格式

工作状态	显示“正在检测”或“结果保持”
电缆长度	显示“XXXX cm”
负载类型	显示“开路”、“电阻”、“电容”中的一种
负载参数	显示电阻或电容的数值及单位

（2）电缆长度 1000cm≤L≤2000cm、终端开路，按“长度检测”键启动检测，装置能够检测并显示电缆长度 L，相对误差的绝对值不大于 5%，一次检测时间不超过 5s。

（3）终端开路条件下完成电缆长度检测后，保持 L 不变，在终端接入电阻、电容中的一种负载，按“负载检测”键启动检测，装置能够正确判断并显示负载类型，一次检测时间不超过 5s。

2. 发挥部分

（1） 提高电缆长度检测精度：电缆长度 1000cm≤L≤2000cm、终端开路，电缆长度检测相对误差的绝对值不大于 1%，一次检测时间不超过 5s。

（2） 终端开路条件下完成长度检测后，保持 L 不变，在终端接入电阻、电容中的一种负载，按“负载检测”键启动检测，装置在正确判断负载类型的基础上检测并显示负载的电阻、电容值，相对误差的绝对值不大于 10%，一次检测时间不超过 5s。

（3） 减小电缆长度检测盲区：终端开路时，在满足电缆长度检测相对误差的绝对值不大于 1%、一次检测时间不超过 5s 的条件下，减小能够检测的电缆长度至 L≤100cm。

（4）其他。

方法

时间法

用 DP 22 测量反射回来的时间差

反射法

反射链接

最后选择的方法，事实证明非常好用，非常简单，非常精准

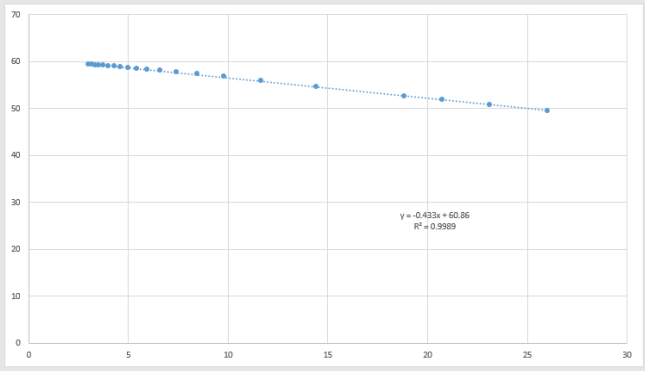
总的思想是先用 NE555 振荡出一个频率合适的方波（大概 100 多 k），再把传输线看成一个大电容（事实上理论就是如此）并联到 555 的电容端，通过新振荡出来的频率来测量这时候传输线的长度

我们目前有三个量

- 线长
- 振荡频率
- 线长和震荡频率的乘积

这时候如果把 x 轴当作振荡频率，y 轴当作二者的乘积
那么可以发现，画出来的图形及其的线性
如下图所示
这个时候就可以通过频率精准的测出长度

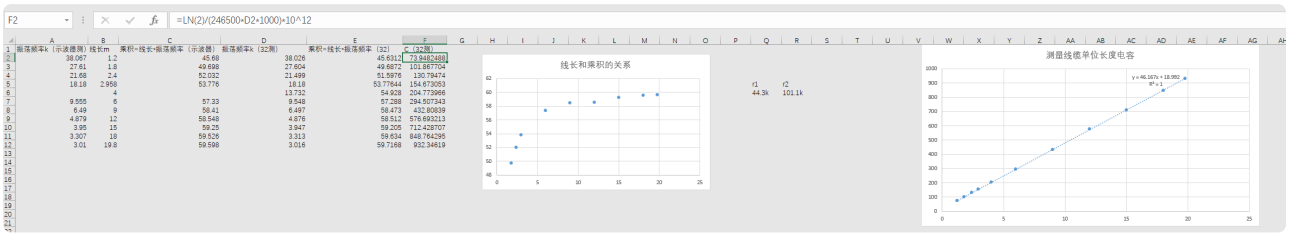
线长m	振荡频率k（32测）	乘积=线长*振荡频率（32）
19.8	3.002	59.4396
18.8	3.159	59.3892
17.8	3.333	59.3274
16.8	3.524	59.2032
15.8	3.744	59.1552
14.8	3.989	59.0372
13.8	4.273	58.9674
12.8	4.597	58.8416
11.8	4.976	58.7168
10.8	5.424	58.5792
9.8	5.955	58.359
8.8	6.603	58.1064
7.8	7.409	57.7902
6.8	8.435	57.358
5.8	9.787	56.7646
4.8	11.662	55.9776
3.8	14.403	54.7314
2.8	18.808	52.6624
2.5	20.735	51.8375
2.2	23.108	50.8376
1.9	26.022	49.4418
1.6	26.871	42.9936
1.3	35.117	45.6521
1	42.56	42.56
0.7	53.93	37.751
0.4	73.19	29.276
0.1		



