机器人感知与交互实验指导书

实验一. 加速度传感器检测速度位置变化

1. **实验内容**

本实验旨在练习通过串行通信方式获得陀螺仪加速度传感器所测得的角度、角速度和加速度值，并对加速度进行积分以得到当前线速度和位置。

1. **实验所需硬件和软件**

本实验需要以下硬件：

1. PC计算机一台，操作系统为Windows或Linux
2. 加速度传感器一套，型号为WT61PC485
3. USB-485通信电缆一套，用于连接传感器和计算机

本实验可采用C/C++、Python语言编写传感器数据采集和处理程序，如采用C/C++语言，计算机上应当预装VS、GCC或其他C/C++开发环境；如采用Python语言，应当预装PyCharm或其他Python开发环境。

1. **硬件接线**

USB-485通信电缆一端为USB接口，应当插入计算机的USB2.0或USB3.0端口。通信电缆另一端为4针扁平口，应当插入加速度传感器的输出端口。

完成接线后，打开计算机的“设备管理器”，在“端口(COM和LPT)”下面会看到新增加了一个USB-SERIAL设备。记住该设备的端口号(在本例中端口号为COM4)。

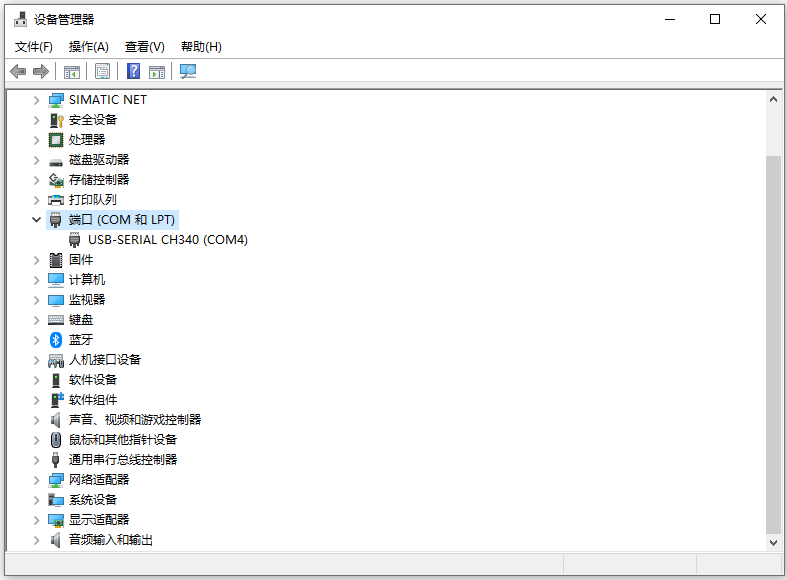


图1. 设备管理器

打开维特智能上位机程序(WitMotion.exe)，点击“搜索设备”按钮，自动搜索传感器。如果接线正确，软件将自动找到传感器，从传感器中读取数据并显示角度(以及加速度、角速度等信息)。如果未能找到传感器，则应当检查接线是否正确，之后再次测试。

