FINAL PROJECT

打地鼠

TEAM 8

108062108 柳廷儒

108062111 劉柔恣

Introduction

一次遊戲2分鐘，共有三隻地鼠，每擊中一隻地鼠加1分。

總共有三個關卡，在關卡一的時候，一次只會升起一隻地鼠；當分數達到20分時，進入關卡二，一次會升起兩隻地鼠；當分數達到50分時，進入關卡三，有機率一次升起三隻地鼠。

FPGA板的七段顯示器上會顯示當前的難度、分數。

LED燈的數量則表示剩餘的時間，遊戲開始時會亮12個燈，接著每10秒熄滅一個燈，當所有燈都暗掉時，表示這局遊戲結束。

當地鼠被擊中時，則會發出音效。

Motivation

在蒐集身旁朋友的意見之後，發現大家最想玩的遊戲是「打地鼠」，於是我們決定動手做出屬於自己的打地鼠，實現大家的願望！

System specification

材料：

FPGA板	1個
電磁鐵模組	3組
鐵尺	3把
多多瓶	3瓶
微控開關	3個
喇叭	1組
行動電源	1個

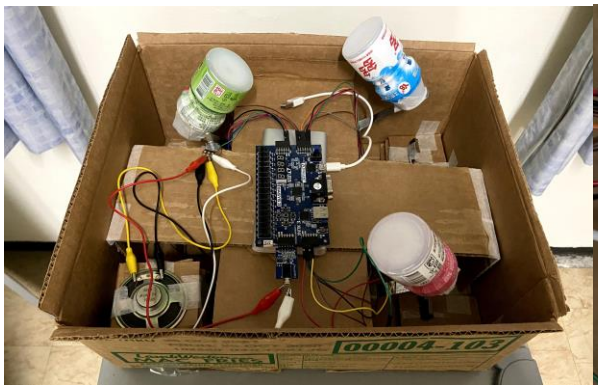
成品圖：

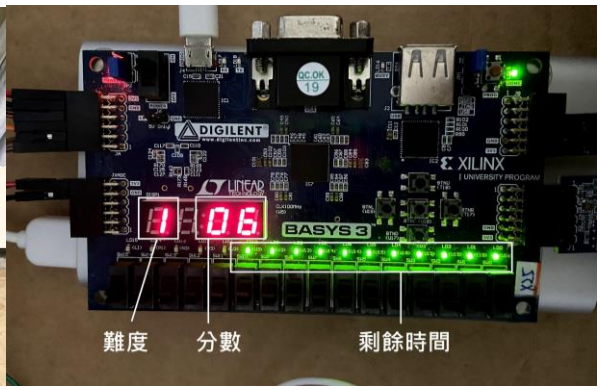
此台打地鼠機由三隻地鼠構成，分別位於三個角落。

FPGA板和行動電源置於中央。

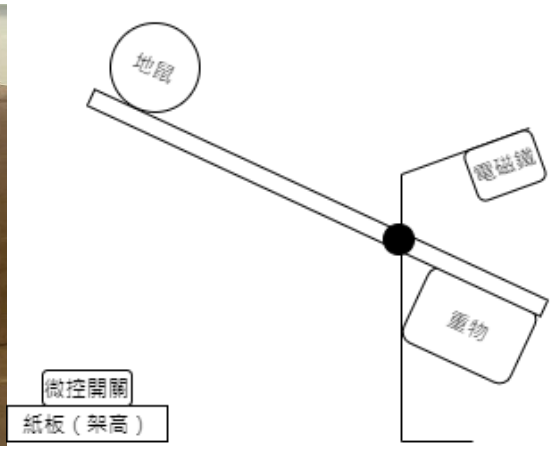
左下角的位置則用來放置喇叭，當地鼠被打到時，會發出擊中的音效。

