使用 STM32 开发心电监测仪

尚硅谷研究院

版本: V1.0

1. 项目需求和技术选型

* 1. 硬件选型

我们的需求是制作一个心电监测仪。心电监测仪需要以下一些硬件设备：

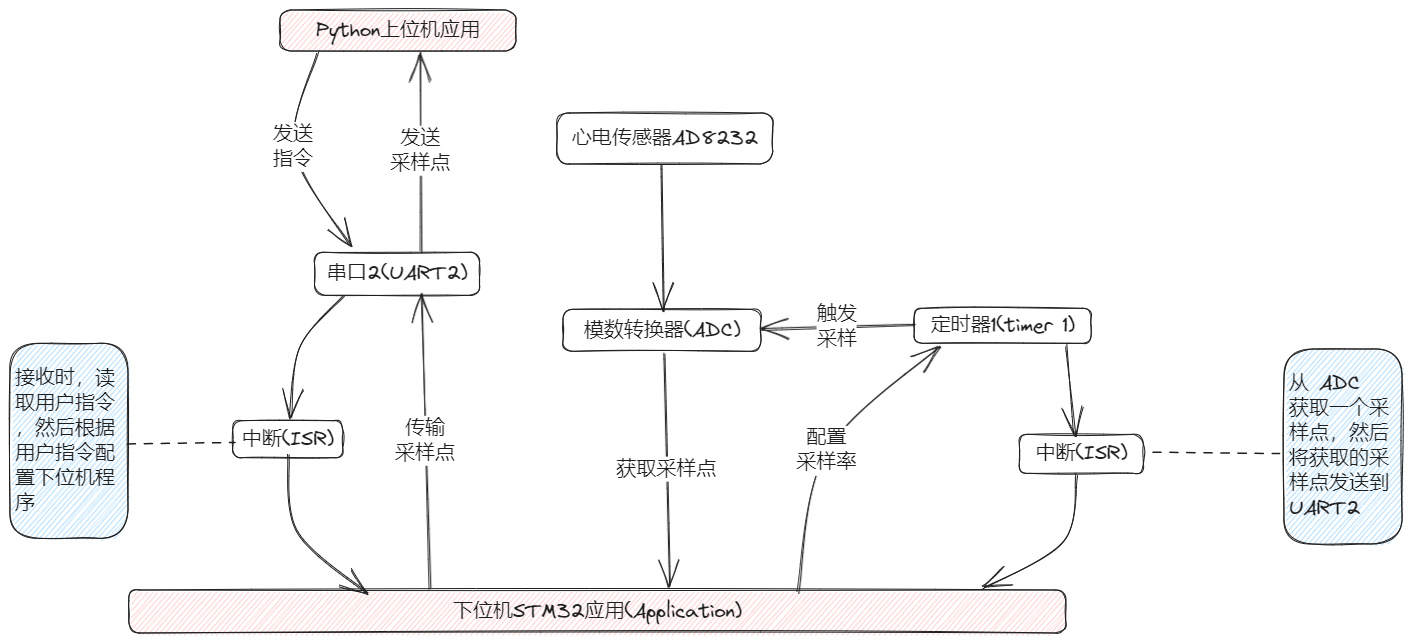
* 导联线：导联线负责从人体采集微弱的电压信号。如果想要采集精确的人体电压信号，可以选择12导联这样的导联线，当然价格是非常感人的。只是想理解原理的话，可以选择便宜的单导联线。
* 电压传感器：负责读取导联线所产生的电压信号。我们可以选择 ADI 公司的 AD8232 芯片。可以采集电压信号并输出。
* 单片机：我们选择最常用的 STM32F103 系列的单片机。这款单片机上面自带了 ADC 功能。可以将 AD8232 输出的模拟电压信号转换成离散的数字信号。方便我们进一步将数字信号值通过串口传递到上位机上去。
* 个人电脑：编写上位机软件，接收单片机通过串口发送过来的数据，并做进一步处理（可视化，滤波，等等）。

想要做更好的心电监测仪产品，需要更好的导联线。由于 STM32 自带的 ADC 的精度和采样率可能不符合要求，可以选择外置的 ADC 芯片，例如 TI 公司生产的 ADS1298 系列的 ADC 芯片。

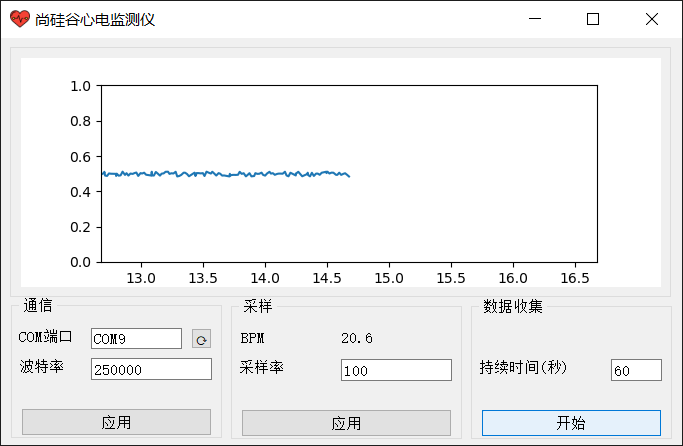
* 1. 软件选型
* STM32CubeMX：用来产生样板代码。
* Clion/VSCode/Keil：用来开发单片机程序。
* PyQt5：用来在上位机开发程序。

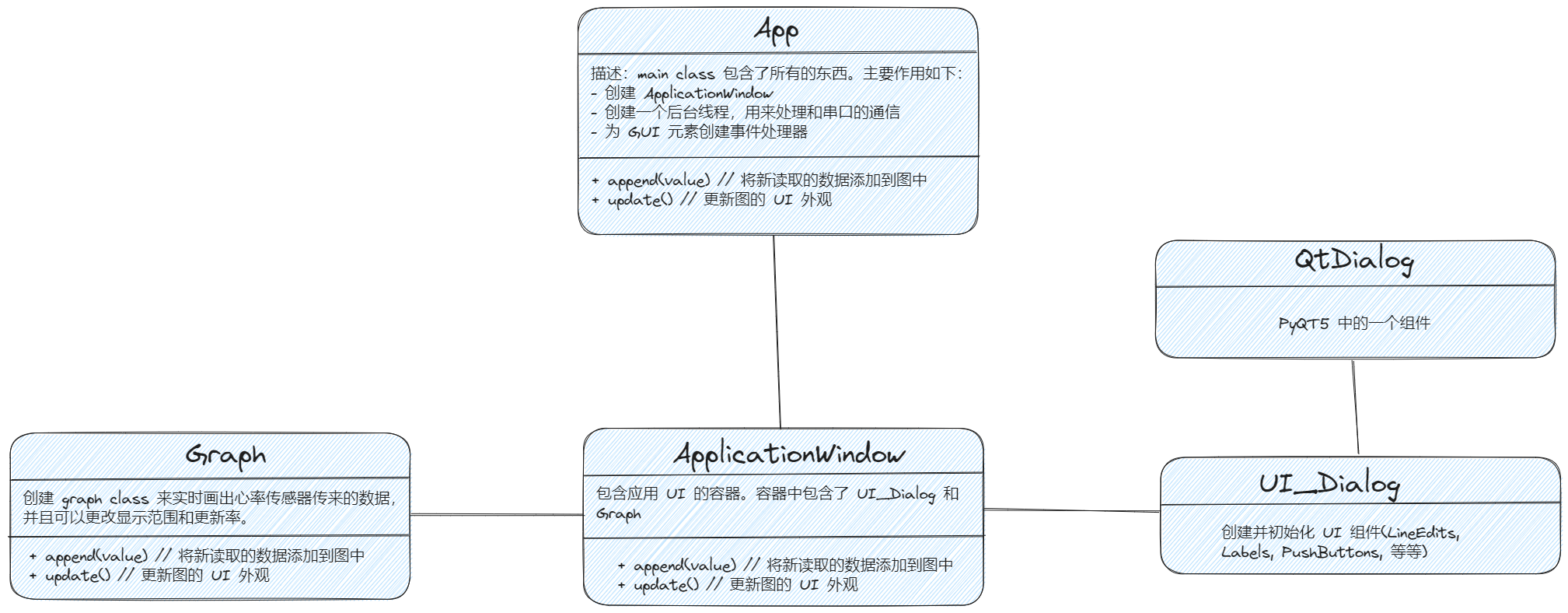
1. 心电监测仪项目架构图

* 1. 硬件架构图



* 1. 软件架构图





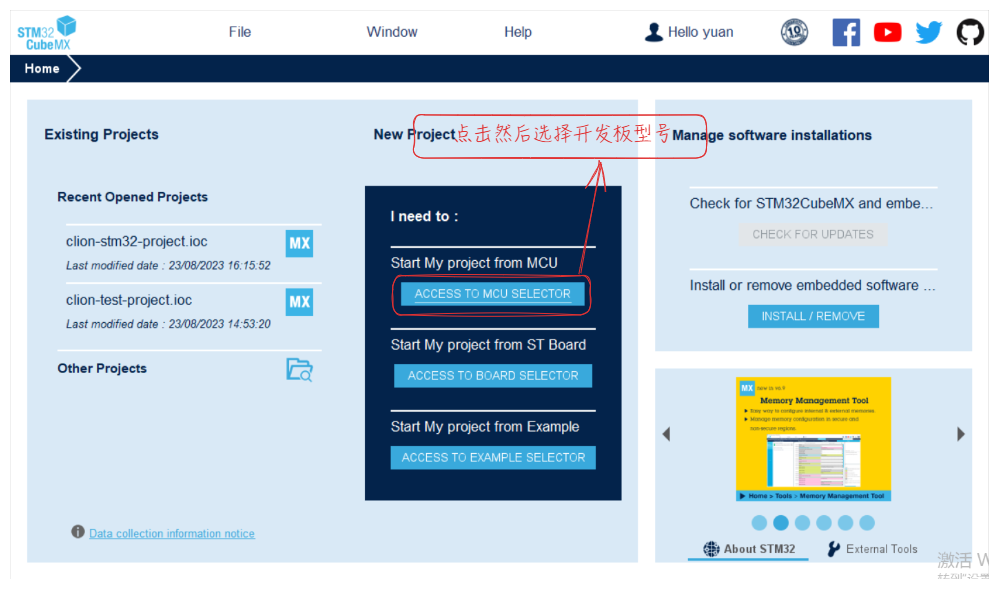
1. 构建开发工具链

传统的 STM32 开发工具一般使用 Keil MDK-ARM IDE 来进行。但 Keil IDE 的编码体验不是很好（只支持 Windows 平台，代码高亮和不全都不好，经常误报错误和警告）。所以我们自己搭建一套开发体验（代码补全，高亮，提示，代码格式化等等）丝滑的工具链。选用的工具如下：

* STM32CubeMX：用来产生包含了 ST 官方 HAL 库的样板代码。非常的方便，基本上是开发 STM32 单片机程序的必备工具。[下载链接](https://www.st.com/zh/development-tools/stm32cubemx.html#overview)
* Clion：JetBrains 公司提供的 C++ IDE。[下载链接](https://www.jetbrains.com/clion/)
* openocd：开源的单片机程序烧写工具。[下载链接](https://gnutoolchains.com/arm-eabi/openocd/)。下载完解压之后，注意将 bin 文件夹的绝对路径添加到环境变量中。打开 PowerShell ，然后输入 openocd -v 确认安装成功。
* gcc-arm-none-eabi：开源的 ARM 指令集编译器。[下载链接](https://developer.arm.com/-/media/Files/downloads/gnu-rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-win32.exe?rev=29bb46cfa0434fbda93abb33c1d480e6&hash=B2C5AAE07841929A0D0BF460896D6E52)。安装完之后，找到安装路径。然后将 bin 文件夹的绝对路径添加到环境变量中。打开 PowerShell ，然后输入 arm-none-eabi-gcc.exe -v 确认安装成功。
* Anaconda：Python集成工具包。用来编写 PC 端的上位机程序。[下载链接](https://www.anaconda.com/download)

