Instituto Tecnológico de Estudios Superiores Monterrey
Campus Monterrey

Ana Lucero Serna Torres. – A01034726
David Kutugata Gastélum. - A01138865
Roberto De la Garza Alvídrez. – A00812611

Proyecto de Robótica: Vehículo Autónomo.
Profesor: Ing. Juan Hinojosa M. Olivares.

Fecha: 5 de mayo de 2016.

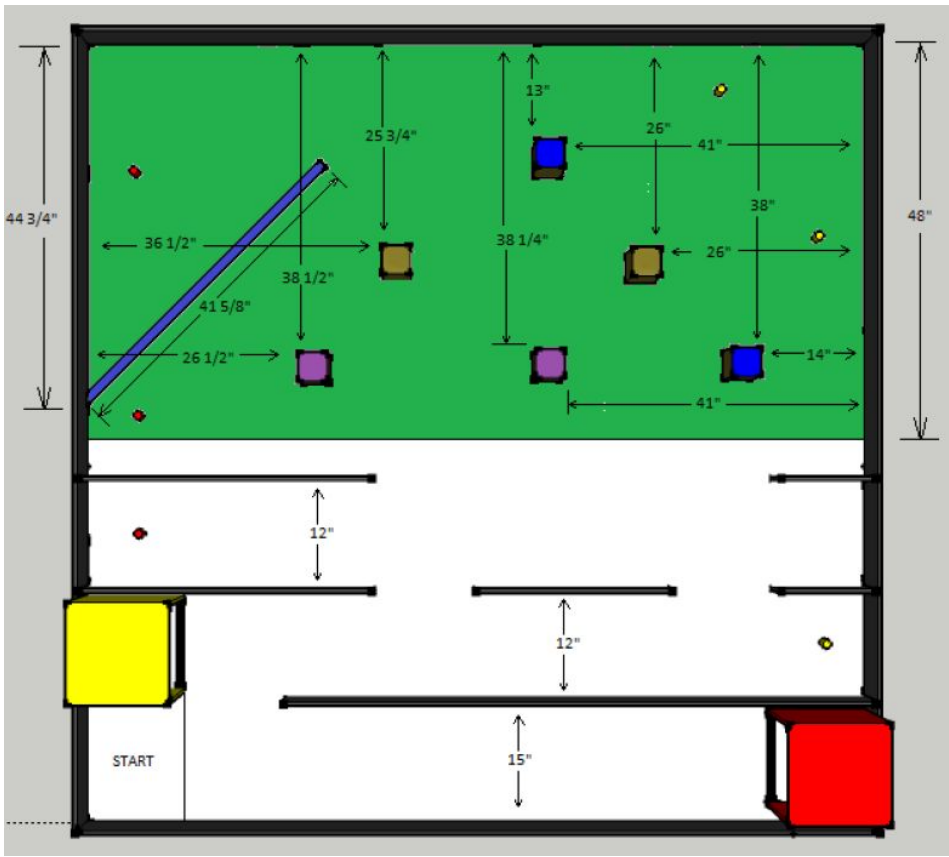
Índice

Introducción

El objetivo del proyecto es diseñar y construir un robot autónomo que busque e identifique víctimas dentro de dos áreas: “ciudad” y “campo”. Las víctimas deben ser transportadas al hospital correspondiente dentro de la ciudad dependiendo del color de dicha víctima dentro de un tiempo límite de 6 minutos. Los dos hospitales, de color rojo y amarillo, ambos están ubicados en la parte sur de la ciudad y están representados por dos cajas de su respectivo color. Las víctimas son representadas por un cilindro de madera de los mismos colores mencionados anteriormente.

Las posiciones de las víctimas son conocidas, hay dos fijas en la ciudad y cuatro variables dentro de la zona rural. La ciudad se representa con calles simuladas con paredes altas y piso de madera. Por otro lado, el campo está representado con pasto sintético, donde se ubican obstáculos azules de madera prismáticos rectangulares grandes y un “río” simulado con una pieza larga, delgada y baja de altura del mismo material y color.

El diseño del robot fue basado en todos los elementos descritos anteriormente para ser ligero y pequeño para maniobrar fácilmente entre las calles de la ciudad y el terreno del campo, además de contar sensores adecuados para la detection de paredes, víctimas y límites del terreno y actuadores para la movilidad y maniobras de recolecta de víctimas, todo funcionando, de manera complementaria y sincronizada, con software embebido en los controladores digitales.



Tablero de la competencia

Propuestas Individuales

Ana

DISEÑO

Se busca tener una base sólida con 4 llantas, con las baterías en medio y una placa para separar eso de lo demás. Las llantas se busca que sean como las que aparecen en la figura 1 y 2.

Ejemplos parecidos:

Figura: 1

Figura: 2

Arriba de la base, se busca colocar la tarjeta microcontroladora, tres sensores ultrasonicos, el gripper y una webcam.

Opciones de tarjeta microcontroladora: CT6811
Características: http://www.iearobotics.com/proyectos/ct6811/ct6811.html

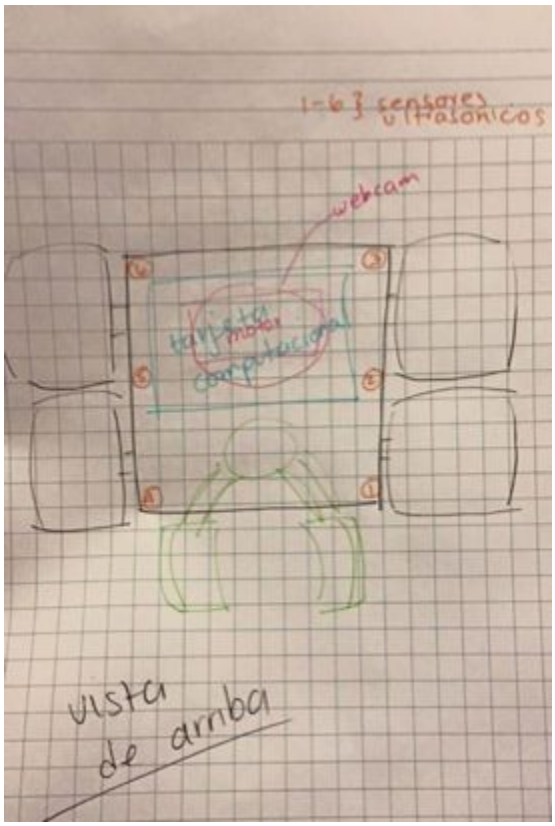
Sensores ultrasónicos: HC-SR04

Gripper:

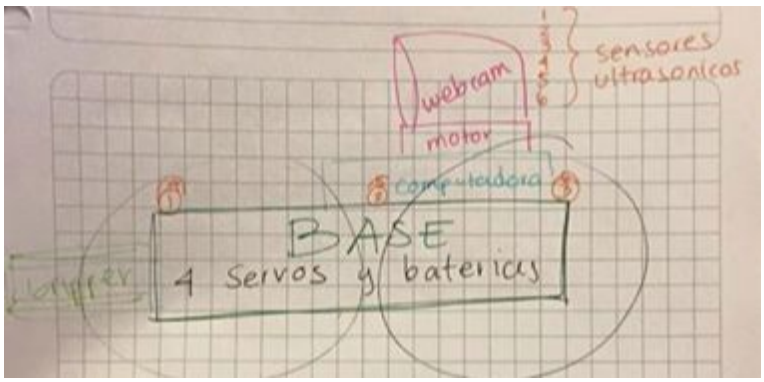
Es necesario agregar un motor que mueva el gripper verticalmente varios centímetros para alegar a la víctima del suelo.

Diseños generales

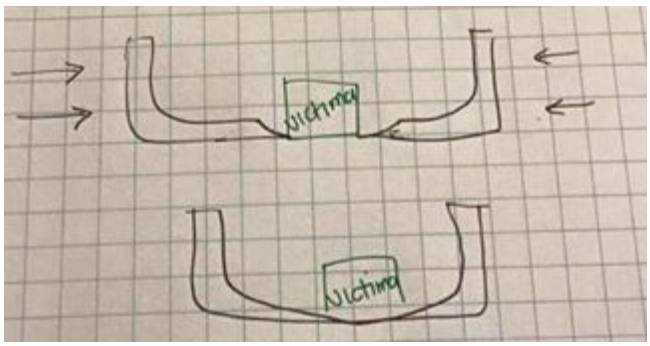
Vista de arriba



Vista lateral



Gripper



El gripper es una pequeña canasta que abre y cierra para recoger y soltar a la victima. La víctima se debe colocar en medio de la canasta para que esta pueda levantarla.

Lista de componentes:

· 4 llantas

- Base solida
- Dentro de esta se encuentra:
  - 4 servo motores para las llantas
  - Baterías
- 6 sensores ultrasónicos: estos son colocados en los lados, adelante y atrás con el fin de permitir que el robot maneje derecho.
- Computadora para visión
- Webcam: Esta tiene el fin de recordar el recorrido que se ha hecho y buscar nuevas rutas para encontrar a las víctimas, así mismo de evitar colisiones.
- Gripper: este está hecho de una canasta de plástico conectada a un motor que hace que estás cierren o abran
- Un motor rotacional para que la webcam tenga visión de 180º

Algoritmo:

Loop
Revisar si hay una victima con la webcam, si sí entrar a la función de recoger víctima
Si no,
Moverse 10 centímetros hacia una dirección viable y no recorrida con anterioridad (con prioridad hacia delante, si no, se busca otra ruta) Siempre se recuerda que ruta se tomó

Función recoger victima
Se avanza hacia ella evitando colisiones con paredes o obstáculos.
Colocarse para que el gripper pueda recoger a la víctima con ayuda de los sensores delanteros
Activar gripper
Función buscar hospital

Función Buscar hospital
Si no se tiene el área del hospital indicado en los datos, se busca por medio de
seguimiento de la pared derecha hasta llegar al área de la ciudad, si aún no se encuentra,
regresar hasta encontrarlo, guardar la ubicación en los datos y desactivar gripper

David

Material

Dispositivo	Cantidad	Uso
Cámara web	1	Visión
Pilas recargables de litio	6	Energía
Microcontroladores Raspberry B o B+	2	Procesamiento
Motores de DC de torque (6V)	2	Movimiento
Llantas	2	
Llanta loca	1	
L298D	2	
Sensores ultrasónicos	5	Navegación
Pistón	1	Sistema de agarre
Servo motor (6V)	1	
Limit switch	2	
Piezas impresas en 3D	X	
Tabla de madera	1	Acomodo

Descripción

Visión

En la parte delantera del robot se encuentra la cámara encima de la tabla de madera. Lo único que se espera sacar de la cámara es si se ve una víctima y de qué color es. Entre el sistema de navegación y agarre se sabrá si se tiene a la víctima a salvo.

Energía

Para energizar los sensores, motores y microcontroladores, se usarán baterías de litio recargables. Es necesario que den al menos un amperio de corriente. Se consideran 6 para usar una para cada microcontrolador (2), una para el pistón (1) y una para cada motor (3).

Procesamiento

Para el procesamiento de los datos se consideran dos microcontroladores Raspberry B. Uno entero será usado para visión y el otro para movimiento, navegación y el sistema de agarre. Ambos microcontroladores se comunicarán por USB.

Movimiento

El robot cuenta con tres llantas, dos en la parte de adelante y una en la parte de atrás. La de atrás es una llanta loca que solo sirve para equilibrio y solo sigue a las otras dos. Las otras dos se encuentran en la parte de adelante y son controladas cada una con un motor de DC de torque de 6V. Deben tener la mayor fricción posible para poder manejar fuera de la ciudad. Los motores serán controlados con ayuda del chip L298D (puente H) para controlar dirección.

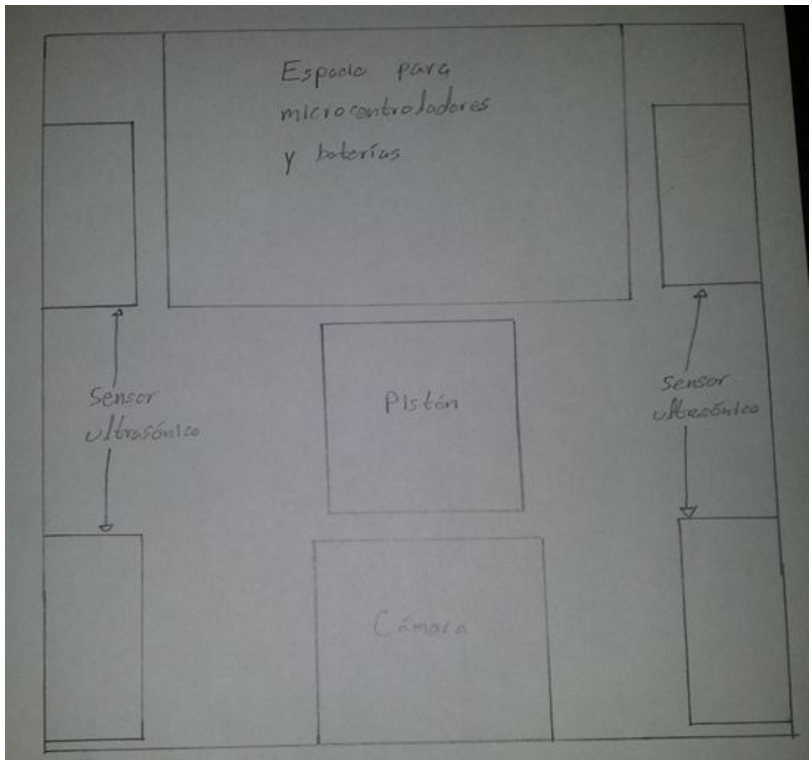
Navegación

La navegación se realizará con los 5 sensores ultrasónicos. Habrá dos a cada lado del robot y uno en frente. Los de los lados van a verificar que el robot va caminando derecho por las calles o pasillos de la ciudad, así como las orillas de la pista en el monte. El de adelante se usará para evitar chocar de frente y también para ver si se está bien alineado con la víctima para poder recogerla.

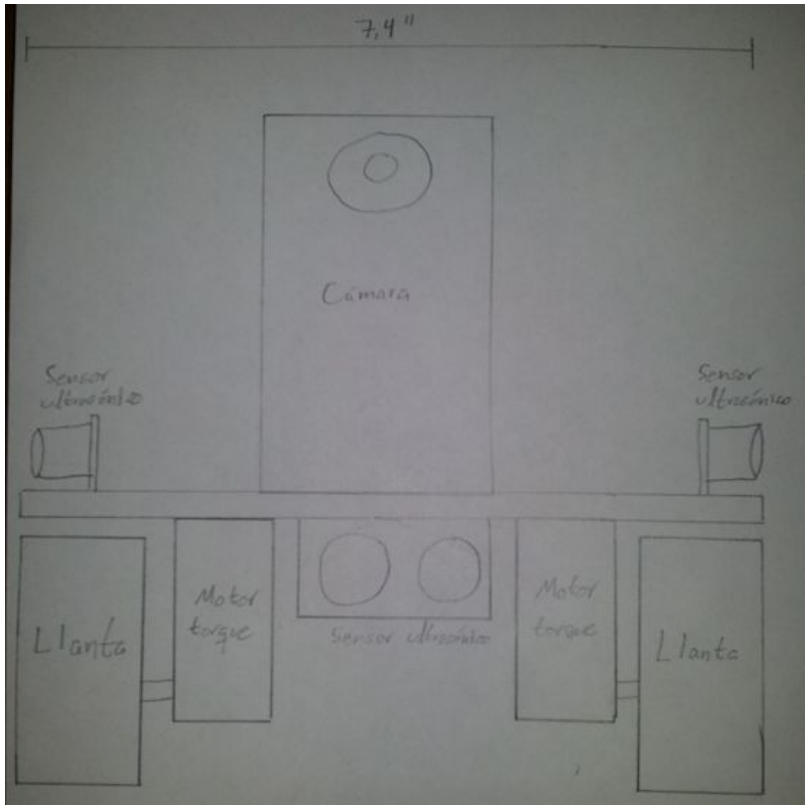
Sistema de agarre

El sistema de agarre será controlado con un pistón y un servomotor. Se encuentra en la parte de en medio del robot y atraviesa la tabla de madera. La víctima va a ser cargada en la parte inferior del robot. Por la parte de arriba se encuentra el pistón, que con bajar sus pocos centímetros será suficiente para levantar la víctima y no arrastrarla. La recibirá un medio cilindro que está fijo por la parte de abajo del robot. Por dentro tiene dos limit switches que indicarán al robot cuando puede cerrar su mano. La parte que va a cerrar es controlada por un servomotor y su posición inicial es levantada por debajo del robot. Éste servomotor baja otro medio cilindro que atrapa a la víctima dentro de un cilindro hueco, luego es levantado por el pistón para el traslado. Para bajar a la víctima solo hace falta abrir el cilindro hueco con el servomotor.

