* 本周计划（9.18——9.24）
* 熟悉git、vim，vim-rails以及byobu多窗口。
* 熟悉unity基本操作。
* 学习ruby语言

周一（9.19）

早：

1. 看unity 入门
2. 熟悉vim操作
3. 安装slack、kanbar等工作软件。熟悉dahema工作目录

午：

1. 看unity 视频教程

周二（9.20）

早：

1. 看unity视频

午：

1. 看unity 案例
2. 咨询蓝牙模块与小车对接

周三

1. 看完星际游戏
2. 配置翻墙
3. Git使用尝试

周四

1. 配置好环境

周五：

1. 界面大小，视角等配置好
2. 清楚调用框架

本周计划（9月26-9月30）

游戏模块蓝牙联通，熟悉unity 3d.

周一：

1. 讨论游戏接下来思路，明确要做的事。
2. 找时间熟悉git.

午：

1. 熟悉unity与android的交互，然后明确调试蓝牙思路，调通蓝牙

周二：

Unity与android交互已经可以实现，打包出来的android包也可以实现开启蓝牙和蓝牙连接等。

待办任务：

1. 调试界面的按钮与小车动作的联通情况，目前主要测试动作、速度、时间的控制，后续等协议定义好才能实现灯光、声音等控制。

周三:

1. 捋清楚目前unity3d的游戏框架
2. 试着实现运动模块

周四

1. 弄清楚游戏的代码框架
2. 学习好git的操作

周五

1. 提交代码多提交了buttonController组件。需更正。
2. 弄清楚长按的逻辑。梳理框架。

本周任务（10.8~10.14）

1. 调试智能坦克功能，熟悉框架。
2. 熟悉git、vim操作，抽空学习rubi语言。

周六：

1. 完成长按、组合按键等功能。
2. 了解unity 中加调试宏。

晚：

1. 上传长按部分代码。
2. 思考组合键的实现

存在问题：

1. 通过log确认unity端长按指令可以实现(已确认)
2. 确定指令发送间隔时间（已确认）

* “长按”问题的解决

1. 长按持续运动的实现：之前出现长按断断续续的问题，源于每个动作的运动时间（200ms）太短，虽然发送指令的时间为100ms,但是有可能传输过程中出现延误，导致连续的指令执行起来并不连续。

解决方案是让每个动作的运动时间长一些（2s），然后持续发送指令的间隔依然为100ms。经测试不再出现断断续续的问题。

1. 长按放开后运动停止的实现：通过BtnExit函数来实现，出现的问题是虽然放开按钮一定会调用到该函数，然后通过该函数向小车发送停止指令（速度限定为0），但是小车也是可能指令传输过程中的问题导致单发一条停止指令很多时候不会实现的情况。

解决方案是使用计时器函数invoke，每隔0.1s发送一条停止指令，发送5~6条。

* “长按”或“长按后放开”出现指令不连续的原因：

根源在于安卓系统处理两条相隔很近的蓝牙指令时会忽略后面那条。针对这一情况，如果两条蓝牙指令间隔时间较短，需要使用队列发送。（这个需要学习）

待完成任务：

1. 组合按键的实现
2. 目前遗留的两个问题：
3. 按钮平滑时，从上一个按钮平滑到下一个按钮时的切换过程延迟过长。（很可能时安卓不能处理两条间隔较短的蓝牙指令）
4. 使用队列处理蓝牙丢包问题。
5. 考虑长按以及按钮切换的逻辑是否有更优的策略。

周日：

1. 解决安卓两条蓝牙指令间隔较短时造成丢包的问题。思考队列解决策略。
2. 梳理框架，线程设计，弄明白线程、协程。

关于android的写操作：

1. 如果要发送数据，是写Characteristic ，要在回调函数onCharacteristicWrite中等待写的结果，才能继续写，否则会写失败，所以需要做一个FIFO形式的写缓存队列才能实现可靠的发送数据。

* 修复android连续写丢包问题

目前android端写的实现操作是

public void SendBleCommand(String cmd)

{

if(mBluetoothGatt == null){

return;

}

byte[] data = Utils.hexStringToBytes(cmd);

DataMOSI.setValue(data);

mBluetoothGatt.writeCharacteristic(DataMOSI);

}

解决思路是：1）将要写的data放到缓存队列中，通过onCharacteristicWrite等待写的结果（只有成功时才会返回结果），前一条发完才进行下一条。

周一：

1. 解决“长按”功能
2. 解决“组合按键”功能

“长按”解决：

根源在于之前的蓝牙写操作需要有response才会执行下一条写的指令，目前改为write without response方式，便可以实现间隔较短的多条蓝牙写指令的发送。

“组合按键”思路：

1. 目前“组合按键”需求仅是方向键的组合。配置一个“方向键控制模块”，设置成单例模式。
2. “方向键控制模块”中配置一个全局状态flag。Flag中的4位分别指示前、后、左、右四个按键的状态。
3. 当某个按钮按下或者离开时，需要触发SetFlag（方向控制模块提供的接口），置位当前按钮的flag,然后再调用StartBtnOpr(方向控制模块提供的开始接口)。
4. 各个按钮的运动指令是通过StartBtnOpr来实现，该函数通过判断按钮全局flag的状态，配置相应的运动参数，然后发送蓝牙指令。

Ps:

关于“组合键”中滑键的思考：

滑键是指从“上”方向键平滑到“下”方向键，或者“左”方向键平滑到“右”方向键的过程。涉及3个状态的切换：“上”—》“上”“下”并存—》”下”

1）1、3状态都好处理。

2）2状态的处理方式：保持上一个状态。即只存在两种状态，“上”和“下”。当从“上”切换到“上下并存”时，依然触发“上”操作指令。当从“上下并存”切换到“下”状态才触发“下”操作指令。

实现：

1）“方向键控制模块”有lastBtnFlag,curBtnFlag两个状态变量，分别指示上一个按键组合状态和当前按键组合状态。

2）当判断curBtnFlag中含有“上下”或“左右”标志位时，讲curBtnFlag的相应两位替换为lastBtnFlag的两位。设定lastBtnFlag=curBtnFlag;

3）目前允许的方向键组合方式有8种：“↑”、“↓”、“←”、“→”、“↑←”、“↓←”、“↓→”、“↑→”

4）StartBtnOpr中判断curBtnFlag是否存在上述两种状态之后，设定好curBtnFlag状态，判断是8种中的哪一种，进行相应的操作。

5)StartBtnOpr中每次运动指令的触发间隔要比每次运动的运行时间短，例如每50ms发送一次运动指令（发送当前flag状态对应的指令），运行时间设为80ms,这是为了保证运行的流畅，因为实际真的触发可能比50ms长，所以留些余量。

周二：

1. 整理“长按”“组合键”功能部分代码，上传代码。
2. 考虑属性的数据结构、数据库等逻辑。

修改“长按”逻辑：

1）每个状态至少执行50ms,才能执行下一个状态的执行。