

# 基于 ART-PI 的智能手势控制系统

郑有才；尹傲；宋明非

## 第一部分 设计概述

### 1.1 设计目的

本文提出了一种基于深度学习的智能算法来识别手势控制智能小车，提出的方案展示出有效的开放性和拓展性，无论是手势识别还是控制小车都可以根据需求做出调整。我们开发了基于 RT-Thread 平台设计的一款手势控制系统，该系统可以在不用走动的情况下，用手势控制，用 Lora 无线通讯的 ART-PI 智能控制系统实现实时、远程的控制智能车的运行，满足现实中人机交互的功能。

### 1.2 应用领域

主要应用场所：家与办公场所。该控制系统可成为智能家居的一部分。随着人们生活水平的提高和高科技的发展，智能家居已成为一种必然趋势而深入千家万户，智能化家居也逐渐普及，基于 Art-pi 的手势控制系统控制小车也可将其推广控制其他智能家居设备，代替原有的人为的无意义的劳动，从而使自己的生活更加方便，工作更加高效。该设备应用范围大，且日常使用次数频繁，实用性强，具有良好的发展前景。

### 1.3 主要技术特点

这款设备包含一个 MPU6050 传感器，两个 lora 模块、一个 art-pi、一个智能小车。Art-pi 可以实时解读手势信号，同时将信息通过 WiFi 发送到 onenet 平台进行记录，通过 Lora 模块可持续将这些信号发送到智能小车，从而控制小车运行。

#### 1.3.1 Art-pi 用 ai 算法处理传感器数据

当 MPU6050 传感器数据三个轴向上的加速度之和超过了一定的阈值时，开始采集数据。收集到  $70 \times 6$  一组的数据后，Art-pi 进行 AI 手势识别，为了识别的准确性，我们的训练数据数目可观。ai 模型传入的手势数据每组是  $70 \times 6$ ，我们先将所有的线性加速度和旋转加速度都加 2000，使得所有的数据都为正数。然后再除以各种取值范围的两倍，使得无论是旋转加速度还是线性加速度的值都保持在  $[0, 1]$  的范围内，归一化处理后然后进行手势判断。

### 1.3.2 板载 WiFi 上传数据至 onenet

利用板载 WiFi 的实时传输特性将 art-pi 处理后的传感器数据传输到 onenet 平台，在 onenet 平台上进行手势数据的记录，方便用户远程查看设备的使用情况。

### 1.3.3 LoRa 实现无线通讯

利用 Lora 的低功耗和高灵敏度与传输距离远的特点，连接了 artpi 和 stm32 小车。为了实现信息互传，采用了透明广播模式，再利用串口进行信息交换，实现了远程信息交互。

## 1.4 关键性能指标

手势识别精度是本系统关键性指标之一。本系统采用了高精度陀螺仪，然后对传感器的零漂做了补偿处理，减小其误差。同时收集不同实验者的手势的多组数据，使得每个手势的数据集达到了 14000，然后对这些大量数据做归一化处理，我们新建一个代码块利用八二法则将数据集拆分成评估数据集和训练数据集。其中评估用数据集为 2，训练用数据集为 8。使用模型进行推理，使得手势判断的正确率达到了 98%-100%之间，手势判断精度高。

## 1.5 主要创新点

本产品为手势智能控制系统，实现了手势控制智能小车运动，使得“挥”之即来，“挥”之即去成为现实。本作品主要具有如下优势：

(1) 为了减小数据误差，函数处理 mpu6050 每次上电后会发生飘移；同时 mpu 设备传入的数据过大，不便在 ai 模型中训练及测试，因此归并了数据，使

其全部介于 0-1 之间，便于机器学习，详情见附录；

（2）通过采集大量数据做归一化处理建立数据库，对不同使用者的初始数据做线性回归，实现了手势不受使用者限制，匹配、自适应任意使用者的习惯，实现了通用化，减小了个体差异性带来的误差，泛化能力强。

（3）充分利用了板载 WiFi，将 art-pi 处理好的数据发送到 onenet 平台，在 onenet 平台上对数据进行实时记录，方便用户可以远程监控设备使用情况。

（4）采用了 LoRa 无线通讯，LoRa 是一种基于扩频技术的远距离无线传输技术，其实也是是诸多 LPWAN 通信技术中的一种。LoRa 为我们提供一种简单的能实现远距离、低功耗无线通信手段。

## 第二部分 系统组成及功能说明

### 2.1 整体介绍

本系统基于 STM32H750 为微控制器，MPU6050、stm32 的智能小车、lora 等组成，能够实现手势语音控制物联网家电功能。手势识别部分通过传感器来捕获手指的动作和姿态，可实现手的弯曲度检测、手的空间运动轨迹检测等功能，能够及时准确地处理手势转换为数字信息与智能小车进行交互。本系统具有手势操作、低功耗、速度快，实时性好，识别率高，应用性高，实用性强，适应能力强等显著优点。极大地方便了人们日常的生活，且易于普及。系统框图如图 1 所示。