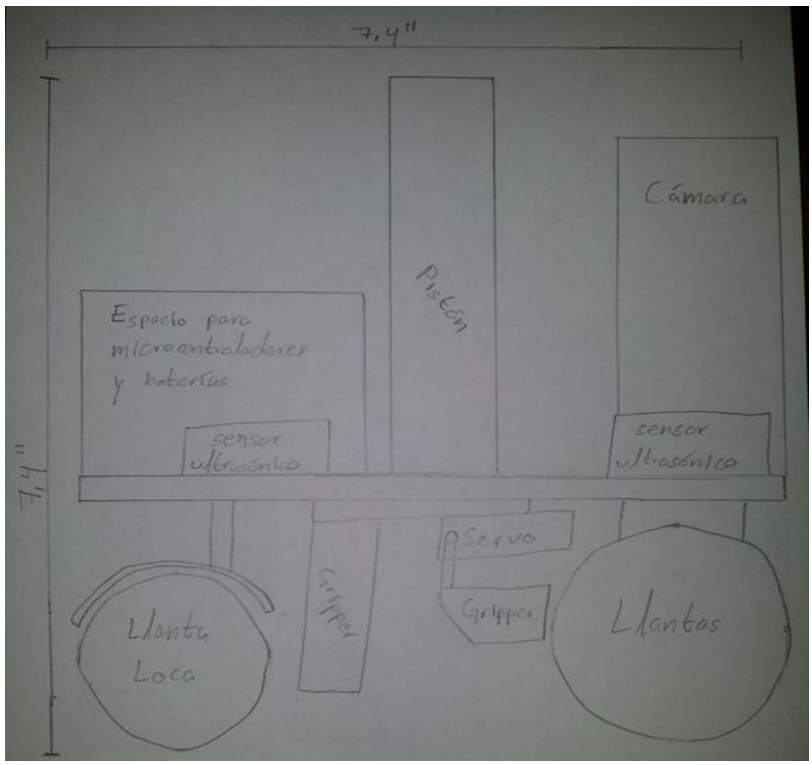
Vista Superior



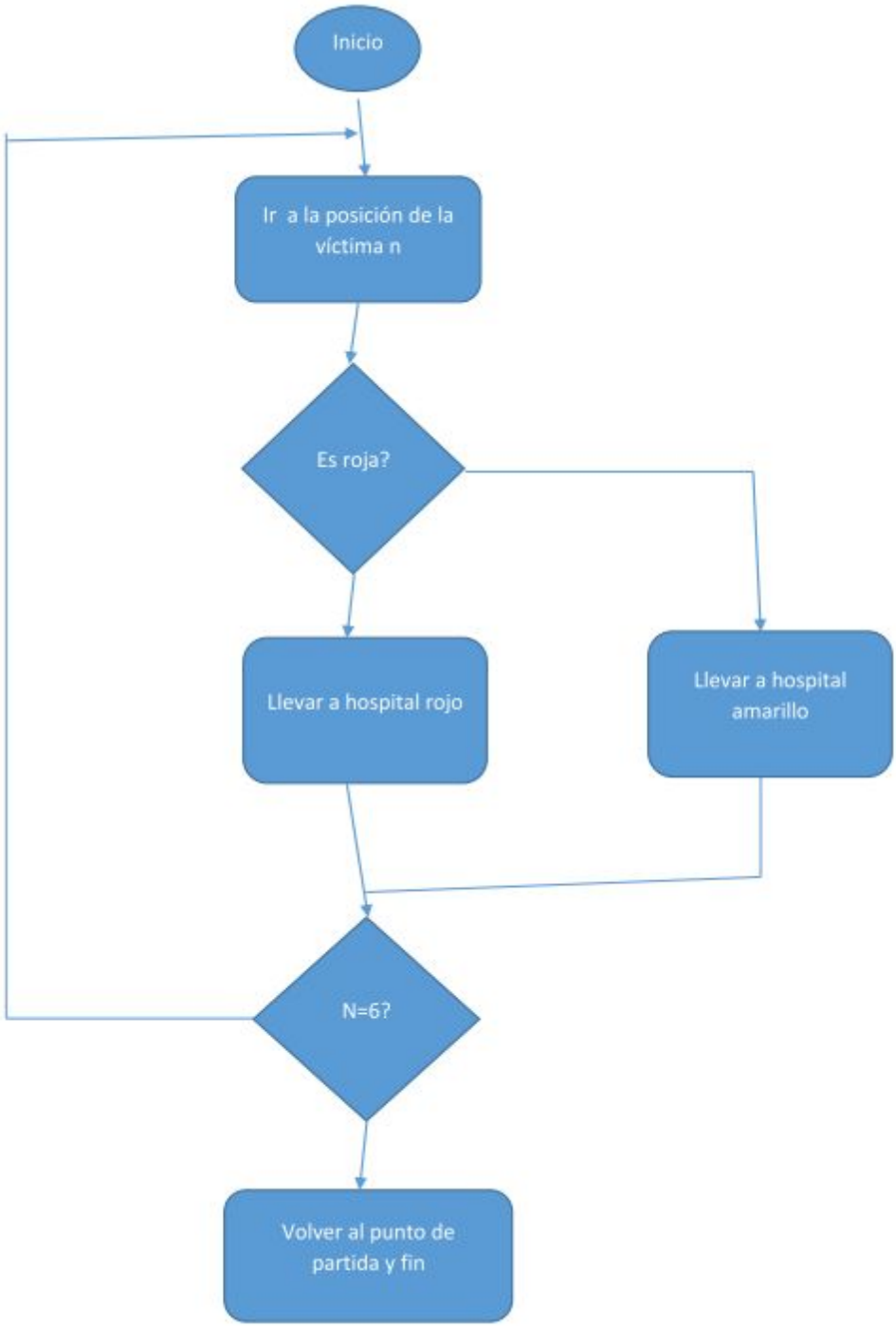
Vista Frontal



Vista Lateral



Algoritmo

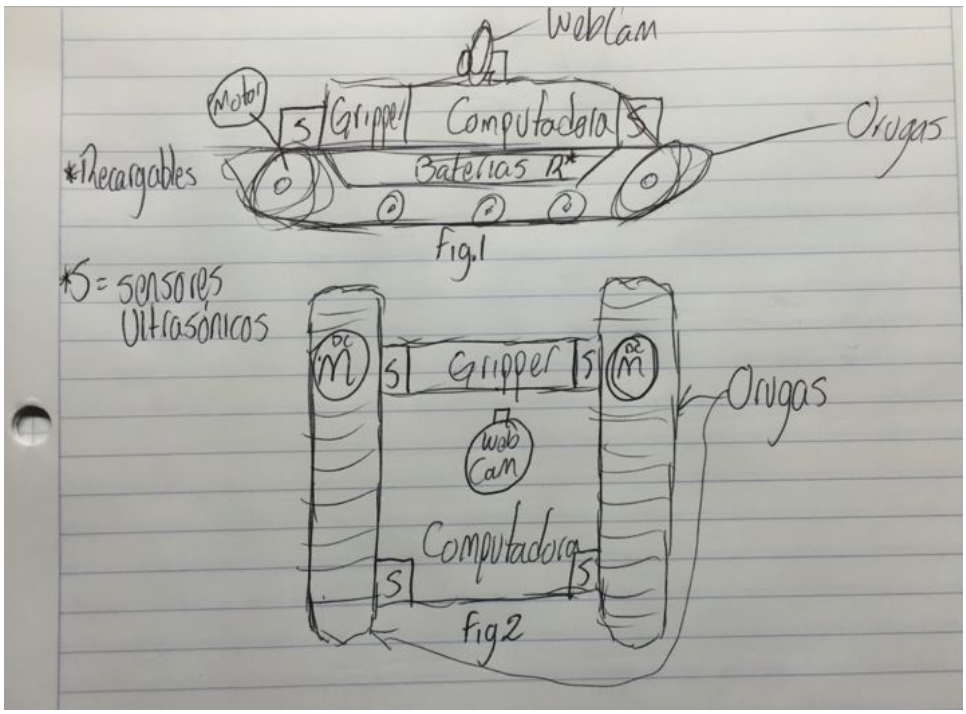

Inicio
Ir a la posición de la víctima n
Es roja?
Llevar a hospital rojo
Llevar a hospital amarillo
N=6?
Volver al punto de partida y fin

Roberto

Material

Dispositivo	Cantidad	Uso
Cámara web	1	Visión
Pilas recargables de litio	2	Energía
Computadora Windows XP o Raspberry Pi Model B	1	Procesamiento
Motores de DC (6V)	2	Movimiento
Llantas Oruga	2	
Rines dentados	2	
Sensores ultrasónicos	4	Navegación
Servo motor (6V)	3	Sistema de agarre
Piezas impresas en 3D	2	
Balero	1	
Tabla de madera	X	Acomodo

Diseño tentativo



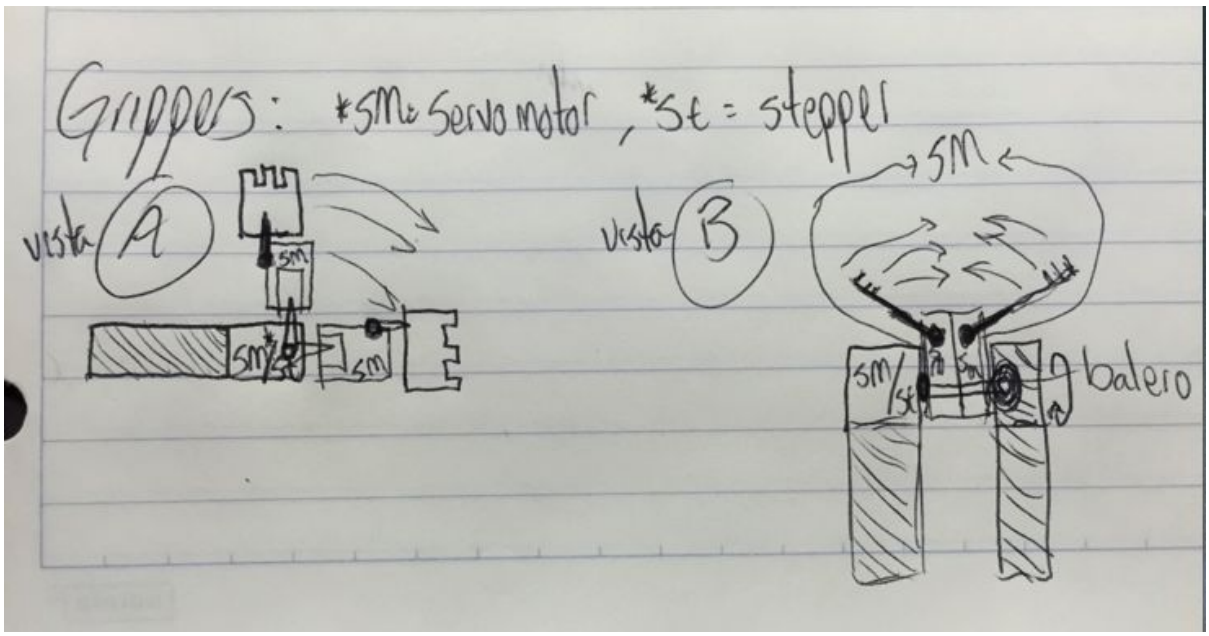
Se pensó en los dos tipos de terreno, por lo que se optó por usar orugas, siendo las bandas efectivas para la fricción, simple de manejar y rotar para los reducidos espacios dentro del laberinto.

Las baterías serian de 9-12 Vdc recargables para ahorrar en cuestión de gastos en energización.

4 sensores ultrasónicos en las esquinas para identificarse a sí mismo dentro del laberinto y sensores adicionales como de color para identificar el terreno.

La computadora sería una tarjeta Windows para mayor procesamiento y mayor posibilidad de manejar la mayor cantidad de puertos.

“Gripper”



El gripper básicamente consiste en un motor conectado con una pequeña placa apoyada del otro extremo con un balero y sobre la tablilla descansan dos servomotores, permitiendo un movimiento de 90 grados. En estos servomotores se adhieren una especie de materia áspero para mayor fricción. En conjunto seria como tenaza o pinza con un movimiento de 90 grados ubicada horizontalmente a la mitad de la altura de la “víctima”, de preferencia.

Algoritmo:

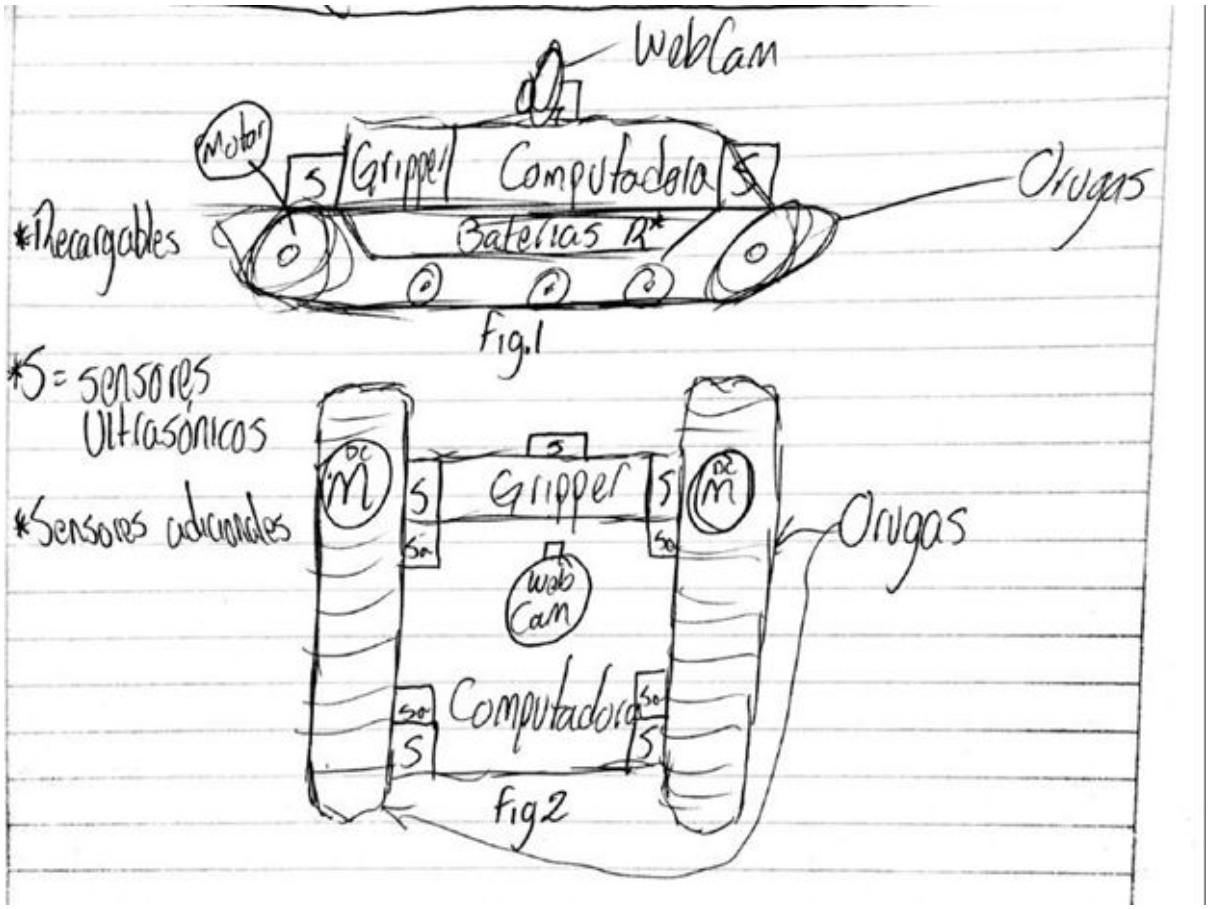
Inicio
Loop(1){
Revisar que haya victimas en pasillo hacia ambas direcciones con la cámara
Revisar paredes y fines de pasillo con sensores sonicos

Si se detecta una victima con la cámara
Ignorar cruces de pasillos con sensores
Avanzar hacia ella hasta que el sensor delantero se active (sónico o infrarrojo)
Detenerse
Activar gripper
Levantar
Llevarlo a su hospital correspondiente mediante el seguimiento de paredes
Avanzar hasta encontrar un cruce
Si se encuentra un cruce con un sensor sónico lateral delantero de un lado
Esperar que se active el sensor lateral trasero del mismo lado
Rotar hacia donde no hubo pared
Avanzar un determinado tiempo
Rotar hacia cualquier sentido
}

Propuesta Final

Plan A

Navegación

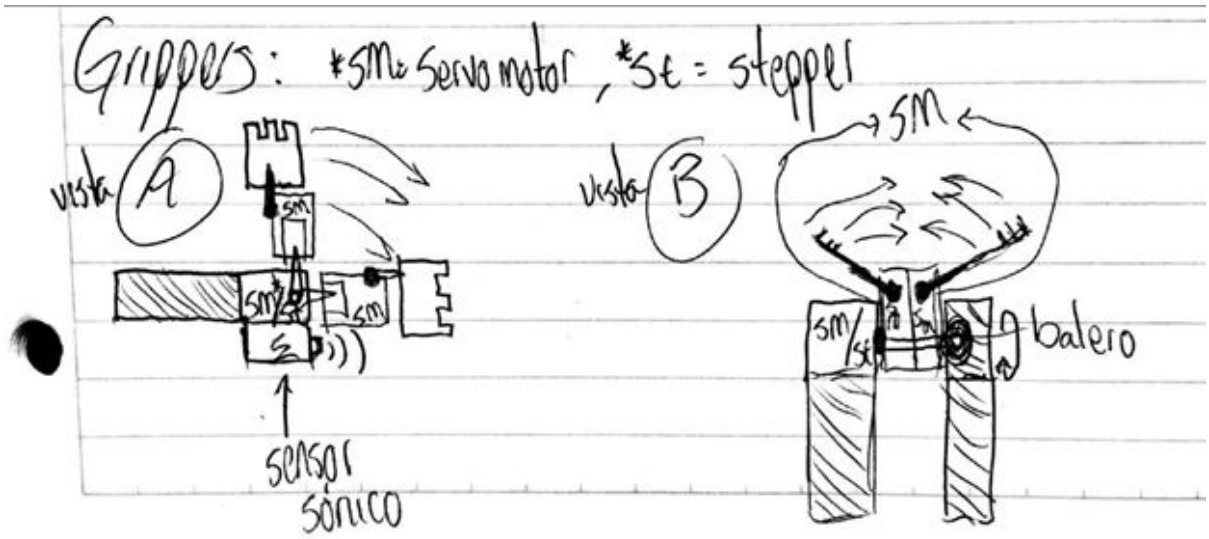


Se pensó en los dos tipos de terreno, por lo que se optó por usar orugas, siendo las bandas efectivas para la fricción, simple de manejar y rotar para los reducidos espacios dentro del laberinto. Se utilizarán para la navegación sensores infrarrojos, 4 en cada esquina del robot apuntando hacia las paredes para identificarse a sí mismo dentro del laberinto y dos sensores adicionales en las esquinas frontales apuntando hacia delante, esto para asegurar que no haya obstáculo o pared que aún obstruya al robot para moverse.

El cerebro del robot se compondrá de una tarjeta Raspberry Pi (de preferencia la más actualizada) para la visión y un microcontrolador pic32 para el control.

El objetivo es tener ya el mapa identificado, sabiendo donde se encuentran los hospitales y víctimas, avanzando con las orugas por cada pasillo revisando que se encuentre la víctima esperada. El movimiento se hará en base a la medición de distancia recorrida apoyándose con los sensores infrarrojos para el seguimiento de paredes.

Gripper



El gripper básicamente consiste en un motor conectado con una pequeña placa base apoyada del otro extremo con un balero y sobre la base descansan dos servomotores, permitiendo un movimiento de 90 grados. En estos servomotores se adhieren una especie de material áspero para mayor fricción. En conjunto el diseño sería similar al de una tenaza o pinza con un movimiento de 90 grados vertical, ubicada de preferencia horizontalmente a la mitad de la altura de la “víctima”. Para identificar a la víctima que está ya en posición para ser tomada, debajo de la estructura del gripper se encuentra un sensor sónico adicional, el cual medirá la distancia entre el gripper y la víctima, calculando la distancia estimada para ejecutar el agarre y levantamiento.