* 1. 使用 STM32CubeMX 创建 STM32F103 项目

1. 打开 STM32CubeMX

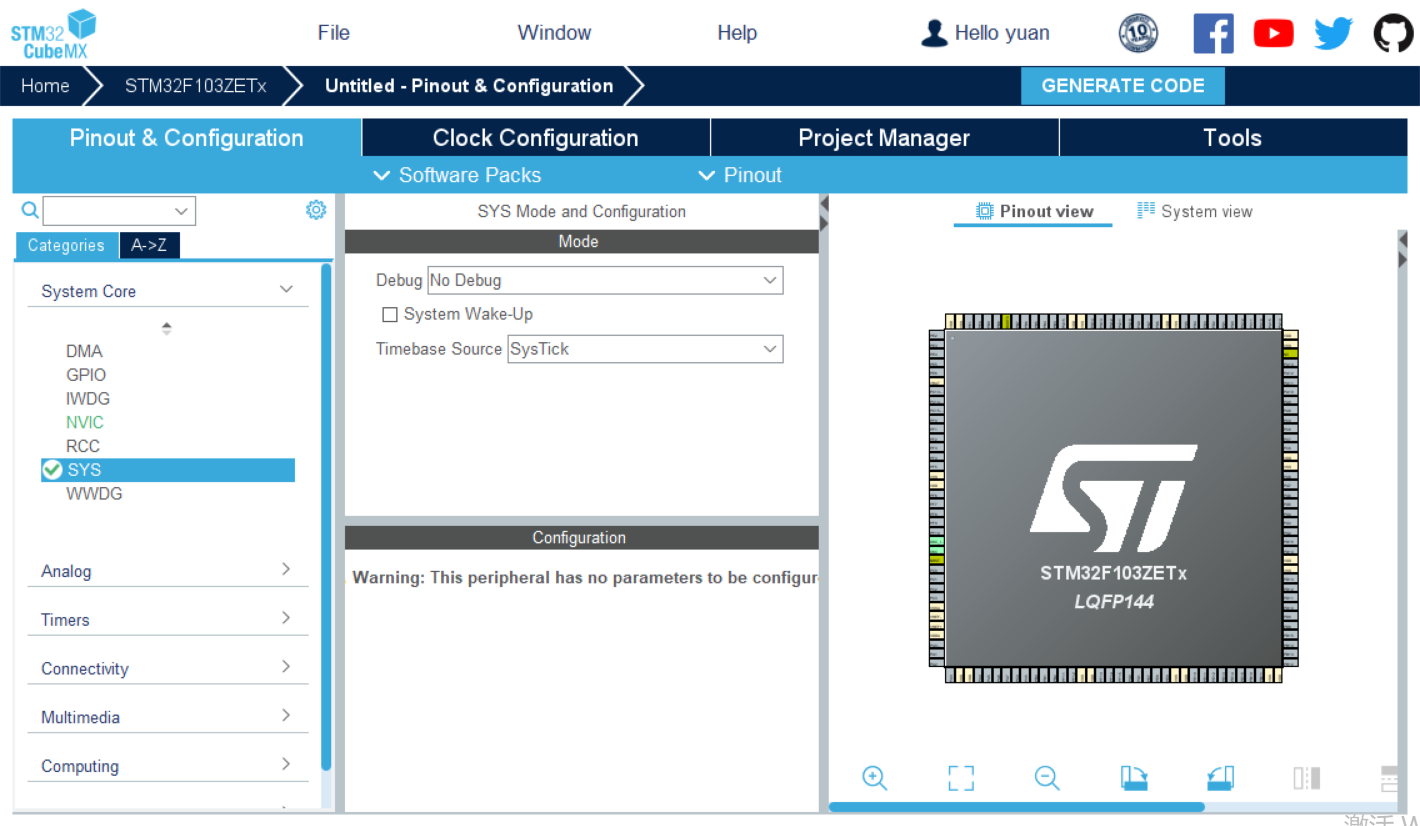


1. 选择要使用的芯片

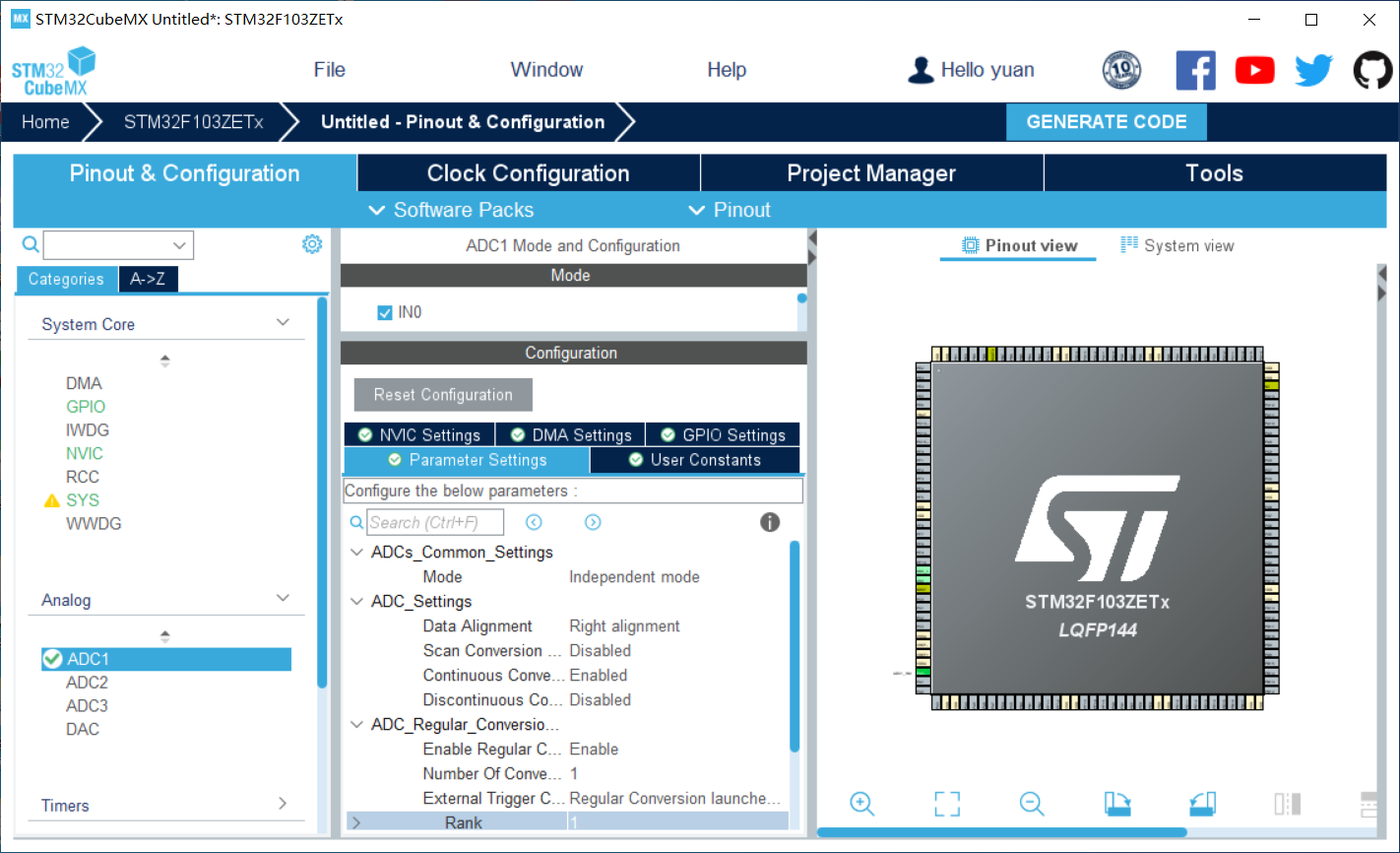


1. 配置 STM32 的系统设置以及一些外设

首先设置单片机的时钟源。

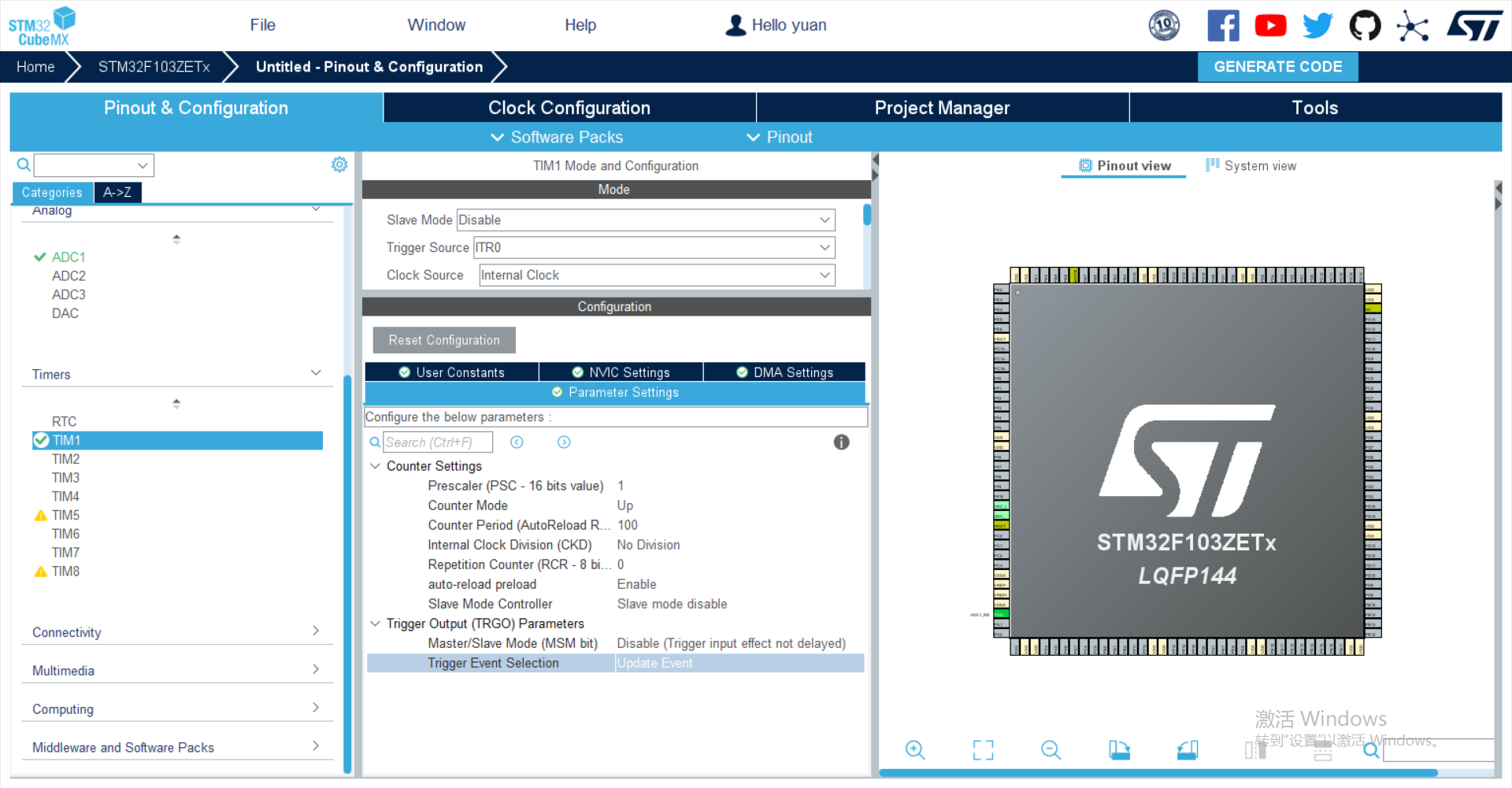