FPGA板的七段顯示器上會顯示當前的難度、分數，而LED燈的數量表示剩餘的時間。

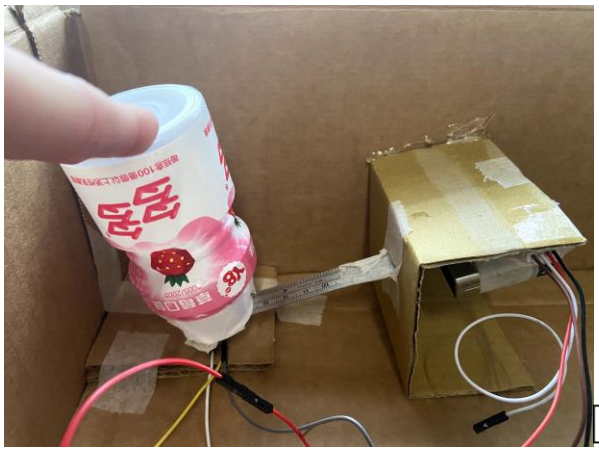


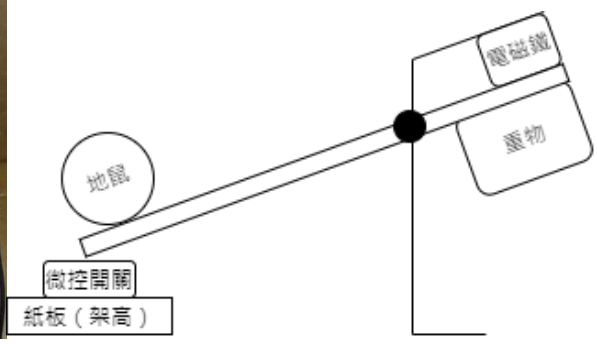


UP狀態，此時電磁鐵不通電、沒有磁力，重物端受重力下降，使地鼠端升至高處：



將地鼠打下去後，進入DOWN狀態，此時電磁鐵通電、產生磁力，將重物端吸住：



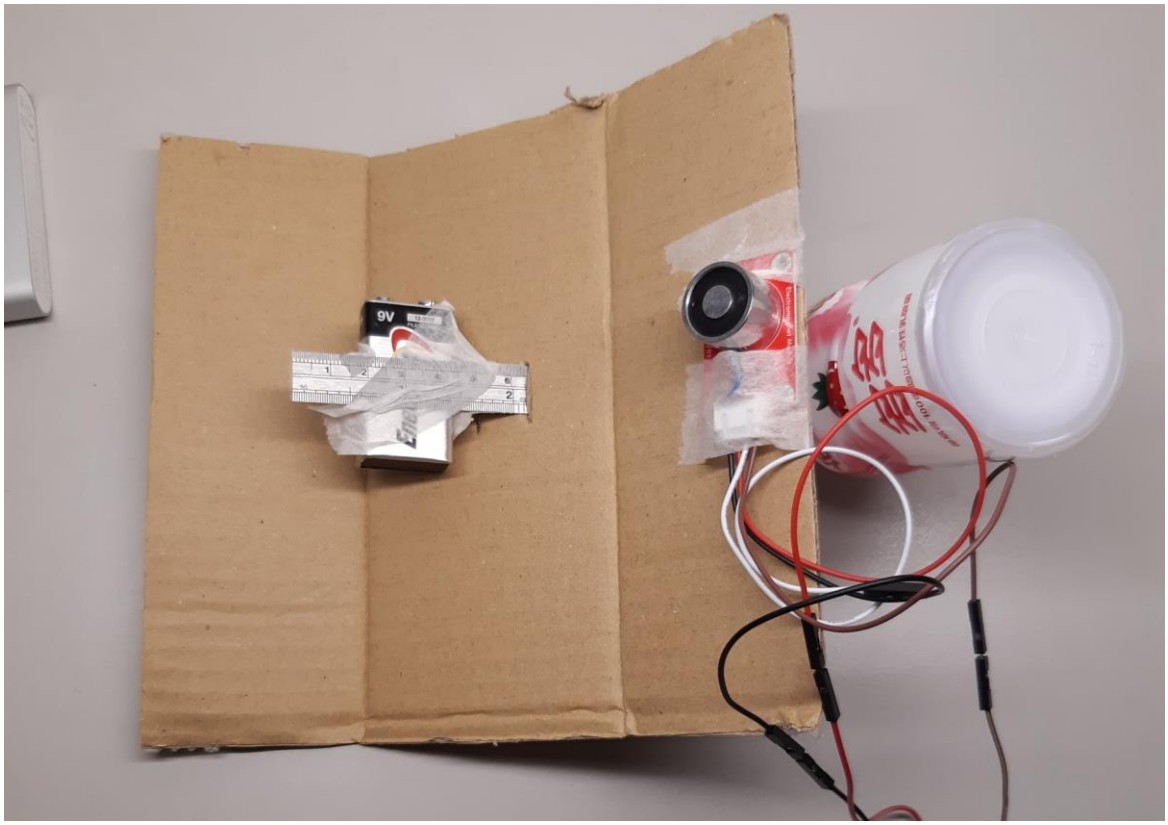


內部構造：

以紙板作為支架，電磁鐵黏在紙板上方，使用鐵尺作為槓桿。

外側端黏上多多瓶作為地鼠。

內側端黏上lab6使用完的電池作為重物。

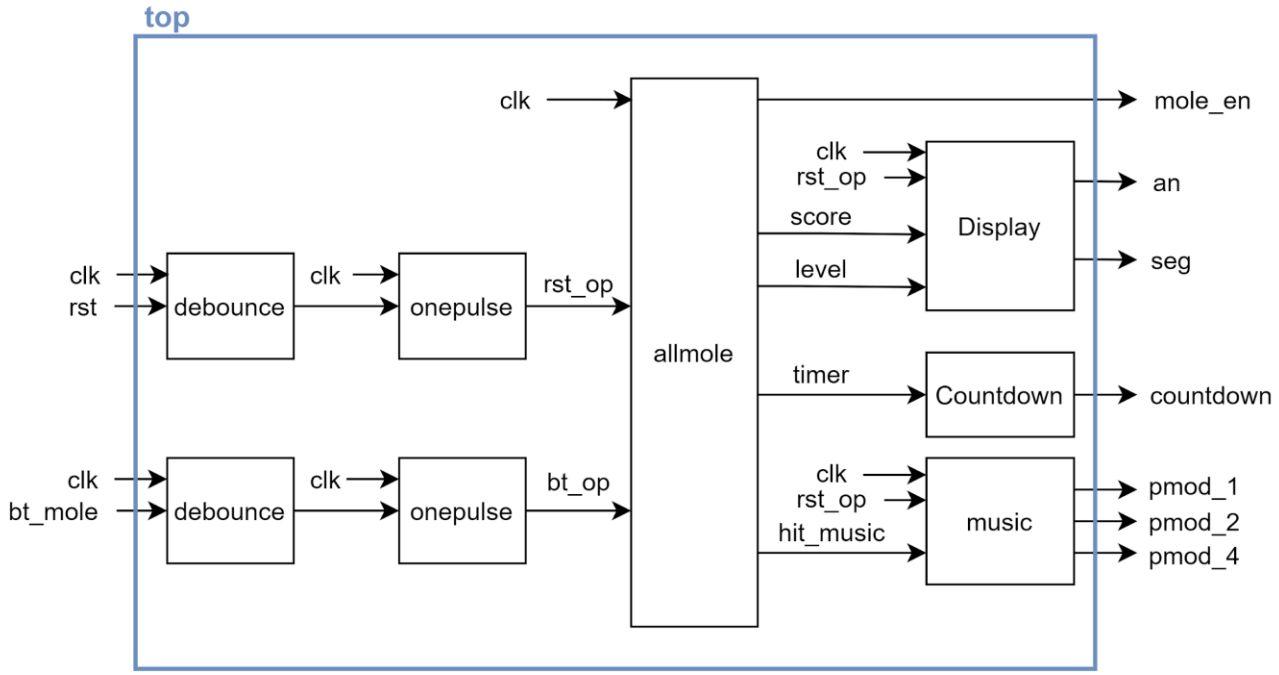


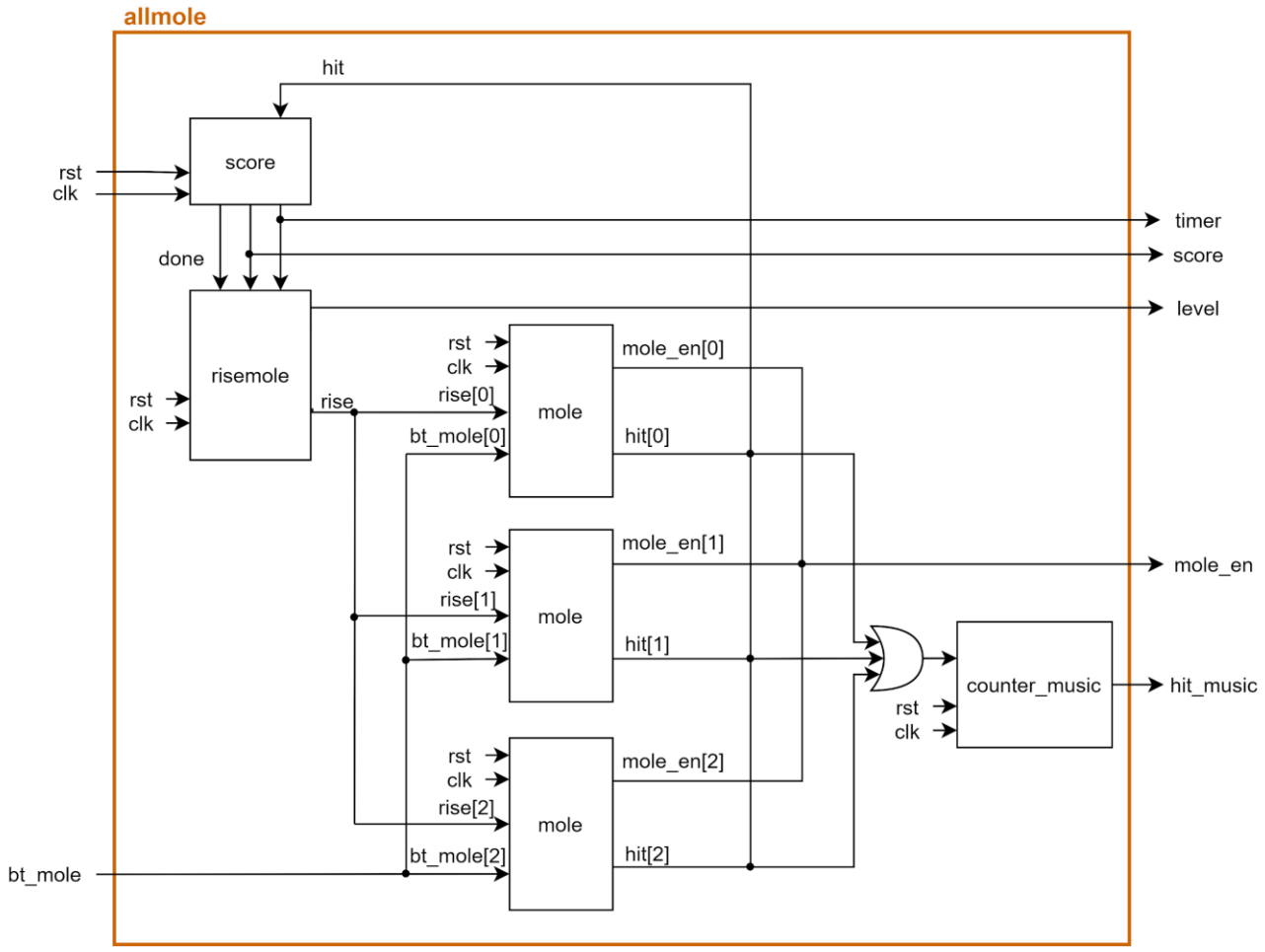
我們在此設計中，主要運用了電流磁效應、槓桿原理。

當地鼠在UP狀態時，將電磁鐵的enable設為0，讓電磁鐵失去磁力，此時鐵尺的重物端便會受到重力作用下降，使地鼠端升起。

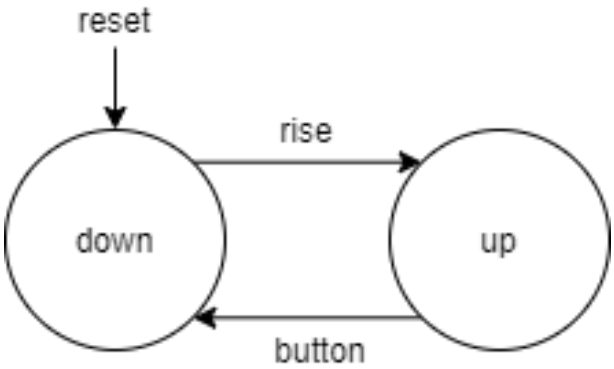
當地鼠被打下去後，會進入DOWN狀態，此時將電磁鐵的enable設為1，讓電磁鐵通電產生磁力，吸附住鐵尺，進而讓地鼠保持在下方。

Block diagram：