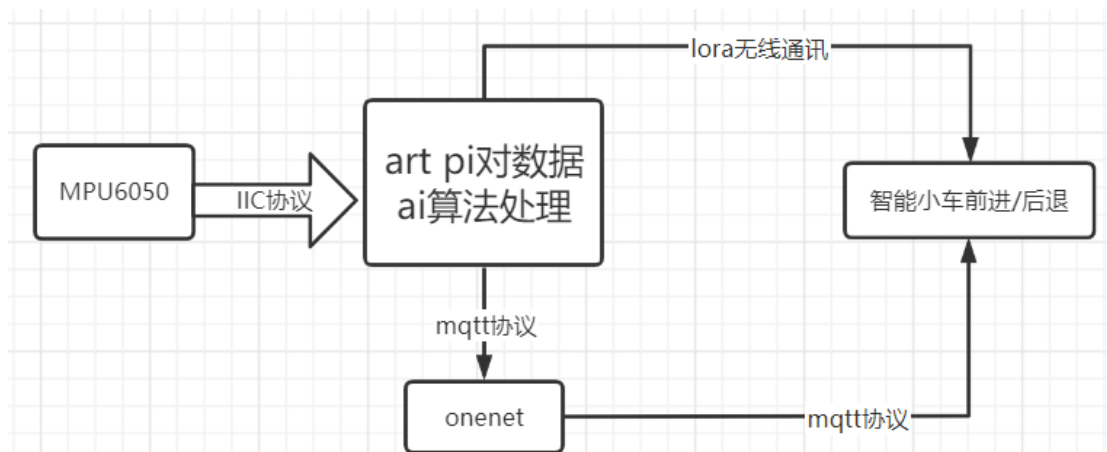


图 1. 系统框图

### 2.2 各模块介绍

#### (1) MPU6050 倾轴传感器

MPU6050 内部整合了 3 轴陀螺仪和 3 轴加速度传感器，并且含有一个 IIC 接口，可用于连接外部磁力传感器即 AUX\_CL 和 AUX\_DA，并利用自带的数字运动处理器（DMP: DigitalMotion Processor）硬件加速引擎，通过主 IIC 接口，向应用端输出完整的 9 轴融合演算数据。

#### (2) STM32H750 单片机

STM32H750 单片机片上资源如下：

内核：ARM - Cortex-M7 工作频率：400MHZ

空间：ROM (flash): 128KB 字节 RAM: 1M 字节

定时器：14 个

5 个低功耗定时器

12 个通用 16 位定时器

6 个 SPI 通讯接口

4 个 IIC 通讯接口

4 个 UART

超过 15 路 PWM(支持高精度定时器 HGTIM)

### (3) 智能两轮平衡小车

内核：ARM 32 位的 Cortex -M3 CPU

存储器：从 16K 到 32K 字节的闪存程序存储器；

从 6K 到 10K 字节的 SRAM；

80 个快速 I/O 端口；

6 个定时器；

6 个通信接口

### (4) LoRa 扩频无线串口模块

AS32-TTL-100 是一款 433MHZ, 100mW, 具有高稳定性, 工业级的无线数传模块。LoRa 扩频调制, TTL 电平输出, 大大提高了模块自由切换, 在省电工作状态下, 消耗电流极低, 非常适合超低功耗应用。

- 具有定点传输、透明传输和空中唤醒功能
- 自动中继、连续传输
- 休眠电流低至 1.5uA
- 超低接收功耗
- 接收灵敏度高达 130dBm, 传输距离 3000 米
- 数据加密

## 第三部分 完成情况及性能参数

### 3.1 手势动作解析

两个基本的控制手势： 出拳、抬手，如图 2。



图 2. 动作示范

为排除系统偶然性，以连续多次采样数据作为判决数据。当几次数据均满足起始条件，判断手势开始；同时满足终止条件判断手势结束；其他情况下，判断为有效数据段。不同的操作者的手指长度、手掌大小等因素会导致采集的传感器手势数据具有一定的差异，但是传感器的绝对大小和手掌活动范围的大小是相对固定的，因此可以建立一个通用的手势模板。采集 100 组手势数据为标准手势数据文本描述，在这里我们把打拳动作的标签号定义为 0，抬手动作定义为 1，图 3 所示为抬手的手势预期标准输出。

```
[D/main] -70, 323, 755, 247, 254, 80
[D/main] -95, 247, 770, 259, 253, 84
[D/main] -123, 173, 782, 277, 254, 86
[D/main] -150, 100, 794, 287, 251, 71
[D/main] -167, 32, 793, 313, 248, 70
[D/main] -178, -19, 782, 328, 241, 71
[D/main] -181, -40, 772, 355, 231, 52
[D/main] -164, -79, 769, 333, 249, 60
[D/main] -125, -125, 731, 375, 251, 90
[D/main] -155, -158, 701, 385, 250, 79
[D/main] -152, -178, 677, 395, 248, 64
[D/main] -138, -195, 652, 391, 255, 61
[D/main] -135, -222, 617, 392, 257, 57
[D/main] -129, -262, 576, 400, 252, 74
[D/main] -116, -293, 538, 402, 248, 59
[D/main] -82, -319, 492, 386, 240, 63
[D/main] 36, -378, 450, 443, 303, 19
[D/main] 22, -373, 409, 389, 277, 85
[D/main] 6, -390, 402, 390, 291, 72
[D/main] 2, -364, 351, 378, 295, 70
[D/main] -11, -336, 308, 347, 291, 89
[D/main] -8, -323, 273, 319, 301, 87
[D/main] -19, -329, 245, 294, 322, 94
[D/main] -30, -330, 232, 277, 324, 87
```

图 3. 抬手预期标准输出

由多组实验可得抬手的数值区间为：(-2000,+2000)