Timebase Source 我们选择 SysTick 。



接下来我们设置模数转换器相关功能，我们选择 STM32 自带的 ADC1 的第 0 个通道 IN0 。

Parameter Settings 如图所示。

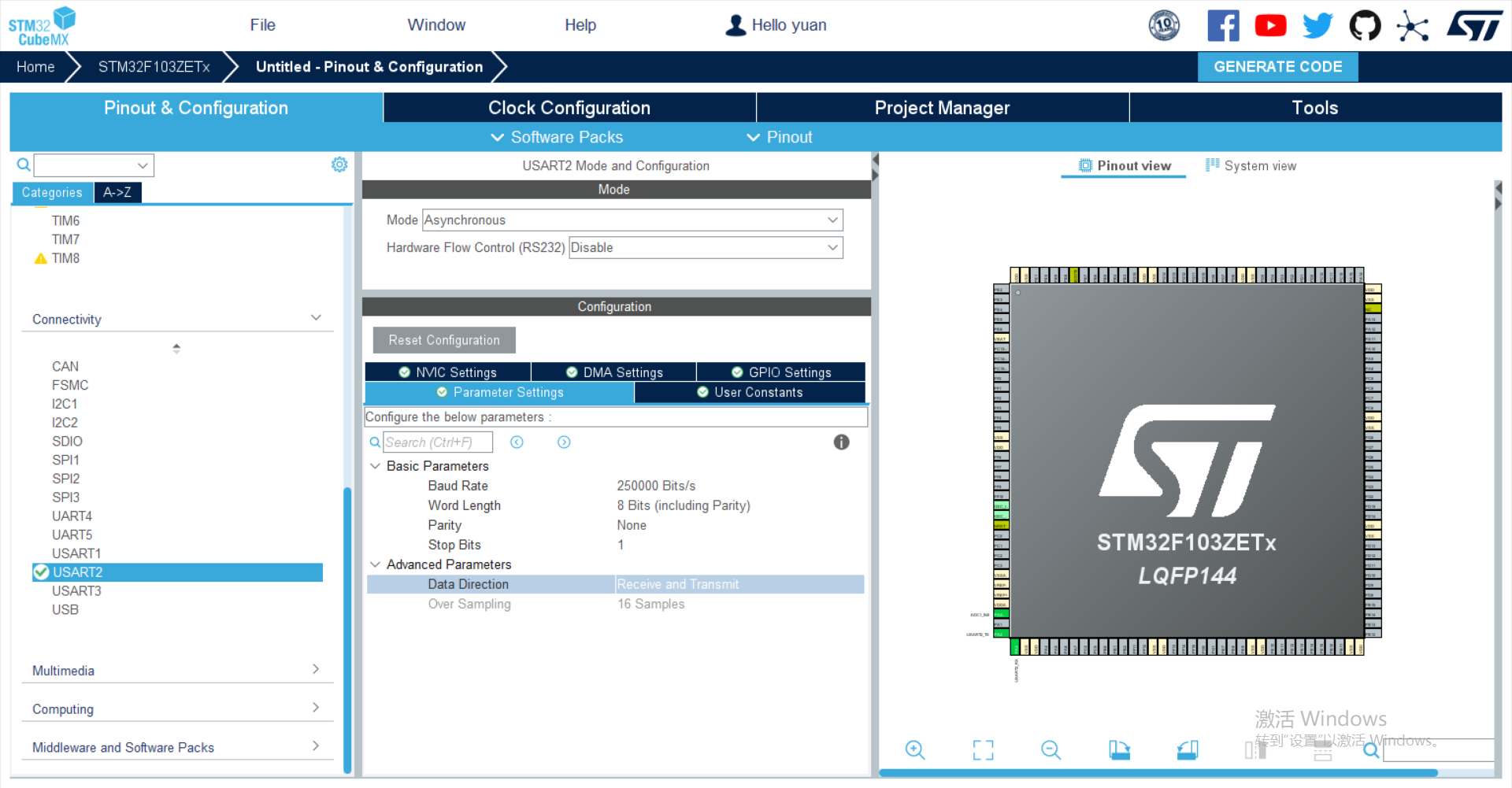


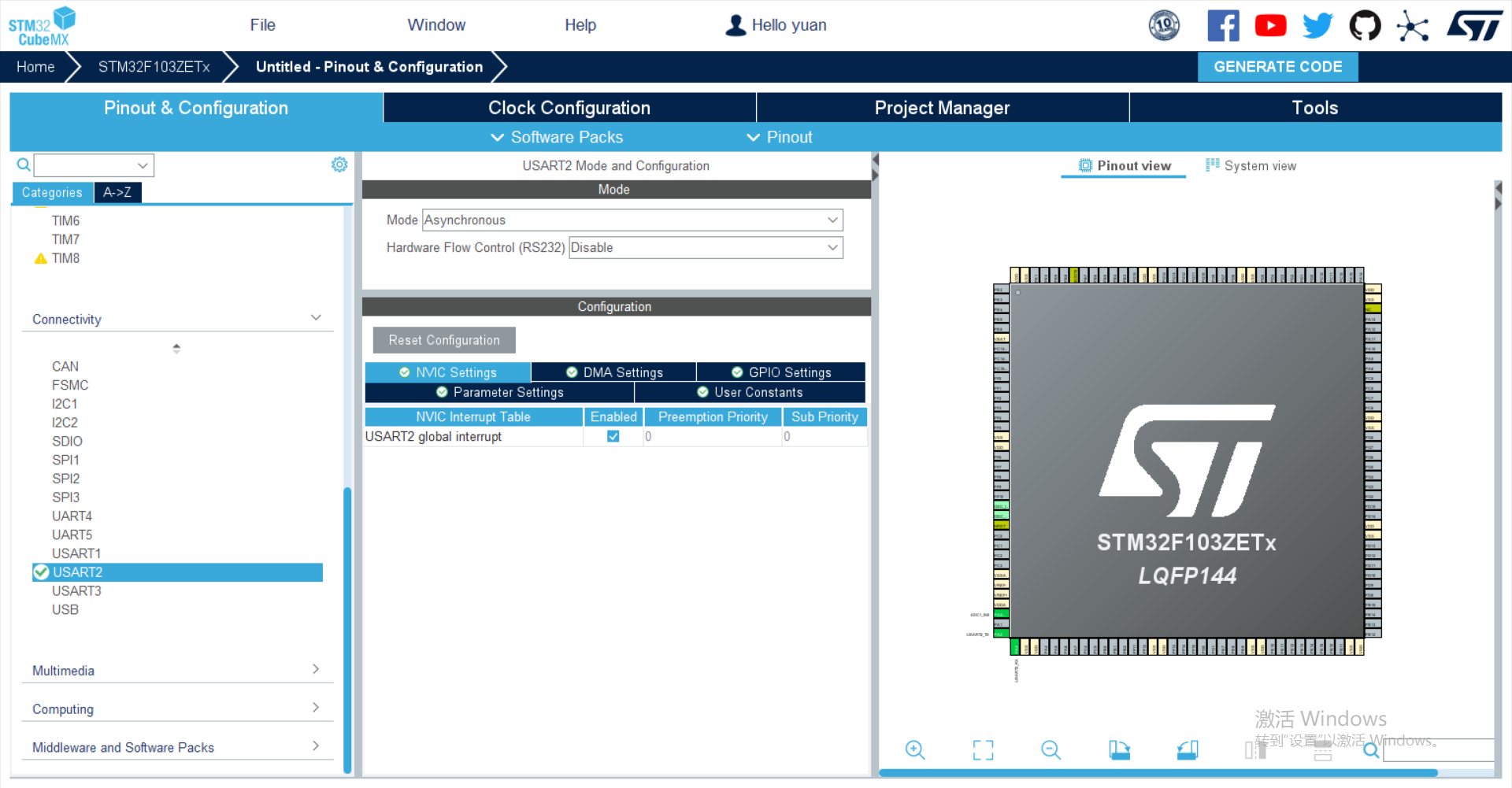
接下来我们配置定时器，我们选用 STM32 提供的定时器1，也就是 TIM1 。