维特智能上位机程序连接并读取传感器数据成功后，关闭软件，编写程序读取传感器数据并加以处理。

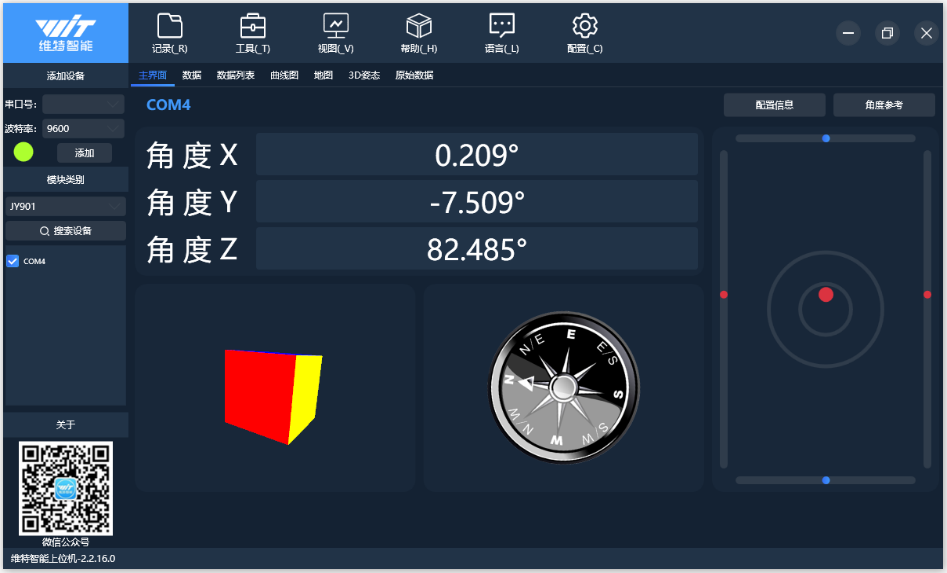


图2. 维特智能上位机程序

1. **传感器数据采集**

传感器通过RS-485总线，以串行通信方式向计算机传输加速度、角速度和角度数据。当计算机通过RS-485总线和传感器连接后，设置通信端口号、波特率(默认为9600)，将持续接收到传感器发出的检测数据。以下为数据传输协议：

1. 加速度数据传输协议

加速度数据报文由11个字节组成，用于传输X、Y、Z三个方向的加速度检测数据(Ax, Ay, Az)，以及传感器温度(T)。为了避免传输受到干扰出现错误，报文末端包含一个校验和(SUM)。

报文格式及数据计算方法如下：

| 0x55 | 0x51 | AxL | AxH | AyL | AyH | AzL | AzH | TL | TH | SUM |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

计算方法：

Ax = ((AxH << 8) | AxL) / 32768 \* 16g (g为重力加速度，可取9.8m/s2)

Ay = ((AyH << 8) | AyL) / 32768 \* 16g (g为重力加速度，可取9.8m/s2)

Az = ((AzH << 8) | AzL) / 32768 \* 16g (g为重力加速度，可取9.8m/s2)

T = ((TH << 8) | TL) / 100 (℃)

Sum = 0x55+0x51+AxH+AxL+AyH+AyL+AzH+AzL+TH+TL

1. 角速度数据传输协议

角速度数据报文格式与计算方法类似于加速度数据报文，具体说明如下：

| 0x55 | 0x52 | wxL | wxH | wyL | wyH | wzL | wzH | TL | TH | SUM |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

计算方法：

Wx = ((wxH << 8) | wxL) / 32768 \* 2000(°/s)

Wy = ((wyH << 8) | wyL) / 32768 \* 2000(°/s)

Wz = ((wzH << 8) | wzL) / 32768 \* 2000(°/s)

T = ((TH << 8) | TL) / 100 (℃)

Sum = 0x55+0x52+wxH+wxL+wyH+wyL+wzH+wzL+TH+TL

1. 角度数据传输协议

| 0x55 | 0x53 | RollL | RollH | PitchL | PitchH | YawL | YawH | VL | VH | SUM |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

滚转角（x轴）Roll = ((RollH << 8) | RollL) / 32768 \* 180(°)

俯仰角（y轴）Pitch = ((PitchH << 8) | PitchL) / 32768 \* 180(°)

偏航角（z轴）Yaw = ((YawH << 8) | YawL) / 32768 \* 180(°)

固件版本计算：Version = (VH << 8) | VL

Sum=0x55+0x53+RollH+RollL+PitchH+PitchL+YawH+YawL+VH+VL

基于C/C++或Python语言编写计算机程序，通过串行端口持续接收加速度传感器发出的检测数据，从中提取出加速度、角速度和角度。读取传感器数据的示例程序(基于Python语言编写)如下：

1. 建立一个加速度传感器类

class AccSensor(object)