拟合2

至于线缆单位长度的电容，也可以通过 NE555 的振荡公式算出来后进行拟合

可以看到，R^2 直接干到 1 了，很准，说明电缆等效电容分布很均匀



之后判断负载可以通过测量接入负载后的振荡频率来区分

- 振荡频率和接入前基本一致是**开路**
- 振荡频率比接入前低是**电容**
- 振荡频率变成 0 是**电阻**

  代码如下

```
int judge_type(float connect_freq)

{

//如果不震荡了，那就是接入了电阻

if(connect_freq == 0)

{

return r_flag;

}

//如果频率和开路的基本频率基本一样，那么就是开路，这个容忍度是40kHz

else if((basic_freq-connect_freq)<40e3)
```

```
{

return open_flag;

}

//否则就是电容接入

else

{

return c_flag;

}

}
```

- 电容测量就根据变化的电容值来测量，接入电容相当于并联了电容，直接换算成传输线的长度来运算，最后反解回电容
- 电阻测量就加入一个继电器，判断为电阻就切换一下测量电路，然后根据分压法测量

其他说明

- TIM 的捕获频率设置了 2 分频（84 MHz），这个频率越大那么频率测量的就越精确，但没有取 1 分频是因为 168 M 太快了，会导致 65535 的最大重装载溢出，造成某些低频无法测量，所以选择了 2 分频
- 还有一些拟合的数据见最后的整体代码

整体代码

宏定义

```
/* USER CODE BEGIN PD */

//定义判断状态标志位

#define open_flag 0

#define c_flag 1

#define r_flag 2

//定义连接负载板电容

#define add_C 1.9

//定义基础开路震荡频率

#define basic_freq 132e3

//乘积和振荡频率的一次函数关系

#define freq_slope -0.465

#define freq_intercept 60.803

//线缆等效电容线密度

#define line_density_slope 46.167

#define line_density_intercept 18.992

//定义分压测量电阻的串联已知电阻
```

```
#define Rs 20.5

/* USER CODE END PD */
```

全局变量

```
/* USER CODE BEGIN PV */

int time[3];

int i = 0;

int flag = 1;

//定义第一次测量电路频率

double freq = 0;

//定义第一次测量的乘积=线长*振荡频率

double sum = 0;

//定义第一次测量线长

double length = 0;

//定义第一次测量电缆的电容值

double C = 0;

//定义接入负载后的振荡频率

double connect_freq = 0;
```

```
double R_vol = 0;

//负载类型判断

int type = 0;

/* USER CODE END PV */
```