参数设置如图所示。

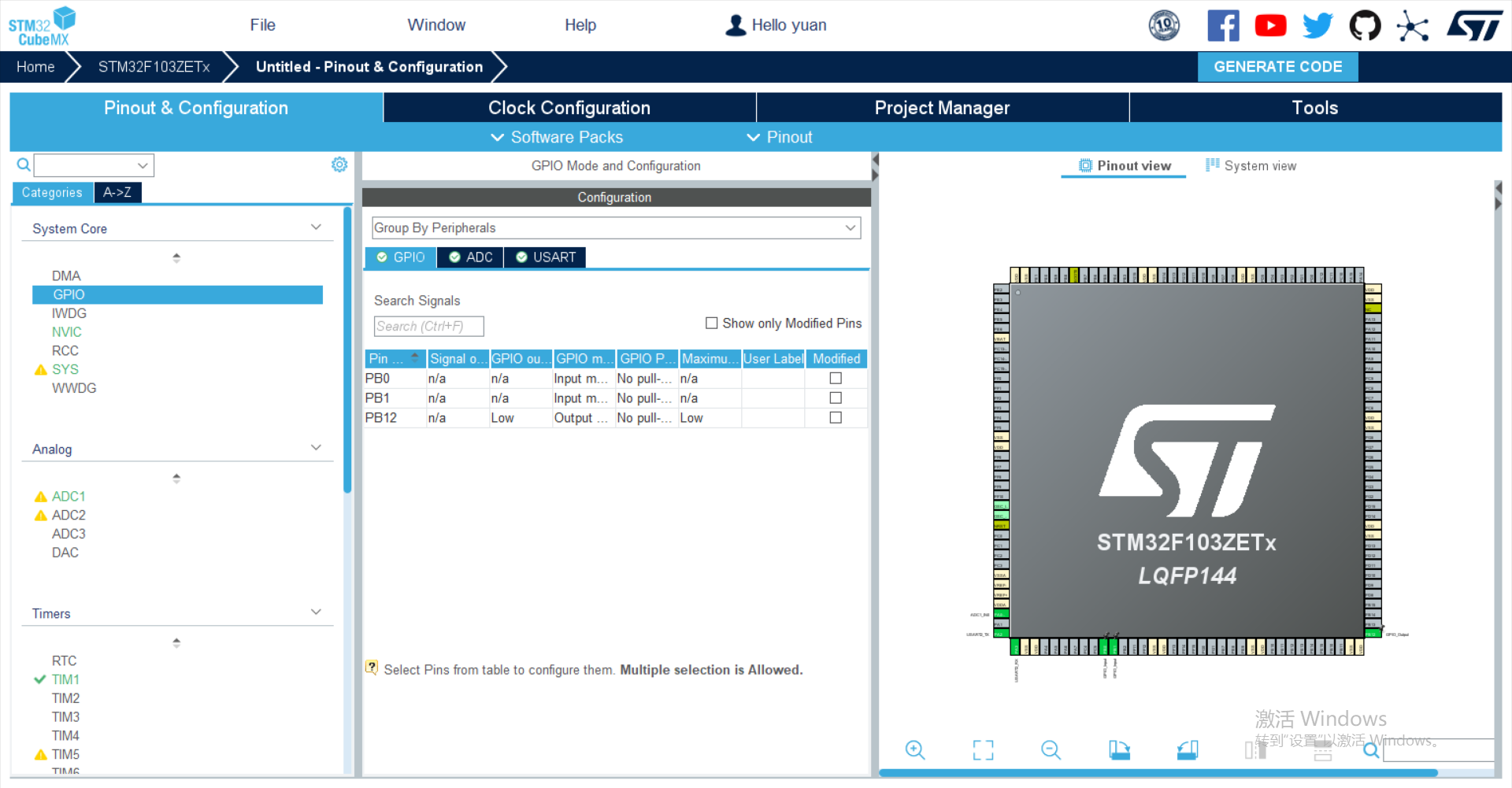
接下来我们配置串口通信相关功能，我们使用 STM32 的 USART2 串口。

打开中断，并将波特率设置为 250000 。

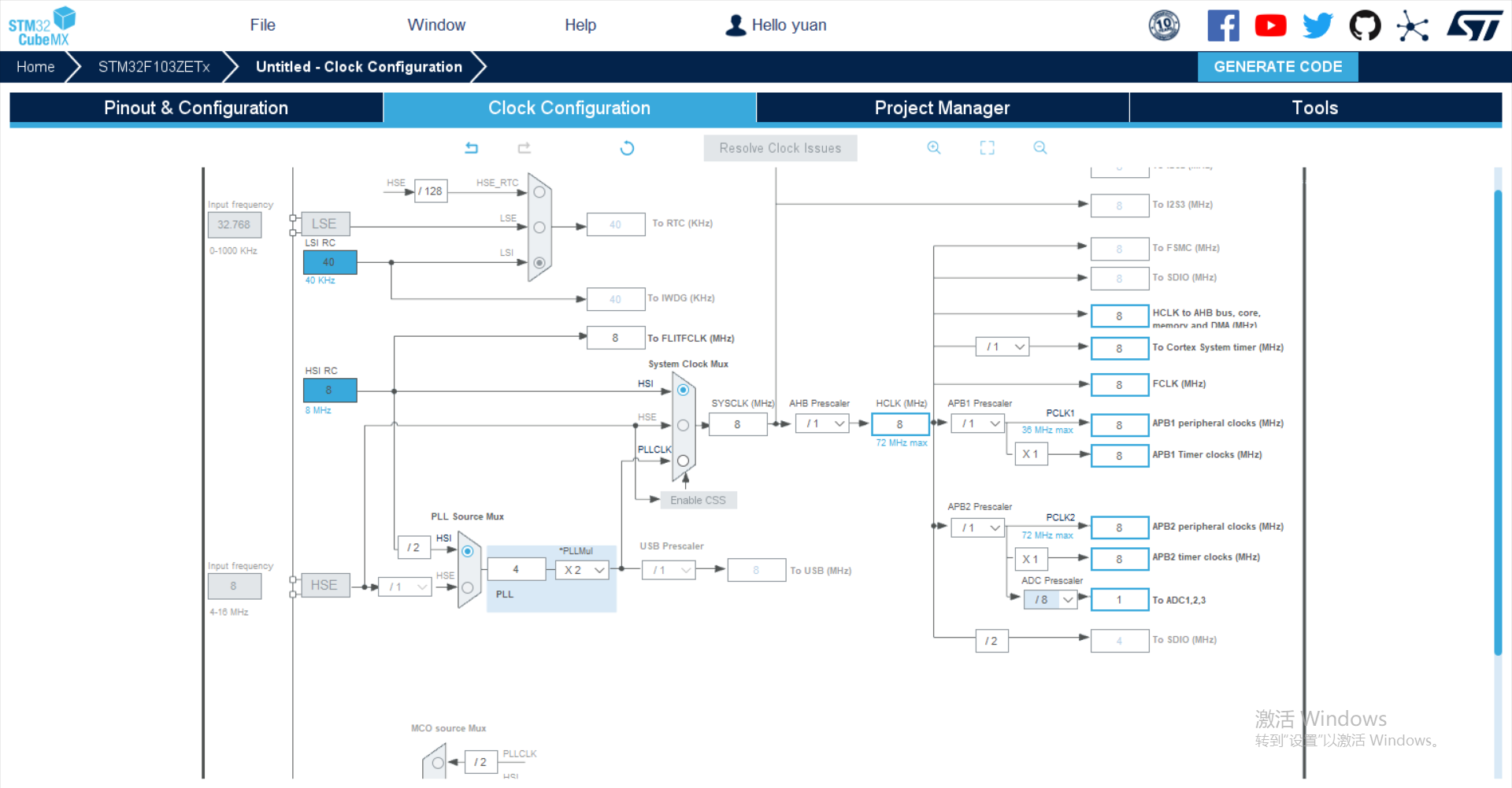




然后将相关的 GPIO PIN 打开并设置 PIN 的各种特性（输入，输出等）。



最后我们配置 STM32 的时钟树。

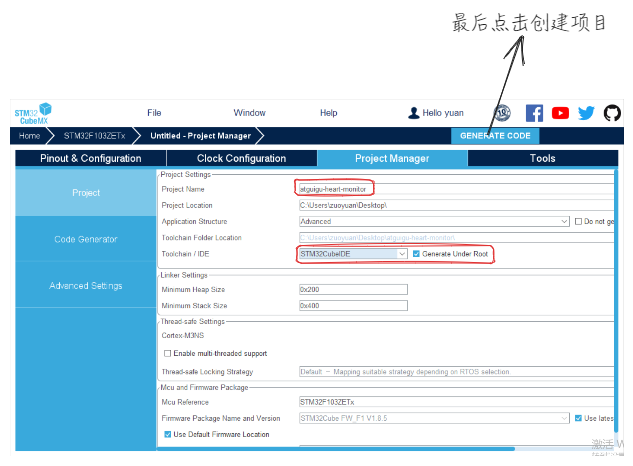


主要是将 ADC Prescaler 设置为 /8 。

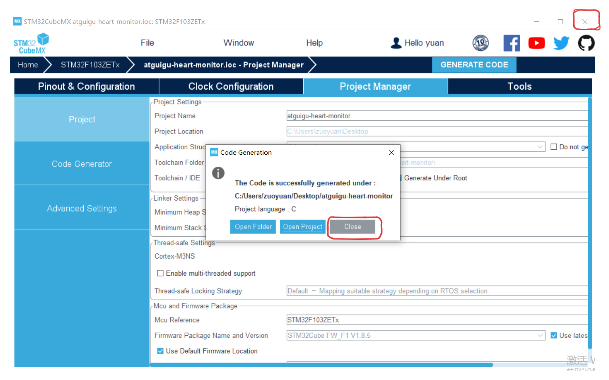
1. 创建项目



然后输入项目名称，并创建项目



1. 关闭 STM32CubeMX



* 1. 编写烧写工具配置文件

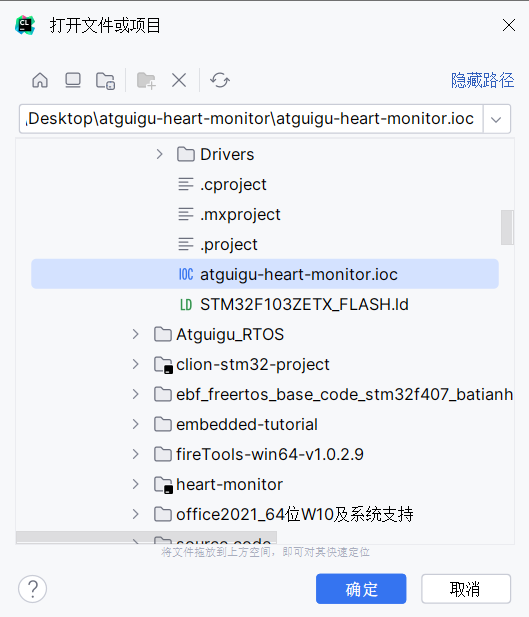
我们使用的仿真器是 cmsis-dap 协议的仿真器。

在路径 解压openocd的绝对路径\openocd-20230712\OpenOCD-20230712-0.12.0\share\openocd\scripts\board 文件夹中新建文件 stm32f1discovery.cfg 文件，在里面写入以下内容：

source [find interface/cmsis-dap.cfg]  
  
transport select swd  
  
# 待烧写单片机是 stm32f1x 型号  
source [find target/stm32f1x.cfg]  
  
# 设置调试器,最大速度为24MHz  
adapter speed 12000

* 1. 使用 Clion 打开 STM32CubeMX 创建的项目

使用 Clion 打开创建的项目中的 .ioc 文件，注意要 **作为项目打开** 。

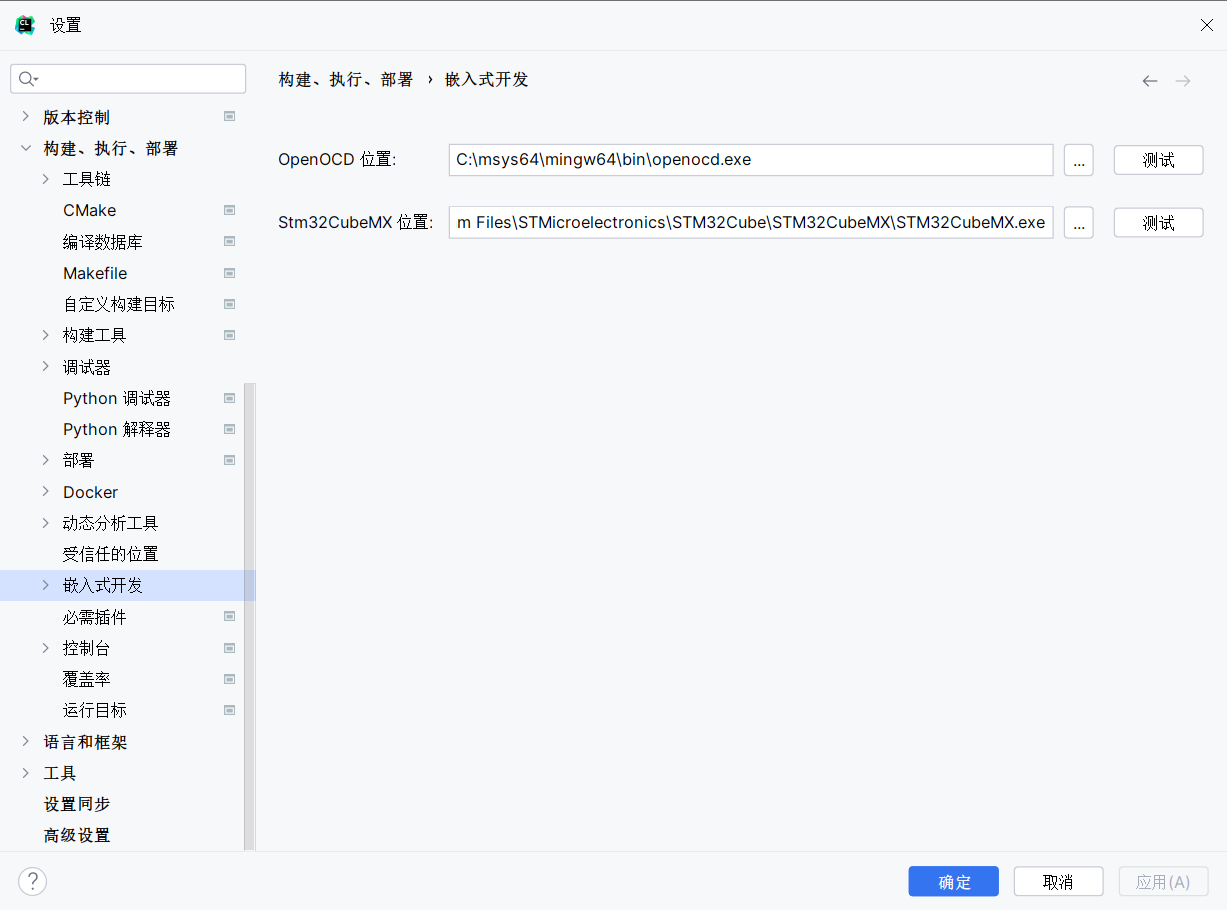




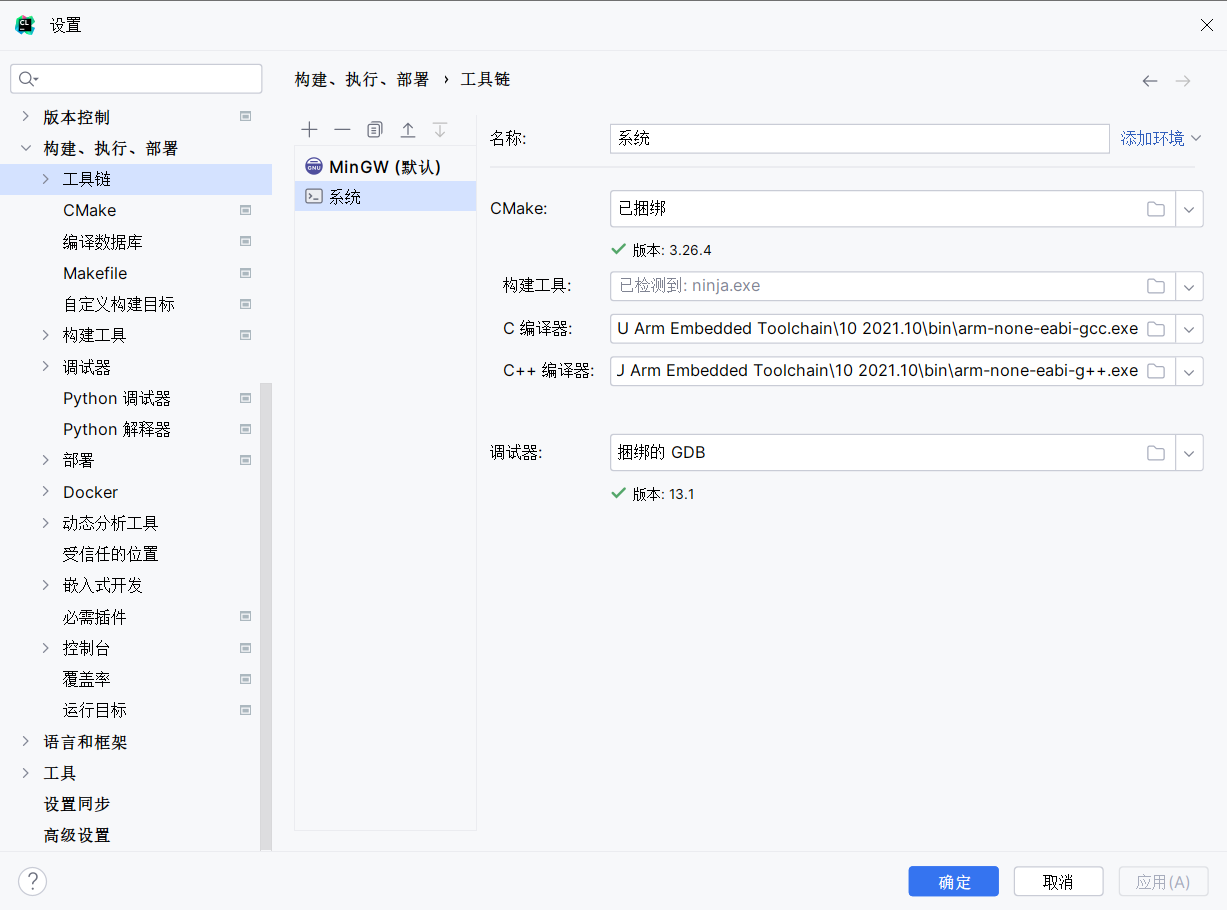
此时会弹出选择 openocd 配置文件的面板，选择 stm32f1discovery.cfg 。