图 4 所示为出拳的手势预期标准输出。

```
[D/main] 738, -427, -391, -28, -179, 326
[D/main] 804, -512, -429, -15, -163, 343
[D/main] 805, -535, -455, -35, -141, 325
[D/main] 801, -529, -451, -44, -94, 315
[D/main] 762, -513, -423, -31, -54, 313
[D/main] 771, -478, -337, -25, -16, 294
[D/main] 772, -453, -264, -15, 8, 282
[D/main] 770, -446, -214, -5, 41, 286
[D/main] 782, -476, -139, -6, 24, 232
[D/main] 863, -498, -76, 47, 27, 189
[D/main] 899, -490, -32, 64, -4, 121
[D/main] 944, -500, 9, 90, -16, 87
[D/main] 998, -465, 12, 147, -35, 66
[D/main] 1036, -439, 20, 234, -42, 76
[D/main] 1064, -382, 88, 221, -76, 93
[D/main] 1055, -314, 35, 233, -98, 61
[D/main] 1069, -271, -24, 257, -105, 59
[D/main] 1077, -223, -69, 292, -145, 70
[D/main] 1098, -168, -123, 333, -154, 63
[D/main] 1111, -135, -187, 328, -155, 70
[D/main] 1127, -118, -240, 351, -131, 106
[D/main] 1114, -91, -310, 342, -103, 110
[D/main] 1084, -52, -359, 349, -89, 108
```

图 4. 出拳预期标准输出

由多组实验可得出拳的数值区间也为：(-2000,+2000)

### 3.2 手部运动分析

手部运动分析为了跟踪手部运动，我们使用陀螺仪。陀螺仪传感器测量角速度的变化。如果任何轴发生变化，则它将发送电信号输出。陀螺仪传感器提供大量的漂移值输出。我们编写了函数补偿的方法减小了陀螺仪传感器零漂的误差。计算可以得出，阈值处理之后的角度偏差变化基本是呈线性关系变化，MATLAB 拟合出该直线方程为： $\text{angle}=0.0008055-0.02175(t \text{ 的单位是秒, angle 的单位是度})$ 。可见，可采用拟合直线方程对 xv8000 的数据进行补偿，可进一步将角度误差减小。所以可以做多组实验，然后取前十组偏差变化的均值作为角度的补偿值。

### 3.3 手势识别 h5 模型搭建及部署

用 np、pd 科学计算库进行数据的处理

```
punchX, punchY = processData(punch, 0)
flexX, flexY = processData(flex, 1)
dataX = np.concatenate((punchX, flexX), axis = 0)
dataY = np.concatenate((punchY, flexY), axis = 0)
```

用 keras 生成一个 Sequential 模型

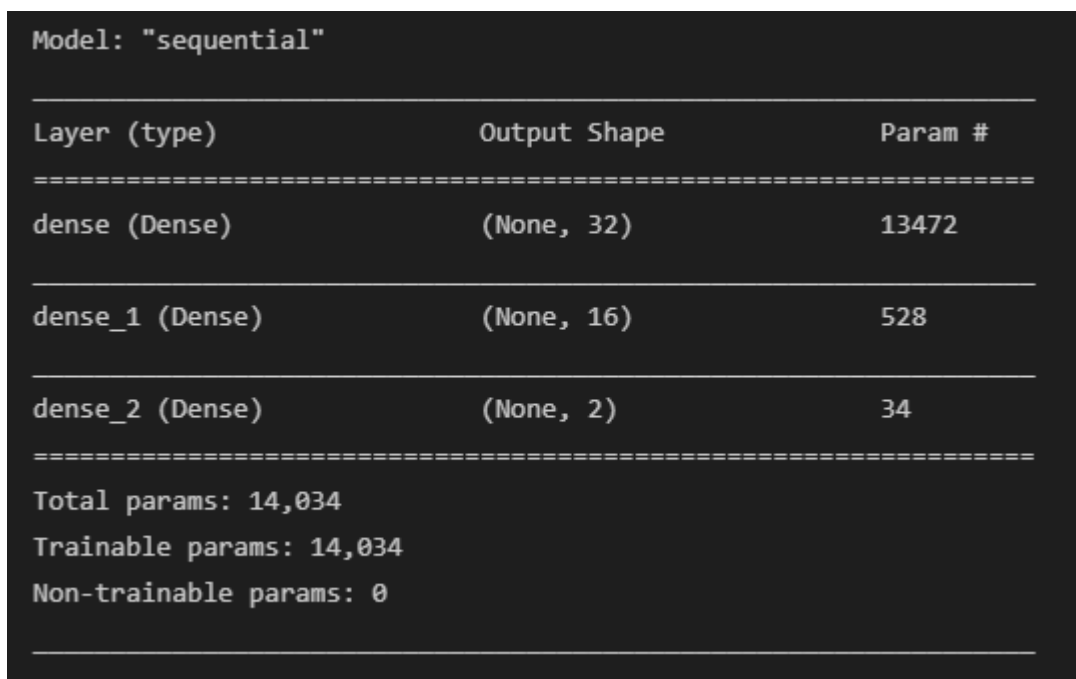
```
model = keras.Sequential()
```

用以下代码搭建模型

```
model.add(keras.layers.Dense(32, input_shape=(6*SAMPLES_PER_GESTURE,),
activation='relu'))
model.add(keras.layers.Dense(16, activation='relu'))
model.add(keras.layers.Dense(2, activation='softmax'))
```