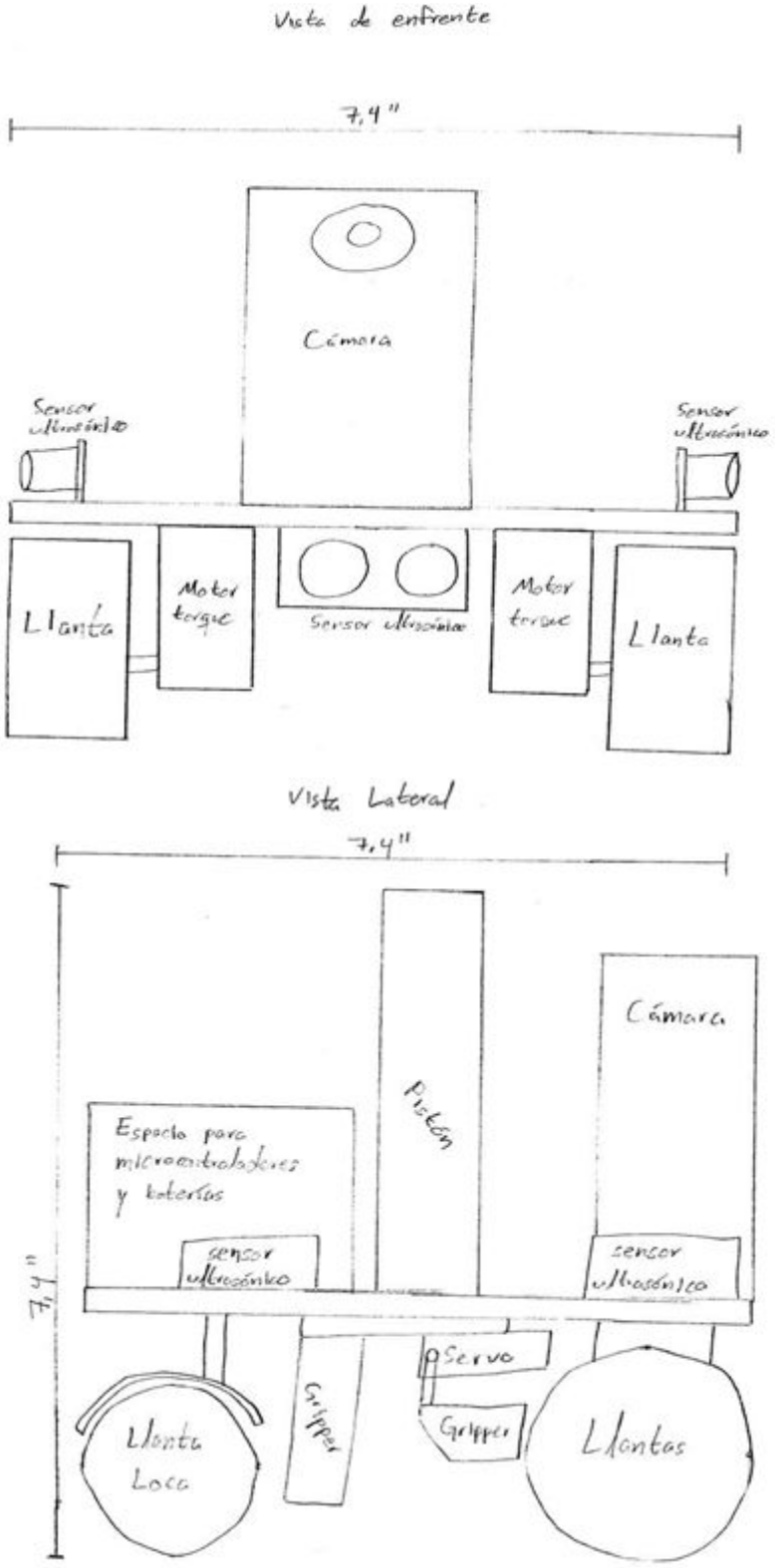
Visión

Para la visión se usará una cámara web genérica, conectado a una de las Raspberry y ubicada en la parte superior del robot. El objetivo principal de la cámara es la identificación de las víctimas en los pasillos por su determinado color. Tendrá la función básica de presencia de objeto e identificación de color utilizando métodos de filtrado y área de objeto en 2D. Al ser identificada la víctima, la cámara se desactivará y el robot se moverá al objeto en base a la navegación y la activación del gripper según los apartados anteriores lo especifican.

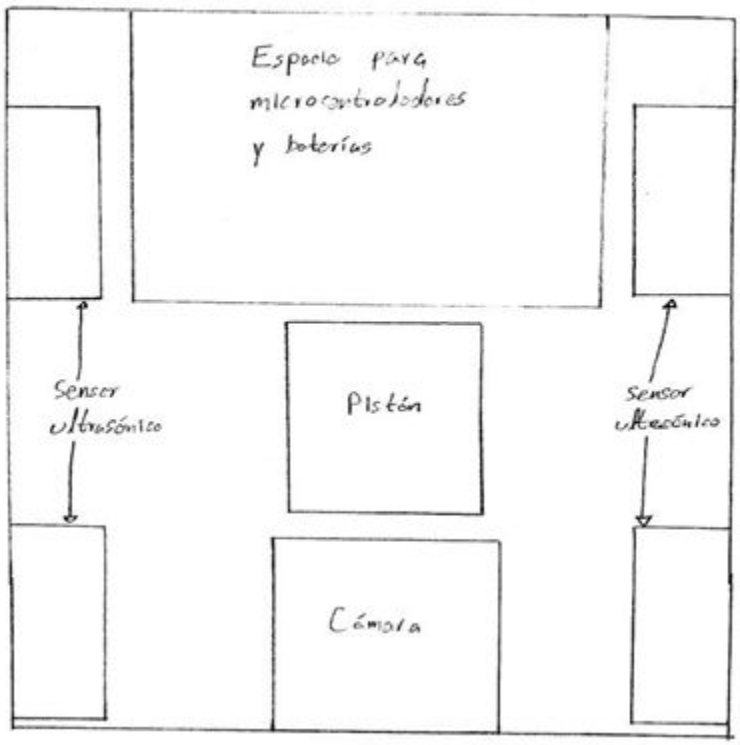
Lista de materiales

Material	**Uso**	**Cant**	**Almacén**	**Precio**	**Modelo**
Oruga	Navegación	1	no	$16.16 USA	Amazon
Motores	Navegación	1	no	$10.98 USA	Amazon
Placa	Navegación	1	no	$8.48 USA	Amazon
Sensores infrarrojos	Navegación	6	no	---	HC-SR04
Puente H	Navegación	2	si (dr. sedas)	---	L293D
Servo motor (12 V)	Gripper	1	si	---	---
Motor DC (12 V)	Gripper	1	si	---	---
Sensor ultrasónico	Gripper	1	si	---	HC-SR04
Mano	Gripper	1	no	---	Impreso 3D
Cámara	Visión	1	si	---	
Raspberry Pi B+	Procesamiento	1	si	---	---
Pic 32	Procesamiento	1		---	---
Pilas de Litio	Fuente	2	si	---	---

Plan B


Vista de enfrente
7,4"
Cámara
Sensor ultrasónico
Sensor ultrasónico
Llanta
Motor torque
Sensor ultrasónico
Motor torque
Llanta
Vista Lateral
7,4"
7,4"
Espacio para microcontroladores y baterías
Pistón
Cámara
sensor ultrasónico
sensor ultrasónico
Servo
Gripper
Gripper
Llanta Loca
Llantas

Vista superior

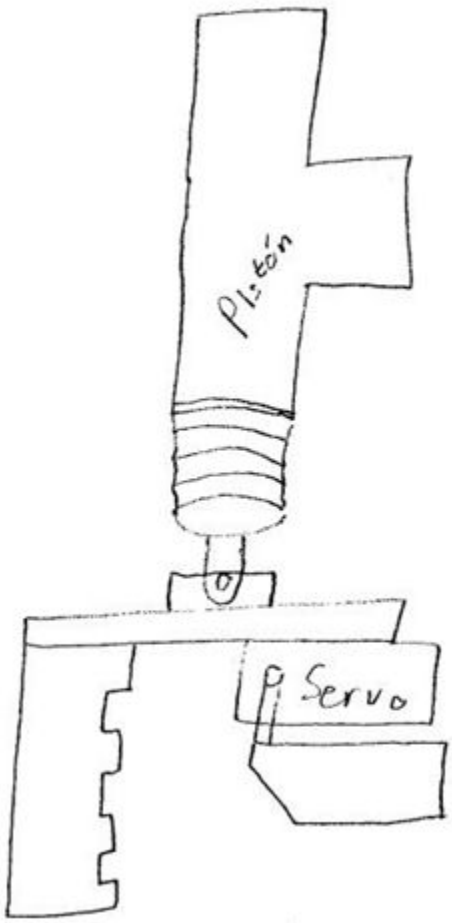


Navegación

El robot cuenta con tres llantas, dos en la parte de adelante y una en la parte de atrás. La de atrás es una llanta loca que solo sirve para equilibrio y solo sigue a las otras dos. Las otras dos se encuentran en la parte de adelante y son controladas cada una con un motor de DC de torque de 6V. Deben tener la mayor fricción posible para poder manejar fuera de la ciudad. Los motores serán controlados con ayuda del chip L298D (puente H) para controlar dirección.

La navegación se realizará con los 5 sensores infrarrojos. Habrá dos a cada lado del robot y uno en frente. Los de los lados van a verificar que el robot va caminando derecho por las calles o pasillos de la ciudad, así como las orillas de la pista en el monte. El de adelante se usará para evitar chocar de frente y también para ver si se está bien alineado con la víctima para poder recogerla.

Gripper



El sistema de agarre será controlado con un pistón y un servomotor. Se encuentra en la parte de en medio del robot y atraviesa la tabla de madera. La víctima va a ser cargada en la parte inferior del robot. Por la parte de arriba se encuentra el pistón, que con bajar sus pocos centímetros será suficiente para levantar la víctima y no arrastrarla. La recibirá un medio cilindro que está fijo por la parte de abajo del robot. Por dentro tiene dos limit switches que indicarán al robot cuando puede cerrar su mano. La parte que va a cerrar es controlada por un servomotor y su posición inicial es levantada por debajo del robot. Éste servomotor baja otro medio cilindro que atrapa a la víctima dentro de un cilindro hueco, luego es levantado por el pistón para el traslado. Para bajar a la víctima solo hace falta abrir el cilindro hueco con el servomotor.

Visión

En la parte delantera del robot se encuentra la cámara encima de la tabla de madera. Lo único que se espera sacar de la cámara es si se ve una víctima y de qué color es. Entre el sistema de navegación y agarre se sabrá si se tiene a la víctima a salvo.

Lista de materiales

Material	Uso	Cant	Almacén	Precio	Modelo
Motor DC de torque (12 V)	Navegacion	2	si	---	
Llantas	Navegacion	2	si	---	
Llanta loca	Navegacion	1	no		Home depot
Puente H	Navegacion	2	si	---	L298D
Sensores infrarrojos	Navegacion	6	si	---	HC-SR04
Tabla de madera	Navegacion	1	no		Home depot
Servo motor (12 V)	Gripper	1	si	---	
Pistón	Gripper	1	no		steren
Limit switch	Gripper	2	si	---	
Sensor ultrasónico	Gripper	1	si	---	HC-SR04
Mano	Gripper	1	no		Impreso 3D
Cámara	Visión	1	si	---	
Raspberry Pi B+	Procesamiento	1	si	---	
Pic 32	Procesamiento	1			
Pilas de Litio	Fuente	2	si	---	

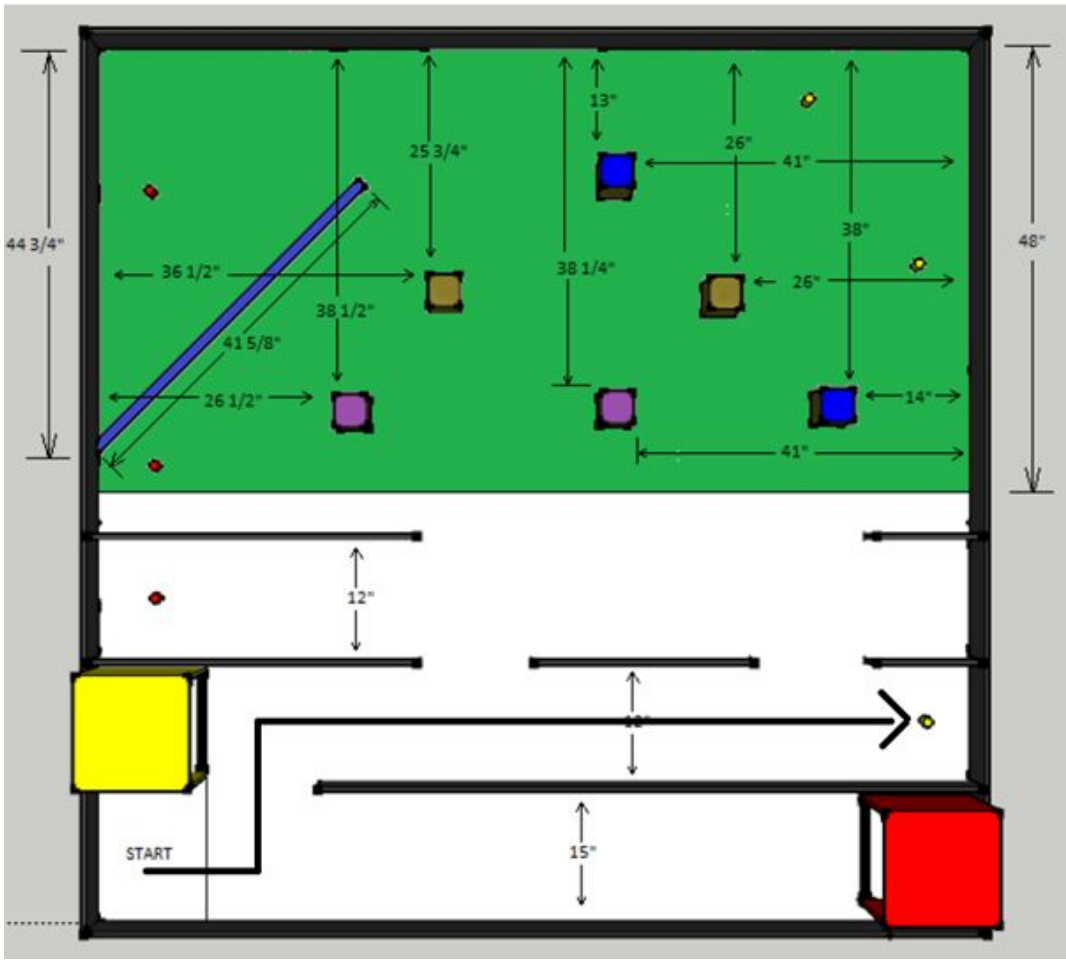
Algoritmo General

1. Ir a la posición P1 de la imagen.
2. Si la víctima es amarilla, llevarla al hospital amarillo, si no, llevarla al hospital rojo.
3. Ir a la posición P2 de la imagen.
4. Si la víctima es amarilla, llevarla al hospital amarillo, si no, llevarla al hospital rojo.
5. Ir a la posición P3 de la imagen.
6. Si hay víctima, pasar al paso 7, si no, pasar al paso 8.
7. Si la víctima es amarilla, llevarla al hospital amarillo, si no, llevarla al hospital rojo.
8. Ir a la posición P4 de la imagen.
9. Si hay víctima, pasar al paso 10, si no, pasar al paso 11.
10. Si la víctima es amarilla, llevarla al hospital amarillo, si no, llevarla al hospital rojo.
11. Si ya se tienen 4 victimas, pasar al paso 19, si no, continuar.
12. Ir a la posición P5 de la imagen.
13. Si hay víctima, pasar al paso 14, si no, pasar al paso 15.
14. Si la víctima es amarilla, llevarla al hospital amarillo, si no, llevarla al hospital rojo.
15. Si ya se tienen 4 victimas, pasar al paso 19, si no, continuar.
16. Ir a la posición P6 de la imagen.
17. Si hay víctima, pasar al paso 18, si no, pasar al paso 19.
18. Si la víctima es amarilla, llevarla al hospital amarillo, si no, llevarla al hospital rojo.
19. Fin

Rutas de Navegación

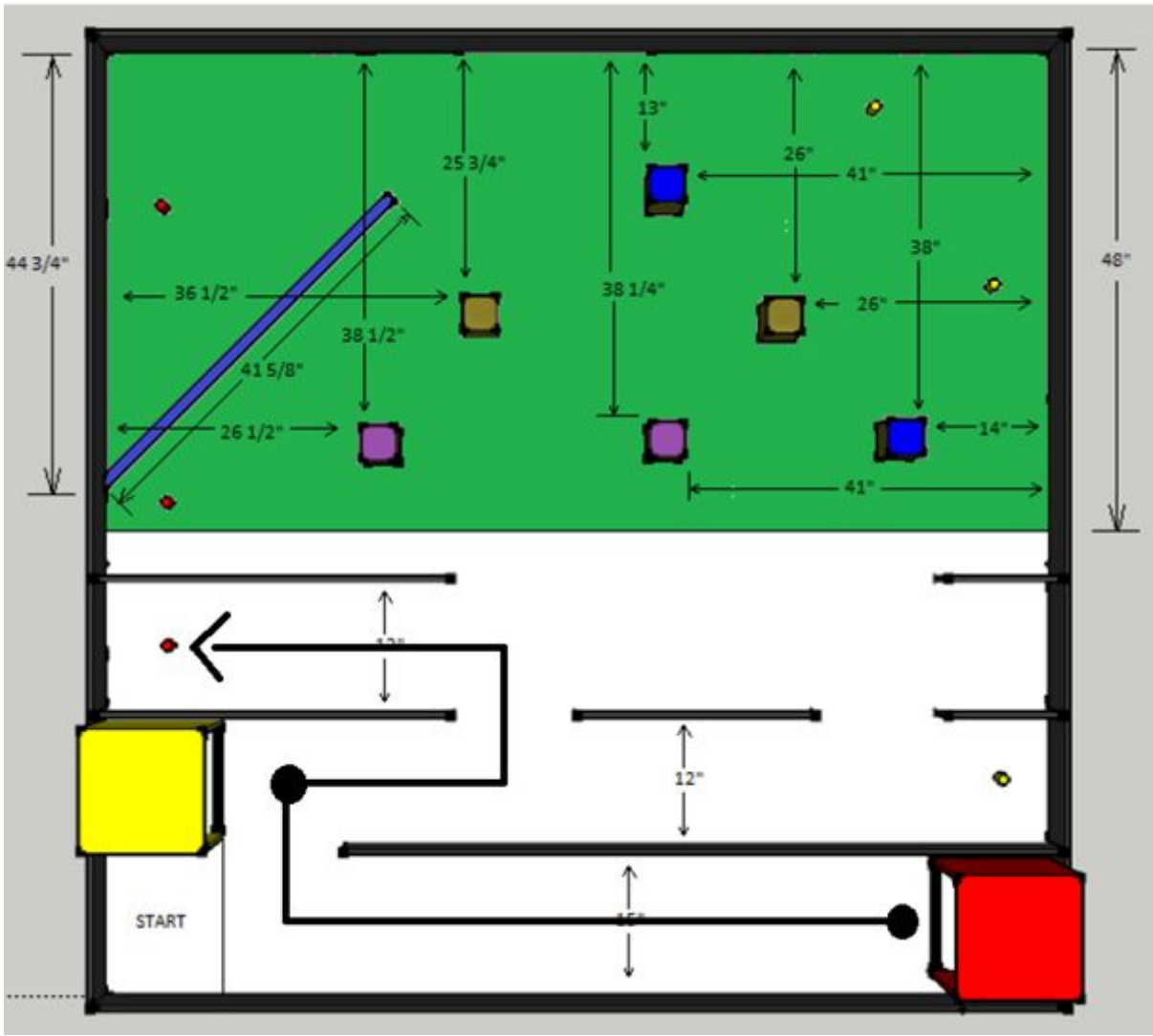
Todas las siguientes rutas cuentan con paredes como referencia para poder guiarnos con los seis sensores ultrasónicos con los que cuenta el robot.

Ruta a la posición 1



1. Usa los sensores izquierdos para saber cuando pasa el hospital amarillo.
2. Gira a la izquierda y comienza a avanzar.
3. Usa sus sensores derechos para saber cuando pasó el pasillo.
4. Gira a la derecha.
5. Usa sus sensores a la derecha para ir derecho a la víctima.

