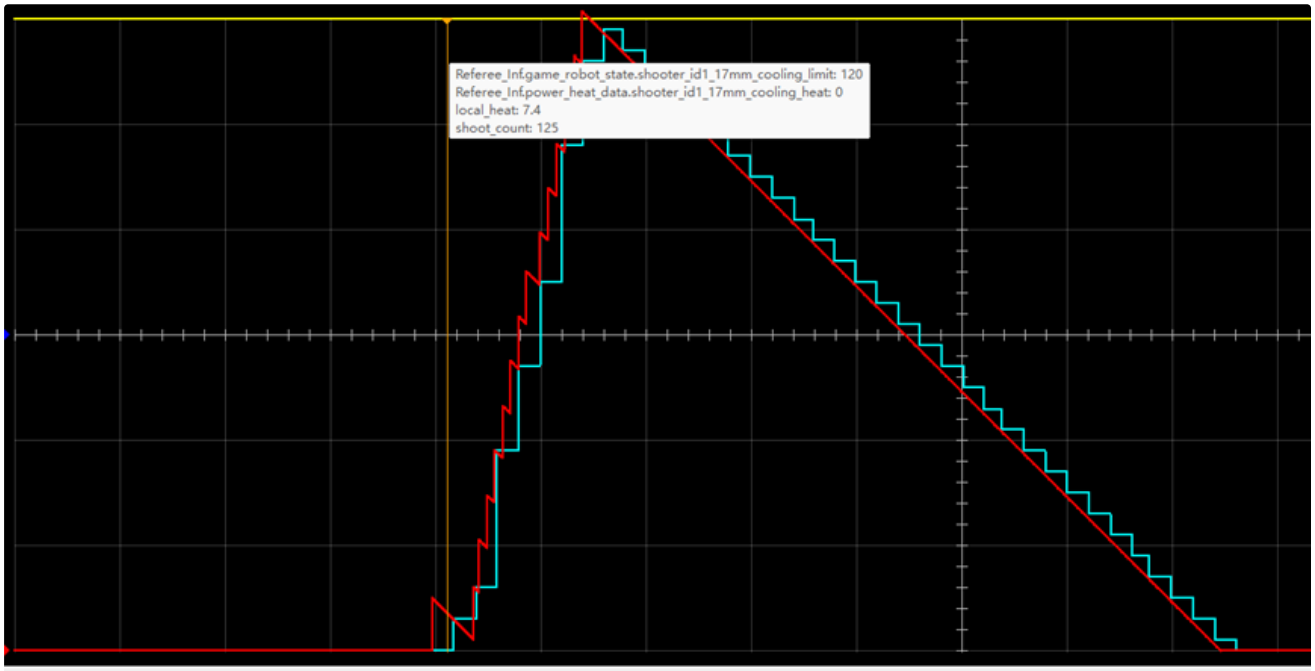
本地热量管理V3

采用摩擦轮转速变化作为发弹判断依据

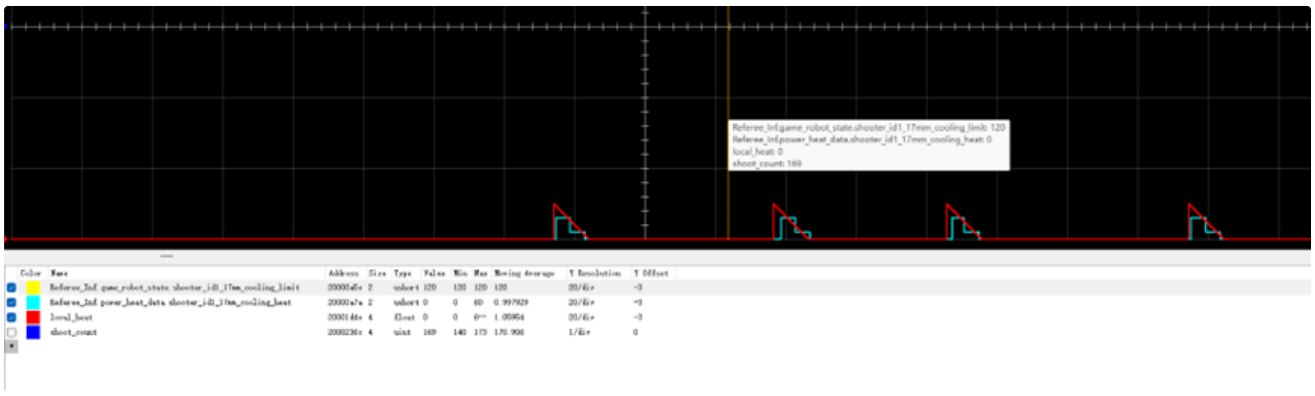
数据处理：

效果：

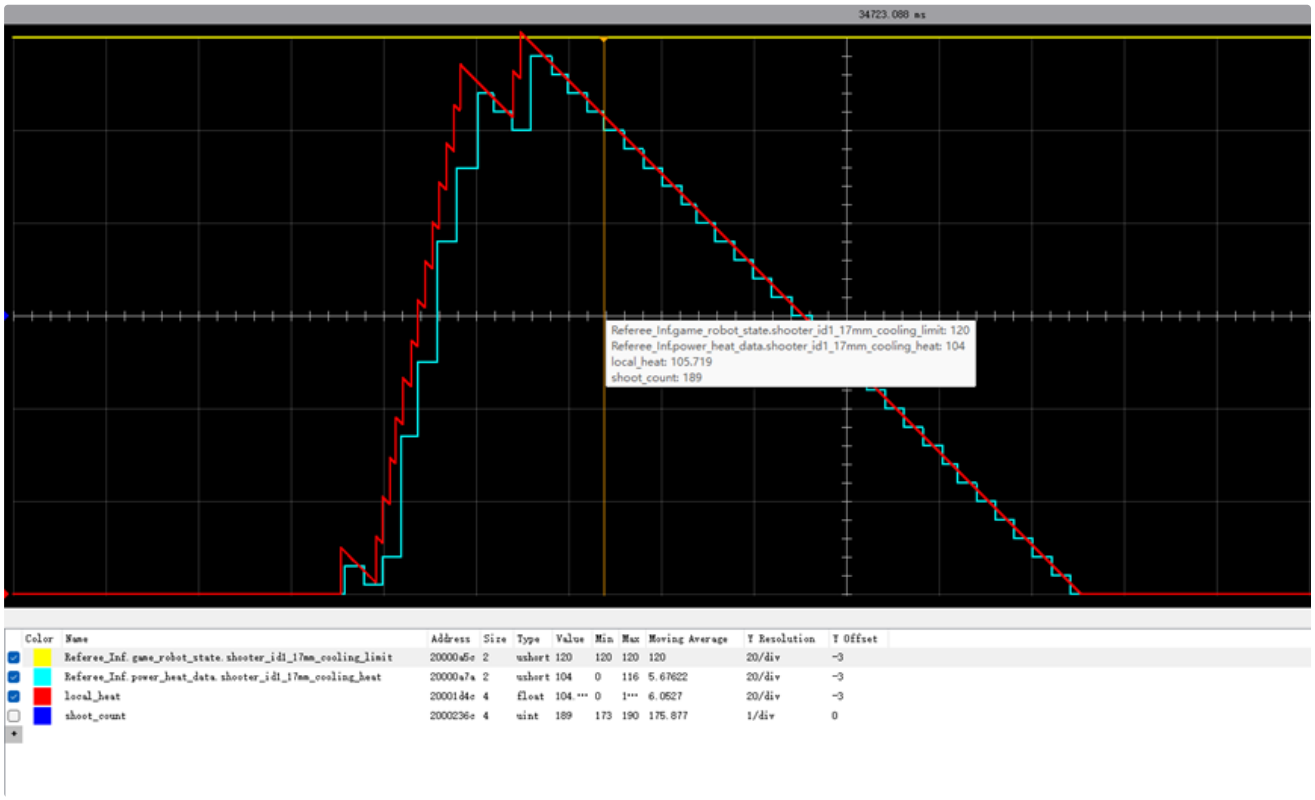
18m/s



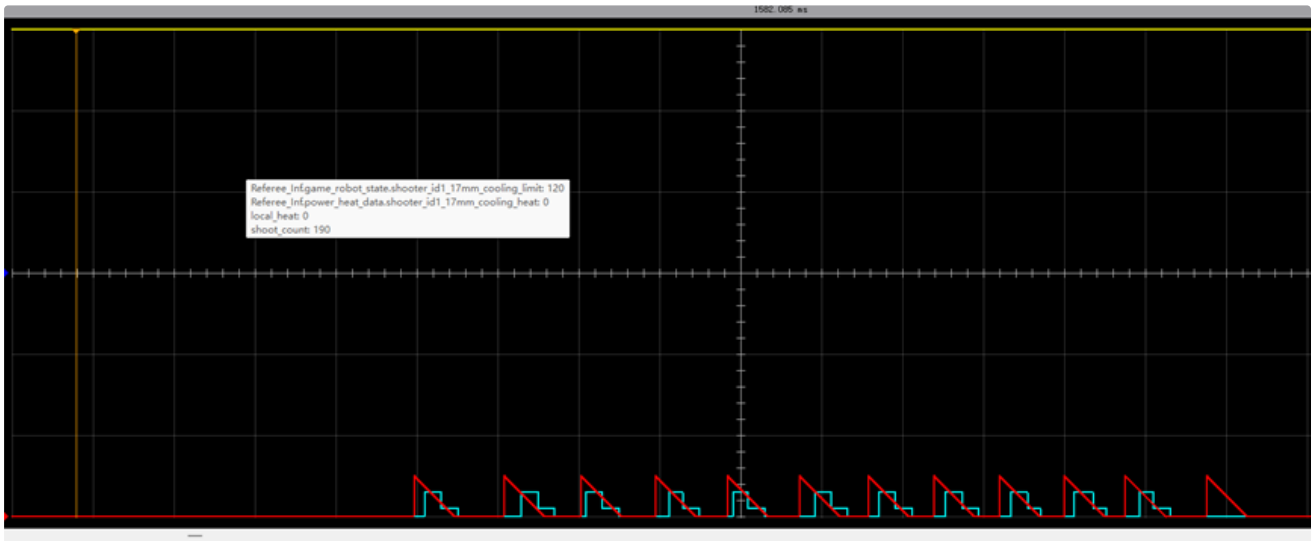
Color	Name	Address	Size	Type	Value	Min	Max	Moving Average	Y Resolution	Y Offset
	Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_limit	20000a5c	2	ushort	120	120	120	120	20/div	-3
	Referee_Inf.power_heat_data.shooter_id1_17mm_cooling_heat	20000a7a	2	ushort	0	0	118	8.74469	20/div	-3
	local_heat	20001d4c	4	float	7.24	0	1···	8.74001	20/div	-3
	shoot_count	2000236c	4	uint	125	29	140	104.171	1/div	0