State transition diagram：



程式碼說明：

- top

包含處理button訊號的debounce和onepulse、處理全部地鼠的allmole、將分數與關卡顯示在七段顯示器的Display、將倒數時間顯示在LED燈上的Countdown、播放擊中音效的music。

- allmole

用來決定關卡、分數、時間、地鼠狀態等。

```
// mole
allmole allmole_0 (
    .rst (rst_op),
    .clk (clk),
    .bt_mole (bt_op),
    .score (score),
    .level (level),
    .mole_en (mole_en),
    .hit_music(hit_music),
    .timer(timer)
);
```

裡面包含mole、Score、risemole、counter_music，以下一一對這些module進行說明。

- mole

每個地鼠透過button和rise各自判斷目前的state是DOWN或UP，並輸出控制電磁鐵的enable和表示地鼠是否有被打到的hit。

```
mole mole0(
    .rst (rst),
    .clk (clk),
    .button (bt_mole[0]),
    .rise (rise[0]),
    .hit (hit[0]),
    .enable (mole_en[0])
);
```

```
mole mole1(
    .rst (rst),
    .clk (clk),
    .button (bt_mole[1]),
    .rise (rise[1]),
    .hit (hit[1]),
    .enable (mole_en[1])
);
```

```
mole mole2(
    .rst (rst),
    .clk (clk),
    .button (bt_mole[2]),
    .rise (rise[2]),
    .hit (hit[2]),
    .enable (mole_en[2])
);
```

DOWN：

enable為1，讓電磁鐵有磁力。

當rise為1時，表示地鼠要升起，因此next_state為UP，next_hit為0；

當rise為0時，表示地鼠維持下降狀態，因此next_state為DOWN，next_hit為0。

```
DOWN: begin
    enable = 1'b1;
    if(rise) begin
        next_state = UP;
        next_hit = 1'b0;
    end
    else begin
        next_state = DOWN;
        next_hit = 1'b0;
    end
end
```