Ruta a la posición 2

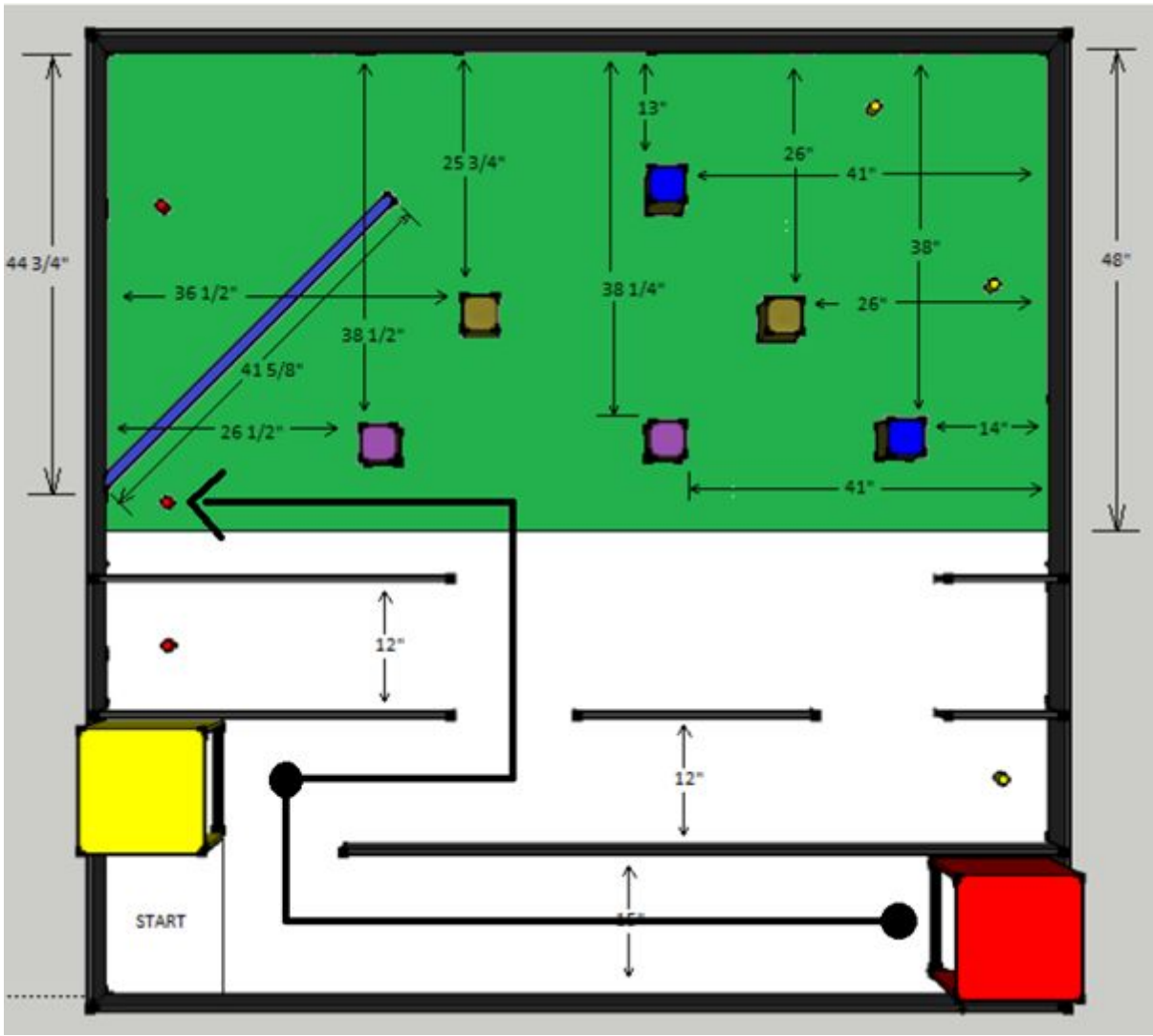


Si la víctima era roja se realiza la navegación siguiente:

1. Se gira 180º sobre su eje
2. Usa los sensores derechos para saber cuando pasó el pasillo
3. Gira 90º a la derecha y comienza a avanzar
4. Usa los sensores derechos para saber cuando pasó el pasillo
5. Gira 90º a la derecha y comienza a avanzar
6. Usa los sensores izquierdos para saber cuando pasó el pasillo
7. Gira 90º a izquierda y usa los sensore izquierdos para saber cuando pasó el pasillo
8. Gira 90º a la izquierda
9. Usa los sensores izquierdos para ir derecho a la víctima

Si la víctima era amarilla se realizan los pasos 1, 6, 7, 8, y 9

Ruta a la posición 3

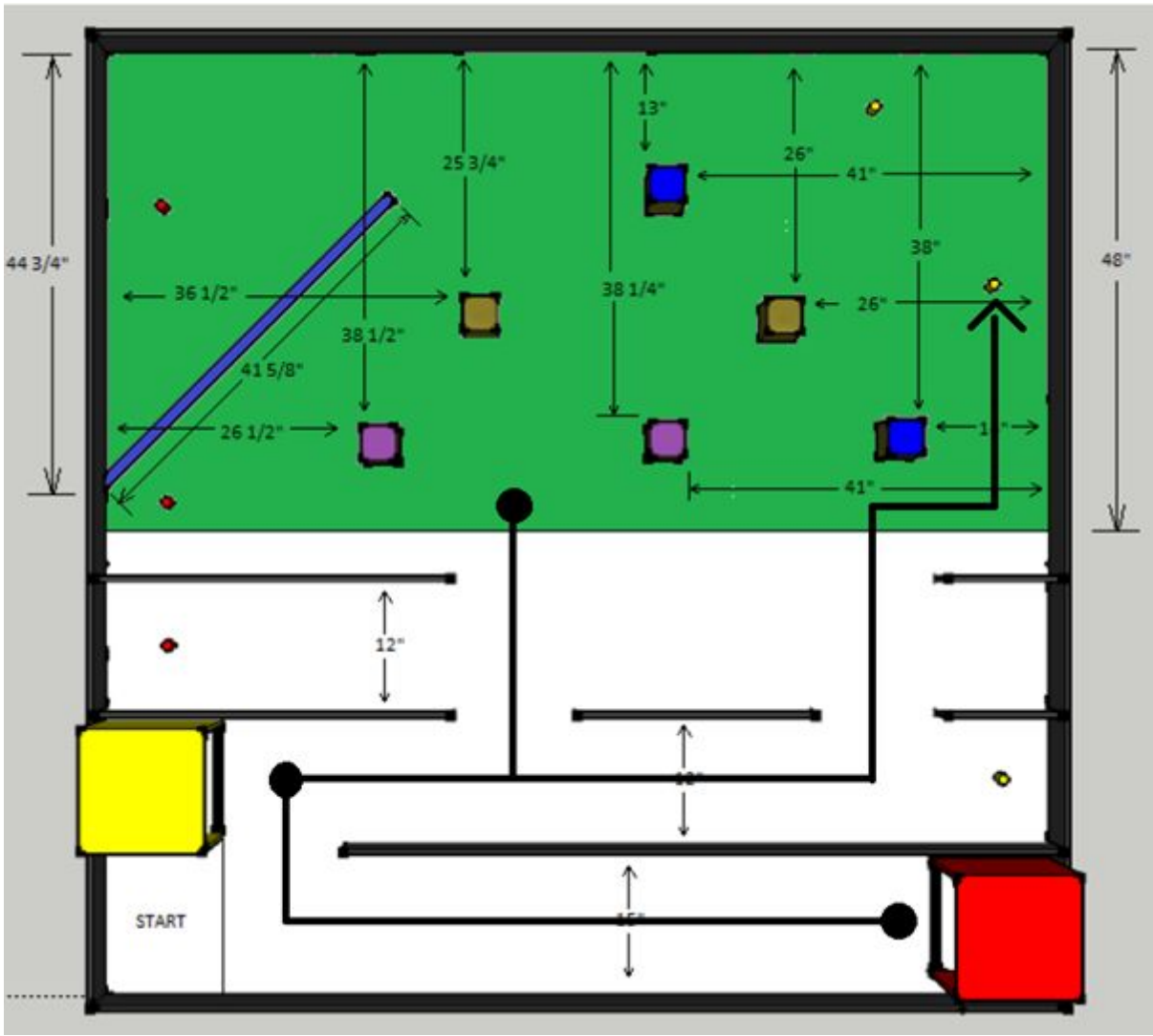


Si la víctima era roja se realiza la navegación siguiente:

1. Se gira 180º sobre su eje
2. Usa los sensores derechos para saber cuando pasó el pasillo
3. Gira 90º a la derecha y comienza a avanzar
4. Usa los sensores derechos para saber cuando pasó el pasillo
5. Gira 90º a la derecha y comienza a avanzar
6. Usa los sensores izquierdos para saber cuando pasó el pasillo
7. Gira 90º a izquierda y usa los sensore izquierdos para saber cuando pasó el segundo pasillo
8. Gira 90º a la izquierda
9. Usa los sensores izquierdos para ir derecho a la víctima

Si la víctima era amarilla se realizan los pasos 1, 6, 7, 8, y 9

Ruta a la posición 4

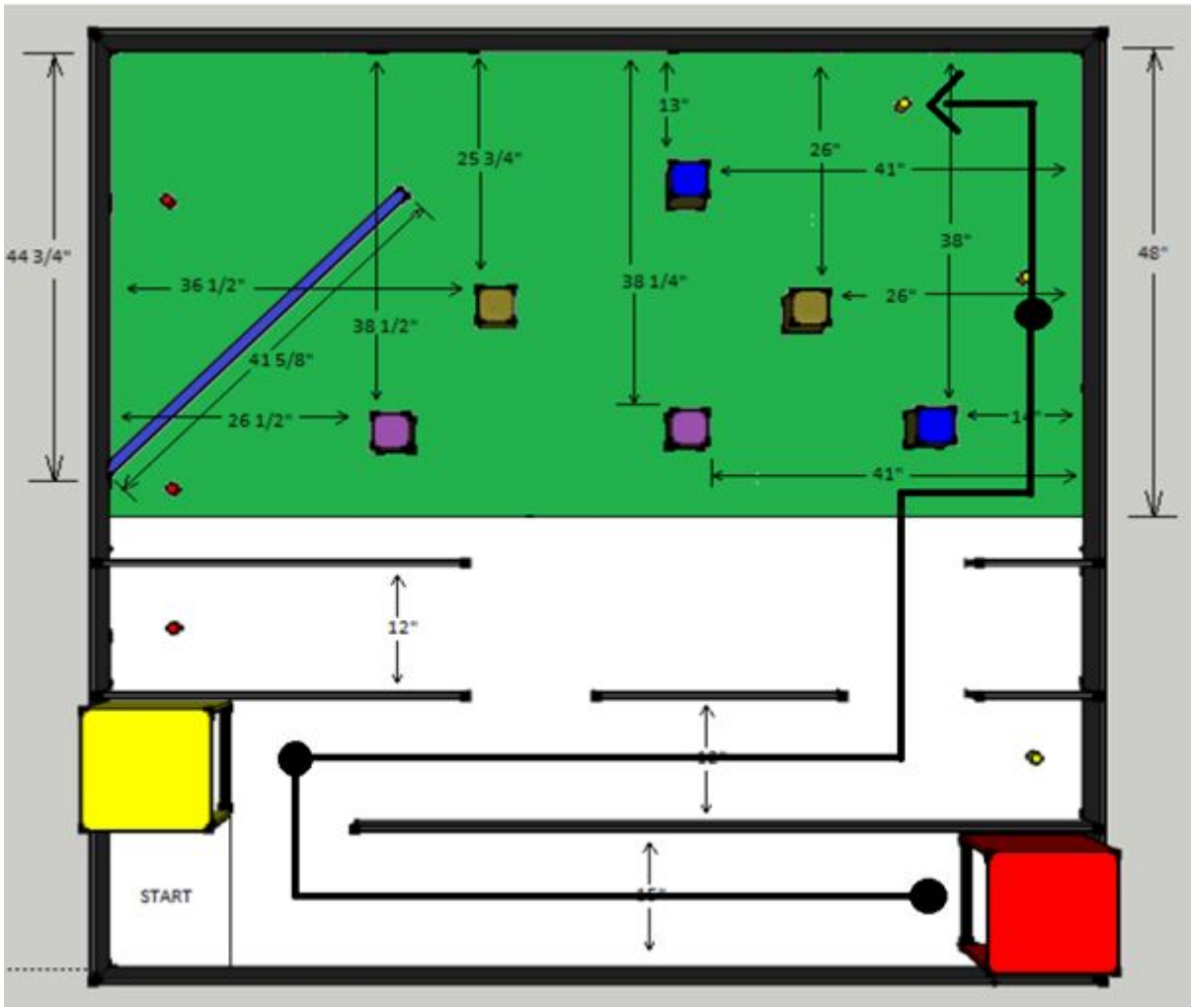


Si la víctima era roja se realiza la navegación siguiente:

1. Se gira 180º sobre su eje
2. Usa los sensores derechos para saber cuando pasó el pasillo
3. Gira 90º a la derecha y comienza a avanzar
4. Usa los sensores derechos para saber cuando pasó el pasillo
5. Gira 90º a la derecha y comienza a avanzar
6. Usa los sensores izquierdos para saber cuando pasó el pasillo
7. Sigue avanzando derecho
8. Usa los sensores izquierdos para saber cuando pasó el segundo pasillo
9. Gira 90º a izquierda y usa los sensore derechos para saber cuando pasó el segundo pasillo
10. Gira 90º a la derecha y avanza
11. Usa los sensores delanteros para llegar hasta la pared
12. Gira 90º a la izquierda
13. Usa los sensores derechos para ir derecho a la víctima

Si la víctima era amarilla se realizan los pasos 1, 6, 7, 8, 9, 10, 11, 12, y 13

Ruta a la posición 5

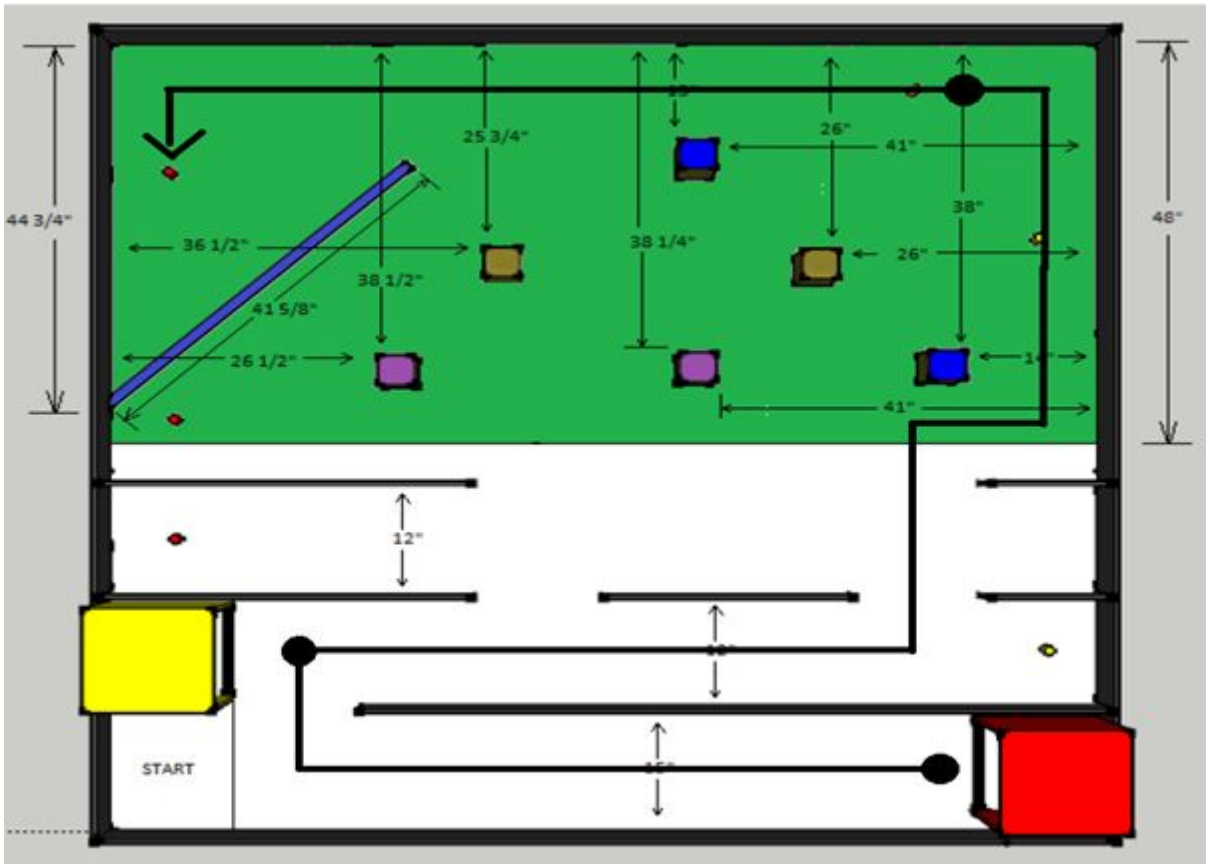


Si la víctima era roja se realiza la navegación siguiente:

1. Se gira 180º sobre su eje
2. Usa los sensores derechos para saber cuando pasó el pasillo
3. Gira 90º a la derecha y comienza a avanzar
4. Usa los sensores derechos para saber cuando pasó el pasillo
5. Gira 90º a la derecha y comienza a avanzar
6. Usa los sensores izquierdos para saber cuando pasó el pasillo
7. Sigue avanzando derecho
8. Usa los sensores izquierdos para saber cuando pasó el segundo pasillo
9. Gira 90º a izquierda y usa los sensore derechos para saber cuando pasó el segundo pasillo
10. Gira 90º a la derecha y avanza
11. Usa los sensores delanteros para llegar hasta la pared
12. Gira 90º a la izquierda
13. Usa los sensores derechos para avanzar y usa los delanteros para llegar hasta la pared
14. Gira 90º a la izquierda
15. Usa los sensores derechos para ir a la víctima

Si la víctima era amarilla se realizan los pasos 1, 6, 7, 8, 9, 10, 11, 12, 13, 14 y 15

Ruta a la posición 6



Si la víctima era roja se realiza la navegación siguiente:

1. Se gira 180º sobre su eje
2. Usa los sensores derechos para saber cuando pasó el pasillo
3. Gira 90º a la derecha y comienza a avanzar
4. Usa los sensores derechos para saber cuando pasó el pasillo
5. Gira 90º a la derecha y comienza a avanzar
6. Usa los sensores izquierdos para saber cuando pasó el pasillo
7. Sigue avanzando derecho
8. Usa los sensores izquierdos para saber cuando pasó el segundo pasillo
9. Gira 90º a izquierda y usa los sensore derechos para saber cuando pasó el segundo pasillo
10. Gira 90º a la derecha y avanza
11. Usa los sensores delanteros para llegar hasta la pared
12. Gira 90º a la izquierda
13. Usa los sensores derechos para avanzar y usa los delanteros para llegar hasta la pared
14. Gira 90º a la izquierda
15. Usa los sensores derechos para avanzar
16. Usa los sensores delanteros para llegar hasta la pared
17. Gira 90º hasta la izquierda
18. Usa los sensores derechos para ir a la víctima

