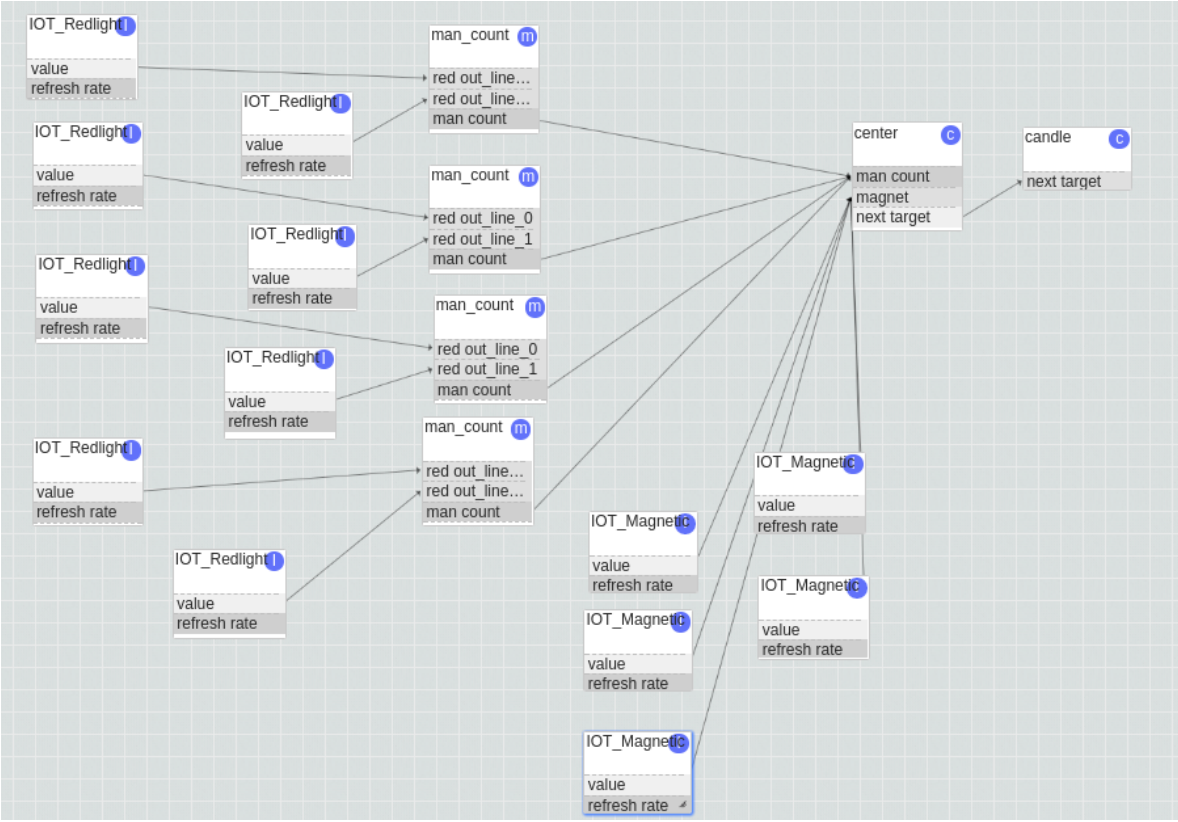
智慧室內盲人引導系統

第三組 b03902109 賴昱憲 b03902060江昶翰

1.系統架構圖：



IOT_Redlight：紅外線測距器，當有物體擋住使電壓值改變時回傳1其他時候則回傳0

man_count:利用兩個紅外線測距氣所得到的值來判斷是否有人通過、通過的人是往哪邊走，並回傳至center。

IOT_Magnetic:每個磁力感應器在執行程式時都會使用argv賦予它編號，若盲人經過此則回傳他自己的編號，若沒有則回傳0。

center:利用man_count回傳的值來計算各個區域的人數，將人離開的區域人數減一，將人前往的區域人數加一。同時利用磁力感應器回傳的盲人位置與自己算出各個區域的人數來為盲人計算較短且人數較少之路徑。

Candle:接收center回傳的路徑並利用陀螺儀所得到的面向來計算接下來要往哪個方向前進，利用震動馬達來提示方向。

2.系統硬體：

磁力感應器：DBRobot DFR0033，理論上每個區域需要依照實際空間配置足夠多個磁力感應器以用來偵測盲人位置，而在demo中每個區域只放置一組，共四組。

紅外線測距：Sharp GP2D12，每個節點配置兩台，依照實際空間配置而定，在demo中我們將空間分為四個區域，所以總共有8台sensor。

振動馬達：Grove - Vibration Motor，手仗上配置四個以用來指示盲人四個方向，而在demo中我們則是使用led燈來表示方向。

六軸電子陀螺儀：MPU6050，裝置於手仗上以得到盲人面向哪裡，由於需計算三軸角速度的積分並利用三軸加速度來矯正積分的偏差值，我們在最後demo中實現。

Raspberry pi 3B: 依照空間大小配置足夠個Raspberry pi 3，在demo中配置了四台

3.資料傳輸方式

我們每一個class所應該要收到的input跟output都是由wifi來做傳輸，因為考慮到我們的scale可能都在室內，而現今的公共建設都會設有wifi，所以將所有的device根據wifi傳輸是沒問題的。