之后进行编译并输出

```
model.compile(loss='sparse_categorical_crossentropy',
optimizer=adam,
metrics=['sparse_categorical_accuracy'])
model.summary()
```



```
Model: "sequential"
_____
Layer (type)                 Output Shape              Param #
=====
dense (Dense)                 (None, 32)                13472
_____
dense_1 (Dense)               (None, 16)                 528
_____
dense_2 (Dense)               (None, 2)                  34
=====
Total params: 14,034
Trainable params: 14,034
Non-trainable params: 0
_____
```

图 5. 模型

训练模型并保存

```
history = model.fit(xTrain, yTrain, batch_size=1, validation_data=(xTest, yTest),
epochs=200, verbose=1)
model.save('gs_mnist.h5')
```

部署 H5 模型到 ART-PI

```
python aitools.py --project="E:\RT-ThreadStudio\workspace\mpuTest" --
model="./Models/gs_mnist.h5" --platform=stm32 --ext_tools="F:\stm32ai-windows-
5.2.0\windows" --clear
```



```
PS F:\ai_download\flusbonading\RT-AK-main\RT-AK\rt_ai_tools> python aitools.py --project= E:\RT-ThreadStudio\workspace\mpuTest --model= ./Models\gs_mnist.h5 --platform=stm32 --ext_tools=F:\stm32ai-windows-5.2.0\windows --clear
2021-09-27 20:43:08 git_plugin [INFO]: git version 2.32.0.windows.2
No stash entries found.
2021-09-27 20:43:09 git_plugin [INFO]: Install plugin_stm32 successfully...
2021-09-27 20:43:09 is_valid_model [INFO]: The model is 'gs_mnist.h5'
2021-09-27 20:43:09 is_valid_cpu [INFO]: The cpu is 'M7'
2021-09-27 20:43:09 set_env [INFO]: stm32ai - Neural Network Tools for STM32 v1.4.0 (AI tools v5.2.0)
2021-09-27 20:43:09 pre_sconsript [INFO]: Create two dirs: Middlewares K-CUBE-AI successfully...
2021-09-27 20:43:13 stm32ai [INFO]: Model convert to c-model successfully...
2021-09-27 20:43:13 rt_ai_model_gen [INFO]: Generate rt_ai_network_model.h successfully...
2021-09-27 20:43:13 load_rt_ai_example [INFO]: Load rt_ai examples successfully...
2021-09-27 20:43:13 load_lib [INFO]: Loading stm32ai libs successfully...
2021-09-27 20:43:13 load_to_project [INFO]: Middlewares loading to project successfully...
2021-09-27 20:43:13 load_to_project [INFO]: K-CUBE-AI loading to project successfully...
2021-09-27 20:43:13 enable_hal_crc [INFO]: Don't need to enable HAL_CRC again!!!
2021-09-27 20:43:13 enable_platform [INFO]: No need to enable RT-AK Lib again...
2021-09-27 20:43:13 load_rt_ai_lib [INFO]: RT-AK Libs loading successfully...
```

### 3.4 手势动作判断

使用模型进行推理，我们在之前收集数据集的过程是先判断和加速度是否超过阈值，如果超过，就开始记录数据，直到记录 70 条数据后进行数据的输出。那么我们就可以将原本要输出的数据输入到模型中去推理。找到原本输出数据的代码区域进行迭代，我们使用 `rt_memcpy()` 将一组 70 条每条 6 种的数据输入到模型中。当然这些数据在输入之前需要先经过与模型训练时一样的正规化操作。通过 `rt_ai_run()` 完成推理后，我们将推理结果通过 `rt_ai_output(model, 0)` 进行输出，我们用 `sprintf()` 和 `rt_kprintf()` 方法结合输出浮点型两个动作的准确率，利用 for 语句来判断属于哪个动作。如图 6 为抬手手势测试输出，图 7 为出拳手势测试输出。

```
[D/main] 47, 2182, 995, 187, 162, -110
[D/main] 67, 1908, 991, 85, 130, -157
[D/main] 16, 1647, 1046, 36, 141, -152
[D/main] 47, 1386, 1047, -103, 136, -151
[D/main] 62, 1079, 1097, -186, 138, -147
[D/main] 33, 729, 1113, -252, 92, -171
[D/main] -14, 352, 1133, -285, 55, -190
[D/main] -54, -25, 1162, -282, 43, -212
[D/main] -105, -377, 1185, -250, 60, -210
[D/main] -164, -699, 1211, -184, 77, -196
[D/main] -193, -958, 1201, -89, 81, -186
[D/main] -179, -1162, 1187, 101, 119, -179
[D/main] -135, -1303, 1146, 205, 126, -172
[D/main] -89, -1313, 1127, 317, 163, -161
[D/main] -43, -1262, 1072, 425, 200, -155
[D/main] 15, -1152, 1005, 533, 233, -164
[D/main] 49, -998, 935, 615, 266, -166
[D/main] 69, -835, 859, 693, 285, -172
[D/main] 81, -622, 770, 736, 302, -196
[D/main] 60, -401, 676, 732, 333, -188
[D/main] 18, -199, 584, 731, 355, -166
[D/main] -1, -104, 540, 709, 374, -142
punch: 0.000074, flex: 0.999926 [AI.LOG]The prediction is : flex
[D/main] [0m[D/onenet.http] buffer : {"zhengyc":1}[0m
[D/main] -8, 4, 6, 2, 8, 0
[D/main] -7, 4, 7, -2, 6, -4
[D/main] -8, 2, 6, 2, 6, -5
```