UP：

enable為0，電磁鐵沒有磁力。

當button為1時，表示地鼠被打下去，因此next_state為DOWN，next_hit為1；

當button為0時，表示維持上升狀態，因此next_state為UP，next_hit為0。

```
UP: begin
    enable = 1'b0;
    if(button) begin
        next_state = DOWN;
        next_hit = 1'b1;
    end
    else begin
        next_state = UP;
        next_hit = 1'b0;
    end
end
```

- Score

產生timer訊號，並藉由hit算出score。

```
// decide score by hit
Score sc0(.rst(rst), .clk(clk), .hit(hit), .done(done), .score(score), .timer(timer));
```

利用counter_timer module產生用來計時的timer訊號。

counter_timer module會在start為1時，每兩秒將輸出的done設為1，因此把timer加上done，就可以讓timer的值每兩秒增加1。

```
// timer (add 1 every 2 sec)
counter_timer cnt_0 (.clk (clk), .rst (rst), .start (start), .done (done));
assign next_timer = timer + done;
```

一局遊戲為兩分鐘，而timer是每兩秒加一，因此當timer < 60時表示遊戲正在進行中，start設為1，score依照打到的地鼠數量加分；當timer >= 60表示遊戲結束，start設為0，score不變。

```
// start (2 min game)
// score (add 1 point when hit a mole)
always @(*) begin
    if(timer < 7'd60) begin
        start = 1'b1;
        next_score = score + hit[0] + hit[1] + hit[2];
    end
    else begin // finish
        start = 1'b0;
        next_score = score;
    end
end
```

- risemole

依照Score module輸出的timer和score，決定level和rise。

```
// decide level by score / decide which mole to rise by timer
risemole rm0 (.rst(rst), .clk(clk), .timer(timer), .done(done), .score(score), .level(level), .rise(rise));
```

首先，使用selectmole產生0~3中的一個數字，來決定要升起哪個地鼠。

selectmole的算法是timer/3加上(timer%5的平方)，再%4，經過這樣的運算後，selectmole會不規則出現0~3的數字，達到隨機升起地鼠的效果，且timer在0~59時，selectmole出現0~3的次數剛好相同，各為15次。

```
// select (decided by timer)
assign selectmole = (timer/3 + (timer%5) * (timer%5)) % 4;
```

下表列出timer為0~59時，selectmole的值：

timer	selectmole	timer	selectmole	timer	selectmole	timer	selectmole	timer	selectmole	timer	selectmole
0	0	10	3	20	2	30	2	40	1	50	0
1	1	11	0	21	0	31	3	41	2	51	2
2	0	12	0	22	3	32	2	42	2	52	1
3	2	13	1	23	0	33	0	43	3	53	2
4	1	14	0	24	0	34	3	44	2	54	2
5	1	15	1	25	0	35	3	45	3	55	2
6	3	16	2	26	1	36	1	46	0	56	3
7	2	17	1	27	1	37	0	47	3	57	3
8	3	18	3	28	2	38	1	48	1	58	0
9	3	19	2	29	1	39	1	49	0	59	3

接著，依照分數判定目前的關卡，並決定地鼠的升起狀況。

當分數小於20時為第一關，在done為1的情況下，一次升起一隻地鼠，當selectmole為0時，rise[0]為1；selectmole為1時，rise[1]為1；selectmole為2時，rise[2]為1；selectmole為3時，則不升起地鼠。

```
if(score < 10'd20) begin
    next_level = 2'd1;
    if(done) begin // every 2 sec
        case(selectmole)
            2'd0: next_rise = 3'b001;
            2'd1: next_rise = 3'b010;
            2'd2: next_rise = 3'b100;
            default: next_rise = 3'b000;
        endcase
    end
    else begin
        next_rise = 3'b000;
    end
end
```

當分數為20~50時為第二關，在done為1的情況，一次升起兩隻地鼠，當selectmole為0~2時，將兩個rise設為1；selectmole為3時，則不升起地鼠。

```
else if(score < 10'd50) begin
    next_level = 2'd2;
    if(done) begin
        case(selectmole)
            2'd0: next_rise = 3'b110;
            2'd1: next_rise = 3'b101;
            2'd2: next_rise = 3'b011;
            default: next_rise = 3'b000;
        endcase
    end
    else begin
        next_rise = 3'b000;
    end
end
```