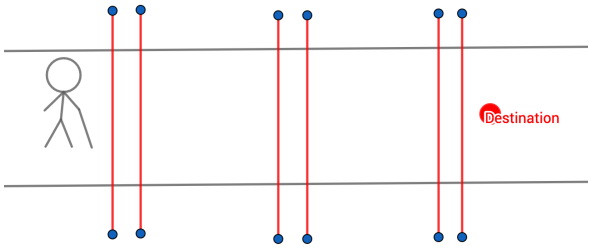
傳輸的資料量不大，唯一可能會有限制的地方在於，因為我們的紅外線感測的class的fresh rate為0.1秒，所以如果有一個人經過同一組（兩個）紅外線感測的時間間隔在0.1秒之內，就有可能出錯。但是這個問題可以經由加大兩個紅外線感測之間的距離而得到改善。經由我們測試，我們把兩個紅外線感測的距離相距5~10cm，即便手很快地畫過都是偵測得到的。

4.演算法

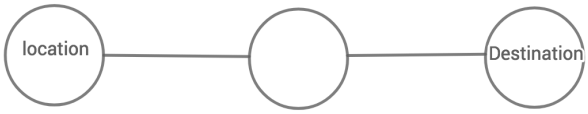
在這個架構中重要的演算法有兩個，分別是man_count與center，下面將會簡單說明。

man_count:這個wuclass會接收A、B兩台紅外線測距器得回傳值，由於A、B兩台感測器有間隔一小段距離，所以當有人經過時不會同時回傳1，而是先後回傳1，利用這項特性來判斷人往哪裡走，當人其中也有很多複雜的判斷式來處理走到一半停下來與折返的情況。

Center:將每組(兩個)紅外線測距器視為一個節點，而節點與節點之間邊上的權重則是節點與節點之間的人數，如圖一圖二所示。當然權重不只考慮人數，也加上實際上的距離，以避免少量人潮導致盲人繞超大一圈遠路。



圖一



圖二

化為圖形的方法後就可以用圖論的方法算出盲人到目的地的最短(最少人數)距離，此圖論的演算法的概念大概為，將所有節點分為兩類，加入最短路徑樹與沒有加入最短路徑樹兩種，一開始只有start，即盲人所在的節點有加入最短路徑樹，之後重複下列步驟：

1.找出所有與最短路徑樹上節點 i 相鄰的點 j ，找出 j 使 start 到 i 的距離加上 i 到 j 的距離是最小值

2.將 j 加入最短路徑樹，若 j 是目的地則停止迴圈

末頁有附pseudo code

5.系統延展性

由於我們是把所有的device都加在我們的FBP裡面，而根據我們的設計，只要建築物內有岔路的地方都得安裝一組紅外線感測，加上man_count 跟磁力感測的class，一個點就得加四個class。所以路線太負責可能在拉FBP的時候就會有點辛苦。

舉例：

1.台大教務處一樓：如果把它放大到台大教務處一樓那個回字形的走廊上是ok的，因為它路線簡單只要在四個轉角設置，我們的iot設備就有辦法work。

2.資工系館：如果我們要指引盲人經由樓梯或是電梯上下樓，這個叉路點有點太多，要拉其實也是可以，但是就會很花時間，成本很高。

附錄.

```
int d[] //起點到點d的最短距離
int parent[] //以start為根的最短路徑樹父親是誰
int visit[] = 0 //判斷此點是否已經加入最短路徑樹，一開始設為0
int w[i][j] //兩相鄰節點i j 的距離
min_path(int start,int destination)
{
    d[start] = 0;//設定起點的最短路徑長度
    parent[start] = start;//設定起點是根
    visit[start] = true; //將起點加入到最短路徑樹
    for (int k=0; k<node-1; k++){
        int a = -1, b = -1, min = inf;
        for (int i=0; i<node; i++)
            if (visit[i])
                for (int j=0; j<node; j++)
                    if (!visit[j])
                        if (d[i] + w[i][j] < min){
                            a = i;
                            b = j;
                            min = d[i] + w[i][j];
                        }
        if (a == -1 || b == -1) break;
        d[b] = min;
        parent[b] = a;
        visit[b] = true;
        if b == destination
            break;
    }
}
```