图 6. 抬手手势测试输出

```
[D/main] -374,-1728,-157,-52,18,123
[D/main] -449,-1556,-172,-74,133,250
[D/main] -508,-1250,22,-87,136,231
[D/main] -551,-1000,224,-102,75,131
[D/main] -511,-857,288,-72,5,94
[D/main] -481,-780,270,-42,-17,87
[D/main] -439,-661,208,10,1,40
[D/main] -256,-488,173,34,33,29
[D/main] -168,-245,165,54,49,102
[D/main] -141,4,172,34,32,70
[D/main] -136,103,189,63,82,126
[D/main] -168,252,203,75,93,154
[D/main] -201,320,186,83,105,126
[D/main] -208,335,157,90,110,103
[D/main] -240,335,132,136,110,103
[D/main] -205,323,97,136,107,57
[D/main] -90,273,71,133,116,62
[D/main] -52,239,66,136,120,107
[D/main] -78,170,51,119,113,99
[D/main] -135,80,46,117,121,86
[D/main] -172,-17,44,115,130,98
[D/main] -182,-93,43,124,134,100
punch:0.995719, flex:0.004281 [AI.LOG]The prediction is : punch
[D/main] [0m[D/onenet.http] buffer : {"zhengyc":0} [0m
[D/main] -8,4,6,1,8,3
[D/main] -7,2,5,1,10,8
[D/main] -8,3,5,0,8,0
```

图 7. 出拳手势测试输出

### 3.5 将数据上传至 onenet

通过 art-pi 的板载 WiFi 将手势数据上传至 onenet 平台进行数据同步与控制，方便用户远程观测数据。图 8 为手势判断。在用户手势判断结束后，将数据同步至 onenet 如图 9 所示。其中 0 代表 punch（出拳），1 代表 flex（抬手）。

```
[32m[I/WWD] wifi initialize done. wicd version 3.3.1 [0m
I/WLAN.dev] wlan init success
I/WLAN.lwip] eth device init ok name:w0
I/WLAN.mgmt] wifi connect success ssid:Geohot
I/WLAN.lwip] Got IP address : 192.168.43.70
[0m[D/mqtt] restart! [0m
[0m[D/onenet.mqtt] Enter mqtt_connect_callback! [0m
[0m[D/mqtt] ipv4 address port: 6002 [0m
[0m[D/mqtt] HOST = '183.230.40.39' [0m
[32m[I/mqtt] MQTT server connect success. [0m
[0m[D/onenet.mqtt] Enter mqtt_online_callback! [0m
I/mpu6xxx] Find device: mpu6050!
I/mpu6xxx] Device init succeed!
D/main] -----init over-----
```

图 8. 手势判断

设备ID	设备名称	设备状态	最后在线时间	操作
787339942	RTT1	在线	2021-09-26 21:43:53	<a href="#">详情</a> <a href="#">数据流</a> <a href="#">更多操作</a> <span>▼</span>
共1项		<div> <span>&lt;</span> <span>1</span> <span>&gt;</span> <span>跳至</span> <input type="text" value="1"/> <span>页</span> </div>		

设备数据总数(个)	昨日新增(个)	最近7日新增(个)	<a href="#">数据流模板管理</a>
2542	205	205	

■ 面板

■ 列表

实时刷新: ☒

置顶区域

收起置顶

zhengyc

2021-09-26 21:44:48

...

0

共1页

<
1
>
跳至

页

图 9. 上传数据

## 第四部分 总结

### 4.1 可扩展之处

高实时性的手势语音控制系统可以应用于很多其它设备，将 LoRa 连接不同的设备，同时丰富手势，实现不同手势控制不同设备功能，真正做到万物互联，提高系统的应用性和普及度。如，将手势语音控制系统应用于智能窗帘，通过 ART-PI，用 LoRa 连接窗帘，实现手势与语音控制窗帘的开关；还可以将手势语音控制系统应用于空调开关。总之，可以将此设备应用的智能化家居，提高家具的智能度，这些都可以极大的提高人们的幸福感，让家成为休闲娱乐的场所，从而解放人类的双手，增强人们的生活体验。

### 4.2 心得体会

智慧家居的迅速发展和广泛应用，使得手势语音控制系统更容易普及与推广，真正的做到了用技术改变生活。手带传感器，便可做到‘呼’之即来，‘挥’之即去，让我们深刻体会到了手势语音控制系统的极大便利。通过本次实验我们加深了对 TensorFlow 的学习，它的高度灵活性和可移植性为我们的学习使用提供了诸多便利。使得数据更加真实和更贴切我们的实验需求。虽然收获颇多但是在这中间也发现了我们存在的不足，我们的探索方式还有待改善，当应对一些复杂实验时我们还不能很快很好的完成以及数据处理能力还有待提高，不能用最佳的处理手段将实验误差减小到最小程度。在今后的学习中我们将不断完善，提高自己的能力。同时，我们也将对此设备进行不断完善，以供以后进一步使用和推广。

## 第五部分 参考文献

- [1]倪训博.基于手语语言学与人体运动学的手语识别研究[D].哈尔滨: 哈尔滨工业大学, 2019
- [2]强威, 李柏辉, 张钧杰, 刘凯, 高宏峰.基于角加速度的手势识别系统[J].山西电子技术, 2018, (04): 16-18
- [3]王剑.关于网络数据归一化处理探讨[J].信息系统工程, 2015, (12): 60
- [4]周世健, 张立亭, 鲁铁定.测量数据处理的实时序贯算法[J].测绘学院学报, 2002, (03): 165-167+173
- [5]薛先贵, 黎路.深度学习工具 Tensorflow 的解析[J].福建电脑, 2021, 37(01): 105-106
- [6]文海龙, 王小华.一种基于 LoRa 的 LED 智能照明远程控制系统设计[J].中国科技信息, 2021, (15): 58-59
- [7]田旭飞, 姚凯学, 王凯鹏, 王运峰.基于 LoRa 和 STM32 的路灯自动控制系统研究[J].计算机工程与科学, 2021, 43(08): 1470-1478

