文章编号:1671-4598(2022)07-0246-09 DOI:10.16526/j.cnki.11-4762/tp.2022.07.037 中图分类号:TP36 文献标识码:B

激光飞秒手势识别算法建模及其简易导航微系统实现

怯肇乾，史继峰
（南昌职业大学 信息技术学院，南昌 330500）

摘要：为了解决演示场合展示办公应用的高性价比手势导航产品系统的需求，提出并阐述了基于高性能的激光飞秒测距传感和人工智能深度学习自适应进行手势识别的算法模型，做到了准确可靠、工效卓著和低成本，有力的突破了现有传统手势识别的局限；以此激光飞秒手势识别为核心，结合 Lora 短距离无线传输、WinUSB 便捷接口和视窗控制软件处理的组合优势，独辟蹊径，实现了高性价比的简易导航微产品系统，使手势识别走向大众化常规应用，达到了教学、论坛、会展、会议等演示办公廉价便捷运用的期求。

关键词：激光飞秒测距；人工智能深度学习；BP_NN；手势识别算法模型；LoRa 长距离无线通信；WinUSB 接口；视窗展示控制

Modeling of Laser Femtosecond Gesture Recognition Algorithm and Implementation of Simple Navigation Microsystem

KAI Zhaoqian, SHI Jifeng
(Institute of Information Technology, Nanchang Vocational University, Nanchang 330500, China)

Abstract: In order to solve the demand of the high cost performance gesture navigation product system for the office application in demonstration occasions, an algorithm model of the gesture recognition based on high-performance laser femtosecond ranging sensor and artificial intelligence deep learning adaptation is proposed and described, which is accurate, reliable, efficient and low-cost, and effectively breaks through the limitations of existing traditional gesture recognition. The laser femtosecond gesture recognition is taken as the core, combined with the combined advantages of the Lora short-distance wireless transmission, the WinUsb convenient interface and window control software processing, a unique way is used to realize a cost-effective simple navigation micro product system, which makes the gesture recognition popular and conventional applications, and meets the expectation of cheap and convenient use of demonstration offices such as teaching, forum, exhibition and meeting.

Keywords: laser femtosecond ranging; artificial intelligence deep learning; back propagation neural networks; gesture recognition; LoRa long-range wireless communication; WinUsb interface; window display control

0 引言

现有手势识别导航，最常见的是红外收发/分析识别，虽然经典，但距离短，通常最远只有 150 mm，若作为论坛、会展、会议等演示场合应用，使用局限，不现实际；更有传统的激光笔导航，虽然简便，但需要正对接收操作，客户体验不佳；还有重大国际/国家类论坛、会展中运用的摄像识别，尽管操控便捷而且技术高大上，但价格不匪，难以拓展应用到广泛的课堂教学等寻常场景。

教学、论坛、会展、会议市场需求空间巨大，寻找和建立更为可靠、准确、高效、廉价的检测传感器和手势识别算法模型意义重大。经反复查阅和对比，最终聚焦了激光飞秒测距 ToF（Time of Flight）传感器和“人工智能 AI（Artificial Intelligence）深度学习自适应”算法建模，于是有了这个省厅科技项目——激光飞秒手势导航简易微系统实现。

1 激光飞秒手势识别算法建模

采用高性价比的激光飞秒测距 ToF 传感器，得到原始距离数据。客户体验考虑，使用 3 个传感器呈上、左、右布局［避免 1 个传感器不合常规的特定手势定义与强制学习］，以此 3 路距离数据组成采集数据输入。以最简洁的“人工智能 AI（Artificial Intelligence）深度学习自适应”算法模型——“后向反馈神经元网络 BP_NN（Back Propagation Neural Networks）”为核心，辅以“智能模糊控制”和“专家经验控制”，构成“运算处理分析”中心，最终得

收稿日期:2022-01-28； **修回日期**:2022-03-11。
基金项目:江西省教育厅科研项目(GJJ206302)。
作者简介:怯肇乾(1969-),男,河南汝州人,教授,高级工程师,设计师/培训师、架构师,主要从事软硬件体系及其大数据监控系统软件设计实现方向的研究。
引用格式:怯肇乾,史继峰.激光飞秒手势识别算法建模及其简易导航微系统实现[J].计算机测量与控制,2022,30(7):246-254.

到并输出“手势数据信息”[1-3,18]。整个“ToF 手势识别算法”模型如图 1 所示。BP _ NN 运算处理分析框架，主要由适配器和调整器构成；适配器，前向传输（Feed Forward），逐层波浪式的传递输出值［左右、上下、前后］；调整器，逆向反馈（Back Propagation），反向逐层调整权重和阈值，修正误差至最小。“智能模糊控制”和“专家经验控制”，进行前级“预处理”和后级“结果识别”；前级“预处理”，通过“滤波”，得到 50～600 mm 内的传感器“视场出入及其运动的距离和时刻数据”，并对快速移动情形完成数据插入，便利 BP _ NN 更加有效运算；后级“结果识别”处理，即后处理，对 BP _ NN 运算得到的左右、上下、前后数据进一步处理分析，得到左右、上下、点击等手势直接数据。借助专家经验，可以把实际操控中多数习惯的“斜上再斜下”的动作解释为前后运动的“点击”。