1. 在初始化函数中打开串行通信端口，设置波特率和超时时间。初始化用于存储通信报文的缓冲区，以及用于存储加速度、角速度和角度的变量

|  |
| --- |
| def \_\_init\_\_(self, comm\_port):  self.seriel = serial.Serial(comm\_port, 9600, timeout=0.5)   self.seriel\_buffer = []  self.a = None  self.w = None  self.angle = None |

1. 将连续的6个字节转换为3个传感器检测值

|  |
| --- |
| def get\_float\_values(self, data\_hex, k):  """  从串行端口读取到的原始报文中读取三个短整型数，转换为浮点数。乘以量程，得到加速度、角速度或角度值。  :param data\_hex: 原始报文数据  :param k: 量程  :return: 转换结果  """  # 将报文中的字节拼接为短整型数，转换为0.0~2.0之间的浮点数  v1 = (data\_hex[1] << 8 | data\_hex[0]) / 32768.0  v2 = (data\_hex[3] << 8 | data\_hex[2]) / 32768.0  v3 = (data\_hex[5] << 8 | data\_hex[4]) / 32768.0   # 转换为-1.0~1.0之间的浮点数，乘以量程，得到结果  v1 = (v1 - 2.0) \* k if v1 > 1.0 else v1 \* k  v2 = (v2 - 2.0) \* k if v2 > 1.0 else v2 \* k  v3 = (v3 - 2.0) \* k if v3 > 1.0 else v3 \* k  return v1, v2, v3 |

1. 从传感器中读取检测数据

|  |
| --- |
| def read\_from\_sensor(self):  """  从串口中获取传感器输出数据  :return: True: 读取成功, False: 读取失败  """  while True:  # 从串口中读取一个字节  # 如果超时读不到，则返回False  pkg = self.seriel.read(1)  if len(pkg) == 0:  self.seriel\_buffer.clear()  return False   # 将读到的字节写入缓冲区以便于后续处理  # 传感器上电后会持续上传数据报文，程序刚启动时，很可能收到的报文不完整。  # 因此，需要确保缓冲区的头两个字节分别是0x55、0x51，即一个包含了加速度、角速度和角度数据的完整报文的起始字节。  # 当接收到一个33字节完整报文后，检查加速度、角速度和角度数据的校验和是否正确，之后从中提取数据。  self.seriel\_buffer.append(pkg[0])  if len(self.seriel\_buffer) == 1 and self.seriel\_buffer[0] != 0x55 or \  len(self.seriel\_buffer) == 2 and self.seriel\_buffer[1] != 0x51:  self.seriel\_buffer.clear()  elif len(self.seriel\_buffer) == 33:  if sum(self.seriel\_buffer[0:10]) & 0x0FF == self.seriel\_buffer[10] and \  sum(self.seriel\_buffer[11:21]) & 0x0FF == self.seriel\_buffer[21] and \  sum(self.seriel\_buffer[22:32]) & 0x0FF == self.seriel\_buffer[32]:  ax, ay, az = self.get\_float\_values(self.seriel\_buffer[2:8], k=16.0 \* 9.8)  wx, wy, wz = self.get\_float\_values(self.seriel\_buffer[13:19], k=2000.0)  tx, ty, tz = self.get\_float\_values(self.seriel\_buffer[24:30], k=180.0)  self.a = (ax, ay, az)  self.w = (wx, wy, wz)  self.angle = (tx, ty, tz)  self.seriel\_buffer.clear()  return True |

1. 传感器数据处理

从串行通信端口读出加速度传感器数据后，对加速度进行处理，从而获得速度和位置信息。加速度传感器可以检测X、Y、Z三个方向的加速度信号，每个方向的信号处理方式都一致，因此可以定义一个加速度通道类，从而实现代码复用。