## 第六部分 附录

### 1.numpy科学计算处理数据

```
def processData(d, v):
    dataX = np.empty([0,SAMPLES_PER_GESTURE*6])
    dataY = np.empty([0])

    data = d.values
    dataNum = data.shape[0] // SAMPLES_PER_GESTURE

    for i in tqdm(range(dataNum)):
        tmp = []
        for j in range(SAMPLES_PER_GESTURE):
            tmp += [(data[i * SAMPLES_PER_GESTURE + j][0] + 2000.0) / 4000.0]
            tmp += [(data[i * SAMPLES_PER_GESTURE + j][1] + 2000.0) / 4000.0]
            tmp += [(data[i * SAMPLES_PER_GESTURE + j][2] + 2000.0) / 4000.0]
            tmp += [(data[i * SAMPLES_PER_GESTURE + j][3] + 2000.0) / 4000.0]
            tmp += [(data[i * SAMPLES_PER_GESTURE + j][4] + 2000.0) / 4000.0]
            tmp += [(data[i * SAMPLES_PER_GESTURE + j][5] + 2000.0) / 4000.0]
        tmp = np.array(tmp)
        tmp = np.expand_dims(tmp, axis = 0)

        dataX = np.concatenate((dataX, tmp), axis = 0)
        dataY = np.append(dataY, v)

    return dataX, dataY

punchX, punchY = processData(punch, 0)
flexX, flexY = processData(flex, 1)
dataX = np.concatenate((punchX, flexX), axis = 0)
dataY = np.concatenate((punchY, flexY), axis = 0)

permutationTrain = np.random.permutation(dataX.shape[0])
print(permutationTrain)
```

```
[ 27 101  53  37 132  58 172 149 114  11 153  26  81 174  13  98 147  65
 95  51  66 110  82  42  67  39 154 104 156 165  88 170 186 119  78  43
 74 145  16  57  50 133  14  61 178 181 179  54 189 125 105  71  20 122
192 124   2 183 144 198  32  56 151  97  68 155  40 163 107  25   6  10
171  73 175   0 184 161  24   8  85 102  45 180 109  17  29  44   7  99
121 103 197 193 130   3 148 166 150 137  12 176 106 138  84  87 113 111
 47  90  31 134  69 152 167  46   4 182  33  92 190 146  15  60  52  35
129 195 162  55 187  59 177  94  49  76 196  72 164 199  80 131  91 158
 34 126  63  93  19 194  77  79  22 128  28  30 157  38  64 169  41 141
173 112  70  23 100 139 120  48 108 140 142 188  96  36 127 185 116   1
159 136 191  75  83 168  18  86  89 135 123 160   9  62 118   5 115  21
117 143]
```

```
dataX = dataX[permutationTrain]
dataY = dataY[permutationTrain]
print(dataY)
```

```
[1.  0.  1.  1.  0.  1.  1.  0.  0.  0.  0.  0.  0.  0.  1.  0.  1.  1.  0.  1.  0.  0.  1.
 0.  0.  0.  1.  0.  1.  0.  0.  0.  0.  0.  1.  1.  1.  1.  0.  1.  0.  1.  1.  0.  1.  1.  0.
 1.  1.  1.  1.  0.  1.  0.  0.  0.  0.  1.  1.  0.  1.  1.  1.  0.  1.  1.  0.  1.  1.  0.  1.
 1.  1.  0.  0.  0.  0.  1.  0.  0.  1.  0.  1.  1.  0.  0.  1.  0.  1.  0.  1.  1.  1.  1.  1.
 0.  1.  0.  1.  0.  0.  1.  0.  1.  0.  0.  1.  0.  0.  1.  1.  0.  1.  1.  0.  1.  1.  0.  1.
 1.  1.  1.  0.  0.  1.  0.  1.  1.  1.  1.  0.  0.  0.  1.  1.  0.  1.  0.  0.  0.  1.  0.  1.
 1.  1.  0.  1.  0.  0.  0.  1.  0.  1.  1.  1.  1.  1.  0.  0.  1.  0.  0.  1.  0.  0.  1.  0.
 0.  0.  1.  1.  1.  0.  0.  1.  1.  1.  0.  0.  0.  0.  0.  1.  1.  1.  1.  0.  0.  0.  1.
 1.  0.  0.  1.  1.  0.  0.  0.]
```

```
vfoldSize = int(dataX.shape[0]/100*20)
```

```
xTest = dataX[0:vfoldSize]
yTest = dataY[0:vfoldSize]
```

```
xTrain = dataX[vfoldSize:dataX.shape[0]]
yTrain = dataY[vfoldSize:dataY.shape[0]]
```

## 2.模型构建

```
model = keras.Sequential()
model.add(keras.layers.Dense(32, input_shape =(6*SAMPLES_PER_GESTURE,), activation='relu'))
model.add(keras.layers.Dense(16, activation='relu'))
model.add(keras.layers.Dense(2, activation='softmax'))
adam = keras.optimizers.Adam()
```

```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=adam,
              metrics=['sparse_categorical_accuracy'])
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	13472
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 2)	34

Total params: 14,034  
 Trainable params: 14,034  
 Non-trainable params: 0

### 3.模型训练