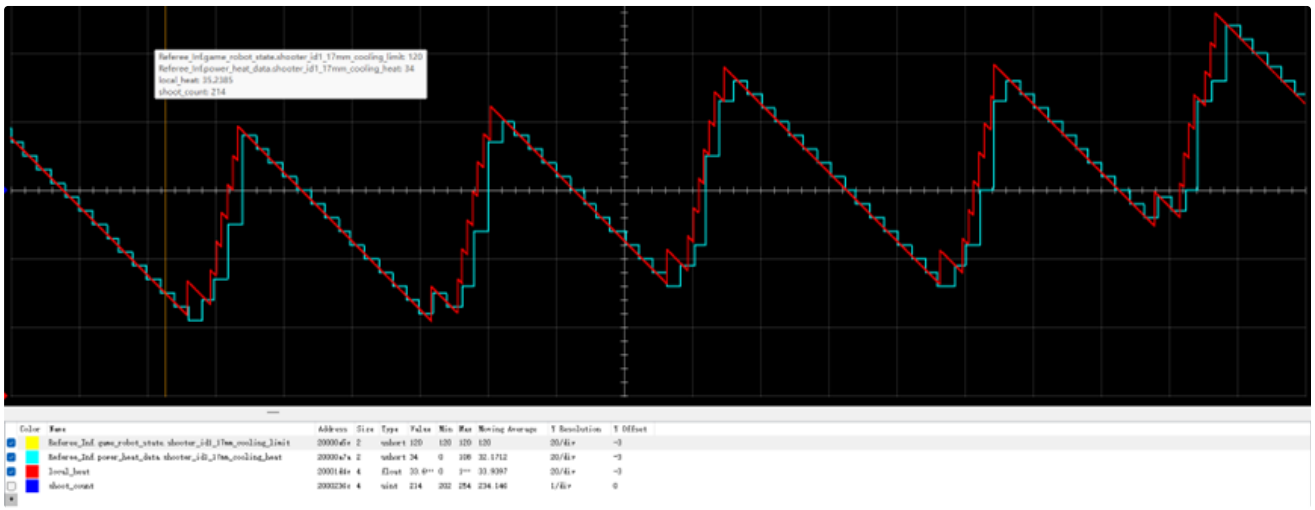
30m/s



Color	Name	Address	Size	Type	Value	Min	Max	Moving Average	Y Resolution	Y Offset
	Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_limit	20000a5c	2	ushort	120	120	120	120	20/div	-3
	Referee_Inf.power_heat_data.shooter_id1_17mm_cooling_heat	20000a7a	2	ushort	104	0	116	5.67622	20/div	-3
	local_heat	20001d4c	4	float	104.***	0	1***	6.0527	20/div	-3
	shoot_count	2000236c	4	uint	189	173	190	175.877	1/div	0



Color	Name	Address	Size	Type	Value	Min	Max	Moving Average	Y Resolution	Y Offset
	Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_limit	20000a5c	2	ushort	120	120	120	120	20/div	-3
	Referee_Inf.power_heat_data.shooter_id1_17mm_cooling_heat	20000a7a	2	ushort	0	0	6	0.816475	20/div	-3
	local_heat	20001d4c	4	float	0	0	10	1.40656	20/div	-3
	shoot_count	2000236c	4	uint	190	190	202	195.03	1/div	0



Color	Name	Address	Size	Type	Value	Min	Max	Moving Average	Y Resolution	Y Offset
	Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_limit	20000a5c	2	ushort	120	120	120	120	20/div	-3
	Referee_Inf.power_heat_data.shooter_id1_17mm_cooling_heat	20000a7a	2	ushort	34	0	108	32.1712	20/div	-3
	local_heat	20001d4c	4	float	33.6***	0	1***	33.9397	20/div	-3
	shoot_count	2000236c	4	uint	214	202	254	234.146	1/div	0

主函数(可加在shoot_task末尾)

```c
uint32_t shoot_count = 0;
void Shoot_Fric_data_process(void)
{
    /*-----------------------------------变量常量-----------------------------------*/
    static bool bullet_waiting_confirm = false;
// 等待比较器确认
    uint8_t shoot_speed = Referee_Inf.game_robot_state.shooter_id1_17mm_speed_limit; // 获取弹速
    int16_t data = CAN_Shoot[1].Current_Speed;
// 获取摩擦轮转速
    static uint16_t data_histroy[MAX_HISTROY];
// 做循环队列
    static uint8_t head = 0, rear = 0;
// 队列下标
    float moving_average[2];
// 移动平均滤波
    uint8_t data_num;
// 循环队列元素个数
    float derivative;
// 微分
    /*-----------------------------------逻辑控制-----------------------------------*/
    data = abs(data);
    /*入队*/
    data_histroy[head] = data;
    head++;
    head %= MAX_HISTROY;
    /*判断队列数据量*/
    data_num = (head - rear + MAX_HISTROY) % MAX_HISTROY;
    if (data_num ⩾ Fliter_windowSize + 1) // 队列数据量满足要求
    {
        moving_average[0] = 0;
        moving_average[1] = 0;
        /*同时计算两个滤波*/
        for (uint8_t i = rear, j = rear + 1, index = rear; index < rear + Fliter_windowSize; i++, j++, index++)
        {
            i %= MAX_HISTROY;
            j %= MAX_HISTROY;
            moving_average[0] += data_histroy[i];
            moving_average[1] += data_histroy[j];
        }
        moving_average[0] ⩵ Fliter_windowSize;
        moving_average[1] ⩵ Fliter_windowSize;
        /*滤波求导*/
        derivative = moving_average[1] - moving_average[0];
        /*导数比较*/
        if (derivative < -shoot_speed * 2)
        {
            bullet_waiting_confirm = true;
        }
```

```
        else if (derivative > -shoot_speed * 1.35)
        {
            if (bullet_waiting_confirm == true)
            {
                local_heat += One_bullet_heat; // 确认打出
                shoot_count++;
                bullet_waiting_confirm = false;
            }
        }
        rear++;
        rear %= MAX_HISTROY;
    }
}
```