當分數達到50時進入第三關，在done為1的情況下，一次升起兩隻或三隻地鼠，當selectmole為0~2時，將兩個rise設為1；selectmole為3時，三個rise都設為1。

```
else begin
    next_level = 2'd3;
    if(done) begin
        case(selectmole)
            2'd0: next_rise = 3'b110;
            2'd1: next_rise = 3'b101;
            2'd2: next_rise = 3'b011;
            default: next_rise = 3'b111;
        endcase
    end
    else begin
        next_rise = 3'b000;
    end
end
```

- counter_music

使用counter_music module，當有任何一隻地鼠被打到時，hit_music訊號會變為1，維持0.25秒。

```
// music (when hit, keep hit_music 0.25 sec)
counter_music cm0 (.clk(clk), .rst(rst), .start(hit), .done(hit_music));
```

使用|hit將hit的4個bits進行OR運算，接著將|hit接到counter_music的start訊號。當|hit為1時，代表有地鼠被打到，要發出被擊中的音效。

因為|hit只會維持一個clock，所以使用counter_music產生出時長為0.25秒的訊號。

當start訊號（也就是allmole中的|hit訊號）為1時，將next_count設為0並開始數、並將next_done設為1。

而當done為1時，則判斷是否已經數到0.25秒，如果還沒，則讓next_count繼續往上數，並將next_done設為1；若已到0.25秒，則將next_count和next_done都設為0。

```
always@(*) begin
    if (start) begin
        next_done = 1'b1;
        next_count = 30'b0;
    end
    else begin
        if (done) begin
            if(count < 30'd2500_0000) begin
                next_done = 1'b1;
                next_count = count + 30'b1;
            end
            else begin
                next_done = 1'b0;
                next_count = 30'b0;
            end
        end
        else begin
            next_done = 1'b0;
            next_count = 30'b0;
        end
    end
end
```

- music

藉由hit_music訊號產生pmod_1。

```
// music
music m0 (.clk(clk), .reset(rst_op), .tone(hit_music), .pmod_1(pmod_1), .pmod_2(pmod_2), .pmod_4(pmod_4));
```

將counter_music產生的done訊號（也就是allmole中的hit_music訊號）接到music的tone。

而music中的Decoder，會根據tone訊號輸出對應的freq訊號。當tone為0時，將頻率設為32'd20000，是人耳聽不見的頻率；當tone為0時，將頻率設為32'd262 << 2，是C6的音。

接著將freq訊號傳入PWM_gen中，產生出對應的pmod_1訊號。

```
always @(*) begin
    case (tone)
        1'b1: freq = 32'd262 << 2;  //C6
        default : freq = 32'd20000; //Do-dummy
    endcase
end
```

- Display

透過從allmole輸出的level和score，產生七段顯示器的an和seg。

```
// display (7 seg)
Display dis_0 (.rst(rst_op), .clk(clk), .score(score), .level(level), .an(an), .seg(seg));
```

將score的個位數字轉換為seg0，十位數字轉換為seg1，level轉換為seg3。

```
Num_To_Seg n2s_0 (.num (score%10), .seg (seg0));
Num_To_Seg n2s_1 (.num (score/10), .seg (seg1));
Num_To_Seg n2s_3 (.num (level), .seg (seg3));
```

將seg0、seg1顯示在最右兩位數、seg3顯示在最左邊，左邊第二位數不顯示數字。

```
always @(*) begin
    case(cnt)
        2'd0: begin
            next_an = 4'b1110;
            next_seg = seg0;
        end
        2'd1: begin
            next_an = 4'b1101;
            next_seg = seg1;
        end
        2'd2: begin
            next_an = 4'b1111;
            next_seg = seg2;
        end
        2'd3: begin
            next_an = 4'b0111;
            next_seg = seg3;
        end
    endcase
end
```

- Countdown

藉由allmole輸出的timer產生決定是否亮燈的countdown訊號。

```
//countdown (LED)
Countdown cntdn(.timer(timer), .countdown(countdown));
```

根據allmole中timer的算法為每兩秒加一，因此timer/5等於0時表示遊戲時間為0~9秒，12個LED燈全亮；timer/5等於1時表示遊戲時間為10~19秒，亮11個LED燈；timer/5等於2時表示遊戲時間為20~29秒，亮10個LED燈；以此類推，列出所有情況。

```
always @(*) begin
    case(timer/7'd5)
        7'd0: countdown = 12'b111111111111;
        7'd1: countdown = 12'b011111111111;
        7'd2: countdown = 12'b001111111111;
        7'd3: countdown = 12'b000111111111;
        7'd4: countdown = 12'b000011111111;
        7'd5: countdown = 12'b000001111111;
        7'd6: countdown = 12'b000000111111;
        7'd7: countdown = 12'b000000011111;
        7'd8: countdown = 12'b000000001111;
        7'd9: countdown = 12'b000000000111;
        7'd10: countdown = 12'b000000000011;
        7'd11: countdown = 12'b000000000001;
        default: countdown = 12'b000000000000;
    endcase
end
```