Si la víctima era amarilla se realizan los pasos 1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 y 18

Plan de Trabajo

Actividad	Descripción	Fecha de entrega
ACT 1	Propuesta por cada miembro del equipo.	Semana 2
ACT 2	Actividad de Ética.	Semana 3
ACT 3	Propuesta final dividida en módulos (Plan A y B).	Semana 4
ACT 4.1	Pruebas sobre la interfaz y software de la computadora de control.	Semana 5
ACT 4.2	Integración de la computadora, driver eléctrico, motores, y software de control.	Semana 6
ACT 4.3	Diseño del chassis e integración del sistema de control.	Semana 7
ACT 4.4	Pruebas funcionales de movimiento del vehículo.	Semana 8
ACT 4.5	Integración del chassis con sensores: detección de paredes, víctimas y obstáculos.	Semana 9
ACT 4.6	Código que sigue una pared en paralelo.	Semana 10
ACT 5.1	Diseño de sensor de detección de víctima, se debe detectar si la víctima está presente o no.	Semana 5
ACT 5.2	Diseño de sensor de detección de obstáculo, se debe detectar si el obstáculo está presente o no.	Semana 6
ACT 5.3	Diseño del sensor de detección del color de la víctima, se debe detectar si la víctima es roja o amarilla.	Semana 7
ACT 5.4	Diseño del sensor de detección de pasillos en la pared, se debe detectar si se encuentra un pasillo.	Semana 8
ACT 5.5	Diseño de rutinas que detecten que tipo de objeto se encuentra en frente de la cámara.	Semana 9
ACT 5.6	Mejoras en el sistema de visión.	Semana 10
ACT 6.1	Diseño en un paquete CAD, de el mecanismo que sostendrá a la víctima.	Semana 5
ACT 6.2	Fabricación del mecanismo anterior. Diseño del mecanismo que levantará a la víctima.	Semana 6
ACT 6.3	Fabricación del mecanismo anterior.	Semana 7

ACT 6.4	Integración de los mecanismos para sostener y levantar la víctima.	Semana 8
ACT 6.5	Integración de los mecanismos anteriores con el vehículo.	Semana 9
ACT 6.6	Mejoras para el sistema de agarre.	Semana 10
ACT 7.1	Desarrollo de software que vaya por la víctima en el segundo corredor, la deje el hospital correcto y regrese al punto de inicio.	Semana 11
ACT 7.2	Desarrollo de software que haga la rutina anterior y vaya también por la víctima en el tercer corredor, la deje en el hospital correcto y regrese al punto de inicio.	Semana 12
ACT 7.3	Desarrollo de software que haga la rutina anterior y vaya también por la víctima en la esquina inferior-izquierda de la zona fuera del camino, la deje en el hospital correcto y regrese al punto de inicio.	Semana 13
ACT 7.4	Desarrollo de software que haga la rutina anterior y vaya también por la víctima en la esquina superior-derecha de la zona fuera del camino, la deje en el hospital correcto y regrese al punto de inicio.	Semana 14
ACT 7.5	Desarrollo de software que haga la rutina anterior y vaya también por la víctima pasando el río, la deje en el hospital correcto y regrese al punto de inicio.	Semana 15
ACT 8	Competencia	Semana 16

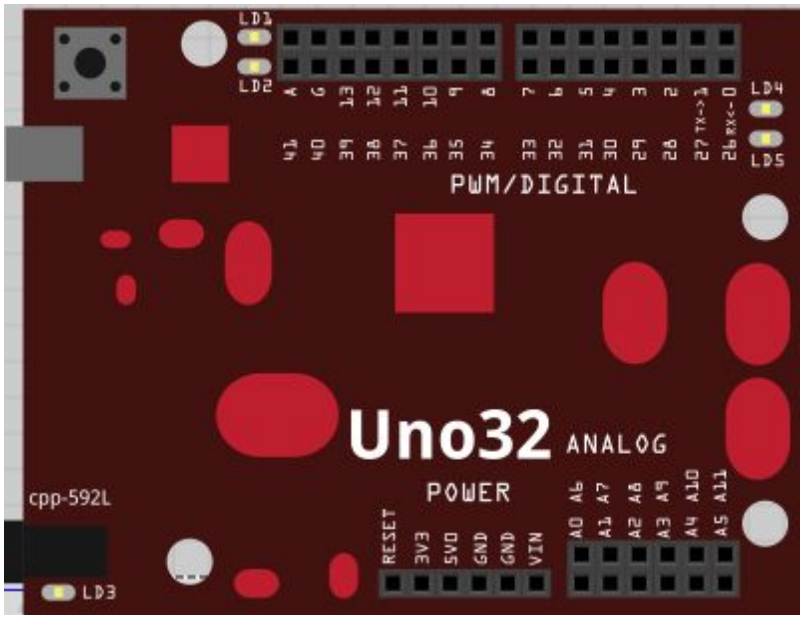
Material Usado

A continuación se presenta una lista del material utilizado para el robot:

Material	**Cantidad**
Raspberry Pi Model B	1
Tarjeta SD 8GB	1
Cámara Web Logitech	1
Pololu Step-Down a 5V	1
Cable de alimentación microUSB	1
Microcontrolador PIC32	1
Sensores ultrasónicos hc-sr04	5
Servomotores (5V)	2
Motores de Corriente Directa (5V)	2
Rueda loca	1
Llantas	2
Batería de celdas de Litio	1
Piezas de acrílico	2
Puente H	2
Interruptor	1
“Gripper”	1
Circuito soldado	1
Reguladores de voltaje LM7805	2

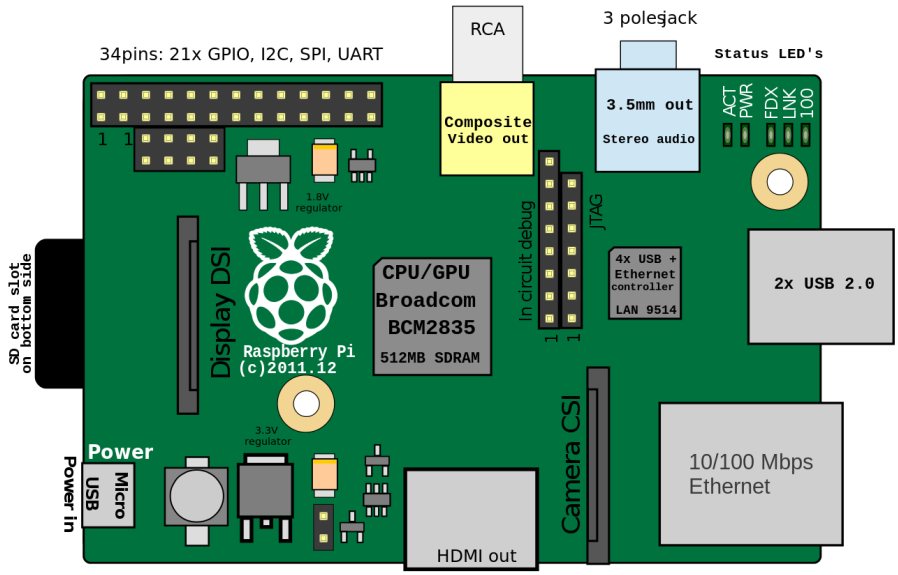
Tarjetas Usadas

PIC32



- Microcontrolador PIC32MX320F128H de Microchip
- 80 Mhz 32-bit MIPS
- 128K Flash
- 16K SRAM
- 42 pins de entrada/salida
- 2 LEDs incluidos
- Connexión USB mini
- 12 entradas análogas
- Voltaje de operación de 3.3V
- Frecuencia de operación de 80Mhz
- Corriente de operación de 75mA
- Voltaje de entrada recomendado de 7V a 15V
- Voltaje de entrada máximo de 20V
- Rango de voltaje de entrada análogo de 0V a 3.3V
- 18mA DC de corriente por pin

Raspberry Pi Modelo B



- Procesador ARM dual core @700MHz.
- Alimentacion: 5 Vdc (+-5%) - 3.5W.
- Memoria RAM: 512MB.
- Almacenamiento: Tarjeta SD Card - 8GB.
- General Purpose Input|Output (GPIO): 34 pines configurables.
- Tolerancia de señales de entrada: 5 Vdc.
- Magnitud de señales de salida: 3.3 Vdc.
- Puertos USB 2.0 (2).
- Puerto HDMI.
- 10/100 Ethernet.
- Salida Video de protocolo estandar.
- Entrada de alimentación microUSB.
- Interfaz para conexión de cámara de marca propia.
- Sistema Operativo basado en Linux (Raspbian - Wheezy).

Tablas de Entradas y Salidas

PIC32

Interfaz	Entrada - Salida	Acción
OC1 (RD0)	Salida	Onda PWM para Servomotor
OC2 (RD1)	Salida	Onda PWM para Servomotor
RD5	Salida	Señal de control para Puente H
RD6	Salida	Señal de control para Puente H
RD7	Salida	Señal de control para Puente H
RD11	Salida	Señal de control para Puente H
RD3	Salida	Señal Trigger para sensor delantero
RD4	Salida	Señal Trigger para sensor de la izquierda adelante
RG8	Salida	Señal Trigger para sensor de la derecha adelante
RG7	Salida	Señal Trigger para sensor de la izquierda atrás
RG6	Salida	Señal Trigger para sensor de la derecha atrás
RE0	Entrada	Señal Echo para sensor delantero
RE1	Entrada	Señal Echo para sensor de la izquierda adelante
RE2	Entrada	Señal Echo para sensor de la derecha adelante
RE3	Entrada	Señal Echo para sensor de la izquierda atrás
RE4	Entrada	Señal Echo para sensor de la derecha atrás
RD2	Salida	Señal que ordena a la Raspberry Pi a tomar una foto
RE5	Entrada	Señal que indica que la víctima se encuentra en frente un poco a la derecha
RE6	Entrada	Señal que indica que la víctima se encuentra en frente un poco a la izquierda
RE7	Entrada	Señal que indica que la víctima es roja
RE8	Entrada	Señal que indica que la víctima es amarilla

Raspberry Pi Model B

Interfaz	Entrada - Salida	Acción
GPIO4	Entrada	Señal para tomar siguiente “frame” y analizarlo
GPIO17	Salida	Indicador de víctima roja
GPIO22	Salida	Ubicación de víctima a la derecha
GPIO23	Salida	Ubicación de víctima a la izquierda
GPIO27	Salida	Indicador de víctima amarilla

Hardware

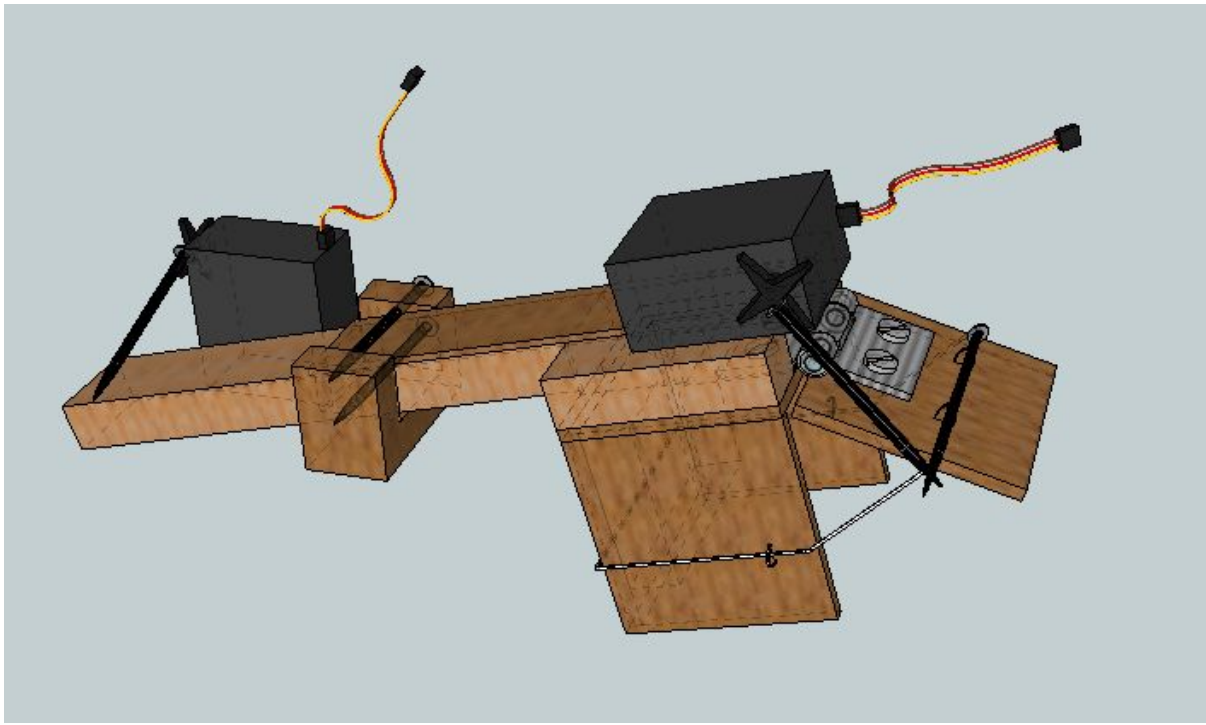
Diseño Mecánico

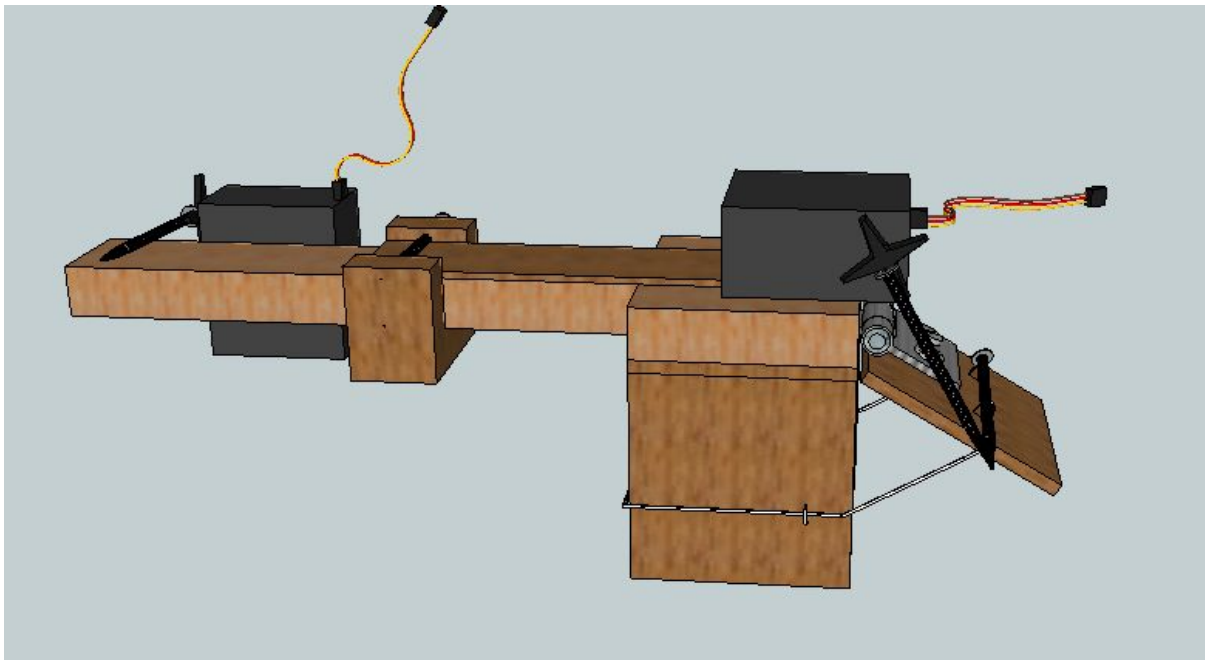
El cuerpo del robot está hecho con dos pedazos de acrílico de 12 cm por 15 cm. Ambos están unidos con cuatro tornillos que actúan como columnas para sostener el segundo acrílico. Se cuenta con dos pisos para componentes y la parte de abajo donde se ubican las ruedas.

En cuanto a la parte de abajo para las ruedas, el robot cuenta con dos motores de corriente directa con llantas en la parte de en frente, controlados con un puente H. En la parte trasera se encontraba una rueda loca, que solo seguía los movimientos de los motores. Junto con la rueda loca se encuentra la batería del robot.

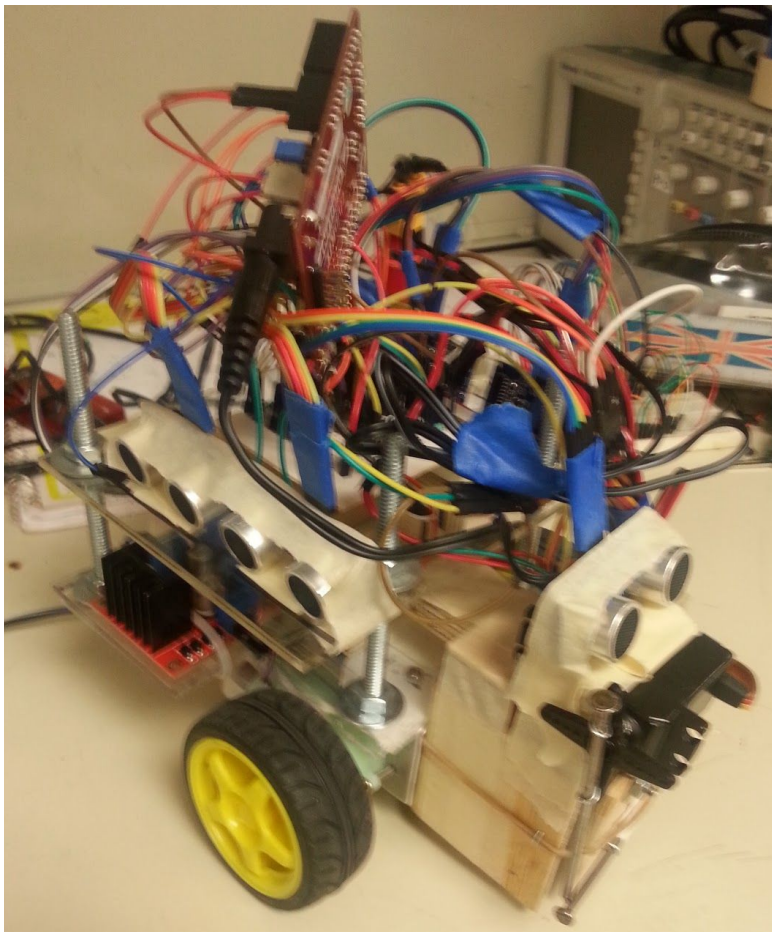
En el primer piso se encuentra el sistema de agarre y el puente H que controla a los motores. El sistema de agarre consta de un pivote, una tabla y una caja de madera. La caja se encuentra cerrada mecánicamente con una liga. Para controlar el mecanismo se usaron dos Servomotores, uno para abrir la puerta y otro para hacer contrapeso en la tabla y levantar todo el sistema.