还有两个特别重要的地方需要配置，也就是 openocd 、编译工具链 的配置。

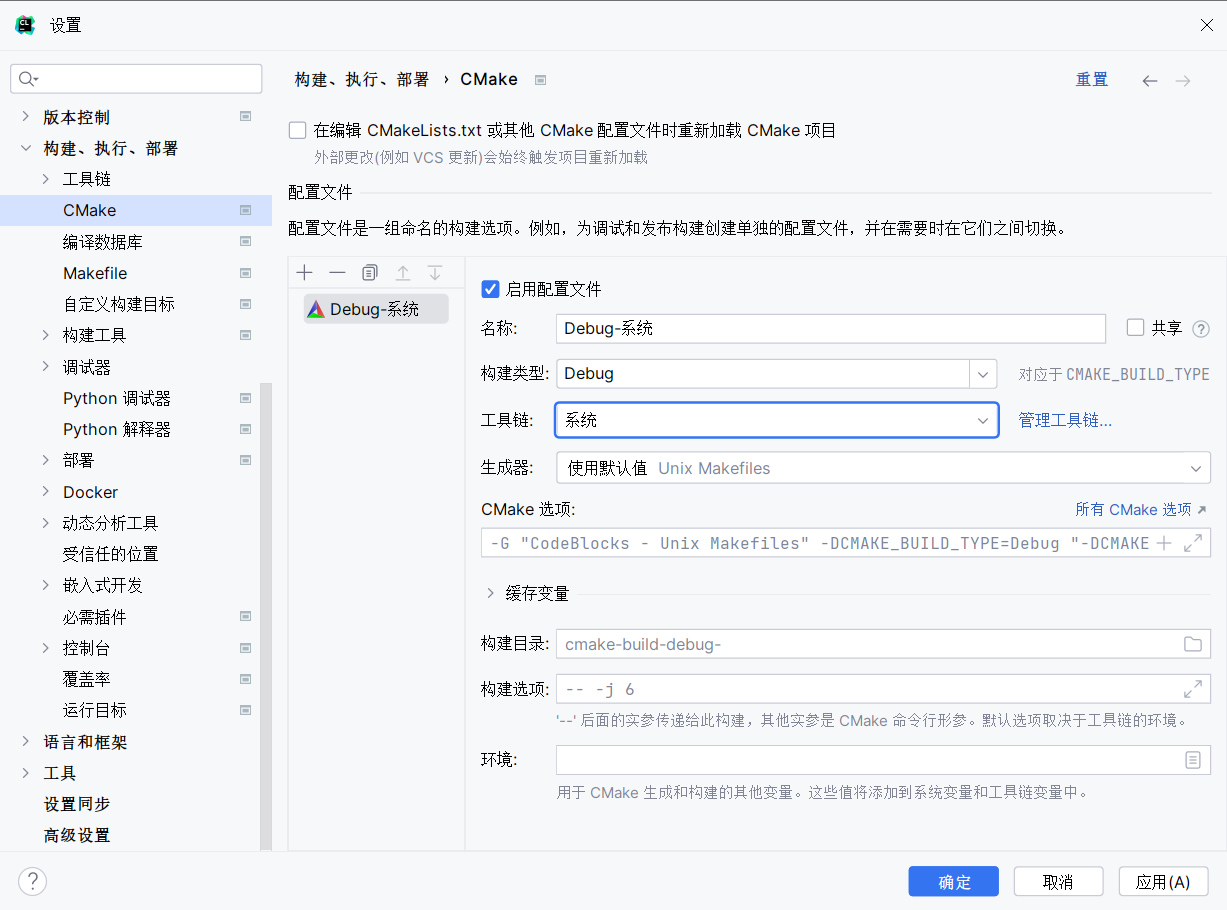
首先检查一下 openocd 和 stm32cubemx 的路径是否正确。



然后新建一个名为 **系统** 的工具链，主要要确保 arm-none-eabi-gcc 和 arm-none-eabi-g++ 的路径是正确的。



最后将 CMake 选项卡的工具链设置为 **系统** 。



然后就可以愉快的写代码了。点击运行看一下样板代码是否可以烧写成功。

1. 心电监测仪项目代码编写

* 1. 下位机代码编写

根据我们的一系列配置，stm32cubemx 帮助我们生成了大部分代码，如果有某些配置我们设置错了，可以直接在代码中改动。接下来我们将要写的代码去掉注释，贴在下面，可以和自动生成的代码对比，我们修改了哪些内容。

我们首先编写初始化 STM32 各种外设的代码。stm32f1xx\_hal\_msp.c 代码内容如下：

#include "main.h"  
  
void HAL\_MspInit(void) {  
 \_\_HAL\_RCC\_AFIO\_CLK\_ENABLE();  
 \_\_HAL\_RCC\_PWR\_CLK\_ENABLE();  
  
 \_\_HAL\_AFIO\_REMAP\_SWJ\_NOJTAG();  
}  
  
void HAL\_ADC\_MspInit(ADC\_HandleTypeDef \*hadc) {  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 if (hadc->Instance == ADC1) {  
 /\* Peripheral clock enable \*/  
 \_\_HAL\_RCC\_ADC1\_CLK\_ENABLE();  
 \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  
 /\*\*ADC1 GPIO Configuration  
 PA0-WKUP ------> ADC1\_IN0  
 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_0;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 HAL\_NVIC\_SetPriority(ADC1\_2\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(ADC1\_2\_IRQn);  
 }  
}  
  
void HAL\_ADC\_MspDeInit(ADC\_HandleTypeDef \*hadc) {  
 if (hadc->Instance == ADC1) {  
 \_\_HAL\_RCC\_ADC1\_CLK\_DISABLE();  
 HAL\_GPIO\_DeInit(GPIOA, GPIO\_PIN\_0);  
 HAL\_NVIC\_DisableIRQ(ADC1\_2\_IRQn);  
 }  
}  
  
void HAL\_TIM\_Base\_MspInit(TIM\_HandleTypeDef \*htim\_base) {  
 if (htim\_base->Instance == TIM1) {  
 \_\_HAL\_RCC\_TIM1\_CLK\_ENABLE();  
 HAL\_NVIC\_SetPriority(TIM1\_UP\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(TIM1\_UP\_IRQn);  
 }  
}  
  
void HAL\_TIM\_Base\_MspDeInit(TIM\_HandleTypeDef \*htim\_base) {  
 if (htim\_base->Instance == TIM1) {  
 \_\_HAL\_RCC\_TIM1\_CLK\_DISABLE();  
 HAL\_NVIC\_DisableIRQ(TIM1\_UP\_IRQn);  
 }  
}  
  
void HAL\_UART\_MspInit(UART\_HandleTypeDef \*huart) {  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 if (huart->Instance == USART2) {  
 \_\_HAL\_RCC\_USART2\_CLK\_ENABLE();  
  
 \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  
 /\*\*USART2 GPIO Configuration  
 PA2 ------> USART2\_TX  
 PA3 ------> USART2\_RX  
 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_2;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;  
 GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_3;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 /\* USART2 interrupt Init \*/  
 HAL\_NVIC\_SetPriority(USART2\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(USART2\_IRQn);  
 }  
}  
  
void HAL\_UART\_MspDeInit(UART\_HandleTypeDef \*huart) {  
 if (huart->Instance == USART2) {  
 \_\_HAL\_RCC\_USART2\_CLK\_DISABLE();  
  
 /\*\*USART2 GPIO Configuration  
 PA2 ------> USART2\_TX  
 PA3 ------> USART2\_RX  
 \*/  
 HAL\_GPIO\_DeInit(GPIOA, GPIO\_PIN\_2 | GPIO\_PIN\_3);  
  
 HAL\_NVIC\_DisableIRQ(USART2\_IRQn);  
 }  
}

然后编写中断函数的代码，代码文件为 stm32f1xx\_it.c ，内容如下：

#include "main.h"  
#include "stm32f1xx\_it.h"  
  
extern ADC\_HandleTypeDef hadc1;  
extern TIM\_HandleTypeDef htim1;  
extern UART\_HandleTypeDef huart2;  
  
void NMI\_Handler(void) {  
}  
  
void HardFault\_Handler(void) {  
 while (1) {  
 }  
}  
  
void MemManage\_Handler(void) {  
 while (1) {  
 }  
}  
  
void BusFault\_Handler(void) {  
 while (1) {  
 }  
}  
  
void UsageFault\_Handler(void) {  
 while (1) {  
 }  
}  
  
void SVC\_Handler(void) {  
}  
  
void DebugMon\_Handler(void) {  
}  
  
void PendSV\_Handler(void) {  
}  
  
// 外设中断  
void ADC1\_2\_IRQHandler(void) {  
 HAL\_ADC\_IRQHandler(&hadc1);  
}

最后我们编写 main.c 文件，里面包含了业务逻辑的代码。

#include "main.h"  
#include "string.h"  
#include "stdlib.h"  
#include "stdio.h"  
  
ADC\_HandleTypeDef hadc1;  
  
TIM\_HandleTypeDef htim1;  
  
UART\_HandleTypeDef huart2;  
  
void SystemClock\_Config(void);  
  
static void MX\_GPIO\_Init(void);  
  
static void MX\_ADC1\_Init(void);  
  
static void MX\_TIM1\_Init(void);  
  
static void MX\_USART2\_UART\_Init(void);  
  
  
// 全局变量的定义  
uint32\_t sample;  
char TxBuffer[8];  
char param\_s[7];  
int param;  
char RxBuffer[8];  
int RxIndex = 0;  
char c;  
  
int ticks = 0;  
int isSending = 0;  
int duration = 0;  
int takeSample = 0;  
int command\_flag = 0;  
  
// 系统时钟频率我们使用的是：8MHz  
void configureSamplingRate(int sampling\_rate) {  
 volatile int prescaler = 128 / sampling\_rate + 1;  
 volatile int load = 8388608 / (sampling\_rate \* prescaler);  
 \_\_HAL\_TIM\_SET\_PRESCALER(&htim1, prescaler);  
 \_\_HAL\_TIM\_SET\_AUTORELOAD(&htim1, load);  
}  
  
void TIM1\_UP\_IRQHandler() {  
 HAL\_TIM\_IRQHandler(&htim1);  
 if (!isSending) return;  
 else if (ticks > 1000 \* duration) {  
 ticks = 0;  
 isSending = 0;  
 }  
 takeSample = 1;  
}  
  
void SysTick\_Handler(void) {  
 HAL\_IncTick();  
 if (isSending) ticks++;  
}  
  
void USART2\_IRQHandler(void) {  
 HAL\_UART\_IRQHandler(&huart2);  
 int result = HAL\_UART\_Receive(&huart2, (uint8\_t \*) &c, 1, 500);  
 if (result == HAL\_OK) {  
 if (c != 0x0A)  
 RxBuffer[RxIndex++] = c;  
 else  
 command\_flag = 1;  
 }  
}  
  
void process\_command() {  
 \_\_HAL\_UART\_DISABLE\_IT(&huart2, UART\_IT\_RXNE);  
 strcpy(param\_s, RxBuffer + 1);  
 param = atoi(param\_s);  
 char command = RxBuffer[0];  
 \_\_HAL\_UART\_ENABLE\_IT(&huart2, UART\_IT\_RXNE);  
 switch (command) {  
 case 's':  
 configureSamplingRate(param);  
 break;  
 case 'c':  
 isSending = 1;  
 duration = param;  
 break;  
 default:  
 break;  
 }  
 memset(RxBuffer, 0, 8);  
 RxIndex = 0;  
}  
  
int main(void) {  
 HAL\_Init();  
  
 SystemClock\_Config();  
  
 HAL\_SYSTICK\_Config(8192);  
  
 MX\_GPIO\_Init();  
 MX\_ADC1\_Init();  
 MX\_TIM1\_Init();  
 MX\_USART2\_UART\_Init();  
  
 HAL\_TIM\_Base\_Start\_IT(&htim1);  
 configureSamplingRate(100);  
 HAL\_ADC\_Start(&hadc1);  
  
 \_\_HAL\_UART\_ENABLE\_IT(&huart2, UART\_IT\_RXNE);  
  
 while (1) {  
 if (takeSample) {  
 HAL\_TIM\_Base\_Stop\_IT(&htim1);  
 takeSample = 0;  
 HAL\_TIM\_Base\_Start\_IT(&htim1);  
 sample = HAL\_ADC\_GetValue(&hadc1);  
 sprintf(TxBuffer, "%04lu\n", (unsigned long) sample);  
 HAL\_UART\_Transmit(&huart2, (uint8\_t \*) TxBuffer, sizeof(TxBuffer), HAL\_MAX\_DELAY);  
 }  
 if (command\_flag) {  
 process\_command();  
 command\_flag = 0;  
 }  
 if (!isSending) {  
 HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_12, 1); //turn off  
 HAL\_TIM\_Base\_Stop(&htim1);  
 HAL\_ADC\_Stop(&hadc1);  
 HAL\_SuspendTick();  
 HAL\_PWR\_EnterSLEEPMode(PWR\_LOWPOWERREGULATOR\_ON, PWR\_SLEEPENTRY\_WFI);  
 HAL\_ResumeTick();  
 SystemClock\_Config();  
 HAL\_TIM\_Base\_Start(&htim1);  
 HAL\_ADC\_Start(&hadc1);  
 HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_12, 0); //turn on  
 }  
 }  
}  
  
