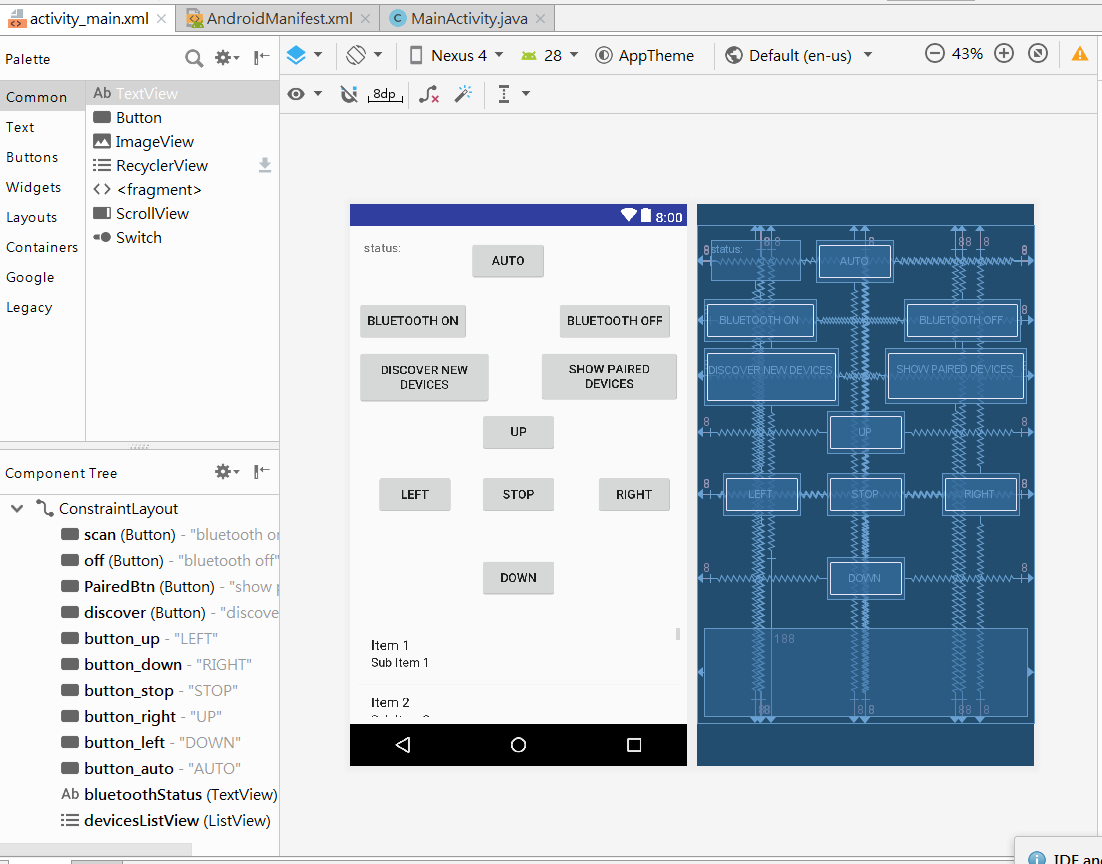
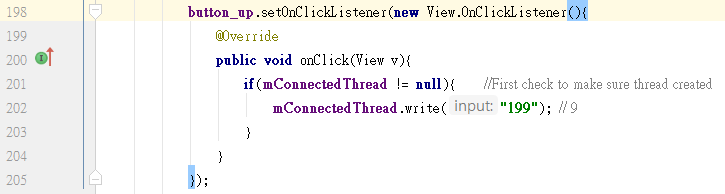
* 藍芽

在寫APP來控制我們的車子方面，我們使用了Android studio這個開發環境來編寫我們的APP，我在這邊簡單介紹一下我們藍芽訊號是怎麼從手機到達FPGA版上的。

在Android studio中我們定義了一些button，如下:



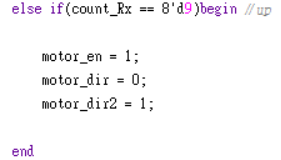
我以讓車子前進的 ”UP” 這個按鈕來舉例說明，當UP這個按鈕被按下時就會觸發下圖第200行的onClick這個function，此外，在傳送資料前我們得先檢查藍芽是否已經接上裝置，不然有可能會出現bug使得我們的APP在執行時shut down，如果藍芽沒有成功連接到裝置，那麼下圖第201行的mConnectedThread就會等於null，當確認完藍芽有正確連接到FPGA上的藍芽模組後，我們就用write這個function將傳送資料出去(當UP這個按鈕被按下時，我傳送字串”199” 到fpga版上的藍芽模組)，如下圖第202行所示:



至於我們為什麼要傳送199呢? 為什麼不是其他的字串? 其實我也不太清楚，199這個字串是trial & error所得到的答案，我有查過ascii code，不過也對不起來，而且很多字串都會傳送同一筆資料到FPGA上的藍芽模組，不過這並不影響我們的設計，因為我們只需要讓FPGA能夠分辨6種我所傳送出去的資料就足夠了(因為我們的APP有6個按鍵會送資料到FPGA版上，分別是AUTO、UP、DOWN、LEFT、RIGHT、STOP這6個按鈕)，trial & error 的結果如下表所示:

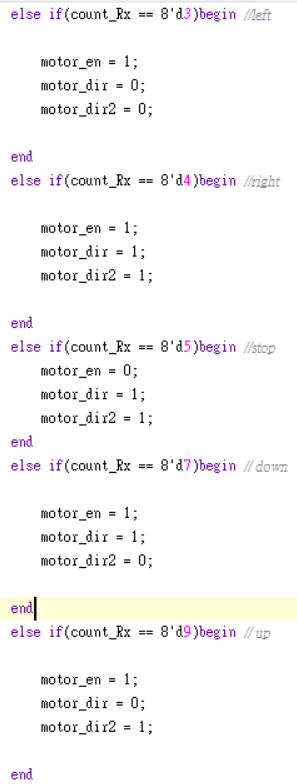
|  |  |
| --- | --- |
| APP傳送出去的資料(字串) | FPGA版上藍芽模組接收到的資料(整數) |
| 1、2、3、4、6、7、9、a | 3 |
| 5、m | 4 |
| 0、8 | 2 |
| 10、18、20、30、40 | 5 |
| 11、12、13、14、16、17、19、21、22、23、50 | 6 |
| 15、100 | 7 |
| 199 | 9 |
| fuck | 14 |

在verilog的TOP.v 中，我們利用count\_Rx這個變數來接收手機所傳送的資料，當我們在手機上按下UP這個按鈕時，我們從手機傳送字串”199”出去，根據上表，FPGA接收到的信號會是整數9，所以當count\_Rx這個變數等於9時，我們就設置一些控制馬達轉動方向的訊號，讓車子向前行進，如下圖所示(motor\_en為enable訊號，如果等於0馬達就不會轉動，motor\_dir是控制第一個馬達轉動的方向，motor\_dir2是控制第二個馬達轉動的方向，至於為什麼一個是0一個是1呢?，這個我在後面講到馬達時會多做說明)。

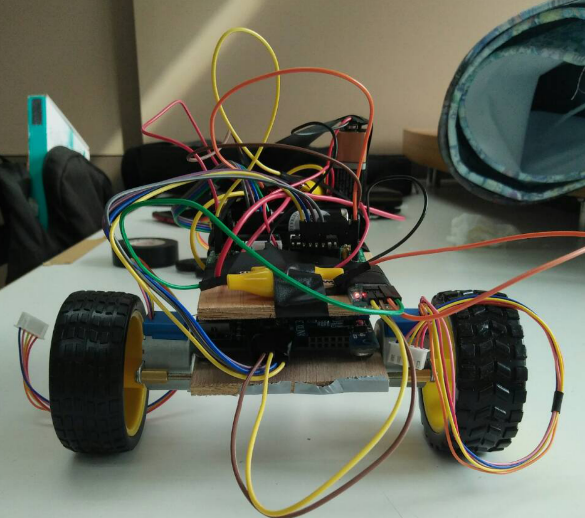


* 馬達:

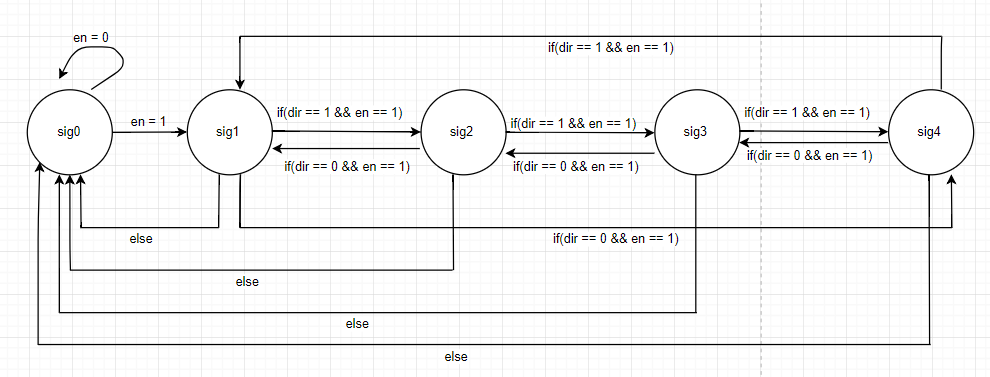
承上所述，在接收到藍芽不同的信號以後，我在verilog(TOP.v)會設置不一樣的訊號給馬達以控制它的轉動，如下圖所示:



motor\_en的運作原理相當簡單，等於1時馬達能動，等於0時馬達不能動，motor\_dir、mortor\_dir2則是分別用來控制第一個馬達與第二個馬達的轉動方向。我以前進為例(上圖最後一個else if，count\_Rx = 8’d9)，為什麼在前進時兩個馬達的旋轉方向不一樣呢? 其實這個原理也相當簡單，因為兩個輪子安裝在相反的方向上，他們是面對面的(如下圖)，所以要前進時，轉動方向要不同，轉彎時轉動方向相同。



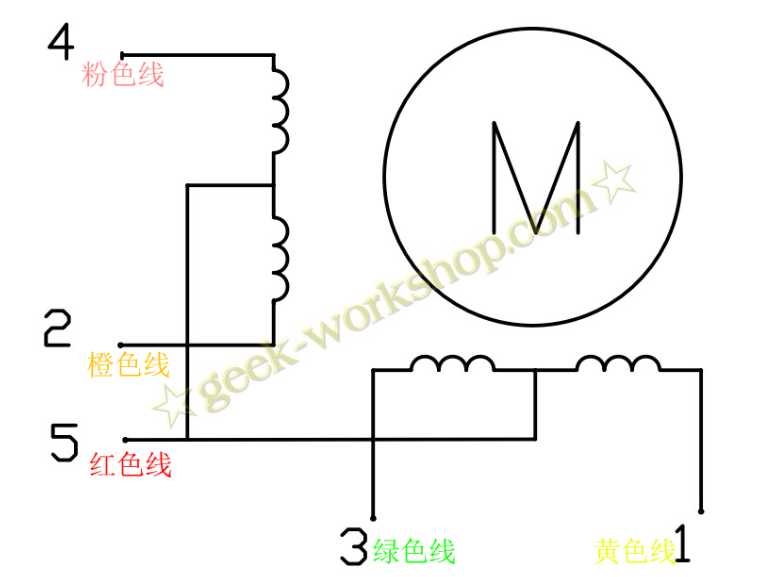
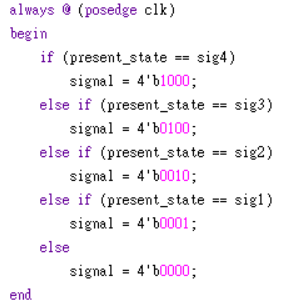
而TOP.v中motor\_en、motor\_dir、motor\_dir2這三個變數會傳送給pmod\_step\_interface這個module，然後這個module會把clk進行除頻後把新的clk以及motor\_en、motor\_dir、motor\_dir2這三個訊號傳給pmod\_step\_driver這個module，它是用來控制步進馬達旋轉的，其state transistion diagram如下:



上圖中sig0是當馬達停止時的state，也就是當en=0時都會回到sig0。

sig1 ~ sig4是控制左下圖中這四個磁鐵的state，如果dir == 1就是正轉，dir == 0則是反轉。

這個finite state machine是一個moore machine，會根據state給馬達訊號，如右下圖所示。

步進馬達結構圖