Librerías:

comunicacion.h:

- void EsperarInformacion(): Bucle infinito, se rompe cuando llega informacion de la rasp
- void SegmentarInformacion(): Divide la informacion recibida en 4 variables:
    - DireccionX: Primer bit
    - NivelesX: bits 2 y 3
    - DireccionY: bit 4
    - NivelesY: bits 5 y 6
    - Se asume que una dirección de 1 equivale a derecha en X y abajo en Y, mientras que 0 es izquierda en X y arriba en Y, **aunque esto podría no ser correcto**, dependerá de cómo se ajusten los motores en la estructura física.
    - Hay un "problema" importante, las variables en las que se separa el dato son necesarias en el main, pero al ser multiples datos no basta un return, hay varias formas de solucionarlo
        - Crear una header (entiendo que es lo mismo que una librería) que simplemente almacena variables globales y usarlo tanto en esta libería como en el main
        - Meter los datos separados en una tupla, lista, array (aún no sé muy bien) y hacer el return con esto
        - Lo más sencillo a mi parecer, meter esta función en el main (ya que las variables están declaradas ahí), pero la verdad va a hacer que se va a ver un poco feo.
- Bool RecibirModo(): **Devuelve 1 o 0 que viene de la rasp**, en el main se interpreta como: 1 = Reacomodo, 0 = Patrón
- void EnviarColor(): Se presume que enviará al rasp el color leído actualmente, no sé si Isaac pensaba hacer que se envíaran todos los colores a la vez (cosa que solo serviría para el modo reacomodo), yo pensaría que es mejor solo enviar el color que se tiene al frente ya que así la función es útil tanto para reacomodo como para patrón.
- void EnviarAlerta(): Las funciones Enviar son distintas para no mezclar formatos de bits al comunicarse con la rasp (color podría requerir solo 3 bits, mientras que alerta 2, por ejemplo).

sensores.h:

- void setup_sensores(): Nada especial
- void leerColores(unsigned int *rgb): No sé cómo funciona, pregúntenle a Isaac
- bool alarmaDistancia(): Antes se llamaba leerDistancia(), lo cambié ya que para lo único que ocupamos el ultrasonico es para la alarma, solo nos interesa si algo estorba o no.
    - Tiene una subfuncion que da la distancia, no sé como manejará el profe los obstáculos, pero sería necesario configurar un poco más el condicional.
    - Devuelve 1 si detectó un obstáculo y 0 si no
    - Parte de que los obstáculos ocuparán el mismo espacio que los elementos pero con una altura distinta (esperemos que una menor para no preocuparnos de que peguen con el pistón)

actuadores.h:

Al inicio se declaran un montón de variables, estas son para que los valores que setean a los motores, imán y pistón trabajen de manera global en la propia librería, de forma tal que las funcionces de seteo alteran estos valores y así se pueden usar fácil.

- void ConfigurarMotores(): A esto me refiero, nótese que lo único que hace es definir los variables del incio con los valores de los parámetros de entrada, los cuales son:
    - int PinMotorX: Obvio
    - int PinMotorY: Obvio
    - int PinDireccion: Obvio
    - float RelacionRevTras: Una revolución de motor = cuantos centímetros de traslación de la grúa?
    - int PulsosRev: Configuración de switches de los stepper, cuando lo probé estába a 6400
    - int Velocidad: No es un valor literal de m/cm, es el valor de un delayMicroseconds que hace funcionar a los stepper, mientras más pequeño el número, más velocidad, para una confguración de 6400 pulsos/rev, 50 está bien
    - int DistanciaEntreEspacios: Distancia en cm entre las casillas de elementos
- void CalibrarCero(int CalibrarX, int CalibrarY): Mueve los motores a la izquierda y arriba hasta que topen con los switches de límite mecánico en X y Y, cuyos pines estarán en CalibrarX y CalibrarY.
- void MoverMotor (int Motor, int Nivel, bool Direccion): Mueve el motor.
    - Motor: 1 si es el X, 0 si es el Y
    - Nivel: Cuantas casillas se moverá
    - Dirección: 0 si es izquierda o arriba, 1 si es derecha o abajo (recordemos que esto podría no cumplirse)
    - Esta misma función hace la mate de cuantos pulsos se necesitan para lograr las casillas deseadas, son puras reglas de 3 con los valores que se ven.
- void EscaneoGeneral():
    1. Baja el piston
    2. Revisa si es obstáculo
    3. Lee color
    4. Lo envía a la rasp
    5. Se mueve a la derecha
    6. Repite de 2 a 5 hasta pasar por todas las columnas (4 veces)
    7. Se mueve hasta la primera columna
    8. Baja una fila
    9. Repite de 2 a 8 hasta pasar por todas las filas (4)
- Los demás son bastante sencillos, estoy seguro de que se entienden.

Main:

El reacomodo tiene la forma de:

1. Espero que me digan donde ir
2. Me llega el dato
3. Lo descompongo en cantidad de casillas y en que direccion
4. Me muevo
5. Espero
6. Bajo el piston
7. Agarro
8. Espero
9. Subo el piston
10. Espero que me digan donde ir
11. Descompongo
12. Voy
13. Espero
14. Bajo el piston
15. Suelto
16. Subo
17. Repito

Las esperas son simplemente para que le de chance a la parte mecánica de quedarse quieta, agarrar bien, etc.

El patrón tiene forma de

1. Voy al almacen
2. Leo color
3. Envío
4. Espero que me digan donde ir
5. Me llega el dato
6. Descompongo
7. Me muevo
8. Espero
9. Bajo el piston
10. Suelto
11. Vuelvo al almacen
12. Repito

“Vuelvo al almacen” implica igual que debo esperar informacion y descomponerla ya que el arduino nunca sabe donde está, así que debe esperar que la rasp le diga