/\*\*  
 \* @brief System Clock Configuration  
 \* @retval None  
 \*/  
void SystemClock\_Config(void) {  
 RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};  
 RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};  
 RCC\_PeriphCLKInitTypeDef PeriphClkInit = {0};  
  
 /\*\* Initializes the RCC Oscillators according to the specified parameters  
 \* in the RCC\_OscInitTypeDef structure.  
 \*/  
 RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;  
 RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;  
 RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;  
 RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;  
 if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK) {  
 Error\_Handler();  
 }  
  
 /\*\* Initializes the CPU, AHB and APB buses clocks  
 \*/  
 RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_SYSCLK  
 | RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;  
 RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;  
 RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;  
 RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;  
 RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;  
  
 if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK) {  
 Error\_Handler();  
 }  
 PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC;  
 PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV8;  
 if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief ADC1 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_ADC1\_Init(void) {  
 ADC\_ChannelConfTypeDef sConfig = {0};  
  
 /\*\* Common config  
 \*/  
 hadc1.Instance = ADC1;  
 hadc1.Init.ScanConvMode = ADC\_SCAN\_DISABLE;  
 hadc1.Init.ContinuousConvMode = ENABLE;  
 hadc1.Init.DiscontinuousConvMode = DISABLE;  
 hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;  
 hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;  
 hadc1.Init.NbrOfConversion = 1;  
 if (HAL\_ADC\_Init(&hadc1) != HAL\_OK) {  
 Error\_Handler();  
 }  
  
 /\*\* Configure Regular Channel  
 \*/  
 sConfig.Channel = ADC\_CHANNEL\_0;  
 sConfig.Rank = ADC\_REGULAR\_RANK\_1;  
 sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5;  
 if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief TIM1 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_TIM1\_Init(void) {  
 TIM\_ClockConfigTypeDef sClockSourceConfig = {0};  
 TIM\_MasterConfigTypeDef sMasterConfig = {0};  
  
 htim1.Instance = TIM1;  
 htim1.Init.Prescaler = 1;  
 htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;  
 htim1.Init.Period = 100;  
 htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  
 htim1.Init.RepetitionCounter = 0;  
 htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;  
 if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK) {  
 Error\_Handler();  
 }  
 sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;  
 if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK) {  
 Error\_Handler();  
 }  
  
 sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;  
 sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;  
 if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief USART2 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_USART2\_UART\_Init(void) {  
 huart2.Instance = USART2;  
 huart2.Init.BaudRate = 250000;  
 huart2.Init.WordLength = UART\_WORDLENGTH\_8B;  
 huart2.Init.StopBits = UART\_STOPBITS\_1;  
 huart2.Init.Parity = UART\_PARITY\_NONE;  
 huart2.Init.Mode = UART\_MODE\_TX\_RX;  
 huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  
 huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;  
 if (HAL\_UART\_Init(&huart2) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief GPIO Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_GPIO\_Init(void) {  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
  
 /\* GPIO Ports Clock Enable \*/  
 \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();  
  
 /\*Configure GPIO pin Output Level \*/  
 HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_12, GPIO\_PIN\_RESET);  
  
 /\*Configure GPIO pins : PB0 PB1 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_0 | GPIO\_PIN\_1;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  
  
 /\*Configure GPIO pin : PB12 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_12;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;  
 HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  
}  
  
/\*\*  
 \* @brief This function is executed in case of error occurrence.  
 \* @retval None  
 \*/  
void Error\_Handler(void) {  
}

* 1. 上位机代码编写

首先打开 Anaconda PowerShell Prompt 程序，然后在终端中输入

pip install pyserial

安装 Python 的串口程序包。然后就可以写代码了。

首先编写 ui.py

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'design.ui'  
#  
# Created by: PyQt5 UI code generator 5.14.2  
#  
# WARNING! All changes made in this file will be lost!  
  
  
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_Dialog(object):  
 def setupUi(self, Dialog):  
 Dialog.setObjectName("Dialog")  
 Dialog.resize(681, 407)  
 icon = QtGui.QIcon()  
 icon.addPixmap(QtGui.QPixmap("icon.ico"),  
 QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 Dialog.setWindowIcon(icon)  
 self.communicationGroupBox = QtWidgets.QGroupBox(Dialog)  
 self.communicationGroupBox.setGeometry(QtCore.QRect(10, 260, 211, 141))  
 self.communicationGroupBox.setObjectName("communicationGroupBox")  
 self.portLineEdit = QtWidgets.QLineEdit(self.communicationGroupBox)  
 self.portLineEdit.setGeometry(QtCore.QRect(80, 30, 91, 21))  
 self.portLineEdit.setObjectName("portLineEdit")  
 self.portRefreshButton = QtWidgets.QPushButton(  
 self.communicationGroupBox)  
 self.portRefreshButton.setGeometry(QtCore.QRect(180, 30, 21, 21))  
 self.portRefreshButton.setObjectName("portRefreshButton")  
 self.portLabel = QtWidgets.QLabel(self.communicationGroupBox)  
 self.portLabel.setGeometry(QtCore.QRect(8, 30, 55, 16))  
 self.portLabel.setObjectName("portLabel")  
 self.communicationApplyButton = QtWidgets.QPushButton(  
 self.communicationGroupBox)  
 self.communicationApplyButton.setGeometry(  
 QtCore.QRect(10, 110, 191, 28))  
 self.communicationApplyButton.setObjectName("communicationApplyButton")  
 self.baudRateLineEdit = QtWidgets.QLineEdit(self.communicationGroupBox)  
 self.baudRateLineEdit.setGeometry(QtCore.QRect(80, 60, 121, 22))  
 self.baudRateLineEdit.setObjectName("baudRateLineEdit")  
 self.baudRateLineEdit.setText("250000")  
 self.baudRateLabel = QtWidgets.QLabel(self.communicationGroupBox)  
 self.baudRateLabel.setGeometry(QtCore.QRect(8, 60, 61, 16))  
 self.baudRateLabel.setObjectName("baudRateLabel")  
 self.communicationErrorLabel = QtWidgets.QLabel(  
 self.communicationGroupBox)  
 self.communicationErrorLabel.setGeometry(QtCore.QRect(10, 90, 191, 16))  
 self.communicationErrorLabel.setText("")  
 self.communicationErrorLabel.setObjectName("communicationErrorLabel")  
 self.communicationErrorLabel.setStyleSheet(  
 'QLabel#communicationErrorLabel {color: red}')  
 self.samplingGroupBox = QtWidgets.QGroupBox(Dialog)  
 self.samplingGroupBox.setGeometry(QtCore.QRect(230, 261, 231, 141))  
 self.samplingGroupBox.setObjectName("samplingGroupBox")  
 self.samplingRateLineEdit = QtWidgets.QLineEdit(self.samplingGroupBox)  
 self.samplingRateLineEdit.setGeometry(QtCore.QRect(110, 60, 111, 22))  
 self.samplingRateLineEdit.setObjectName("samplingRateLineEdit")  
 self.samplingRateLineEdit.setText("100")  
 self.samplingRateLabel = QtWidgets.QLabel(self.samplingGroupBox)  
 self.samplingRateLabel.setGeometry(QtCore.QRect(8, 60, 91, 16))  
 self.samplingRateLabel.setObjectName("samplingRateLabel")  
 self.samplingApplyButton = QtWidgets.QPushButton(self.samplingGroupBox)  
 self.samplingApplyButton.setGeometry(QtCore.QRect(10, 110, 211, 28))  
 self.samplingApplyButton.setObjectName("samplingApplyButton")  
 self.BPMlabel = QtWidgets.QLabel(self.samplingGroupBox)  
 self.BPMlabel.setGeometry(QtCore.QRect(10, 30, 91, 16))  
 self.BPMlabel.setObjectName("BPMlabel")  
 self.BPMValue = QtWidgets.QLabel(self.samplingGroupBox)  
 self.BPMValue.setGeometry(QtCore.QRect(110, 30, 91, 16))  
 self.BPMValue.setObjectName("BPMValue")  
 self.samplingErrorLabel = QtWidgets.QLabel(self.samplingGroupBox)  
 self.samplingErrorLabel.setGeometry(QtCore.QRect(10, 90, 211, 16))  
 self.samplingErrorLabel.setText("")  
 self.samplingErrorLabel.setObjectName("samplingErrorLabel")  
 self.samplingErrorLabel.setStyleSheet(  
 'QLabel#samplingErrorLabel {color: red}')  
 self.dataCollectionGroupBox = QtWidgets.QGroupBox(Dialog)  
 self.dataCollectionGroupBox.setGeometry(  
 QtCore.QRect(470, 261, 201, 141))  
 self.dataCollectionGroupBox.setObjectName("dataCollectionGroupBox")  
 self.durationLineEdit = QtWidgets.QLineEdit(  
 self.dataCollectionGroupBox)  
 self.durationLineEdit.setGeometry(QtCore.QRect(140, 60, 51, 22))  
 self.durationLineEdit.setObjectName("durationLineEdit")  
 self.durationLineEdit.setText("60")  
 self.durationLabel = QtWidgets.QLabel(self.dataCollectionGroupBox)  
 self.durationLabel.setGeometry(QtCore.QRect(8, 60, 121, 16))  
 self.durationLabel.setObjectName("durationLabel")  
 self.dataCollectionStartButton = QtWidgets.QPushButton(  
 self.dataCollectionGroupBox)  
 self.dataCollectionStartButton.setGeometry(  
 QtCore.QRect(10, 110, 181, 28))  
 self.dataCollectionStartButton.setObjectName(  
 "dataCollectionStartButton")  
 self.dataCollectionErrorLabel = QtWidgets.QLabel(  
 self.dataCollectionGroupBox)  
 self.dataCollectionErrorLabel.setGeometry(  
 QtCore.QRect(10, 90, 181, 16))  
 self.dataCollectionErrorLabel.setText("")  
 self.dataCollectionErrorLabel.setObjectName("dataCollectionErrorLabel")  
 self.dataCollectionErrorLabel.setStyleSheet(  
 'QLabel#dataCollectionErrorLabel {color: red}')  
 self.graphGroupBox = QtWidgets.QGroupBox(Dialog)  
 self.graphGroupBox.setGeometry(QtCore.QRect(9, 9, 661, 251))  
 self.graphGroupBox.setTitle("")  
 self.graphGroupBox.setObjectName("graphGroupBox")  
  
 self.retranslateUi(Dialog)  
 QtCore.QMetaObject.connectSlotsByName(Dialog)  
  
 Dialog.setWindowIcon(QtGui.QIcon('icon.ico'))  
  
 def retranslateUi(self, Dialog):  
 \_translate = QtCore.QCoreApplication.translate  
 Dialog.setWindowTitle(\_translate("Dialog", "Heart Monitor"))  
 self.communicationGroupBox.setTitle(  
 \_translate("Dialog", "Communication"))  
 self.portLabel.setText(\_translate("Dialog", "COM Port"))  
 self.portRefreshButton.setText(\_translate("Dialog", "⟳"))  
 self.communicationApplyButton.setText(\_translate("Dialog", "Apply"))  
 self.baudRateLabel.setText(\_translate("Dialog", "Baud Rate"))  
 self.portRefreshButton.setText(\_translate("Dialog", "⟳"))  
 self.samplingGroupBox.setTitle(\_translate("Dialog", "Sampling"))  
 self.samplingRateLabel.setText(\_translate("Dialog", "Sampling Rate"))  
 self.samplingApplyButton.setText(\_translate("Dialog", "Apply"))  
 self.BPMlabel.setText(\_translate("Dialog", "BPM"))  
 self.BPMValue.setText(\_translate("Dialog", "20.6"))  
 self.dataCollectionGroupBox.setTitle(  
 \_translate("Dialog", "Data Collection"))  
 self.durationLabel.setText(\_translate("Dialog", "Duration (seconds)"))  
 self.dataCollectionStartButton.setText(\_translate("Dialog", "Start"))