函数

自定义函数

```
/* USER CODE BEGIN PFP */

//判断负载类型

int judge_type(float connect_freq);

//第一次计算电缆长度

void calculate_length();

//第二次计算

void recalculate();

/* USER CODE END PFP */
```

```
/* USER CODE BEGIN 4 */

int judge_type(float connect_freq)

{
```

```
//如果不震荡了，那就是接入了电阻
if(connect_freq == 0)
{
return r_flag;
}
//如果频率和开路的基本频率基本一样，那么就是开路，这个容忍度是40kHz
else if((basic_freq-connect_freq)<40e3)
{
return open_flag;
}
//否则就是电容接入
else
{
return c_flag;
}
}

void calculate_length()
{
```

```
//第一次计算，算出振荡频率、线长和等效电容

time[2]=time[1]-time[0];

printf("%d\r\n",time[2]);

freq= (84*1000000)/time[2];

printf("freq=%f k\r\n",freq/1000);

sum = freq_slope*freq/1000+freq_intercept;

length = sum/(freq/1000);

printf("length=%f\r\n",length);

C = length*46.167+18.992;

printf("C=%f\r\n",C);

}

void recalculate()

{

//第二次计算，考虑到可能不震荡从而触发不了输入捕获，先把两个time置零

time[1] = time[0] = 0;

i = 0;

__HAL_TIM_SET_COUNTER(&htim1,0);

HAL_Delay(1000);
```

```
time[2]=time[1]-time[0];

//如果算出来是0，那说明没有触发输入捕获，这样就说明没有震荡，接入了电阻

if(time[2] == 0)

{

connect_freq = 0;

printf("电阻");

}

//触发了输入捕获就是开路或者接入电容，可以算出来振荡频率

else

{

connect_freq= (84*1000000)/time[2];

}

}
/* USER CODE END 4 */
```

TIM 回调函数

```
/* USER CODE BEGIN 0 */



void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
```

```
{

// __HAL_TIM_SET_COUNTER(&htim1,0);



if(i==0)

{

time[i] = HAL_TIM_ReadCapturedValue(&htim1, TIM_CHANNEL_3);

i++;

}

else if(i==1)

{

time[i] = HAL_TIM_ReadCapturedValue(&htim1, TIM_CHANNEL_3);

i++;

}

else

{

return;

}
```

```
}

/* USER CODE END 0 */
```

主函数

```
int main(void)

{



/* USER CODE BEGIN 1 */



/* USER CODE END 1 */



/* MCU Configuration--------------------------------------*/

/* Reset of all peripherals, Initializes the Flash interface
and the Systick. */

HAL_Init();



/* USER CODE BEGIN Init */



/* USER CODE END Init */
```

```
/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART1_UART_Init();
MX_DAC_Init();
MX_TIM3_Init();
MX_TIM1_Init();
/* USER CODE BEGIN 2 */
//初始化ADS8688
ADS8688_Init_Mult();
```

```
//初始化定时器一些参数

HAL_TIM_Base_Start(&htim1);

HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_3);

__HAL_TIM_SET_COUNTER(&htim1,0);



//初始化继电器

HAL_GPIO_WritePin(Relays_GPIO_Port, Relays_Pin, RESET);



/* USER CODE END 2 */



/* Infinite loop */

/* USER CODE BEGIN WHILE */

//适当延时，使输入捕获频率测量完毕

HAL_Delay(1000);

//计算第一次的参数：freq,C,length

calculate_length();



HAL_Delay(1000);
```

```
//进行第二次计算

recalculate();

//进行负载判断

type = judge_type(connect_freq);



if(type == c_flag)

{

//计算连接后的乘积

double sum_connect = (-0.4052)*connect_freq/1000+60.846;

//计算连接后的等效电缆长度

double length_connect = sum_connect/(connect_freq/1000);

//根据电缆电容线密度，由等效长度算出等效电容大小

double C_connect =
length_connect*line_density_slope+line_density_intercept;

//算出来的C不对，需要拟合得出正确的结果

double real_c = 2.0755*(C_connect-C-add_C)+8.7644;

printf("newC=%f\r\n",real_c);

}

else if(type == r_flag)
```

```
{

//写后续切换继电器操作

HAL_GPIO_WritePin(Relays_GPIO_Port, Relays_Pin, SET);

HAL_Delay(100);



//分压法测电阻阻值，用8688采集电压更准确

R_vol = get_vol(1);

printf("R_vol=%f\r\n",R_vol);



double R = (R_vol*Rs)/(5-R_vol);

printf("R=%f\r\n",R);

}

else

{

//开路，什么也不操作

}



while (1)
```

```
{

}


/* USER CODE END WHILE */



/* USER CODE BEGIN 3 */



/* USER CODE END 3 */

}
```