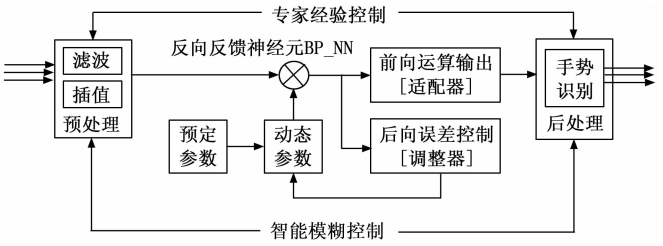

图 1　激光飞秒手势识别算法模型框图

BP _ NN 框架，选用 3-4-3 层次，如图 2 所示，3 个输入层，4 个隐藏层，3 个输出层，便于常用的 Cortex-Mx/RISC-V 类型的微控制器 MCU（Microcontroller Unit）在软件层次上进行快速高效的实现［MCU，优势在控制，数字信号处理是弱势］。BP _ NN 运算，经典成熟，不再赘述相关过程的各个离散处理公式，其关键是误差计算及其对各层权值和阈值的修正，尽管已经采用最简层次框架，放在 MCU 上实现仍然运算繁锁、耗时、耗资源，便于 MCU 硬软件进一步最简化及其成本最少考虑起见，借助 Mablib、Maple 等的 NN 工具函数和仿真功能，针对实际测量数据，训练各层的权值和阈值，形成表格，缩短训练和学习时间，即采用 Mablib 实现适配器，MCU 仅做调整器实现。环境光的变化是手势准确度的最大影响因素，从系统实用和精简方面考虑，训练 4 种常用光环境：夜晚、白炽灯、日光灯、太阳光，MCU 系统在硬件上采用以光敏电阻－模数转换 ADC（Analog-to-Digital Converter）电路进行选择识别。

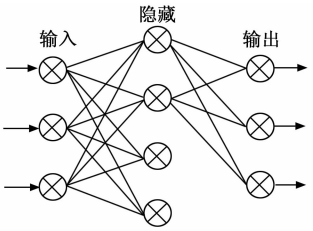

图 2　后向反馈神经元网络层次构成示意图

MCU 在 BP _ NN 中执行前向输出的运算公式如下，这里激励函数采用修正线性单元 ReLU（Rectified Linear Unit），其中 $x/h/y$ 分别对应输入/隐藏/输出各层神经元节点，w/θ 表示相应前级的权重和阀值。

$$y_i = \sum_{i=0}^{n}(w_{hi} * h_i + \theta_i)\ ,h_i = \sum_{i=0}^{n}(w_{xi} * x_i + \theta_i)$$

2　激光飞秒手势导航微系统构造

系统由三部分组成：信号采集识别发送终端（以下简称采发终端）、手势数据收集转发终端（以下简称收转终端）和视窗展示服务软件（以下简称视窗展示软件）。短距离无线通信采用抗干扰强、传输距离大、穿墙能力强、功耗低（发射最大功耗 35 mA）的远距无线 LoRa 通信（Long Range Radio）形式。由于系统主要工作在 Windows、UOS 等通用操作系统下，并且服务以优秀的微软 office 等办公软件，转换接口采用精简易用的 WinUSB 形式。终端微控制核心 MCU，选用高性价比的 32 位的 GD32F103TB（Cortex-M3 内核）或 GD32VF103TB（RISC-V 内核），以 C 语言编程实现 BP _ NN 适配器及其 TOF 数据采集与 LoRa、USB 传输通信。视窗展示软件，以 C＋＋语言编程实现 WinUSB 驱动，以 C＋＋或 Python 语言编程实现手势随动的各个视窗切换，伴随视窗上隐含的微透明动态指示，既可完成基本的 PPT 幻灯选择播放，也可完成包括 office（word、excel、ppt 等）、acrobat（pdf 电子文档）在内常用办公软件的远程切换翻页操作，更可完成涵盖办公软件在内包括 eclipse、notepad 等集成开发环境的远程研发演示，即 3 个版本：幻灯播放版、办公播控版和开发播控版[4-5,17]。完整的系统构成及其核心算法模型与语言编程实现，如图 3 所示，详细的微产品系统如图 4 演示 PPT 截图所示，图 4 还示意了产品系统化的生产测试与配置的演示软件。

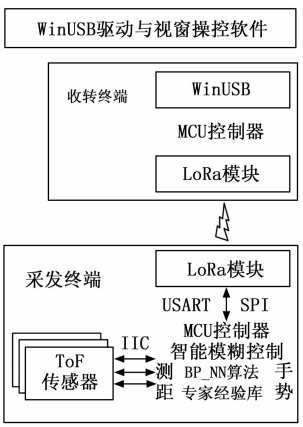

图 3　微系统构成及其核心技术手段示意图

3　激光飞秒手势导航微系统设计

3.1　采集识别发送电子终端构造

3.1.1　电子电路设计

核心功能实现，以 MCU 片上接口——内部集成电路 IIC（Inter-Integrated Circuit）挂接 ToF 传感器、异同步串行收发器 USART（Universal Synchronous/Asynchronous

图 4 微产品系统应用及其测试与配置示意图

Receiver/Transmitter）挂接 LoRa 无线模块和 ADC 片上外设连接光敏电阻，同时以其片内实时时钟 RTC（Real _ Time Clock）完成 ToF 测距时刻记录。便携低功耗设计考虑，采用 1 000～2 500 mA 扁平锂电池供电，以高性价比的 ETA7640 芯片作为充供电管理，同时设置按钮唤醒重启电路。ToF 激光发射测距和 LoRa 数据发送，最大功耗达 100 mA，供电转换电路选用性能优良的大功率 LM1117 _ 3V3，无检测传感时必须预以关断，因而设计 10 min 内无动作时通过软件关闭系统，再次使用时以上述按钮唤醒重启。由于采用 QFN36 最小 MCU 封装，此类 MCU 无 VB 电池待机/休眠管理脚，则控制关断时直接切断 LM117 供电电路。电源开关选用 P 沟通 MOS 场效应管 PDN304P。ToF 传感器 2.8 V 供电时性能最佳，选用 XC6206P28MR 为其集群独立供电，3.3 V 与 2.8 V 之间的 IIC 总线采用 N 沟通 MOS 场效应管 2N7002 过渡。模块化电路设计，MCU 与 ToF 传感器及其唤醒按钮构成小顶板，大底板携带锂电池及其充供电路，并背附 LoRa 无线通信模块。一大一小两板在两侧通过“信号排针”连通[5,11,17]。终端平面大小不超过锂电池大小 35 mm×50 mm，包括锂电池在内整体厚度控制在 8 mm 内，“胸针”或“别针”形式，供胸前佩带，充分体现“便携”。选用透明热缩管外封，既时尚大方又可免去昂贵的模具设制费用。识别模块设计最大持续工作时间不低于 8h@1 000 mA，总体重量控制在 250 gf 内。电路原理及其整体外观造型如图 5 所示。