然后编写 appWindow.py 文件

from matplotlib.figure import Figure  
import time  
import numpy as np  
from matplotlib.backends.qt\_compat import QtCore, QtWidgets  
from matplotlib.backends.backend\_qt5agg import (  
 FigureCanvas, NavigationToolbar2QT as NavigationToolbar)  
from ui import Ui\_Dialog  
  
  
class Graph():  
 def \_\_init\_\_(self):  
 self.startingTime = time.time()  
 self.timeData = []  
 self.data = []  
 self.canvas = FigureCanvas(Figure(figsize=(6.4, 2.3)))  
 self.ax = self.canvas.figure.subplots()  
 self.line, = self.ax.plot(self.timeData, self.data)  
 self.ax.set\_ylim(0, 1)  
 self.updateRate = 24  
 self.lastUpdate = time.time()  
 self.timeRange = 2 #shows only the last 5 seconds  
  
 def append(self, value):  
 self.data.append(value)  
 self.timeData.append(time.time()-self.startingTime)  
 if time.time() - self.startingTime - self.timeData[0] > self.timeRange:  
 self.data.pop(0)  
 self.timeData.pop(0)  
 if (time.time() - self.lastUpdate > 1/self.updateRate):  
 self.updateGraph()  
 self.lastUpdate = time.time()  
   
 def updateGraph(self):  
 self.line.set\_xdata(self.timeData)  
 self.line.set\_ydata(self.data)  
 self.ax.set\_xlim(self.timeData[-1] - self.timeRange, self.timeData[-1] + self.timeRange)  
 self.canvas.figure.canvas.draw()  
  
   
  
class ApplicationWindow(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.ui = Ui\_Dialog()  
 self.ui.setupUi(self)  
   
 gwidget = QtWidgets.QWidget(self.ui.graphGroupBox)  
 self.graph = Graph()   
 layout = QtWidgets.QVBoxLayout(gwidget)  
 layout.addWidget(self.graph.canvas)  
   
  
 def appendValue(self, value):  
 self.graph.append(value)

最后编写主窗口程序 main.py

from appWindow import ApplicationWindow  
import sys, time, serial  
from matplotlib.backends.qt\_compat import QtWidgets, QtGui  
from threading import Thread, Event  
import csv  
from serial.tools import list\_ports  
  
def backgroundThread(app, appWindow):  
 while True:  
 if app.serial is None:  
 continue  
 try:  
 val = app.serial.readline().decode()[3:-1]  
 val = int(val) / 4096  
 appWindow.appendValue(val)  
 # print(val)  
 except:  
 print("Not a valid number!")  
   
class App():  
 def \_\_init\_\_(self):   
  
 #communication  
 self.serial = None  
  
 qapp = QtWidgets.QApplication(sys.argv)  
 self.appWindow = ApplicationWindow()  
  
 self.\_setupEventHandlers()  
 self.\_refreshPortHandler()  
  
 thread = Thread(target=backgroundThread, args=(self, self.appWindow,))  
 thread.daemon = True  
 thread.start()  
  
 self.appWindow.show()  
 sys.exit(qapp.exec\_())  
  
 def \_communicationHandler(self, port, baudRate):  
 val = 0  
 try:  
 val = int(baudRate)  
 except:  
 self.appWindow.ui.communicationErrorLabel.setText(  
 "ERROR: Non numerical value!")  
 return  
 if val <= 0:  
 self.appWindow.ui.communicationErrorLabel.setText(  
 "ERROR: Non positive value!")  
 return  
 try:  
 self.serial.close()  
 except:  
 "Opening a new port.."  
 self.serial = serial.Serial(port=port, baudrate=baudRate)  
 self.appWindow.ui.communicationErrorLabel.setText("")  
 print('Communication settings changed!')  
  
 def \_refreshPortHandler(self):  
 ports = list(list\_ports.comports())  
 port = '' if (len(ports) == 0) else ports[0][0]  
 self.appWindow.ui.portLineEdit.setText(port)  
  
 def \_setSamplingRateHandler(self, rate):  
 val = 0  
 try:  
 val = int(rate)  
 except:  
 self.appWindow.ui.samplingErrorLabel.setText("ERROR: Non numerical value!")  
 return  
 if val <= 0:  
 self.appWindow.ui.samplingErrorLabel.setText(  
 "ERROR: Non positive value!")  
 return  
 self.serial.write(f's{rate}\n'.encode())  
 self.appWindow.ui.samplingErrorLabel.setText("")  
 print('Command for setting sampling rate sent!')  
  
 def \_startCollectingHandler(self, duration):  
 val = 0  
 try:  
 val = int(duration)  
 except:  
 self.appWindow.ui.dataCollectionErrorLabel.setText(  
 "ERROR: Non numerical value!")  
 return  
 if val <= 0:  
 self.appWindow.ui.dataCollectionErrorLabel.setText(  
 "ERROR: Non positive value!")  
 return  
 self.serial.write(f'c{duration}\n'.encode())  
 self.appWindow.ui.dataCollectionErrorLabel.setText("")  
 print('Command for collecting data sent!')  
   
  
 # def \_getSelectedBaudRate(self):  
  
 def \_setupEventHandlers(self):  
 self.appWindow.ui.dataCollectionStartButton.clicked.connect(  
 lambda : self.\_startCollectingHandler(self.appWindow.ui.durationLineEdit.text())  
 )  
 self.appWindow.ui.communicationApplyButton.clicked.connect(  
 lambda : self.\_communicationHandler(self.appWindow.ui.portLineEdit.text(), self.appWindow.ui.baudRateLineEdit.text())  
 )  
 self.appWindow.ui.portRefreshButton.clicked.connect(  
 self.\_refreshPortHandler  
 )  
 self.appWindow.ui.samplingApplyButton.clicked.connect(  
 lambda : self.\_setSamplingRateHandler(self.appWindow.ui.samplingRateLineEdit.text())  
 )  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 App()

然后在终端中运行

python main.py

就可以运行上位机程序了。

1. 程序优化：将 ADC 采集的数据通过 DMA 的方式写入内存

下位机代码只需要很少的改动就可以添加 DMA 的功能。

stm32f1xx\_it.c 文件如下：

#include "main.h"  
#include "stm32f1xx\_it.h"  
  
/\* External variables --------------------------------------------------------\*/  
extern DMA\_HandleTypeDef hdma\_adc1;  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* Cortex-M3 Processor Interruption and Exception Handlers \*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*  
 \* @brief This function handles Non maskable interrupt.  
 \*/  
void NMI\_Handler(void) {  
}  
  
/\*\*  
 \* @brief This function handles Hard fault interrupt.  
 \*/  
void HardFault\_Handler(void) {  
 while (1) {  
 }  
}  
  
/\*\*  
 \* @brief This function handles Memory management fault.  
 \*/  
void MemManage\_Handler(void) {  
 while (1) {  
 }  
}  
  
/\*\*  
 \* @brief This function handles Prefetch fault, memory access fault.  
 \*/  
void BusFault\_Handler(void) {  
 while (1) {  
 }  
}  
  
/\*\*  
 \* @brief This function handles Undefined instruction or illegal state.  
 \*/  
void UsageFault\_Handler(void) {  
 while (1) {  
 }  
}  
  
/\*\*  
 \* @brief This function handles System service call via SWI instruction.  
 \*/  
void SVC\_Handler(void) {  
}  
  
/\*\*  
 \* @brief This function handles Debug monitor.  
 \*/  
void DebugMon\_Handler(void) {  
}  
  
/\*\*  
 \* @brief This function handles Pendable request for system service.  
 \*/  
void PendSV\_Handler(void) {  
}  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* STM32F1xx Peripheral Interrupt Handlers \*/  
/\* Add here the Interrupt Handlers for the used peripherals. \*/  
/\* For the available peripheral interrupt handler names, \*/  
/\* please refer to the startup file (startup\_stm32f1xx.s). \*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\*\*  
 \* @brief This function handles DMA1 channel1 global interrupt.  
 \*/  
void DMA1\_Channel1\_IRQHandler(void) {  
 HAL\_DMA\_IRQHandler(&hdma\_adc1);  
}

stm32f1xx\_hal\_msp.c 如下：

#include "main.h"  
  
extern DMA\_HandleTypeDef hdma\_adc1;  
  
void HAL\_MspInit(void) {  
 \_\_HAL\_RCC\_AFIO\_CLK\_ENABLE();  
 \_\_HAL\_RCC\_PWR\_CLK\_ENABLE();  
}  
  
/\*\*  
\* @brief ADC MSP Initialization  
\* This function configures the hardware resources used in this example  
\* @param hadc: ADC handle pointer  
\* @retval None  
\*/  
void HAL\_ADC\_MspInit(ADC\_HandleTypeDef \*hadc) {  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 if (hadc->Instance == ADC1) {  
 /\* Peripheral clock enable \*/  
 \_\_HAL\_RCC\_ADC1\_CLK\_ENABLE();  
  
 \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  
 /\*\*ADC1 GPIO Configuration  
 PA0-WKUP ------> ADC1\_IN0  
 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_0;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_ANALOG;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 /\* ADC1 DMA Init \*/  
 /\* ADC1 Init \*/  
 hdma\_adc1.Instance = DMA1\_Channel1;  
 hdma\_adc1.Init.Direction = DMA\_PERIPH\_TO\_MEMORY;  
 hdma\_adc1.Init.PeriphInc = DMA\_PINC\_DISABLE;  
 hdma\_adc1.Init.MemInc = DMA\_MINC\_ENABLE;  
 hdma\_adc1.Init.PeriphDataAlignment = DMA\_PDATAALIGN\_WORD;  
 hdma\_adc1.Init.MemDataAlignment = DMA\_MDATAALIGN\_WORD;  
 hdma\_adc1.Init.Mode = DMA\_CIRCULAR;  
 hdma\_adc1.Init.Priority = DMA\_PRIORITY\_LOW;  
 if (HAL\_DMA\_Init(&hdma\_adc1) != HAL\_OK) {  
 Error\_Handler();  
 }  
  
 \_\_HAL\_LINKDMA(hadc, DMA\_Handle, hdma\_adc1);  
 }  
  
}  
  
/\*\*  
\* @brief ADC MSP De-Initialization  
\* This function freeze the hardware resources used in this example  
\* @param hadc: ADC handle pointer  
\* @retval None  
\*/  
void HAL\_ADC\_MspDeInit(ADC\_HandleTypeDef \*hadc) {  
 if (hadc->Instance == ADC1) {  
 /\* Peripheral clock disable \*/  
 \_\_HAL\_RCC\_ADC1\_CLK\_DISABLE();  
  
 /\*\*ADC1 GPIO Configuration  
 PA0-WKUP ------> ADC1\_IN0  
 \*/  
 HAL\_GPIO\_DeInit(GPIOA, GPIO\_PIN\_0);  
  
 /\* ADC1 DMA DeInit \*/  
 HAL\_DMA\_DeInit(hadc->DMA\_Handle);  
 }  
}  
  
/\*\*  
\* @brief TIM\_Base MSP Initialization  
\* This function configures the hardware resources used in this example  
\* @param htim\_base: TIM\_Base handle pointer  
\* @retval None  
\*/  
void HAL\_TIM\_Base\_MspInit(TIM\_HandleTypeDef \*htim\_base) {  
 if (htim\_base->Instance == TIM1) {  
 \_\_HAL\_RCC\_TIM1\_CLK\_ENABLE();  
 HAL\_NVIC\_SetPriority(TIM1\_UP\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(TIM1\_UP\_IRQn);  
 }  
  
}  
  
/\*\*  
\* @brief TIM\_Base MSP De-Initialization  
\* This function freeze the hardware resources used in this example  
\* @param htim\_base: TIM\_Base handle pointer  
\* @retval None  
\*/  
void HAL\_TIM\_Base\_MspDeInit(TIM\_HandleTypeDef \*htim\_base) {  
 if (htim\_base->Instance == TIM1) {  
 \_\_HAL\_RCC\_TIM1\_CLK\_DISABLE();  
 HAL\_NVIC\_DisableIRQ(TIM1\_UP\_IRQn);  
 }  
  
}  
  
/\*\*  
\* @brief UART MSP Initialization  
\* This function configures the hardware resources used in this example  
\* @param huart: UART handle pointer  
\* @retval None  
\*/  
void HAL\_UART\_MspInit(UART\_HandleTypeDef \*huart) {  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 if (huart->Instance == USART2) {  
 \_\_HAL\_RCC\_USART2\_CLK\_ENABLE();  
  
 \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  
 /\*\*USART2 GPIO Configuration  
 PA2 ------> USART2\_TX  
 PA3 ------> USART2\_RX  
 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_2;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;  
 GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_3;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 HAL\_NVIC\_SetPriority(USART2\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(USART2\_IRQn);  
 }  
  