Problem solving

1. 鐵條的選擇

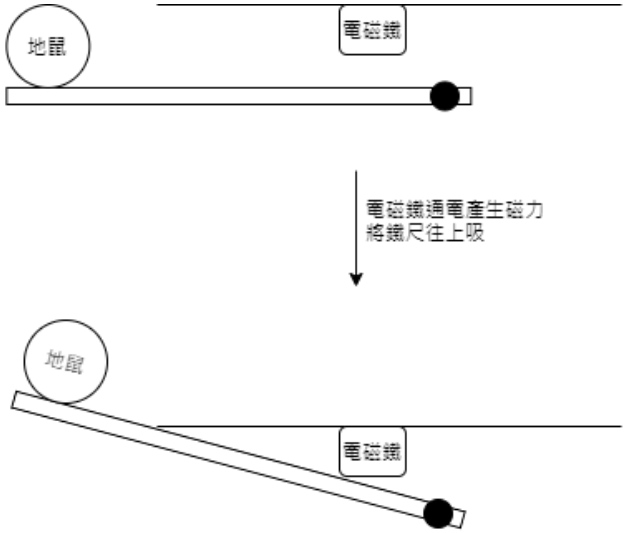
起初，我們去五金行，向老闆詢問是否有能讓磁鐵吸附的鐵條，然後買到了白鐵的鐵條。

但實際上操作時，卻發現買回來的鐵條無法被磁鐵吸引，上網搜尋，發現不是所有的白鐵都能被磁鐵吸住，只有型號是4開頭的白鐵才可以。

接著，我們靈光一閃，想到鉛筆盒中的鋼尺，嘗試後發現：鋼尺重量輕、可輕微彎曲、價格便宜，是非常理想的材料，於是決定使用鋼尺作為放置地鼠的槓桿。

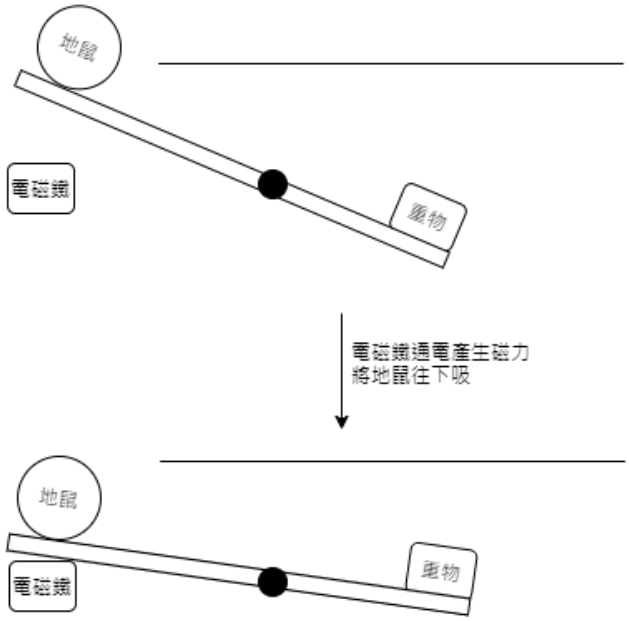
2. 如何運用電磁鐵

我們的第一種設計是將電磁鐵黏在上方，在UP狀態時，讓電磁鐵通電產生磁力，將鐵尺吸上去，進而讓地鼠升起：



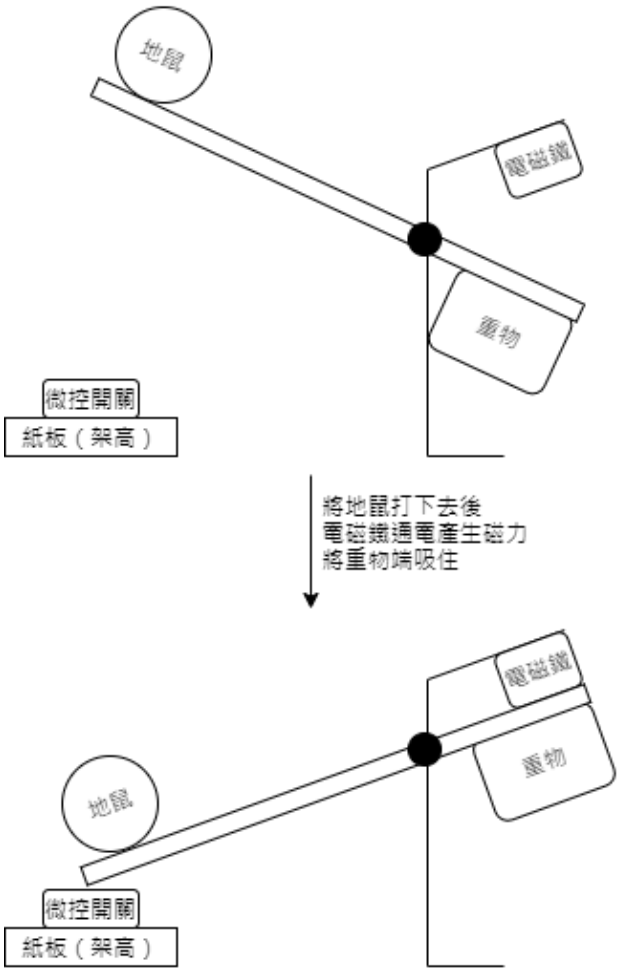
但後來發現，電磁鐵產生的磁場範圍不夠大，電磁鐵和鐵尺的距離太遠了，完全沒有反應，於是這個設計便宣告為不可行。