主函数执行放到main.c定时器中

```
/* USER CODE BEGIN Callback 1 */
if (htim->Instance == TIM6)
{      yuanshuacpu, 7个月前 • first commit
  /*-----------------------------------------热量控制部分-----------------------------------------*/
  local_heat -= (Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_rate / 1000.0f); // 1000Hz冷却
  if (local_heat < 0)
  {
    local_heat = 0;
  }
  Shoot_Fric_data_process();
  /*-----------------------------------------IMU部分-----------------------------------------*/
  mpu_data.imu_tick_counts++;
  Yaw_Angle += mpu_data.processed_data.wz * 0.0576f;
}
/* USER CODE END Callback 1 */
```

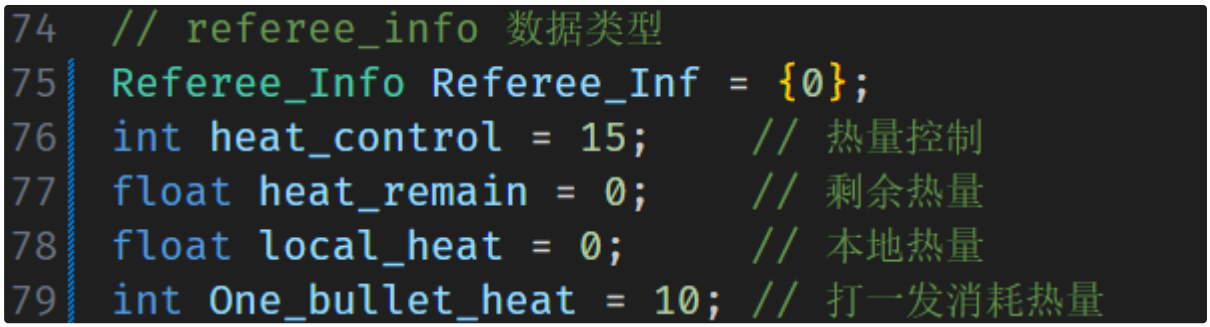
```
/* USER CODE BEGIN Callback 1 */
  if (htim->Instance == TIM6)
  {
    /*-----------------------------------------热量控制部分-----------------------------------------*/
    local_heat -= (Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_rate / 1000.0f); // 1000Hz冷却
    if (local_heat < 0)
    {
      local_heat = 0;
    }
    Shoot_Fric_data_process();
    /*-----------------------------------------IMU部分-----------------------------------------*/
    mpu_data.imu_tick_counts++;
    Yaw_Angle += mpu_data.processed_data.wz * 0.0576f;
  }
  /* USER CODE END Callback 1 */
```

热量控制部分

```
    if (Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_limit - local_heat <= heat_control) // 剩余热量小于留出
    {
        Shoot_Speed = 0;
    }
```

```
if (Referee_Inf.game_robot_state.shooter_id1_17mm_cooling_limit - local_heat ⩽ heat_control) // 剩余热量小于留出的热量
    {
        Shoot_Speed = 0;
    }
```

其他一些变量的初始化




```
// referee_info 数据类型
Referee_Info Referee_Inf = {0};
int heat_control = 20;     // 热量控制
float heat_remain = 0;     // 剩余热量
float local_heat = 0;      // 本地热量
int One_bullet_heat = 10; // 打一发消耗热量
```

heat_control的值经过测试，20是个不错的选项

heat_control的值经过测试，20是个不错的选项

heat_control的值经过测试，20是个不错的选项

每个车摩擦轮体质不一样，如果有必要，需要采集摩擦轮转速进行分析