Imágenes del diseño del sistema de agarre:



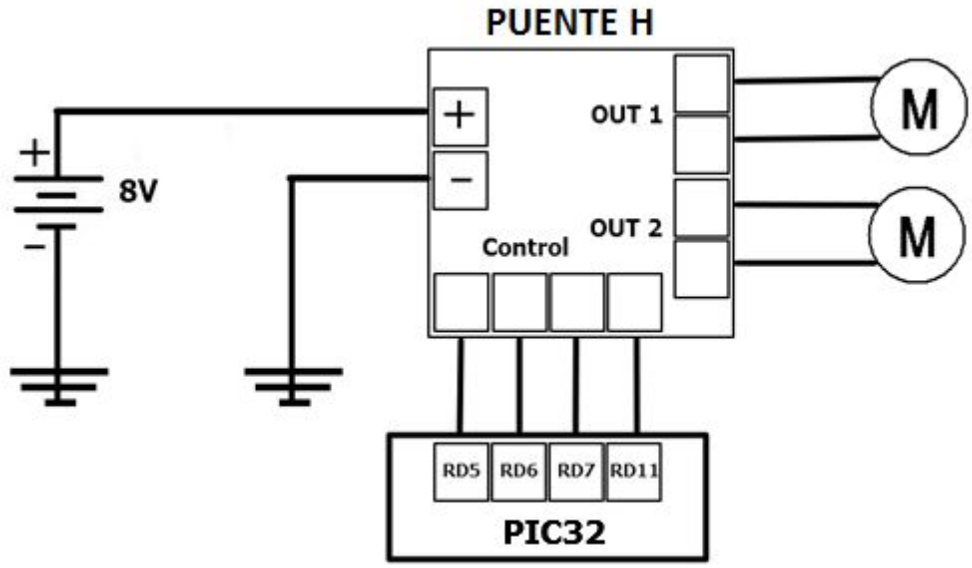


En el segundo piso se encuentra el microcontrolador (PIC32), cinco sensores ultrasónicos para movimiento y un circuito que regulaba señales de voltaje. Los puertos del Microcontrolador son de 3.3V y los sensores son de 5V, por eso la necesidad de regular cada señal que iba y venía de cada sensor. En esta parte se esperaba poner el segundo microcontrolador (Raspberry Pi) el cual hacia el procesamiento necesario para la visión y la cámara sobre el sistema de agarre, pero por problemas de tiempo no se pudo completar.

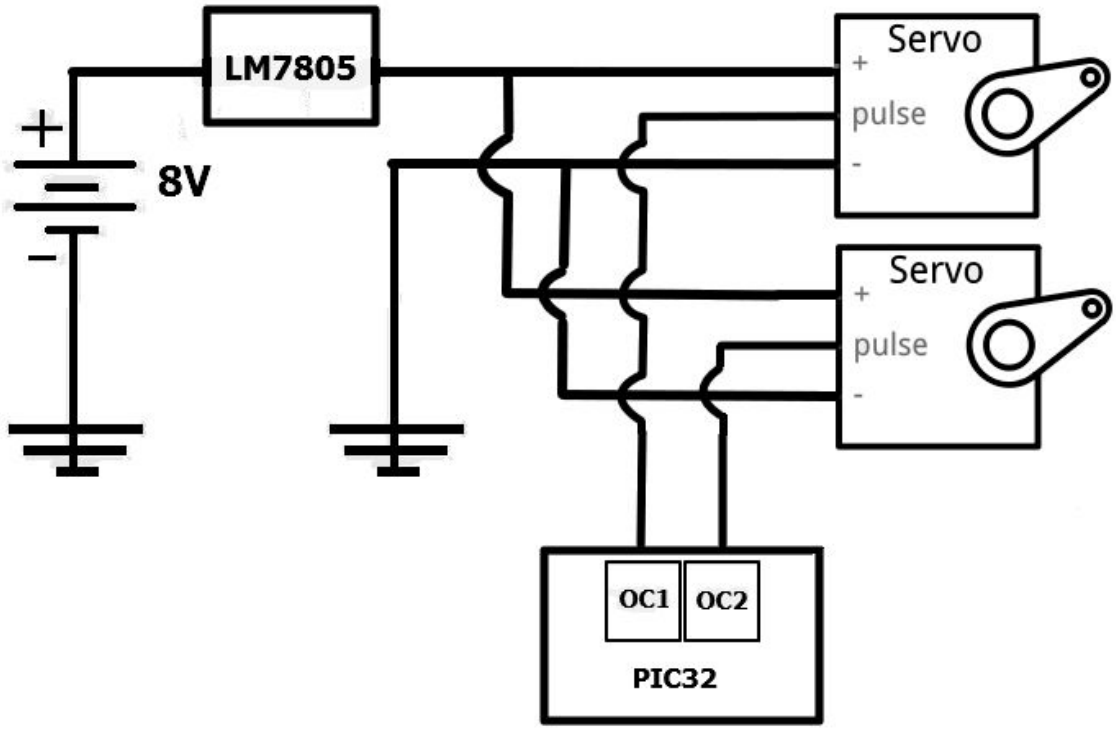


Producto Final

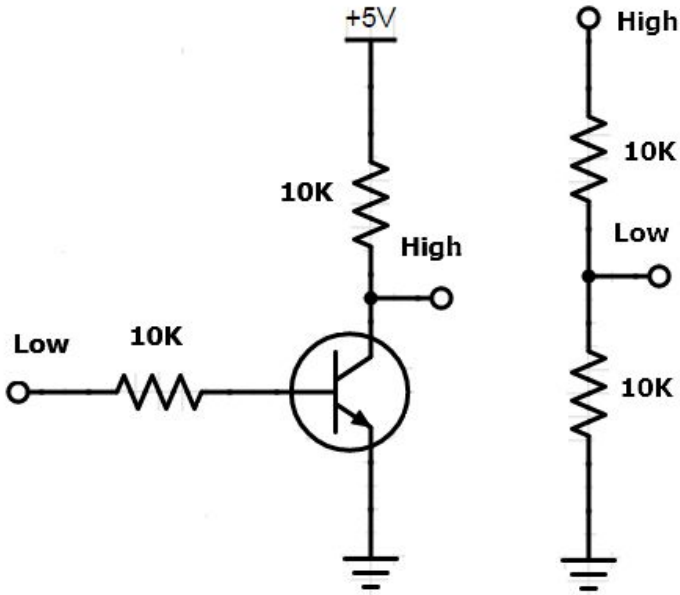
Diseño Eléctrico



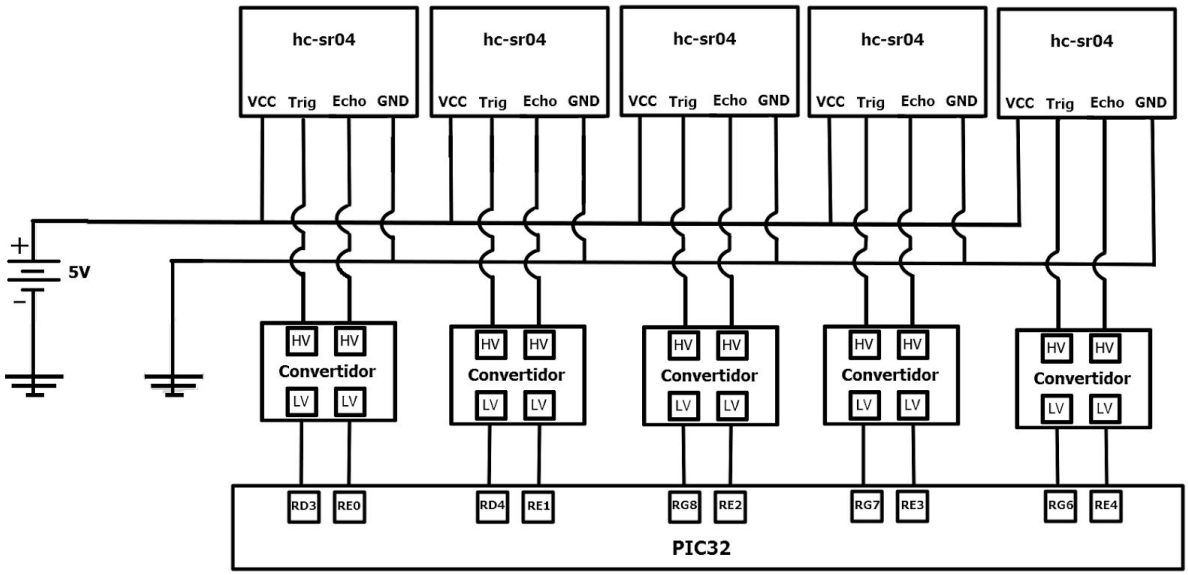
Conexión del puente H a la batería, al PIC32 y a los motores de movimiento.



Conexión de los Servomotores a la batería (por medio del regulador de voltaje) y al PIC32.

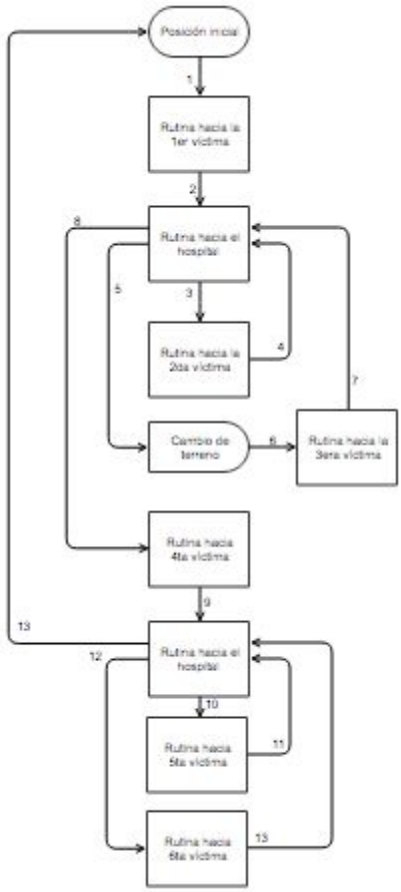


El circuito anterior es un convertidor de nivel lógico para poder comunicar los sensores de 5V con el microcontrolador de 3.3V, es usado 5 veces en el siguiente circuito.



Conexión de los 5 sensores ultrasónicos al microcontrolador, por medio del convertidor de nivel lógico.

Software



Navegación y traslado de víctimas.

El algoritmo consiste en una rutina paso a paso para revisar todos los puntos posibles donde se pueden encontrar victimas y regresar cada una al hospital amarillo. El motivo por el cual no se regresan las victimas a su hospital correspondiente es debido a que no se pudo incorporar visión al microcontrolador.

Funciones:

Tenemos diferentes funciones que hacemos llamar en los diferentes pasos del algoritmo. Estas realizan lo siguiente:

- **pulseOut y pulseIn**: Estas mandan el eco a los sensores y reciben la señal conforme a la distancia en que se encuentran de la pared.
- **close_Gripper, open_Gripper, up_Gripper, down_Gripper:** Estas funciones se encargan de mover los servo motores del gripper para poder abrirlo, cerrarlo, subirlo y bajarlo.

- **moveFwd, y moveBack**: Estas reciben una distancia en centímetros y prenden los motores para avanzar o retroceder hasta cumplir esa distancia.
- **rotClk, rotCount, rotD, rotI**: Reciben los grados que el robot debe rotar. Para esto apaga los motores, prende uno hacia delante y otro hacia atrás por el tiempo necesario para cumplir los grados pedidos.
- **seguirPared**: Recibe un 1 si se desea seguir la pared derecha, un dos para la pared izquierda y un 3 para no seguir una pared.
    - Prende el sensor delantero
    - Si este recibe una distancia menor a 10cm, para los motores y se sale del ciclo.
    - Si se recibió un 1, prende los sensores derechos y manda llamar la función CorrigoDir
    - Si se recibió un 2, prende los sensores izquierdos y manda llamar la función CorrigoDir
- **CorrigoDir**: Recibe dos distancias y el numero indicando que pared se esta siguiendo.
    - Si la diferencia de las distancias es mayor a 2cm, se paran los motores. Seguido se rota el robot los grados necesarios para que las distancias sean las mismas por medio de obtención del ángulo.
      *rotI(atan((dis4-dis2))*180/pi) ó rotD(atan((dis2-dis4))*180/pi)*
    - Luego se revisa si la distancia de ambos es mayor a 8cm o menor a 2cm indicando que se encuentra muy cerca o muy lejos de la pared. Si así es, prende el sensor del lado indicado por el numero que recibió y manda llamar la función PosRobot
- **PosRobot**: Revisa si la distancia de los sensores es muy grande o muy chica y dependiendo de eso gira el robot los grados necesarios por medio de obtención del ángulo, luego avanza 10 cm y vuelve a girar el robot para quedar paralelo a la pared a cierta distancia de ella.

Algoritmo:

Case 0:

· El robot avanza 38 centímetros hacia delante fuera de su zona de inicio

Case 1:

· El robot gira 90 grados hacia la izquierda

Case 2:

· Dentro de un ciclo, prende el sensor delanteros y avanza. Sale del ciclo cuando el sensor detecta a cierta distancia la pared de adelante.

Case 3:

· Gira el robot 90 grados hacia la derecha y avanza 38 centímetros.

Case 4:

· Prende los motores para avanzar
· Entra a un ciclo
· Llama la función seguirPared(1)

Case 5:

· Se baja el gripper y se abre la puerta.

Case 6:

- Avanza 10 cm para tomar la victima

Case 7:

- Cierra la puerta

Case 8:

- Sube el gripper

Case 9:

- Rota 180 grados

Case 10:

- Prende los motores para avanzar
- Entra a un ciclo
- Prende los sensores izquierdos
- Si un sensor izquierdo detecta una distancia mayor a 10cm, se para y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir(2)
- Entra a otro ciclo
- Llama la función seguirPared(1)

Case 11:

- Abre el gripper para soltar la victima

Case 12:

- Retrocede 15 cm hacia atrás
- Rota 180 grados

Case 13:

- Prende los motores para avanzar
- Entra a un ciclo
- Dentro del ciclo: Prende los sensores izquierdos
- Cuando el sensor izquierdo detecta una distancia mayor a 10cm, para los motores y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir()

Case 14:

- Rota a la izquierda 90 grados
- Avanza 35 cm
- Rota a la izquierda 90 grados

Case 15:

- Prende los motores
- Entra a un ciclo
- Manda llamar la función seguirPared(1)

Case 16:

- Se baja el gripper y se abre la puerta.

Case 17:

- Avanza 10 cm para tomar la victima

Case 18:

- Cierra la puerta

Case 19:

- Sube el gripper

Case 20:

- Rota 180 grados

Case 21:

- Prende los motores para avanzar
- Entra a un ciclo
- Dentro del ciclo: Prende los sensores izquierdos
- Cuando el sensor izquierdo detecta una distancia mayor a 10cm, para los motores y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir()

Case 22:

- Rota 90 grados hacia la derecha
- Avanza 35cm hacia delante
- Rota 90 grados hacia la derecha

Case 23:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 24:

- Abre el gripper para soltar la victima

Case 25:

- Retrocede 15 cm hacia atrás
- Rota 180 grados

Case 26:

- Prende los motores para avanzar
- Entra a un ciclo
- Dentro del ciclo: Prende los sensores izquierdos
- Cuando el sensor izquierdo detecta una distancia mayor a 10cm, para los motores y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir()

Case 27:

- Rota a la izquierda 90 grados
- Avanza 70cm
- Rota a la izquierda 90 grados

Case 28:

- Prende los motores
- Entra a un ciclo
- Manda llamar la función seguirPared(1)

Case 29:

- Se baja el gripper y se abre la puerta.

Case 30:

- Avanza 10 cm para tomar la victima

Case 31:

- Cierra la puerta

Case 32:

- Sube el gripper

Case 33:

- Rota 180 grados

Case 34:

- Prende los motores

- Entra a un ciclo
- Prende los sensores derechos
- Si la distancia derecha es mayor a 15cm, para los motores y se sale del ciclo
- Manda llamar la función CorrigoDir

Case 35:

- Rota a la derecha 90 grados
- Avanza 70cm
- Rota a la derecha 90 grados

Case 36:

- Prende los motores
- Entra a un ciclo
- Manda llamar la función seguirPared(1)

Case 37:

- Abre el gripper para soltar la victima

Case 38:

- Retrocede 15 cm hacia atrás
- Rota 180 grados

Case 39:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 40:

- Rota 180 grados a la derecha

Case 41:

- Prende los motores
- Entra a un ciclo
- Prende los sensores derechos
- Si la distancia derecha es mayor a 15cm, para los motores y se sale del ciclo
- Manda llamar la función CorrigoDir

Case 42:

- Rota a la derecha 90 grados
- Avanza 70cm
- Rota a la derecha 90 grados

Case 43:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 44:

- Rota a la izquierda 90 grados

Case 45:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 46:

- Se baja el gripper y se abre la puerta.

Case 47:

- Avanza 10 cm para tomar la victima

Case 48:

- Cierra la puerta

Case 49:

- Sube el gripper

Case 50:

- Rota 180 grados

Case 51:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 52:

- Rota a la derecha 90 grados
- Avanza 35cm
- Rota a la izquierda 90 grados

Case 53:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(3)

Case 54:

- Rota a la derecha 90 grados

Case 55:

- Prende los motores para avanzar
- Entra a un ciclo
- Prende los sensores izquierdos
- Si un sensor izquierdo detecta una distancia mayor a 10cm, se para y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir()
- Prende los motores para avanzar
- Entra a otro ciclo
- Llama la función seguirPared(1)

Case 56:

- Abre el gripper para soltar la victima

Case 57:

- Retrocede 15 cm hacia atrás
- Rota 180 grados

Case 58:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 59:

- Rota a la derecha 90 grados

Case 60:

- Prende los motores
- Entra a un ciclo
- Prende los sensores derechos
- Si la distancia derecha es mayor a 15cm, para los motores y se sale del ciclo

- Manda llamar la función CorrigoDir

Case 61:

- Rota a la derecha 90 grados
- Avanza 70cm
- Rota a la derecha 90 grados

Case 62:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 63:

- Rota a la izquierda 90 grados

Case 64:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 65:

- Rota a la izquierda 90 grados

Case 66:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 67:

- Se baja el gripper y se abre la puerta.

Case 68:

- Avanza 10 cm para tomar la victima

Case 69:

- Cierra la puerta

Case 70:

- Sube el gripper

Case 71:

- Rota 180 grados

Case 72:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(2)

Case 73:

- Rota a la derecha 90 grados

Case 74:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(2)

Case 75:

- Rota a la derecha 90 grados
- Avanza 35cm
- Rota a la izquierda 90 grados

Case 76:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(3)

Case 77:
- Rota a la derecha 90 grados

Case 78:
- Prende los motores para avanzar
- Entra a un ciclo
- Prende los sensores izquierdos
- Si un sensor izquierdo detecta una distancia mayor a 10cm, se para y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir()
- Prende los motores para avanzar
- Entra a otro ciclo
- Llama la función seguirPared(1)

Case 79:
- Abre el gripper para soltar la victima

Case 80:
- Retrocede 15 cm hacia atrás
- Rota 180 grados

Case 81:
- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 82:
- Rota a la derecha 90 grados

Case 83:
- Prende los motores
- Entra a un ciclo
- Prende los sensores derechos
- Si la distancia derecha es mayor a 15cm, para los motores y se sale del ciclo
- Manda llamar la función CorrigoDir

Case 84:
- Rota a la derecha 90 grados
- Avanza 70cm
- Rota a la derecha 90 grados

Case 85:
- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 86:
- Rota a la izquierda 90 grados

Case 87:
- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 88:

- Rota a la izquierda 90 grados

Case 89:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 90:

- Rota a la izquierda 90 grados

Case 91:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(1)

Case 92:

- Se baja el gripper y se abre la puerta.