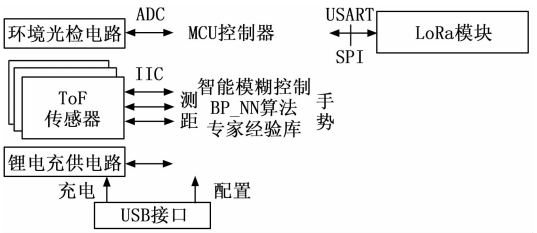

图 5 采发终端电路原理及其整体造型示意图

3.1.2 软件体系构造及其编制

稳定高效设计及其开发考虑，基于 MCU 寄存器和多中断处理机制，构造嵌入式软件应用体系，这里采用项目主持人的发明专利——ARM/RISC-V 系列微控制处理器软件架构工具，快速得到包括启动代码、系统时钟变换配置、片内接口/外设驱动程序、中断分配及其处理程序、多任务调度程序等的基本框架代码，直接在此基础上展开填空式的功能代码设计。

3.1.2.1 TOF 测距及其 BP _ NN 手势识别

TOF 测距，采用意法半的 VL53L0X 模块，需要两级驱动，一级驱动 IIC 传输完成数据读写，二级驱动，移植相应的应用程序接口 API（Application Programming Interface），配合模块内嵌算法处理得到最终距离数据。采用两个中断处理完成核心操作：IIC 中断和距离获取事件中断，优先级最高，仅次任务调度的时钟节拍中断。两级中断的使用，把单次测距时间提升了 18 ms 内（厂家提供的最小值为 22 ms）。

中断处理 IIC 操作，效率最优，但时序节点把握要求苛刻，特别是单字节的读操作，很难驾驭，常常因此造成系统停滞，多采用通用输入输出 GPIO（General-Purpose Input/Output）模拟实现，软件架构工具得到的 IIC 驱动程序，很好解决了这个问题[19-21]。核心的底层 IIC 驱动程序流程如图 6 所示。

关键的单字节读取 IIC 操控时序如图 7 所示，先写指定寄存器地址，再从中读出数据。

距离获取事件中断，设置任务启动标识，处理程序在滤波插值、BP _ NN 适配输出、手势识别 3 个顺序任务中实现，首先是滤波插值，依据专家经验，只取连续（最大时间间隔 100 ms）有效手势距离（110～550 mm 内）构成原始数据队列（长度 30 内），且每个传感器应用至少 3 个数据，数量不足按照进出计量场的数据变化规律插入，然后是 BP _ NN 适配和手势识别[8-9,19-21]，核心的处理函数代码如下：

```
char vldDtLgth = 0, vldDtFlg = 0, lstGst ='z'; // 有效数据数量、标识、上次手势值
VL53L0X_RangingPoint vldDt[30] = {0}; // 有效数据存储
unsigned int lstTs = 0; // 最近测量活动时间记录
void tof_bpnn_process(char maxNum) { // ToF 数据分析处理[传感器最大数量]
uint8_t i, m; unsigned short v; unsigned int ts;
VL53L0X_RangingMeasurementData_t msData[maxNum];
for(i=0; i<maxNum; i++) { // 测量数据处理
if(((alarm_flag>>i)&1)  1) { alarm_flag &= ~(1 << i);
VL53L0X_ GetRangingMeasurementData (&vl53l0x _ dev[i], &msData[i]); // 获取测量距离
msData[i]. TimeStamp = RTC_GetCounter();
ts = msData[i]. TimeStamp;
v = msData[i]. RangeMilliMeter;
if((v>110)&&(v<550)) {
if((vldDtLgth  0)&&(vldDtFlg  0)) {
vldDt[vldDtLgth]. dvcNum = i;
vldDt[vldDtLgth]. TimeStamp = msData[i]. TimeStamp;
vldDt[vldDtLgth]. RangeMilliMeter = msData[i]. RangeMilliMeter;
```