之後想到的第二種設計是將電磁鐵放在下方，把地鼠打下去後，進入DOWN狀態，讓電磁鐵通電產生磁力，把地鼠吸在下方：



但後來發現，電磁鐵和鐵尺之間不能有斜角，一定要平行，才吸得住，於是這個設計也宣告為不可行。

第三種設計，也就是我們現在所使用的方法，則是將電磁鐵貼在紙板上方，把地鼠打下去後，進入DOWN狀態，讓電磁鐵通電產生磁力，吸附住鐵尺，進而讓地鼠保持在下方。



3. 槓桿原理的應用

完成上述的設計後，在DOWN狀態時，地鼠能順利保持在下方了，但是在UP狀態時，地鼠卻沒有升得很高。

我們嘗試過增加另一端的重量、以及改變支點的位置，而最終的版本是將一顆9V電池貼在鐵尺的尾端、支點設在距離其6公分的位置。

4. selectmole的運算

為了要有隨機升起地鼠的效果，我們用timer做一些運算後%4得出的selectmole來決定要升起哪隻地鼠，至於要做怎樣的運算我們嘗試了許多方法後才找到適合的。最先

想到的是將timer加或乘上一個數，但這個想法很快就被否決了，因為這樣做後仍然會照順序出現，不會有隨機的效果。接著又想到平方，但發現這樣只會有兩個數輪流出現而已。之後又試了幾種方法，像是timer的三次方、十位數字加個位數字等等，都覺得不太可行。後來想到可以把timer除以一個數再和timer%另一個數來做運算，而最後試出最滿意的算法是timer%5的平方加timer/3。

5. rise的時機

一開始在risemole module中設定rise訊號時，我們沒有加上if (done)的條件，測試後發現地鼠被打中後，不會變為DOWN state，後來才知道沒有加上if (done)的條件，rise訊號可能一直為1，因此到了DOWN state後，又會馬上變回UP state，而如果在done為1時，將rise設為1，rise就會在每過兩秒時才可能為1，其餘時間都為0，就不會有一開始的錯誤發生。

6. 音效

一開始我們直接將hit傳入music，卻沒有發出聲音，以為是喇叭沒接好。後來使用之前的sample code測試後，發現喇叭沒有問題。思考之後，發現是因為hit的長度太短了，只有一個clock，沒辦法發出一個完整的音，所以後來又加了counter_music，來產生長度為0.25的訊號。

Experimental results

實作出三個地鼠、三個關卡、顯示分數與剩餘時間、擊中音效。

原本計畫做BGM，但因為一個喇叭只能播出擊中音效和BGM其中一個，而我們後來選擇了做擊中音效。

另外，本來還有想讓地鼠一段時間沒打的話會自動降下去，也有寫出了這部分的code，但後來開始硬體組裝後才發現電磁鐵的使用與我們預想的不同（詳見Problem solving第二點），換了方案後，便很難運用電磁鐵做出這個效果，因此放棄了這個部分。

Conclusion

這次final project的過程中，大部分的時間都花在硬體上，主要是因為電磁鐵的使用方法和我們原先預想的非常不同，詳細過程請見Problem solving的第二點，這也讓我們對於磁力的運用有更深的掌握。而終於定下可行的設計後，製作時則要細心謹慎，如組裝時位置要準確、要黏得牢固等等。另外，因為一個電磁鐵有三個port要接、一個微控開關也是三個port，所以三隻地鼠總共有18個port、18條線，必須注意不要接錯、不要鬆脫、不能阻擋到地鼠的升降等等，讓我們在處理硬體的部份更有經驗。

而在程式碼的部分，因為這次的輸出有電磁鐵、七段顯示器、LED燈、音效，所以我們把不同功能分別寫在許多不同的module，這時也有許多小細節需要注意，像是訊號是否有連到正確的地方、同個訊號在不同module間bit的數量是否相同等等，若能注意到這些地方，便能節省許多debug的時間。