Case 93:

- Avanza 10 cm para tomar la victima

Case 94:

- Cierra la puerta

Case 95:

- Sube el gripper

Case 96:

- Rota 180 grados

Case 97:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(2)

Case 98:

- Rota a la derecha 90 grados

Case 99:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(2)

Case 100:

- Rota a la derecha 90 grados

Case 101:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(2)

Case 102:

- Rota a la derecha 90 grados
- Avanza 35cm
- Rota a la izquierda 90 grados

Case 103:

- Prende los motores para avanzar
- Entra a un ciclo
- Llama la función seguirPared(3)

Case 104:

- Rota a la derecha 90 grados

Case 105:

- Prende los motores para avanzar
- Entra a un ciclo
- Prende los sensores izquierdos
- Si un sensor izquierdo detecta una distancia mayor a 10cm, se para y se sale del ciclo
- Dentro del ciclo: Manda llamar la función CorrigoDir()
- Prende los motores para avanzar
- Entra a otro ciclo
- Llama la función seguirPared(1)

Case 106:

- Abre el gripper para soltar la victima

Case 107:

- Retrocede 15 cm hacia atrás
- Rota 180 grados

Case 108:

- Prende los motores
- Entra a un ciclo
- Prende los sensores derechos
- Si la distancia derecha es mayor a 15cm, para los motores y se sale del ciclo
- Manda llamar la función CorrigoDir

Case 109:

- Rota a la derecha 90 grados
- Avanza 35cm
- Rota a la derecha 90 grados

Case 110:

- Dentro de un ciclo, prende el sensor delanteros y avanza. Sale del ciclo cuando el sensor detecta a cierta distancia la pared de adelante.

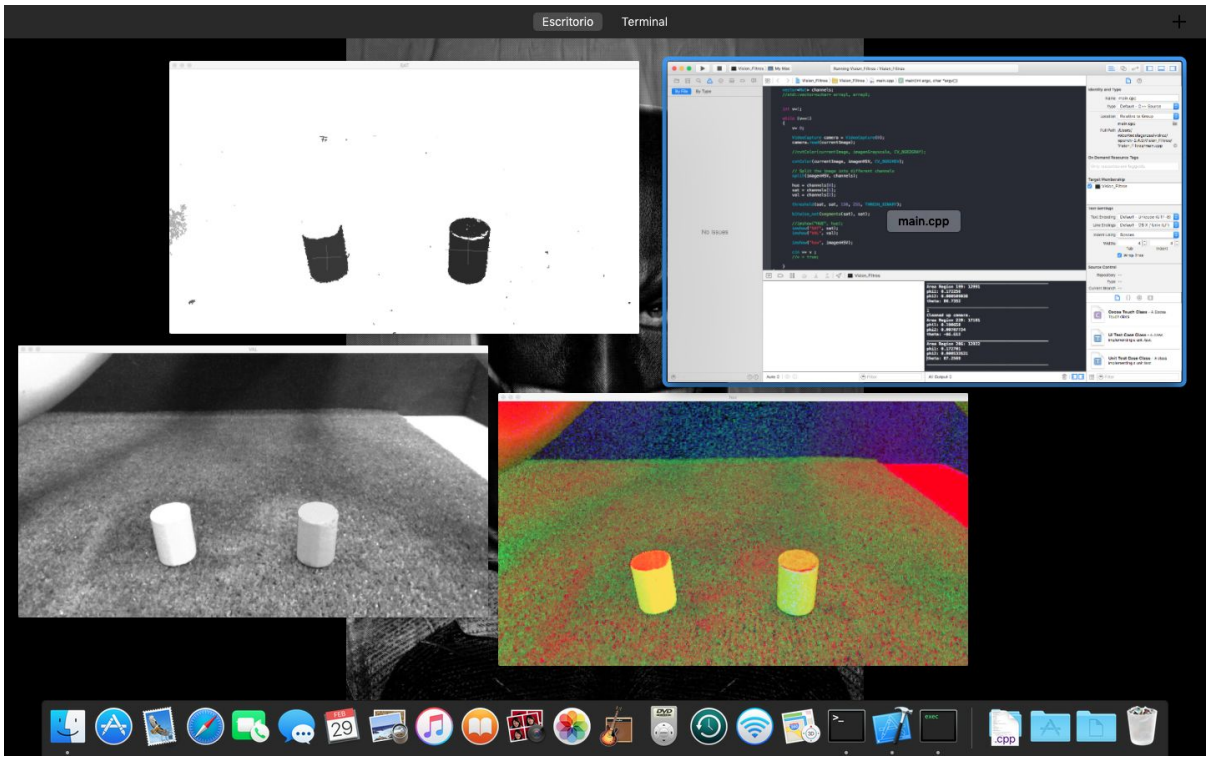
Visión:

El algoritmo de visión comienza con la adquisición de imágenes cada vez que se requiera dentro de un ciclo infinito. Después de adquirir la imagen, se hace una copia de esta y se le aplica un filtro HSV. Después de la aplicación del filtro, se separan los canales de la imagen (Hue, Saturation, Value) con la finalidad de obtener los valores de saturación de color y aplicar posteriormente un filtro de binarización, referenciado con el valor de la saturación adquirida.

Al tener la imagen binarizada, se detectan las regiones de interés (mostradas en negro) con la finalidad de localizar el objeto dentro de la imagen. A esto se le llama segmentación. Después de localizar el objeto dentro de la imagen, esta es explorada para sacar sus dimensiones (medidas en pixeles) como área, centro del objeto, momentos estadísticos y centralizados y su inclinación. En base a esta información se puede caracterizar el objeto, siendo identificado como una víctima al cumplir con un rango establecido según los parámetros anteriormente descritos. Aquí se toma referencia del centro de la imagen si la victim encontrada se localiza a la derecha o izquierda del centro, mandando señales por el GPIO (22 y 23) de la Raspberry.

Recordando la copia hecha de la imagen original, esta se usa para mapear el centro del objeto obtenido en la imagen binarizada, teniendo el punto deseado en un pixel de color y asi descubrir mediante un algoritmo que identifica por canales RGB el color de la victima y asi dar aviso a mediante otro par de señales en la tarjeta Raspberry (GPIO 17 y 27). Al finalizar este análisis, solamente se espera mediante una señal de entrada al GPIO (4) si debe tomar la siguiente fotografía y comenzar de nuevo con el proceso de filtrado y análisis de la imagen.

La declaracion de los GPIO se realiza mediante un biblioteca externa que los inicializa y crea sus objetos para la manipulación dentro del código de C++. Estos están descritos en GPIOClass.

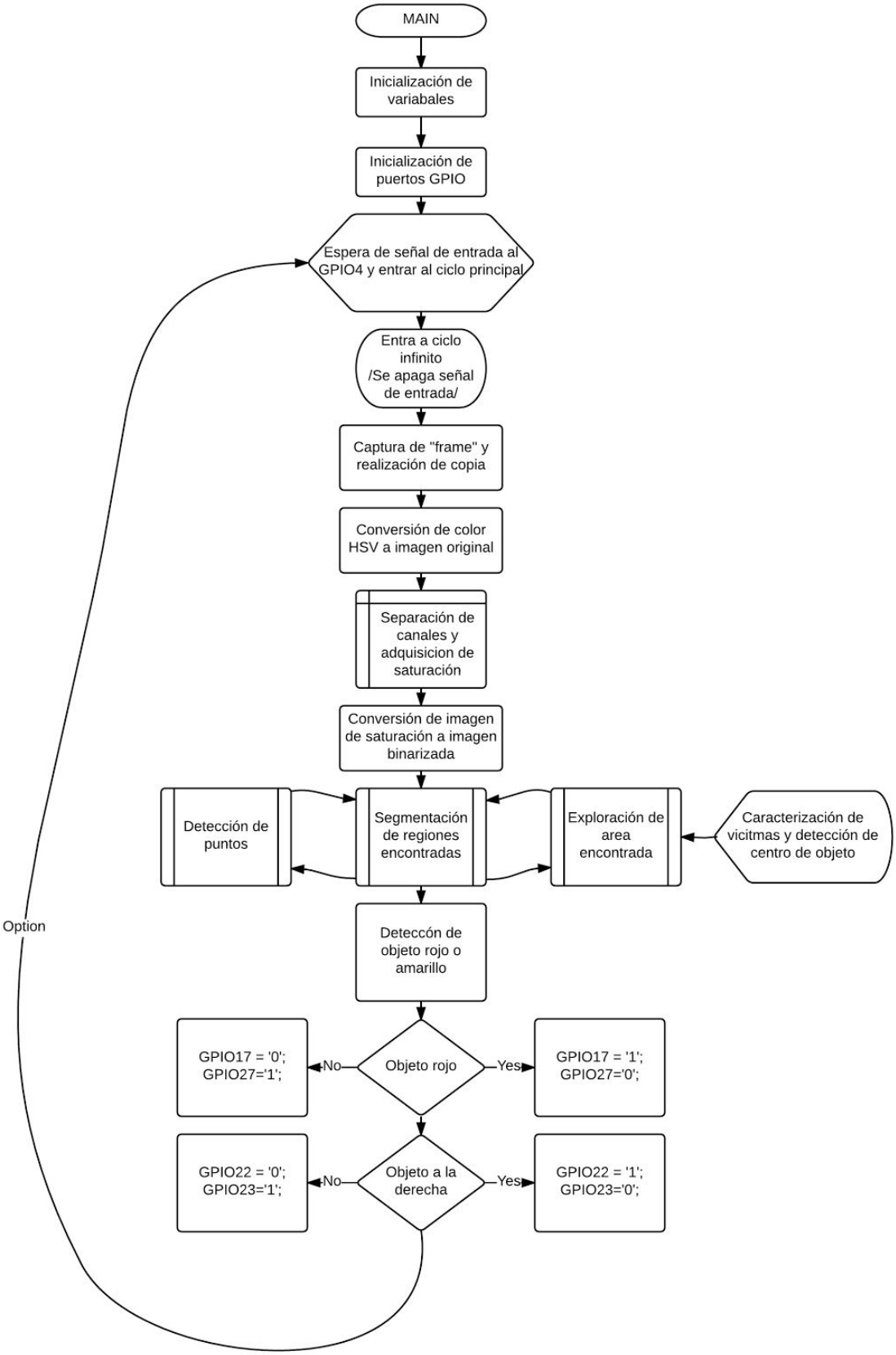


Ambientes de desarrollo de software para visión:

1.- Computadora MacBook Pro 15” (2011) - Almacenamiento SSD 480GB - 8GB RAM.
OS X El Capitan - Xcode 7.3. C++ - OpenCV 2.4.9.

2.- Raspberry Pi Model B - Almacenamiento Tarjeta SD de 8GB - 512MB RAM.
Raspbian Wheezy - Terminal: editor de textos embebido “nano”. C++ - OpenCV 2.4.9.

Diagrama de flujo de código de visión:



Comentarios

Roberto De la Garza: El proyecto sin duda puso a prueba nuestras habilidades para diseñar y construir o implementar código, circuitos eléctricos/electrónicos, y aunque no todo funcionó a la perfección, salimos con el aprendizaje de administrar proyectos y lo importante que es no desfasar lo agendado en actividades y entregas. El profesor fue bastante bueno, proporcionó la asesoría y revisión de manera adecuada, de la misma manera como lo hizo la instructora. Recomiendo ampliamente el curso para ponerse a prueba en aquellas áreas donde uno pudo haber estar débil durante la carrera, como personalmente lo fue la parte de visión.

David Kutugata: Este proyecto representa un gran reto, que aunque fallido, deja mucho aprendizaje. Particularmente lo aprendido es la administración del tiempo y seguimiento de un plan de trabajo establecido, con lo anterior el proyecto pudo haber sido exitoso. Como comentario hacia la clase considero que la información para hacer facturas al nombre de la institución debe ser más accesible. Y finalmente considero que el proyecto requería de al menos dos miembros más en el equipo para sacarlo adelante con mayor fluidez.

Ana: Este proyecto es una muy buena integración de los conocimientos que hemos adquirido durante la carrera. Se me hizo un reto muy interesante y con mucha oportunidad de aprendizaje. Asimismo se concluyó que hace falta varios conocimientos para este proyecto tales como diseño mecánico. Me hubiera gustado contar con más material en el almacén y con más integrantes en el equipo. Creo que este proyecto hubiera tenido un mejor resultado si se hubiera tenido mejor material y más personas que colaboraran.