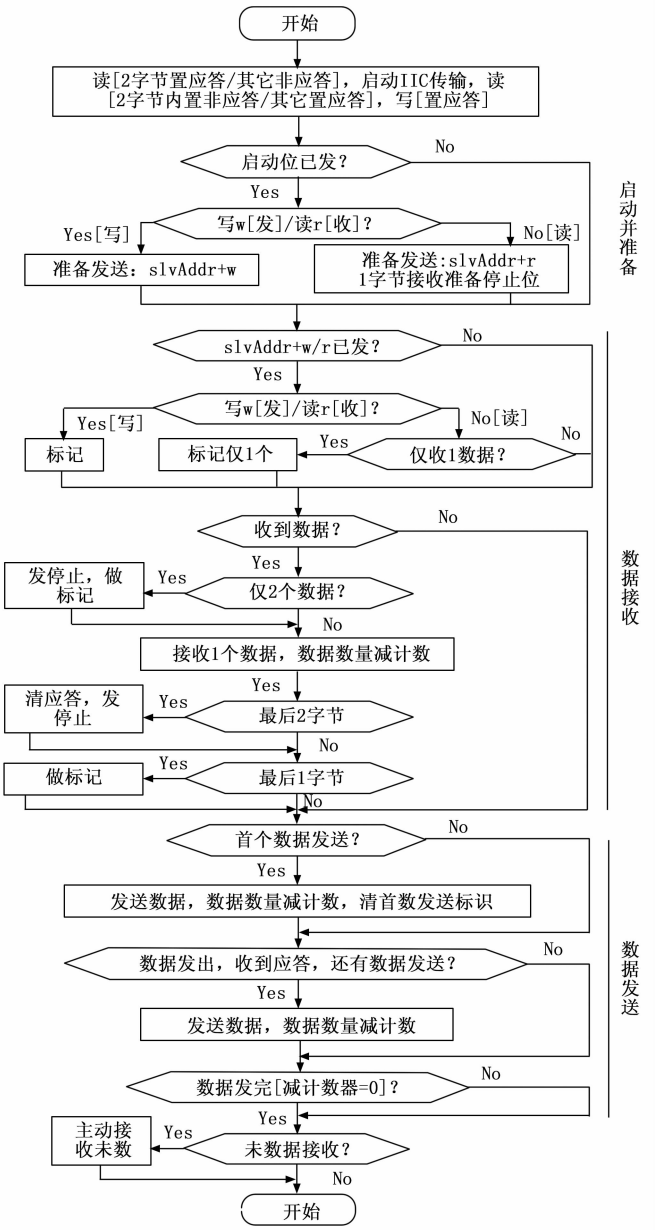

图 6　IIC 中断收发驱动程序流程图

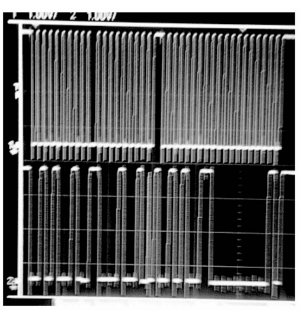

图 7　恰当的片内 IIC 驱动程序流程图

```
vldDtLgth += 1; lstTs = ts; vldDtFlg = 1;
if((optFlg>>1)&1)
printf("**%d-%d: %4dmm,%8d;\r\n",vldDtLgth, i, v, ts);
}
else {
if(((ts-lstTs)<10)&&(vldDtLgth<30)&&(vldDtFlg  1)) {
vldDt[vldDtLgth].dvcNum = i;
vldDt[vldDtLgth].TimeStamp = msData[i].TimeStamp;
vldDt[vldDtLgth].RangeMilliMeter = msData[i].RangeMilliMeter;
vldDtLgth += 1; lstTs = ts;
if((optFlg>>1)&1)
printf("**%d-%d: %4dmm,%8d;\r\n",vldDtLgth, i, v, ts);
}
else vldDtFlg = 2;
}
}
VL53L0X_ClearInterruptMask(&vl53l0x_dev[i],0); // 清除 VL53L0X 中断标志位
}
}
if((vldDtFlg  1)&&((RTC_GetCounter()-lstTs)>53)) vldDtFlg = 2;
if(vldDtFlg  2) { // 手势分析识别
VL53L0X_DtSetInterpolation (); // 插值
VL53L0X_BpNnAdepter(); // BP_NN 适配输出
VL53L0X_gestureRecognition(); // 手势识别
if(gestureFlg) { // Lora 外传
printf("  gesture:%d  \r\n", m);
gesture[5] = m | 0x30;
USART1_SendData(gesture, 6);
}
}
}
```

3.1.2.2　LoRa 配置及其有效数据无线发送

配置包括 LoRa 地址、通道、升级通信速率，并进入易用的透明传输方式，需要反复写入验证特别耗时，仅做一次即可，无需每次启动时都执行，设计快速启动运行，运行中需要修改时通过 UART 串口或 USB 接口通信实现，尽管耗时，却一劳永逸，相关操控函数代码如下：

```
void Lora2G4_smpInit(void) { // Lora2G4 模块快速常规初始化
PIO_DataOutput(0, 5, 1); // 设置进入连续透传工作模式
PIO_DataOutput(0, 6, 0);
PIO_DataOutput(0, 7, 0);
while(PIO_DataInput(0, 4)  0); // 等待模块有效
}
void Lora2G4_vInit(void) { // Lora2G4 模块配置初始化
unsigned char i, b[15] = {0};
unsigned char a[6] = { 0xC0, // 工作模式：参数掉电保存
{0x30, 0x31}, 0x28, 0x03, 0x04 }; // 地址，波特率，通道号,透传/功耗
```

```
inFlshByteRead(0, AddrChnl, 3); // Lora 地址通道号
a[1] = AddrChnl[0]; a[2] = AddrChnl[1]; // Lora 地址通道号
a[4] = AddrChnl[2];
PIO_DataOutput(0, 5, 1); // 设置模块进入配置模式
PIO_DataOutput(0, 6, 1);
PIO_DataOutput(0, 7, 1);
while(PIO_DataInput(0, 4)  0); // 等待模块有效
do { // 配置参数及其验证
Delay(300); USART1_SendData(a, 6); // 发送配置参数
Delay(300); b[0] = b[1] = b[2] = 0xC1; // 验证配置
USART1_SendData(b, 3);
Delay(100); i = USART1_QueueRcvData(b);
}while((b[0]! =a[0]) || (b[1]! =a[1]) || (b[2]! =a[2]) || (b[3]! =a[3]) ||
(b[4]! =a[4]) || (b[5]! =a[5]) || (i! =6));
PIO_DataOutput(0, 6, 0); // 设置进入连续透传模式
PIO_DataOutput(0, 7, 0);
USART1_CR1 &= ~6; // USART 波特率调整：禁止收发
USART1_BRR = 0x00000138; // 波特率 115200
USART1_CR1 |= 6; // 收发使能
while(PIO_DataInput(0, 4)  0); // 等待模块有效
}
void USART0_Process(void)  // USART0 数据收发处理
{ unsigned int status = USART0_SR;
if(status&15) return; // 异常处理：校验错，帧错，噪声，溢出
else if((status>>5)&1) // 数据接收(环形队列存放，队列满则抛掉)
{ status = Usart0RcvPrt + 1;
if(status>Usart0QueueLth) status = 0;
if(status! =Usart0AppPrt) { status = USART0_DR;
switch(Usart0RcvFlg) {
case 0: if(status  '*') Usart0RcvFlg += 1; break;
case 1: if(status  ' ') Usart0RcvFlg += 1;
else Usart0RcvFlg = 0; break;
case 2: case 3: case 4: Usart0RcvFlg += 1; break;
default: break;
}
Usart0RcvQueue[Usart0RcvPrt++] = status;
if(Usart0RcvPrt>=Usart0QueueLth) Usart0RcvPrt = 0;
}
else USART0_DR;
}
}
```