```
history = model.fit(xTrain, yTrain, batch_size=1, validation_data=(xTest, yTest), epochs=200, verbose=1)
```

Train on 160 samples, validate on 40 samples

Epoch 1/200  
 160/160 [=====] - 2s 11ms/sample - loss: 0.7294 - sparse\_categorical\_accuracy: 0.5000 - val\_loss: 0.7323 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 2/200  
 160/160 [=====] - 1s 5ms/sample - loss: 0.7042 - sparse\_categorical\_accuracy: 0.5000 - val\_loss: 0.6894 - val\_sparse\_categorical\_accuracy: 0.6000  
 Epoch 3/200  
 160/160 [=====] - 1s 5ms/sample - loss: 0.7047 - sparse\_categorical\_accuracy: 0.5312 - val\_loss: 0.7446 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 4/200  
 160/160 [=====] - 1s 5ms/sample - loss: 0.6994 - sparse\_categorical\_accuracy: 0.4625 - val\_loss: 0.6958 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 5/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6929 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.6968 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 6/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6929 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.6977 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 7/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6926 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.6990 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 8/200  
 160/160 [=====] - 1s 7ms/sample - loss: 0.6924 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.6997 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 9/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6927 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.7007 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 10/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6923 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.7015 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 11/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6923 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.7021 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 12/200  
 160/160 [=====] - 1s 7ms/sample - loss: 0.6923 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.7016 - val\_sparse\_categorical\_accuracy: 0.4000  
 Show more (open the raw output data in a text editor) ...  
 Epoch 181/200  
 160/160 [=====] - 1s 7ms/sample - loss: 0.6922 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.7044 - val\_sparse\_categorical\_accuracy: 0.4000  
 Epoch 182/200  
 160/160 [=====] - 1s 6ms/sample - loss: 0.6926 - sparse\_categorical\_accuracy: 0.5250 - val\_loss: 0.7040 - val\_sparse\_categorical\_accuracy: 0.4000

### 4.mpu6050 收集并处理数据线程

```
void mpu_get_entry(void *parameter)
```



```
{
    i2c_bus = (struct mpu6xxx_device *)mpu6xxx_init(MPU6050_I2C_BUS_NAME, MPU6050_A
DDR);
    mpu6xxx_set_param(i2c_bus, MPU6XXX_ACCEL_RANGE, MPU6XXX_GYRO_RANGE_250DP
S);
    mpu6xxx_set_param(i2c_bus, MPU6XXX_ACCEL_RANGE, MPU6XXX_ACCEL_RANGE_2G);
    mpu6xxx_set_param(i2c_bus, MPU6XXX_DLPF_CONFIG, MPU6XXX_DLPF_188HZ);
    mpu6xxx_set_param(i2c_bus, MPU6XXX_SAMPLE_RATE, 500);

    while(1){
        /* 获取数据 */
        mpu6xxx_get_gyro(i2c_bus, &gyro1);
        mpu6xxx_get_accel(i2c_bus, &accel1);
        // LOG_D("gyroX:%d\tyroY:%d\tyroZ:%d\t", gyro1.x, gyro1.y, gyro1.z);
        // LOG_D("accelX:%d\taccelY:%d\taccelZ:%d\t", accel1.x, accel1.y, accel1.z);
        /*****
            * 解决 mpu6050 瞎跑问题
            *****/
        if(10<=mpuCount&&mpuCount<20)
        {
            px+=accel1.x;
            py+=accel1.y;
            pz+=accel1.z;
        }
        accel1.x-=px/10;
        accel1.y-=py/10;
        accel1.z-=pz/10;

        if(mpuCount==20) LOG_D("-----init over-----
-----");

        if(mpuCount>=20)
        {
            ///////////////////////////////////
            // LOG_D("px:%d\tpy:%d\tpz:%d", px/10, py/10, pz/10);
            // LOG_D("gyroX:%d\tyroY:%d\tyroZ:%d\t", gyro1.x, gyro1.y, gyro1.z);
            // LOG_D("accelX:%d\taccelY:%d\taccelZ:%d\t", accel1.x, accel1.y, accel1.z);
            // LOG_D("%d,%d,%d,%d,%d,%d", gyro1.x, gyro1.y, gyro1.z, accel1.x, accel1.y, accel1.z);
            //判断阈值或已经进入数据采集模式
            if((accel1.x+accel1.y+accel1.z) >= 10*accelerationThreshold || ai_count!=0)
            {
                if(ai_count==1)
                // LOG_D("start");
                /***** 处理数据，变为 [0,1]，便于机器学习 *****/
            }
        }
    }
}
```

```

*****/

    gx=(gyro1.x+2000.0)/4000.0;
    gy=(gyro1.y+2000.0)/4000.0;
    gz=(gyro1.z+2000.0)/4000.0;
    ax=(accel1.x+2000.0)/4000.0;
    ay=(accel1.y+2000.0)/4000.0;
    az=(accel1.z+2000.0)/4000.0;