加速度信号的处理主要包括：

1. 传感器标定

加速度传感器通常存在一定的零点漂移。将传感器静止在水平面，使其X、Y轴水平于地面，Z轴垂直于地面。理想情况下，X、Y方向的加速度值应当为0，Z方向的加速度值应当等于重力加速度。但是实际情况下，由于存在零点漂移，X、Y、Z方向上的加速度值并不等于以上理想值。通过传感器标定，可以检测并记录出传感器的零点漂移值，在之后的计算中减去该零点漂移，从而获得更为准确的检测结果。传感器标定采用一阶滞后滤波实现。

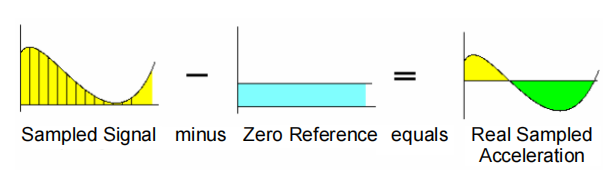


图3. 传感器标定

|  |
| --- |
| def calibrate(self, acc):  zero = self.zero \* self.cali\_k + acc \* (1.0 - self.cali\_k)  delta = math.fabs(zero - self.zero)  self.zero = zero  return self.zero, delta |

1. 信号滤波

对传感器检测得到的原始加速度信号进行滤波处理。首先将加速度信号减去传感器零点，消除可能存在的零点漂移。之后通过一阶滞后滤波，抑制传感器原始信号中的干扰。最后，检查加速度是否小于死区值，如果小于死区值，则将加速度置为零。

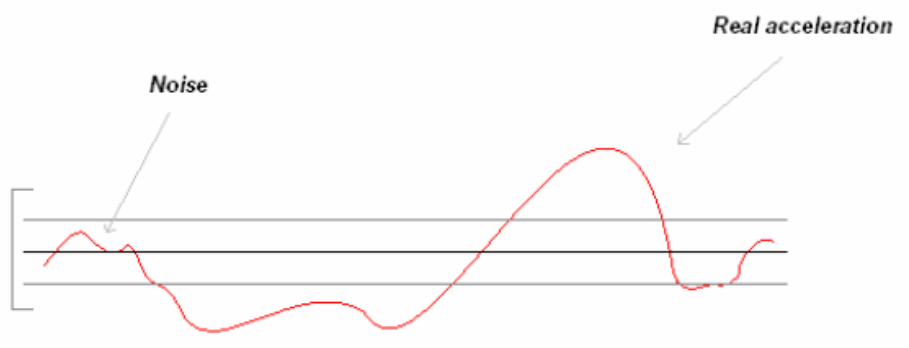


图4. 零点死区

|  |
| --- |
| def filter(self, acc):  acc -= self.zero  acc = self.acc \* self.filter\_k + acc \* (1.0 - self.filter\_k)  if math.fabs(acc) < self.min\_acc:  acc = 0.0  return acc |

1. 速度、位置积分

对加速度积分，得到速度。对速度进一步积分，得到位置。为了提高计算精度，采用梯形积分。

|  |  |
| --- | --- |
|  |  |

图5. 梯形积分

|  |
| --- |
| def accumulate(self, acc, time):  speed = self.speed + (self.acc + acc) \* time \* 0.5  self.position += (self.speed + speed) \* time \* 0.5  self.acc = acc  self.speed = speed  return self.speed, self.position |

1. 持续更新速度和位置

持续对加速度进行积分，从而跟踪传感器的速度和位置。理想情况下，传感器停止运动时，速度积分应当为零。由于存在积分误差，当加速度传感器静止时，很可能速度积分不为零，从而导致位置积分持续增大。为了避免这一问题，当加速度持续一段时间为零时，将速度强制为零，从而避免位置持续积分。



图6. 积分误差导致传感器停止时速度不为零

|  |
| --- |
| def accumulate(self, acc, time):  speed = self.speed + (self.acc + acc) \* time \* 0.5  self.position += (self.speed + speed) \* time \* 0.5  self.acc = acc  self.speed = speed  return self.speed, self.position |