3.2 接收转换传输电子终端构造

3.2.1 电子电路设计

收转终端，USART 串口无线 LoRa 接收，通用串行总线 USB（Universal Serial Bus ） “二传”并供电，采用 MCU 片内外设——USB2. 0 全速模块[5-8][10-12]，电路原理及其正反面 mini 造型如图 8 所示。

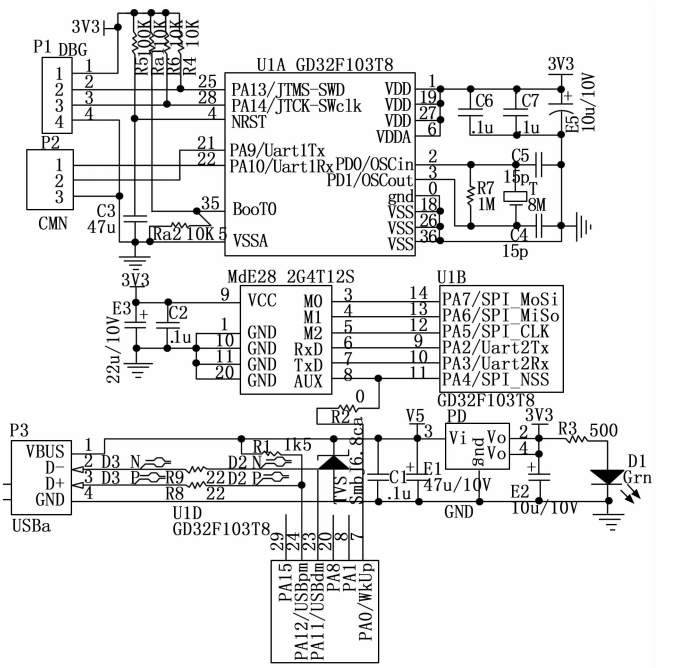

图 8 收转终端原理及其正反面 mini 造型图

3.2.2 软件体系构造及其编制

仍然采用项目主持人的发明专利——ARM/RISC-V 系列微控制处理器软件架构工具，快速得到基本框架代码(包括 USB 驱动程序)，在此基础上展开填空式的功能代码设计。采用 USART 中断和 USB 中断进行 LoRa 无线透传信息接收和 USB 收发通信，中断里接收，相应任务中分析处理[17,22]。

LoRa 手势信息接收，在中断处理程序中设置环形队列高效接收识别，相应任务中转包调用 USB 发送任务上传，关键程序代码如下：

```
int USART1_QueueRcvData(unsigned char *Data) // 环形队列中断数据接收
{ int i = 0;
USART1_CR1 |= 1 << 2; // 使能启动接收
if(Usart1AppPrt<Usart1RcvPrt) // 队列正向增长数据接收
{ do { i++;
*Data++ = Usart1RcvQueue[Usart1AppPrt++];
} while(Usart1AppPrt! =Usart1RcvPrt);
}
else if(Usart1AppPrt>Usart1RcvPrt) // 队列反向增长数据接收
{ do
{ *Data++ = Usart1RcvQueue[Usart1AppPrt++]; i++;
if(Usart1AppPrt  Usart1QueueLth) Usart1AppPrt = 0;
} while(Usart1AppPrt! =Usart1RcvPrt);
}
return i; // 返回数据接收量
}
```

```
void USART1_Process(void) // USART1 数据收发处理
{ unsigned int status = USART1_SR;
if(status&15) return; // 异常处理：校验错，帧错，噪声
if((status>>5)&1) // 数据接收(环形队列存放,满则抛掉)
{ status = Usart1RcvPrt + 1;
if(status>Usart1QueueLth) status = 0;
if(status! =Usart1AppPrt) {
status = USART1_DR;
switch(Usart1RcvFlg) {
case 0: if(status  '*') Usart1RcvFlg += 1; break;
case 1: if(status  '&') Usart1RcvFlg += 1;
else Usart1RcvFlg = 0; break;
case 2: if(status  ~) Usart1RcvFlg += 1;
else Usart1RcvFlg = 0; break;
case 3: if(status  ' ') Usart1RcvFlg += 1;
else Usart1RcvFlg = 0; break;
case 4: if(status  '@') Usart1RcvFlg += 1;
else Usart1RcvFlg = 0; break;
case 5: Usart1RcvFlg += 1; break;
default: break;
}
Usart1RcvQueue[Usart1RcvPrt++] = status;
if(Usart1RcvPrt>=Usart1QueueLth) Usart1RcvPrt = 0;
}
else USART1_DR;
}
}
while(1) {
m = DrvUSB_EpRead(2, tmp, 6); // USB 接收数据处理
if(m  0) {
DrvUSB_EpWrite(1, tmp, 5); // 通过 USB 上传
if((tmp[0]  '*')&&(tmp[1]  ' ')) { // Lora 地址通道号变更
AddrChnl[0] = tmp[2] & 0x0F;
AddrChnl[1] = tmp[3] & 0x0F;
AddrChnl[2] = tmp[4] & 0x0F;
inFlshPageErase(); // 闪存记录修正
inFlshByteWrite(0, AddrChnl, 3);
USART1_CR1 &= ~6; // USART 波特率调整
USART1_BRR = 0x00000E98; // 波特率 9600
USART1_CR1 |= 6;
Lora2G4_vInit(); // Lora 重新初始化
}
}
if(Usart1RcvFlg  6) { // Lora 收到有效数据
m = USART1_QueueRcvData(tmp);
Usart1RcvFlg = 0;
if(m  6) DrvUSB_EpWrite(1, tmp, 6); // 通过 USB 上传
}
}
```