    /***** 打印浮点型数据，RTT未自带
*****/

    //      char float_str[80];
    //      sprintf(float_str,"gx:%f\tyg:%f\tgz:%f\tax:%f\tay:%f\taz:%f\t",gx,gy,gz,ax,ay,az);
    //      LOG_D(float_str);
    ai_data[ai_count][0] = gx;
    ai_data[ai_count][1] = gy;
    ai_data[ai_count][2] = gz;
    ai_data[ai_count][3] = ax;
    ai_data[ai_count][4] = ay;
    ai_data[ai_count][5] = az;
    ai_count++;
    /*****AI数据足以推理一次，并上报数据
*****/

    if(ai_count==72)
    {

        //ai推理
        mnist_app();
        ai_count=0;
        rt_thread_mdelay(5000);
    }

}

mpuCount++;
}
}

```

## 5.AI模型推理线程

```

/*
 * Copyright (c) 2006-2021, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:

```

---

```

* Date      Author    Notes
* 2021-09-11  zhengyc  first version
*/

#include "mnist_app.h"

#define THREAD_PRIORITY    5
#define THREAD_STACK_SIZE 2048
#define THREAD_TIMESLICE  5

static rt_thread_t tid1 = RT_NULL;

//传入的数据集
extern const float ai_data[70][6];
static rt_ai_t model = NULL;
int pred_num = 0;

char *mnist_label[] = {"punch", "flex"};

void ai_run_complete(void *arg){
    *(int*)arg = 1;
}

/* ai 线程 的入口函数 */
void ai_run_entry(void *parameter)
{
    rt_err_t result = RT_EOK;
    int ai_run_complete_flag = 0;
    rt_ai_buffer_t *work_buffer = rt_malloc(RT_AI_NETWORK_WORK_BUFFER_BYTES +
                                             RT_AI_NETWORK_IN_TOTAL_SIZE_BYTES +
                                             RT_AI_NETWORK_OUT_TOTAL_SIZE_BYTES);

    // find a registered model handle
    model = rt_ai_find(RT_AI_NETWORK_MODEL_NAME);
    if(!model){
        rt_kprintf("ai model find err\r\n");
        return -1;
    }

    // init the model and allocate memory
    result = rt_ai_init(model, work_buffer);
    if (result != 0) {
        rt_kprintf("ai init err\r\n");
        return -1;
    }

```

```

    }

    // prepare input data
    rt_memcpy(model->input[0], ai_data, RT_AI_NETWORK_IN_1_SIZE_BYTES);
    result = rt_ai_run(model, ai_run_complete, &ai_run_complete_flag);
    if (result != 0) {
        rt_kprintf("ai model run err\n");
        return -1;
    }

    // get output and post-process the output

    if(ai_run_complete_flag){
        float *out = (float *)rt_ai_output(model, 0);

        char ai_str[80];
        //显示传入的 ai 数据
        for(int i=0; i<70;i++)
        {
            sprintf(ai_str,"aidata: [%f,%f,%f,%f,%f,%f]\r\n",ai_data[i][1],ai_data[i][2],ai_data[i][3],ai_data
[i][4],ai_data[i][5],ai_data[i][6]);
            //      rt_kprintf(ai_str);
        }
        //      /*****打印*****/
        char float_str[80];
        sprintf(float_str,"punch:%f, flex:%f",out[0],out[1]);
        //      AI_LOG(out[0]);
        //      rt_kprintf(float_str);
        //
        //      /*****模型预测*****/
        for(int i = 1 ; i < 2 ; i++){
            if(out[i] > out[pred_num]){
                pred_num = i;
            }
        }
        /* pred_num=1->punch->小车后退 */
        if(out[0]>out[1])
            pred_num = 2;
        else
            pred_num = 1;
        //      AI_LOG("The prediction is : %s\n", mnist_label[pred_num]);

        rt_kprintf("%d",pred_num);

        onenet_http_upload_digit("zhengyc", pred_num);
    }
}

```

```

    }

    rt_free(work_buffer);
    return 0;
}

/* 线程 */
int mnist_app(void)
{
    /* 创建线程，名称是 mnist，入口是 thread_entry*/
    tid1 = rt_thread_create("thread",
                            ai_run_entry, RT_NULL,
                            THREAD_STACK_SIZE,
                            THREAD_PRIORITY, THREAD_TIMESLICE);

    /* 如果获得线程控制块，启动这个线程 */
    if (tid1 != RT_NULL)
        rt_thread_startup(tid1);

    return 0;
}

//MSH_CMD_EXPORT(mnist_app, mnist demo);

```

## 6.平衡小车

```

#include "sys.h"

/*****init*****/
float Voltage;
float pitch,roll,yaw;
short aacx,aacy,aacz;
short gyroX,gyroY,gyroZ;
float UltrasonicWave_Distance;
int Encoder_Left,Encoder_Right;
int Moto1=0,Moto2=0;

int Velocity=0,Turn=0;

u8 key=0;

/******/
int main(void)
{
    LED_Init();
    USB_Init();

```

```

KEY_Init();
delay_init();
uart1_init(115200);
uart3_init(115200);          //=====LORA 无线通讯模块串口
delay_ms(100);

uart3_init(115200);
NVIC_Configuration();
Adc_Init();
Encoder_Init_TIM2();
Encoder_Init_TIM4();
Timer3_Init(5000,7199);
OLED_Init();
OLED_Clear();
MPU_Init();
mpu_dmp_init();
UltrasonicWave_Configuration();
TIM1_PWM_Init(7199,0);
delay_ms(1000);
Motor_Init();
MPU6050_EXTI_Init();
oled_first_show();
while(1)
{
    oled_show();
    delay_ms(50);
}
}

```