}  
  
/\*\*  
\* @brief UART MSP De-Initialization  
\* This function freeze the hardware resources used in this example  
\* @param huart: UART handle pointer  
\* @retval None  
\*/  
void HAL\_UART\_MspDeInit(UART\_HandleTypeDef \*huart) {  
 if (huart->Instance == USART2) {  
 /\* Peripheral clock disable \*/  
 \_\_HAL\_RCC\_USART2\_CLK\_DISABLE();  
  
 /\*\*USART2 GPIO Configuration  
 PA2 ------> USART2\_TX  
 PA3 ------> USART2\_RX  
 \*/  
 HAL\_GPIO\_DeInit(GPIOA, GPIO\_PIN\_2 | GPIO\_PIN\_3);  
 HAL\_NVIC\_DisableIRQ(USART1\_IRQn);  
 }  
  
}

main.c 如下：

#include "main.h"  
#include "stdio.h"  
#include "string.h"  
#include "stdlib.h"  
  
uint32\_t dma\_buffer[1];  
  
ADC\_HandleTypeDef hadc1;  
DMA\_HandleTypeDef hdma\_adc1;  
TIM\_HandleTypeDef htim1;  
UART\_HandleTypeDef huart2;  
  
/\* Private function prototypes -----------------------------------------------\*/  
void SystemClock\_Config(void);  
  
static void MX\_GPIO\_Init(void);  
  
static void MX\_DMA\_Init(void);  
  
static void MX\_ADC1\_Init(void);  
  
static void MX\_TIM1\_Init(void);  
  
static void MX\_USART2\_UART\_Init(void);  
  
uint32\_t sample;  
char TxBuffer[8];  
char param\_s[7];  
int param;  
char RxBuffer[8];  
int RxIndex = 0;  
char c;  
  
// systick  
int ticks = 0;  
int isSending = 0;  
int duration = 0;  
int takeSample = 0;  
int dmaDone = 0;  
int command\_flag = 0;  
  
void configureSamplingRate(int sampling\_rate) {  
 // (2^23/sampling rate)/2^16 = 2^7/sampling rate +1 in case integer division rounds down  
 volatile int prescaler = 128 / sampling\_rate + 1;  
 // #clocks / prescaler = (2^23/sampling rate) / prescaler  
 volatile int load = 8388608 / (sampling\_rate \* prescaler);  
 \_\_HAL\_TIM\_SET\_PRESCALER(&htim1, prescaler);  
 \_\_HAL\_TIM\_SET\_AUTORELOAD(&htim1, load);  
}  
  
void TIM1\_UP\_IRQHandler() {  
 HAL\_TIM\_IRQHandler(&htim1);  
 if (!isSending)  
 return;  
 else if (ticks > 1000 \* duration) {  
 ticks = 0;  
 isSending = 0;  
 }  
 takeSample = 1; // SET flag to true  
}  
  
void SysTick\_Handler(void) {  
 HAL\_IncTick();  
 if (isSending)  
 ticks++;  
}  
  
void USART2\_IRQHandler(void) {  
 HAL\_UART\_IRQHandler(&huart2);  
 int result = HAL\_UART\_Receive(&huart2, (uint8\_t \*) &c, 1, 500);  
 if (result == HAL\_OK) {  
 if (c != 0x0A) { // newline  
 RxBuffer[RxIndex++] = c;  
 } else {  
 command\_flag = 1;  
 }  
 }  
}  
  
void process\_command() {  
 \_\_HAL\_UART\_DISABLE\_IT(&huart2, UART\_IT\_RXNE);  
 strcpy(param\_s, RxBuffer + 1);  
 param = atoi(param\_s);  
 char command = RxBuffer[0];  
 \_\_HAL\_UART\_ENABLE\_IT(&huart2, UART\_IT\_RXNE);  
 switch (command) {  
 case 's':  
 configureSamplingRate(param);  
 break;  
 case 'c':  
 isSending = 1;  
 duration = param;  
 default:  
 break;  
 }  
 memset(RxBuffer, 0, 8);  
 RxIndex = 0;  
}  
  
/\*\*  
 \* @brief The application entry point.  
 \* @retval int  
 \*/  
int main(void) {  
 /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/  
 HAL\_Init();  
  
 SystemClock\_Config();  
  
 HAL\_SYSTICK\_Config(8192);  
  
 /\* Initialize all configured peripherals \*/  
 MX\_GPIO\_Init();  
 MX\_DMA\_Init();  
 MX\_ADC1\_Init();  
 MX\_TIM1\_Init();  
 MX\_USART2\_UART\_Init();  
 /\* USER CODE BEGIN 2 \*/  
  
 HAL\_TIM\_Base\_Start\_IT(&htim1);  
 configureSamplingRate(100);  
 HAL\_ADC\_Start(&hadc1);  
  
 \_\_HAL\_UART\_ENABLE\_IT(&huart2, UART\_IT\_RXNE);  
  
 //HAL\_ADC\_Start\_DMA(&hadc1, dma\_buffer, 1);  
  
 /\* Infinite loop \*/  
 /\* USER CODE BEGIN WHILE \*/  
 while (1) {  
 if (takeSample) {  
 HAL\_TIM\_Base\_Stop\_IT(&htim1);  
 takeSample = 0;  
 HAL\_ADC\_Start\_DMA(&hadc1, dma\_buffer, 1);  
 HAL\_TIM\_Base\_Start\_IT(&htim1);  
 }  
 if (dmaDone) {  
 sprintf(TxBuffer, "%04lu\n", (unsigned long) dma\_buffer[0]);  
 HAL\_UART\_Transmit(&huart2, (uint8\_t \*) TxBuffer, sizeof(TxBuffer), HAL\_MAX\_DELAY);  
 dmaDone = 0;  
 }  
 if (command\_flag) {  
 process\_command();  
 command\_flag = 0;  
 }  
 if (!isSending) {  
 HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_12, 1); // turn off  
 HAL\_TIM\_Base\_Stop(&htim1);  
 HAL\_ADC\_Stop(&hadc1);  
 HAL\_SuspendTick();  
 HAL\_PWR\_EnterSLEEPMode(PWR\_LOWPOWERREGULATOR\_ON, PWR\_SLEEPENTRY\_WFI);  
 HAL\_ResumeTick();  
 SystemClock\_Config();  
 HAL\_TIM\_Base\_Start(&htim1);  
 HAL\_ADC\_Start(&hadc1);  
 HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_12, 0); // turn on  
 }  
 /\* USER CODE BEGIN 3 \*/  
 }  
 /\* USER CODE END 3 \*/  
}  
  
/\*\*  
 \* @brief System Clock Configuration  
 \* @retval None  
 \*/  
void SystemClock\_Config(void) {  
 RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};  
 RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};  
 RCC\_PeriphCLKInitTypeDef PeriphClkInit = {0};  
  
 /\*\* Initializes the RCC Oscillators according to the specified parameters  
 \* in the RCC\_OscInitTypeDef structure.  
 \*/  
 RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;  
 RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;  
 RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;  
 RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;  
 if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK) {  
 Error\_Handler();  
 }  
  
 /\*\* Initializes the CPU, AHB and APB buses clocks  
 \*/  
 RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_SYSCLK  
 | RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;  
 RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;  
 RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;  
 RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;  
 RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;  
  
 if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK) {  
 Error\_Handler();  
 }  
 PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC;  
 PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV8;  
 if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief ADC1 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_ADC1\_Init(void) {  
 ADC\_ChannelConfTypeDef sConfig = {0};  
  
 /\*\* Common config  
 \*/  
 hadc1.Instance = ADC1;  
 hadc1.Init.ScanConvMode = ADC\_SCAN\_ENABLE;  
 hadc1.Init.ContinuousConvMode = DISABLE;  
 hadc1.Init.DiscontinuousConvMode = DISABLE;  
 hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;  
 hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;  
 hadc1.Init.NbrOfConversion = 1;  
 if (HAL\_ADC\_Init(&hadc1) != HAL\_OK) {  
 Error\_Handler();  
 }  
  
 /\*\* Configure Regular Channel  
 \*/  
 sConfig.Channel = ADC\_CHANNEL\_0;  
 sConfig.Rank = ADC\_REGULAR\_RANK\_1;  
 sConfig.SamplingTime = ADC\_SAMPLETIME\_1CYCLE\_5;  
 if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief TIM1 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_TIM1\_Init(void) {  
 TIM\_ClockConfigTypeDef sClockSourceConfig = {0};  
 TIM\_MasterConfigTypeDef sMasterConfig = {0};  
  
 htim1.Instance = TIM1;  
 htim1.Init.Prescaler = 0;  
 htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;  
 htim1.Init.Period = 100;  
 htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  
 htim1.Init.RepetitionCounter = 0;  
 htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;  
 if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK) {  
 Error\_Handler();  
 }  
 sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;  
 if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK) {  
 Error\_Handler();  
 }  
 sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;  
 sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;  
 if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief USART2 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_USART2\_UART\_Init(void) {  
 huart2.Instance = USART2;  
 huart2.Init.BaudRate = 250000;  
 huart2.Init.WordLength = UART\_WORDLENGTH\_8B;  
 huart2.Init.StopBits = UART\_STOPBITS\_1;  
 huart2.Init.Parity = UART\_PARITY\_NONE;  
 huart2.Init.Mode = UART\_MODE\_TX\_RX;  
 huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  
 huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;  
 if (HAL\_UART\_Init(&huart2) != HAL\_OK) {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* Enable DMA controller clock  
 \*/  
static void MX\_DMA\_Init(void) {  
 /\* DMA controller clock enable \*/  
 \_\_HAL\_RCC\_DMA1\_CLK\_ENABLE();  
  
 /\* DMA interrupt init \*/  
 /\* DMA1\_Channel1\_IRQn interrupt configuration \*/  
 HAL\_NVIC\_SetPriority(DMA1\_Channel1\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(DMA1\_Channel1\_IRQn);  
}  
  
/\*\*  
 \* @brief GPIO Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_GPIO\_Init(void) {  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
  
 \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();  
  
 /\*Configure GPIO pin Output Level \*/  
 HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_12, GPIO\_PIN\_RESET);  
  
 /\*Configure GPIO pins : PB0 PB1 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_0 | GPIO\_PIN\_1;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  
  
 /\*Configure GPIO pin : PB12 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_12;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;  
 HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  
}  
  
void HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef \*hadc) {  
 dmaDone = 1;  
}  
  
/\*\*  
 \* @brief This function is executed in case of error occurrence.  
 \* @retval None  
 \*/  
void Error\_Handler(void) {  
 /\* USER CODE BEGIN Error\_Handler\_Debug \*/  
 /\* User can add his own implementation to report the HAL error return state \*/  
 \_\_disable\_irq();  
 while (1) {  
 }  
 /\* USER CODE END Error\_Handler\_Debug \*/  
}  
  
#ifdef USE\_FULL\_ASSERT  
/\*\*  
 \* @brief Reports the name of the source file and the source line number  
 \* where the assert\_param error has occurred.  
 \* @param file: pointer to the source file name  
 \* @param line: assert\_param error line source number  
 \* @retval None  
 \*/  
void assert\_failed(uint8\_t \*file, uint32\_t line)  
{  
 /\* USER CODE BEGIN 6 \*/  
 /\* User can add his own implementation to report the file name and line number,  
 ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/  
 /\* USER CODE END 6 \*/  
}  
#endif /\* USE\_FULL\_ASSERT \*/

1. 将心电数据采集到上位机以后，进行滤波

由于人的心跳频率范围在 50Hz ~ 200Hz 之间，而单导联采集的信号包含了 50Hz 的工频噪声，所以可以在上位机将噪声滤除。

我们使用 50Hz ~ 200Hz 的带通滤波器来对信号进行滤波。Python 代码如下：

import numpy as np  
import matplotlib.pyplot as plt  
  
data\_list = []  
  
with open("data.txt", "r") as f:  
 for line in f.readlines():  
 data\_list.append(float(line))  
  
y = np.array(data\_list)  
  
x = np.linspace(0, 1000, 1000)  
  
plt.plot(x,y[:1000])  
  
from scipy.signal import butter, filtfilt  
  
# 信号采样率和截止频率  
sampling\_rate = 2000  
cutoff\_freq = 50  
  
# 设计高通滤波器  
nyquist\_freq = sampling\_rate / 2  
normalized\_cutoff\_freq = cutoff\_freq / nyquist\_freq  
b, a = butter(4, [0.05, 0.2], btype='bandpass', analog=False)  
  
# 应用滤波器  
filtered\_signal = filtfilt(b, a, y[:1000])  
  
plt.plot(x, filtered\_signal)

滤波前的信号图：



滤波后的信号图：