USB 收发传输，不以传统的人机接口设备 HID（Human Interface Device）/配合主机 USART 转 USB 做“借尸还魂”，直接采用高效的 WinUSB 格式，增加特定的操作系统描述、兼容 ID 特征描述和设备接口 GUID 描述符，并对收发任务函数做超时处理以避免程序阻塞，主要程序代码如下：

```
const unsigned char OsDscrptStr[] =  // 操作系统描述字符串
{ USBStrDscrpt_Lth(8), 3,
USBStrDscrpt_Uncd('M'), USBStrDscrpt_Uncd('S'),
USBStrDscrpt_Uncd('F'), USBStrDscrpt_Uncd('T'),
USBStrDscrpt_Uncd('1'), USBStrDscrpt_Uncd('0'),
USBStrDscrpt_Uncd('0'), USBStrDscrpt_Uncd(1)
};
const CptbIdDscrpt cptbIdDscrpt = // 兼容 ID 特征描述符
{ 0x28,  0x100,  4,  1,  // dwLength,  bcdVersion, wIndex, wCount
{0, 0, 0, 0, 0, 0, 0}, // Reserved[7]
0, 1, // bFirstInterfaceNumber, RESERVED
{'W', 'I', 'N', 'U', 'S', 'B', 0, 0}, // compactiableID[8]
{0, 0, 0, 0, 0, 0, 0, 0}, // subCompactiableID[8]
{0, 0, 0, 0, 0, 0} // Reserved[6]
};
const ItfcGuidDscrpt itfcGuidDscrpt = // 设备接口 GUID 描述符
{ 0x8E,  0x100,  5,  1,  // dwTotalSize,  bcdVersion, wIndex, wCount
0x84, 1, 0x28, // dwSize, dwPropertyDataType, wPropertyNameLength
USBStrDscrpt_ Uncd('DeviceInterfaceGUID'), // bProperyt-Name
0x4E, // dwPropertyDataLength
USBStrDscrpt_ Uncd('{12345678 - 1234 - 1344 - 1234 - 12345678ABCD}0') // bPropertyData
};
// 单缓端点数据发送(查询激发中断发出)，参数：端点号 1－7，预发数据，数量，返回[0－正确/－1 超时]
char DrvUSB_EpWrite(char EpNum, unsigned char * data, unsigned int counts) {
unsigned int i; short sz;
if(EpNum  1) sz = CfgDscrpts.EpIn1.wMaxPcktSz;
while(counts) { i = 0;
do { i++;
if(i>maxTmOut) return 0xFF; // 超时退出
} while((EpDtSts>>EpNum)&1); // 时机查询：等待 USBD 空闲
if(counts>sz) // 超过最大包尺寸的发送
{ USBD_BasicWrite(EpNum, 0, data, sz);
data += sz; counts -= sz;
}
else // 最大包尺寸内的发送
{ USBD_BasicWrite(EpNum, 0, data, counts);
```

```
counts = 0;
}
EpDtSts |= 1 << EpNum; // 标识需要数据发送
}
return 0;
}
// 端点数据接收，参数：端点号 1－7，预发数据，数量，返回[0－正确/－1 超时]
char DrvUSB_EpRead(char EpNum, unsigned char* data, unsigned int counts) {
unsigned int i; RcvPt = data;
while(counts) { i = 0;
do { i++;
if(i>maxTmOut) return 0xFF; // 超时退出
} while(! ((EpDtSts>>EpNum)&1)); // 等待 USB 数据到来
counts -= RcvCnt;
EpDtSts &= ~(1 << EpNum); // 标识已经取得 USBD 收到数据
}
return 0;
}
```

3.3 WinUSB 接收及其视窗展示控制实现

3.3.1 WinUSB 驱动及其数据接收实现

针对常规 Windows、UOS 操作系统应用，直接调用 Win7 以上内嵌的 WinUSB.dll 动态库实现，主要是操控句柄的开关和接收监听，采用两个定时器分别开关线程实现收转 USB 终端的即插即用和动态接收监听。只要底层嵌入式应用软件中添加特定描述支持，Windows 就可以自动识其为 WinUSB 设备，上层可视化测试/应用程序就可以通过 WinUSB. dll 与其进行 USB 数据传输通信。这样，USB 设备的数据采集、监视控制、配置参数、软件刷新等编程操控，就同传统的“RS-232C 通信”一样了，无需 USB 转串口作硬软件转接，而且速度快、实时性强，直截了当[6][10-15]。相关关键代码如下：

```
HANDLE hDeviceHandle = INVALID_HANDLE_VALUE; // 设备文件句柄
WINUSB_INTERFACE_HANDLE hWinUSBHandle // WinUSB 操作句柄
  = INVALID_HANDLE_VALUE;
OVERLAPPED overlapped; // 异步收发缓存结构变量
BOOL flgWinUSB = false; // WinUSB 设备存在与否[true 存在]
char thrdStt = 0; // 线程建立标识状态[0－无/1－建]
void __fastcall TfmShow::SystemInit(void) { // 系统初始化
static const GUID stm32F1xxDvcItfc = { 0x12345678, 0x1234, 0x1344,
{ 0x12, 0x34, 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC } };
GUID guidDeviceInterface = stm32F1xxDvcItfc;
BOOL bResult = TRUE; AnsiString str;
bResult = GetDeviceHandle ( guidDeviceInterface, &hDeviceHandle);
if (! bResult) { inOutTmr->Enabled = true;
sttFlg->Font->Color = clGray; return;
}
bResult = GetWinUSBHandle(hDeviceHandle, &hWinUSBHandle);
if (! bResult) { inOutTmr->Enabled = true;
sttFlg->Font->Color = clGray; return;
}
flgWinUSB = true; // 设备已经正常寻址到
inOutTmr->Enabled = false;
sttFlg->Font->Color = clRed;
if(thrdStt  0) { // 启动并运行线程
rcvThrdDt = new rcvThrd(true, imgDrct,
sttFlg, spcTmr, inOutTmr); thrdStt = 1;
}
rcvThrdDt->FreeOnTerminate = true;
rcvThrdDt->Resume(); UINT timeout = 100; // 最大传输时间[ms]
bResult = WinUsb_SetPipePolicy(hWinUSBHandle,
0x81, PIPE _ TRANSFER _ TIMEOUT, sizeof ( timeout ), &timeout);
if(! bResult)
{ str = "设置超时错误:"; str += GetLastError();
ShowMessage(str); return;
}
ZeroMemory(&overlapped, sizeof(overlapped));
overlapped.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
}
void __fastcall rcvThrd::RcvDt()
{ if(hWinUSBHandle  INVALID_HANDLE_VALUE) return;
ULONG cbRead = 0; UCHAR szBffr[20] = {'0'}; AnsiString str;
BOOL bResult = WinUsb_ReadPipe(hWinUSBHandle, 0x81,
szBffr, 15, &cbRead, &overlapped);
if(bResult) { const char* t = szBffr; str = t; windowChange(str); }
else if(ERROR_IO_PENDING  GetLastError())
{ bResult = WinUsb_GetOverlappedResult(hWinUSBHandle,
&overlapped, &cbRead, true);
if(NULL ! = cbRead)
{ int m = cbRead; cbRead = 0;
if(bResult)
{ const char* d = szBffr; str = d;
char i = 0; windowChange(str);
}
}
}
else if(22  GetLastError()) inOutTmr->Enabled = true; // 设备拔出
```

```
}
```

3.3.2　视窗动态展现及其控制软件编制

开发之初，设想用通用流行的 Python 语言开发，拟用 PyUSB、Kivy/Tkinter、python-pptx/word/excel/acrobat，PyKeyboard/PyMouse 等库，深入之后发现这些库多是用 C++开发打包的，多是针对 WinAPI 的包，而且不新、不全面，作为解释语言也没有编译语言 C++运行效率高，与其特制一些把 C++库弥补 Python 应用，倒不如与底层合二为一，直接嵌入其监听线物程中实现，更为经典直接，于是回到了 C++开发实现。这里选用 Enbarcadero 的 RadStudio 开发环境，主要是两类窗口：引导窗口和展示窗口。引导窗口，小视野透明暗标动态图文指示，控制展示窗口切换和翻页。展示窗口，展示常规办公软件和特定开发软件的窗口，主要通过调用 Windows 视窗变换和键盘模拟操控 API 函数实现[6][10-15]。核心程序函数代码如下：

```
BOOL SetTopWindow(HWND hWnd) { // 设置窗口到最顶层[参数:窗口句柄]
    HWND hForeWnd = GetForegroundWindow();
    DWORD dwForeID = GetWindowThreadProcessId(hForeWnd, NULL);
    DWORD dwCurID = GetCurrentThreadId();
    AttachThreadInput(dwCurID, dwForeID, TRUE);
    ShowWindow(hWnd, SW_MAXIMIZE);
    SetWindowPos(hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOSIZE | SWP_NOMOVE);
    SetWindowPos(hWnd, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOSIZE | SWP_NOMOVE);
    SetForegroundWindow(hWnd);
    AttachThreadInput(dwCurID, dwForeID, FALSE); return TRUE;
}
```

窗口随动变换展示的程序流程图如图 9 所示。

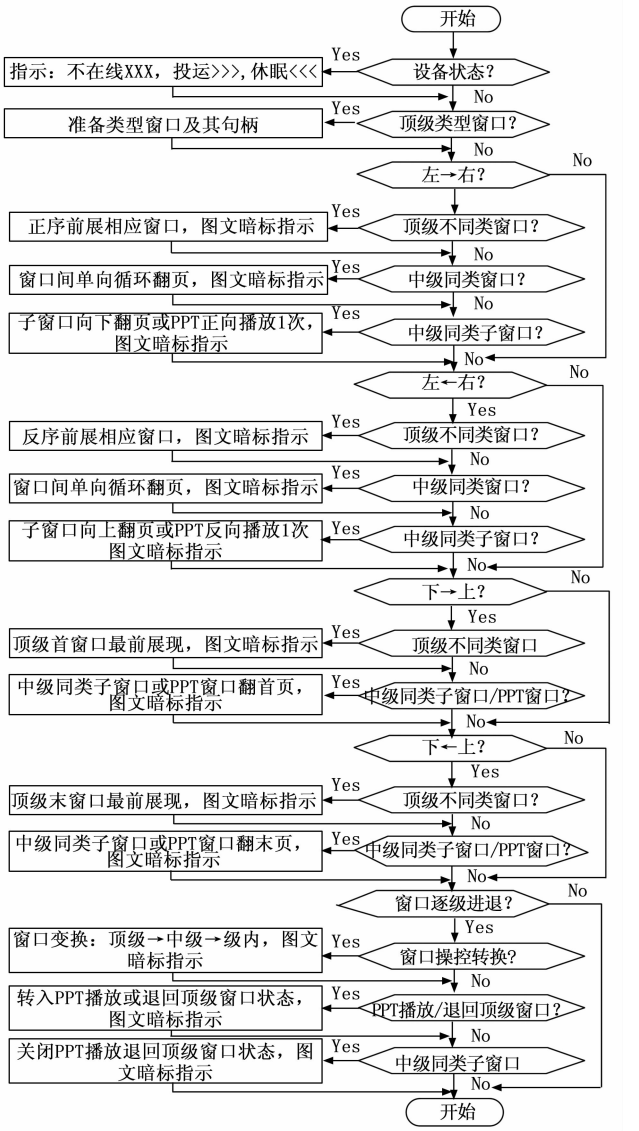

图 9　窗口随动变换展示的程序流程图

4　产品系统化测试及其配置考虑

生产出厂测试，输出指示原始的手势分析结果和通信通道的畅通性，配置 LoRa 无线通信地址、通道及其是否输出原始测距数据。设计有专用的可视化 USB 软件，完成手势分析原始数据的即时查看和 LoRa 无线通信配置。同时底层采发终端还支持使用 USB 串口助手查看手势分析原始数据、原始测距数据详细查看和 LoRa 无线通信配置[6-8]。专用可视化 USB 软件和 USB 串口助手操控界面，如图 10～11 所示。

图 10　专用可视化 USB 软件实现的测试与配置示意图

要做到这些，终端软件中需要添加相应支持，采发终端中相关的典型实现函数代码如下：

```
void Lora2G4_chgCfg(void) { // Lora 地址通道变更/系统打印信息[配合 USART0 接收中断]
    unsigned char tmp[10] = {0}, m;
    if(Usart0RcvFlg  5) { // 收到有效数据
    m = USART0_QueueRcvData(tmp);
    Usart0RcvFlg = 0;
    if(m  5) {
    if(tmp[2]<0x41) { // Lora 地址通道变更
    AddrChnl[0] = tmp[2] & 15;
    AddrChnl[1] = tmp[3] & 15;
    AddrChnl[2] = tmp[4] & 15;
    inFlshPageErase(); // 闪存记录修正
```

图 11 USB 串口助手实现的测试与配置示意图

```
inFlshByteWrite(0, AddrChnl, 3);
USART1_CR1 &= ~6; // USART 波特率调整
USART1_BRR = 0x00000E98; // 波特率 9600
USART1_CR1 |= 6;
Lora2G4_vInit(); // Lora 重新初始化
}
else { // 系统输出指示信息变更
if(((optFlg>>1)&1) 0) optFlg |= 1 <<1; // 详细
else optFlg &= ~(1 << 1);
}
}
}
}
```

5 产品系统的应用统计分析

激光手指飞行导航简易微系统，试销以后，以极高的性价比，满足了各类学校、企业、酒店、机关、场所的寻常应用需要，内外循环行销，市场前景颇佳，紧接着就围绕微型化和低功耗设计又推出了二代产品。另外，顶板即主板，还作为独立的 ToF 手势或雷达测距测向模块，单独出售，填补远距离手势导航和低成本雷达测距测向领域的空白。

6 结束语

激光手指飞行导航简易微系统，麻雀虽小，五脏俱全，以神经元深度学习自适应人工智能手势识别核心算法数学处理模型为中心，既涵盖了数电/模电/射电在内的低功耗微型电子电路设计，也包括了嵌入式 C 语言应用软件设计、C++驱动程序设计和 C++/Python 动态视窗切换技术，通用流行的局部长距离无线通信和各类局部总线接口技术一应俱全，既是可探索开发项目的过程研发，也产生了可测试生产行销的产品系统。在此基础上，加大深度学习自适应人工智能算法处理，实现鼠标功能的“远距离移动飞鼠”，将有望成为可能，“学研”应用颠覆突破，意义重大，拭目以待，为期不远。

参考文献：

[1] 怯肇乾. 数学分析与系统工程建模及其应用实践：V2.00 [Z]. 2008.
[2] CSDN 博客. 详解 BP 神经网络 [EB/OL]. [2022-01-28]. https://blog.csdn.net/fanxin_i/article/details/80212906,2018.
[3] CSDN 博客. BP 神经网络 C 语言实现 [EB/OL]. [2022-01-28]. https://blog.csdn.net/qq_39545674/article/details/82495569,2018.
[4] 怯肇乾. 江西省教育厅科研项目申请书——手势飞行导航简易微系统构造：V1.1 [Z]. 2020.
[5] 怯肇乾. 激光手指飞行导航科研项目及其产品系统发布演示 PPT：V1.1 [Z]. 2021.
[6] 怯肇乾. 嵌入式系统硬件体系设计 [M]. 北京：航空航天大学出版社，2007.
[7] 怯肇乾. 基于底层硬件的软件设计 [M]. 北京：航空航天大学出版社，2008.
[8] 怯肇乾. GD-CortexM3 系列微处理器软件架构工具使用手册：V2.00 [Z]. 2018.
[9] CSDN 博客. 使用 GD32 内嵌 IIC 控制器 [EB/OL]. [2022-01-28]. https://mp.csdn.net/mp_blog/creation/editor/119082291,2021.
[10] CSDN 博客. CortexM 系列 MCU 操控 2.4G 的 Lora 模块 [EB/OL]. https://mp.csdn.net/mp_blog/creation/editor/119788114,2021.
[11] 怯肇乾，等. WinUSB 驱动实现及其通信传输应用 [J]. 软件，2018，39 (8)：57-64.
[12] CSDN 博客. GD32-MCU-WinUSB 编程问题及其解决 [EB/OL]. [2022-01-28]. https://mp.csdn.net/mp_blog/creation/editor/118701513,2021.
[13] CSDN 博客. C++实现的特定 Window 系统窗口操控 [EB/OL]. [2022-01-28]. https://mp.csdn.net/mp_blog/creation/editor/119784417,2021.
[14] 博客园. SetForegroundWindow 的正确用法 [EB/OL]. [2022-01-28]. https://www.cnblogs.com/ziwuge/archive/2012/01/06/2315342.html,2012.
[15] CSDN 博客. windows 模拟键盘输入 [EB/OL]. [2022-01-28]. https://blog.csdn.net/gaobobo138968/article/details/39455001,2014.
[16] 怯肇乾. 电磁兼容应对及其改进技术 [J]. 电气工程，2015，3 (4)：155-172.
[17] 怯肇乾. 嵌入式网络通信开发应用 [M]. 北京：航空航天大学出版社，2010.
[18] CSDN 博客. 详解 BP 神经网络 [EB/OL]. [2022-01-28]. https://blog.csdn.net/fanxin_i/article/details/80212906,2018.
[19] CSDN 博客. BP 神经网络 C 语言实现 [EB/OL]. [2022-01-28]. https://blog.csdn.net/qq_39545674/article/details/82495569,2018.
[20] 友信网. STM32F103+VL53L0 测距 [EB/OL]. [2022-01-28]. https://www.freesion.com/article/94531023798/,2019.
[21] CSDN 博客. STM32F103 模拟 IIC 总线驱动 4 个 VL53L0 [EB/OL]. [2022-01-28]. https://blog.csdn.net/lx635541878/article/details/102863796,2019.
[22] 杨 杨. 基于 LoRa 物联网技术的火龙果大棚监控系统的设计与应用 [J]. 计算机测量与控制，2019，27 (11)：82-85.