Anexos

Código del PIC:

```
#include <xc.h>
#include <plib.h>
#include <math.h>

//Configuration
#pragma config POSCMOD = HS, FNOSC = PRIPLL, FPLLMUL = MUL_20 //Oscilador de cristal > 4MHZ,Oscilador primario con PLL,Multiplicador de frecuencia x 20
#pragma config FPLLIDIV = DIV_2, FPLLODIV = DIV_1 // //Divisor de entrada 1:2, Divisor de salida 1:1
#pragma config FSOSCEN = OFF, FPBDIV = DIV_1 //Oscilador externo secundario off,PBCLK (PeripheralBusClock) igual a SYSCLK
#pragma config FWDTEN = OFF, CP = OFF, FCKSM = CSECMD //Watch Dog off, Codigo de Programa off

#define SYS_FREQ        (80000000L)
#define PRESCALER       64
#define T1_TICK         1250 //T1_TICK = (0.001 / (T_PBCLK * PRESCALER)) = 1 ms
#define CORE_TICK       40000 //CORE_TICK= (0.001 / (T_SYS_FREQ * 2)) = 1 ms
#define PWM_FREQ        50

float PWM_PERIOD = (SYS_FREQ / (PWM_FREQ * PRESCALER)) - 1;
float pi = 3.14159265359;
unsigned int offset = 100000;   //100000 para piso
                        //150000 para zacate

unsigned long dis0, dis1, dis2, dis3, dis4, dis5;
unsigned int contServo = 0;

void inicializa () {
   OpenTimer1(T1_ON | T1_SOURCE_INT | T1_PS_1_64, T1_TICK); //Configuracion del Timer1 para generar interrupcion cada 1 ms
   OpenCoreTimer(CORE_TICK); // 1 ms
   OpenOC1(OC_ON | OC_TIMER2_SRC | OC_PWM_FAULT_PIN_DISABLE, 0, 0); //Configuracion de OutputCompare1 para generar salidas PWM
   OpenOC2(OC_ON | OC_TIMER2_SRC | OC_PWM_FAULT_PIN_DISABLE, 0, 0); //Configuracion de OutputCompare2 para generar salidas PWM
   OpenTimer2(T2_ON | T2_PS_1_64, PWM_PERIOD); //Configuracion del Timer2 para la generacion del DutyCycle
   //Configuracion de Niveles de Prioridad
   INTSetVectorPriority(INT_TIMER_1_VECTOR, INT_PRIORITY_LEVEL_2);
   INTSetVectorPriority(INT_CORE_TIMER_VECTOR, INT_PRIORITY_LEVEL_3);
```

```
    //Habilitar interrupciones de Timer1,CoreTimer,Timer2 y Timer3
    INTEnable(INT_T1, INT_ENABLED);
    INTEnable(INT_CT, INT_ENABLED);
    INTEnableSystemMultiVectoredInt();}

void delay (int x) {
    while (x > 0)
        x--;}

void pulseOut (int x){
    switch (x)
    {
        case 0 :            //sensor delantero
            PORTDbits.RD2 = 1;
            delay(100);
            PORTDbits.RD2 = 0;
            delay(100);
            break;

        case 1 :            //sensor izq adelante
            PORTDbits.RD3 = 1;
            delay(100);
            PORTDbits.RD3 = 0;
            delay(100);
            break;

        case 3 :            //sensor izq atras
            PORTDbits.RD4 = 1;
            delay(100);
            PORTDbits.RD4 = 0;
            delay(100);
            break;

        case 2 :            //sensor der adelante
            PORTDbits.RD8 = 1;
            delay(100);
            PORTDbits.RD8 = 0;
            delay(100);
            break;

        case 4 :            //sensor der atras
            PORTDbits.RD9 = 1;
            delay(100);
            PORTDbits.RD9 = 0;
            delay(100);
            break;
```

```
      case 5 :              //sensor trasero
         PORTDbits.RD10 = 1;
         delay(100);
         PORTDbits.RD10 = 0;
         delay(100);
         break;
   }}
unsigned long pulseIn (int x) {
   unsigned long width = 0;

   switch (x)
   {
      case 0 :              //sensor delantero
         // wait for any previous pulse to end
         while (PORTEbits.RE0 == 1)  {}

         // wait for the pulse to start
         while (PORTEbits.RE0 != 1)  {}

         // wait for the pulse to stop
         while (PORTEbits.RE0 == 1)
            width++;

         return width;
         break;

      case 1 :              //sensor izq adelante
         // wait for any previous pulse to end
         while (PORTEbits.RE1 == 1)  {}

         // wait for the pulse to start
         while (PORTEbits.RE1 != 1)  {}

         // wait for the pulse to stop
         while (PORTEbits.RE1 == 1)
            width++;

         return width;
         break;

      case 3 :              //sensor izq atras
         // wait for any previous pulse to end
         while (PORTEbits.RE3 == 1)  {}

         // wait for the pulse to start
```

```
    while (PORTEbits.RE3 != 1)  {}

    // wait for the pulse to stop
    while (PORTEbits.RE3 == 1)
       width++;

    return width;
    break;

case 2 :              //sensor der adelante
    // wait for any previous pulse to end
    while (PORTEbits.RE2 == 1)  {}

    // wait for the pulse to start
    while (PORTEbits.RE2 != 1)  {}

    // wait for the pulse to stop
    while (PORTEbits.RE2 == 1)
       width++;

    return width;
    break;

case 4 :              //sensor der atras
    // wait for any previous pulse to end
    while (PORTEbits.RE4 == 1)  {}

    // wait for the pulse to start
    while (PORTEbits.RE4 != 1)  {}

    // wait for the pulse to stop
    while (PORTEbits.RE4 == 1)
       width++;

    return width;
    break;

case 5 :              //sensor trasero
    // wait for any previous pulse to end
    while (PORTEbits.RE5 == 1)  {}

    // wait for the pulse to start
    while (PORTEbits.RE5 != 1)  {}

    // wait for the pulse to stop
    while (PORTEbits.RE5 == 1)
```

```
            width++;

        return width;
        break;
    }}

///Gripper
void close_Gripper() {
    OC1RS = 300;
}
void open_Gripper(){
    OC1RS = 1100;
}
void down_Gripper(){
    OC2RS = 800;
}
void up_Gripper(){
    OC2RS = 300;
}

void moveFwd (int cm) {
    PORTDbits.RD11 = 1;
    delay(5);
    PORTDbits.RD7 = 1;
    delay (cm*offset);

    PORTDbits.RD11 = 0;
    delay(5);
    PORTDbits.RD7 = 0;
    delay(5);}
void moveBack (int cm) {
    PORTDbits.RD5 = 1;
    delay(5);
    PORTDbits.RD6 = 1;
    delay(cm*offset);

    PORTDbits.RD5 = 0;
    delay(5);
    PORTDbits.RD6 = 0;
    delay(5);}

void rotD(int a){
    double grados = a*pi*8/180;
    PORTDbits.RD11 = 1;
    delay(5);
    PORTDbits.RD6 = 1;
```

```
    delay(grados*offset);
    PORTDbits.RD11 = 0;
    delay(5);
    PORTDbits.RD6 = 0;
    delay(5);} //rotar con decimal
void rotI(int a){
    double grados = a*pi*8/180;
    PORTDbits.RD7 = 1;
    delay(5);
    PORTDbits.RD5 = 1;
    delay(grados*offset);
    PORTDbits.RD7 = 0;
    delay(5);
    PORTDbits.RD5 = 0;
    delay(5);}

void rotClk (int deg) {
    int cm = deg*pi*8/180;
    PORTDbits.RD11 = 1;
    delay(5);
    PORTDbits.RD6 = 1;
    delay(cm*offset);

    PORTDbits.RD11 = 0;
    delay(5);
    PORTDbits.RD6 = 0;
    delay(5);}  //rotar entero
void rotCount (int deg) {
    int cm = deg*pi*8/180;
    PORTDbits.RD7 = 1;
    delay(5);
    PORTDbits.RD5 = 1;
    delay(cm*offset);

    PORTDbits.RD7 = 0;
    delay(5);
    PORTDbits.RD5 = 0;
    delay(5);}

void SeguirPared(int lado);{
    //Prendo sensor delantero
    pulseOut(0);
    dis0 = pulseIn(0);
    delay(10000);
    //Reviso si llegué a la pared para para los motores
    if(dis0 < 1500){
```

```
        PORTDbits.RD11 = 0;
        delay(5);
        PORTDbits.RD7 = 0;
        delay(5);
        break;
    }

    if(lado = 1){
        pulseOut(2);
        dis2 = pulseIn(2);
        delay(10000);
        pulseOut(4);
        dis4 = pulseIn(4);
        delay(10000);
        CorrigoDir(dis2, dis4, lado);
    }
    else if(lado = 2){
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        pulseOut(3);
        dis3 = pulseIn(3);
        delay(10000);
        CorrigoDir(dis1, dis3, lado);
    }
    else{ //no quiero revisar pared, solamente quiero revisar cuando llegue a topar con algo

    }} // 1 pared derecha 2 pared iquierda 3 solo pared enfrente
void CorrigoDir(int x1, int x2, int lado); {
    if(abs(x1-x2)>150){
        PORTDbits.RD11 = 0;
        delay(5);
        PORTDbits.RD7 = 0;
        delay(10000);
        if(x1>x2){
            //Roto a la derecha los grados necesarios para quedar paralelo
            rotD(atan(((x1-x2)/10)*180/pi));
        }
        else if(x2>x1){
            rotI(atan(((x1-x2)/10)*180/pi));
        }
    }
    if(dis2>720||dis2<570){
        if(lado = 1){
            pulseOut(2);
            dis2 = pulseIn(2);
```

```
            delay(10000);
            PosRobot(dis2);
        }
        else if(lado = 2){
            pulseOut(1);
            dis1 = pulseIn(1);
            delay(10000);
            PosRobot(dis1);
        }
    }
    PORTDbits.RD11 = 1;
    delay(5);
    PORTDbits.RD7 = 1;
    delay(10000);}
void PosRobot(int x){
    if(x<670){
        double grado = atan((670-x)/10)*180/pi;
        delay(1000);
        rotI(grado);
        delay(1000);
        moveFwd(10);
        delay(1000);
        rotD(grado);
        delay(1000);
    }
    else if(x>670){
        double grado = atan((x-670)/10)*180/pi;
        delay(1000);
        rotI(grado);
        delay(1000);
        moveFwd(10);
        delay(1000);
        rotD(grado);
        delay(1000);
    }}



main()
{
    TRISD = 0x00;   //Puerto D como salidas
    PORTD = 0x00;   //Inicializa salidas en 0
    TRISE = 0xFF;   //Puerto E como entradas

    OC1RS = 300;        //Inicializa servos en 0
    OC2RS = 300;
```

```
init();
int state = 0;

while (1)
{
   switch (state)
   {
      case 0 :    //avanza 38cm
         moveFwd(38);
         state = 4;
         break;

      case 1 :    //vuelta izq de 90 grados
         rotCount(88, 0);
         state = 2;
         break;

      case 2 :    //avanzar sin seguir pared hasta llegar a la pared de enfrente
         PORTDbits.RD11 = 1;
         delay(100);
         PORTDbits.RD7 = 1;
         delay(100);
         while (1)
         {
            SeguirPared(3);
         }
         state = 3;
         break;

      case 3 :    //vuelta derecha de 90 grados
         rotClk(85, 0);
         moveFwd(30);
         state = 4;
         break;

      case 4 :    //avanzar guiado por la pared derecha solamente
          PORTDbits.RD11 = 1;
          delay(100);
          PORTDbits.RD7 = 1;
          delay(100);
         while (1){
            SeguirPared(1);
         }
         contServo =0;
         state = 5;
```

```
    break;

case 5 :    //abre y baja gripper

    if (contServo >= 1000){
        down_Gripper();
        open_Gripper();
        contServo = 0;
        state = 6;
    }
    break;
case 6 :    //avanza para agarrar gripper
    if (contServo >= 5000){
        moveFwd(10);
        contServo = 0;
        state = 7;
    }
    break;
case 7 :    //agarrar victima
    if (contServo >= 4000){
        close_Gripper();
        contServo = 0;
        state = 8;

    }
    break;
case 8 :    //sube el gripper
    if (contServo >= 4000){
        up_Gripper();
        state = 9;
        contServo = 0;
    }
    break;
case 9 :    //vuelta en 180
    rotClk(185, 0);
    state = 10;
    break;

case 10 :   //seguir pared izq luego pared der
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(3);
        dis3 = pulseIn(3);
```

```
        delay(10000);
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis3 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis1, dis3, 2);
    }
    while(1){
        SeguirPared(1);
    }

    state = 11;
    break;

case 11 :
    if (contServo >= 1000){
        open_Gripper();
        contServo = 0;
        state = 12;
    }
    break;

case 12:
    moveBack(15);
    delay(1000);
    rotClk(180);
    delay(1000);
    state = 13;
    break;

case 13:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(3);
        dis3 = pulseIn(3);
        delay(10000);
```

```
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis3 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis1, dis3, 2);
    }
    state = 14;
    break;

case 14:
    rotCount(88);
    delay(1000);
    moveFwd(35);
    delay(1000);
    rotCount(88);
    delay(1000);
    state = 15;
    break;

case 15:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    state = 16;
    break;

case 16:
    if (contServo >= 1000){
        down_Gripper();
        open_Gripper();
        contServo = 0;
        state = 17;
    }
    break;
case 17 :    //avanza para agarrar gripper
```

```
    if (contServo >= 5000){
        moveFwd(10);
        contServo = 0;
        state = 18;
    }
    break;
case 18 :    //agarrar victima
    if (contServo >= 4000){
        close_Gripper();
        contServo = 0;
        state = 19;

    }
    break;
case 19 :    //sube el gripper
    if (contServo >= 4000){
        up_Gripper();
        contServo = 0;
        state = 20;
    }
    break;
case 20 :    //vuelta en 180
    rotClk(185, 0);
    state = 21;
    break;

case 21:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(3);
        dis3 = pulseIn(3);
        delay(10000);
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis3 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
```

```
        CorrigoDir(dis1, dis3, 2);
    }
    state = 22;
    break;

case 22:
    rotClk(88);
    delay(1000);
    moveFwd(35);
    delay(1000);
    rotClk(88);
    delay(1000);
    state = 23;
    break;

case 23:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 5;
    break;

case 24:
    if (contServo >= 1000){
        open_Gripper();
        contServo = 0;
        state = 25;
    }
    break;

case 25:
    moveBack(15);
    delay(1000);
    rotClk(180);
    delay(1000);
    state = 26;
    break;

case 26:
    PORTDbits.RD11 = 1;
    delay(100);
```

```
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(3);
        dis3 = pulseIn(3);
        delay(10000);
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis3 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis1, dis3, 2);
    }
    state = 27;
    break;

case 27:
    rotCount(88);
    delay(1000);
    moveFwd(70);
    delay(1000);
    rotCount(88);
    delay(1000);
    state = 28;
    break;

case 28:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    state = 29;
    break;

case 29:

    if (contServo >= 1000){
```

```
        down_Gripper();
        open_Gripper();
        contServo = 0;
        state = 30;
    }
    break;
case 30 :    //avanza para agarrar gripper
    if (contServo >= 5000){
        moveFwd(10);
        contServo = 0;
        state = 31;
    }
    break;
case 31 :    //agarrar victima
    if (contServo >= 4000){
        close_Gripper();
        contServo = 0;
        state = 32;

    }
    break;
case 32 :    //sube el gripper
    if (contServo >= 4000){
        up_Gripper();
        state = 33;
        contServo = 0;
    }
    break;
case 33 :    //vuelta en 180
    rotClk(185, 0);
    state = 34;
    break;

case 34:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(4);
        dis4 = pulseIn(4);
        delay(10000);
        pulseOut(2);
        dis2 = pulseIn(2);
        delay(10000);
        //Reviso si pase la pared para parar los motores
```

```
        if(dis4 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis2, dis4, 1);
    }
    state = 35;
    break;

case 35:
    rotClk(88);
    delay(1000);
    moveFwd(70);
    delay(1000);
    rotClk(88);
    delay(1000);
    state = 36;
    break;

case 36:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 37;
    break;

case 37:
    if (contServo >= 1000){
        open_Gripper();
        contServo = 0;
        state = 38;
    }
    break;

case 38:
    moveBack(15);
    delay(1000);
    rotClk(180);
```

```
    delay(1000);
    state = 39;
    break;

case 39:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 40;
    break;

case 40:
    rotClk(185, 0);
    state = 41;
    break;

case 41 :   //seguir pared izq luego pared der
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(4);
        dis4 = pulseIn(4);
        delay(10000);
        pulseOut(2);
        dis2 = pulseIn(2);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis4 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis2, dis4, 1);
    }

    state = 42;
    break;
```

```
case 42:
    rotClk(88);
    delay(1000);
    moveFwd(70);
    delay(1000);
    rotClk(88);
    delay(1000);
    state = 43;
    break;
case 43:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 44;
    break;
case 44:
    rotCount(88);
    delay(1000);
    state = 45;
    break;

case 45:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 46;
    break;

case 46:

    if (contServo >= 1000){
        down_Gripper();
        open_Gripper();
        contServo = 0;
        state = 47;
```

```
    }
    break;
case 47 :    //avanza para agarrar gripper
    if (contServo >= 5000){
        moveFwd(10);
        contServo = 0;
        state = 48;
    }
    break;
case 48 :    //agarrar victima
    if (contServo >= 4000){
        close_Gripper();
        contServo = 0;
        state = 49;

    }
    break;
case 49 :    //sube el gripper
    if (contServo >= 4000){
        up_Gripper();
        state = 50;
        contServo = 0;
    }
    break;
case 50 :    //vuelta en 180
    rotClk(185, 0);
    state = 51;
    break;
case 51:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    contServo =0;
    state = 52;
    break;
case 52:
    rotClk(88);
    delay(1000);
    moveFwd(35);
    delay(1000);
    rotCount(88);
    delay(1000);
```

```
    state = 53;
    break;
case 53:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1)
    {
        SeguirPared(3);
    }
    state = 54;
    break;
case 54:
    rotClk(88);
    delay(1000);
    state = 55;
    break;

case 55:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(3);
        dis3 = pulseIn(3);
        delay(10000);
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis3 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis1, dis3, 2);
    }
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while(1){
```

```
        SeguirPared(1);
    }

    state = 56;
    break;

case 56 :
    if (contServo >= 1000){
        open_Gripper();
        contServo = 0;
        state = 57;
    }
    break;

case 57:
    moveBack(15);
    delay(1000);
    rotClk(180);
    delay(1000);
    state = 58;
    break;
case 58:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 59;
    break;

case 59:
    rotClk(185, 0);
    state = 60;
    break;

case 60 :   //seguir pared izq luego pared der
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(4);
        dis4 = pulseIn(4);
```

```
        delay(10000);
        pulseOut(2);
        dis2 = pulseIn(2);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis4 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis2, dis4, 1);
    }

    state = 61;
    break;

case 61:
    rotClk(88);
    delay(1000);
    moveFwd(70);
    delay(1000);
    rotClk(88);
    delay(1000);
    state = 62;
    break;
case 62:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 63;
    break;
case 63:
    rotCount(88);
    delay(1000);
    state = 64;
    break;

case 64:
    PORTDbits.RD11 = 1;
```

```
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 65;
    break;

case 65:
    rotCount(88);
    delay(1000);
    state = 66;
    break;
case 66:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 67;
    break;
case 67:
    if (contServo >= 1000){
        down_Gripper();
        open_Gripper();
        contServo = 0;
        state = 68;
    }
    break;
case 68 :    //avanza para agarrar gripper
    if (contServo >= 5000){
        moveFwd(10);
        contServo = 0;
        state = 69;
    }
    break;
case 69 :    //agarrar victima
    if (contServo >= 4000){
        close_Gripper();
        contServo = 0;
        state = 70;
```

```
    }
    break;
case 70 :    //sube el gripper
    if (contServo >= 4000){
        up_Gripper();
        state = 71;
        contServo = 0;
    }
    break;
case 71 :    //vuelta en 180
    rotClk(185, 0);
    state = 72;
    break;
case 72:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    contServo =0;
    state = 73;
    break;

case 73:
    rotClk(88);
    delay(1000);
    state = 74;
    break;
case 74:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    contServo =0;
    state = 75;
    break;
case 75:
    rotClk(88);
    delay(1000);
    moveFwd(35);
```

```
    delay(1000);
    rotCount(88);
    delay(1000);
    state = 76;
    break;
case 76:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1)
    {
        SeguirPared(3);
    }
    state = 77;
    break;
case 77:
    rotClk(88);
    delay(1000);
    state = 78;
    break;

case 78:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(3);
        dis3 = pulseIn(3);
        delay(10000);
        pulseOut(1);
        dis1 = pulseIn(1);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis3 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis1, dis3, 2);
    }
    PORTDbits.RD11 = 1;
    delay(100);
```

```
    PORTDbits.RD7 = 1;
    delay(100);
    while(1){
        SeguirPared(1);
    }

    state = 79;
    break;

case 79 :
    if (contServo >= 1000){
        open_Gripper();
        contServo = 0;
        state = 80;
    }
    break;

case 80:
    moveBack(15);
    delay(1000);
    rotClk(180);
    delay(1000);
    state = 81;
    break;
case 81:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 82;
    break;

case 82:
    rotClk(185, 0);
    state = 83;
    break;

case 83 :   //seguir pared izq luego pared der
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
```

```
    while (1){
      pulseOut(4);
      dis4 = pulseIn(4);
      delay(10000);
      pulseOut(2);
      dis2 = pulseIn(2);
      delay(10000);
      //Reviso si pase la pared para parar los motores
      if(dis4 > 1826){
        PORTDbits.RD11 = 0;
        delay(5);
        PORTDbits.RD7 = 0;
        delay(5);
        break;
      }
      CorrigoDir(dis2, dis4, 1);
    }

    state = 84;
    break;

  case 84:
    rotClk(88);
    delay(1000);
    moveFwd(70);
    delay(1000);
    rotClk(88);
    delay(1000);
    state = 85;
    break;
  case 85:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
      SeguirPared(1);
    }
    contServo =0;
    state = 86;
    break;
  case 86:
    rotCount(88);
    delay(1000);
    state = 87;
    break;
```

```
case 87:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 88;
    break;

case 88:
    rotCount(88);
    delay(1000);
    state = 89;
    break;
case 89:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 90;
    break;
case 90:
    rotCount(88);
    delay(1000);
    state = 91;
    break;
case 91:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(1);
    }
    contServo =0;
    state = 92;
    break;
case 92:
```

```
    if (contServo >= 1000){
        down_Gripper();
        open_Gripper();
        contServo = 0;
        state = 93;
    }
    break;
case 93 :    //avanza para agarrar gripper
    if (contServo >= 5000){
        moveFwd(10);
        contServo = 0;
        state = 94;
    }
    break;
case 94 :    //agarrar victima
    if (contServo >= 4000){
        close_Gripper();
        contServo = 0;
        state = 95;

    }
    break;
case 95 :    //sube el gripper
    if (contServo >= 4000){
        up_Gripper();
        state = 96;
        contServo = 0;
    }
    break;
case 96 :    //vuelta en 180
    rotClk(185, 0);
    state = 97;
    break;
case 97:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    contServo =0;
    state = 98;
    break;
case 98:
    rotClk(88);
```

```
    delay(1000);
    state = 99;
    break;
case 99:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    contServo =0;
    state = 100;
    break;

case 100:
    rotClk(88);
    delay(1000);
    state = 101;
    break;
case 101:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        SeguirPared(2);
    }
    contServo =0;
    state = 102;
    break;
case 102:
    rotClk(88);
    delay(1000);
    moveFwd(35);
    delay(1000);
    rotCount(88);
    delay(1000);
    state = 103;
    break;
case 103:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1)
```

```
    {
      SeguirPared(3);
    }
    state = 104;
    break;
  case 104:
    rotClk(88);
    delay(1000);
    state = 105;
    break;

  case 105:
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
      pulseOut(3);
      dis3 = pulseIn(3);
      delay(10000);
      pulseOut(1);
      dis1 = pulseIn(1);
      delay(10000);
      //Reviso si pase la pared para parar los motores
      if(dis3 > 1826){
        PORTDbits.RD11 = 0;
        delay(5);
        PORTDbits.RD7 = 0;
        delay(5);
        break;
      }
      CorrigoDir(dis1, dis3, 2);
    }
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while(1){
      SeguirPared(1);
    }

    state = 106;
    break;

  case 106 :
    if (contServo >= 1000){
```

```
        open_Gripper();
        contServo = 0;
        state = 106;
    }
    break;

case 107:
    moveBack(15);
    delay(1000);
    rotClk(180);
    delay(1000);
    state = 108;
    break;
case 108 :   //seguir pared izq luego pared der
    PORTDbits.RD11 = 1;
    delay(100);
    PORTDbits.RD7 = 1;
    delay(100);
    while (1){
        pulseOut(4);
        dis4 = pulseIn(4);
        delay(10000);
        pulseOut(2);
        dis2 = pulseIn(2);
        delay(10000);
        //Reviso si pase la pared para parar los motores
        if(dis4 > 1826){
            PORTDbits.RD11 = 0;
            delay(5);
            PORTDbits.RD7 = 0;
            delay(5);
            break;
        }
        CorrigoDir(dis2, dis4, 1);
    }

    state = 109;
    break;
case 109:
    rotClk(88);
    delay(1000);
    moveFwd(35);
    delay(1000);
    rotClk(88);
    delay(1000);
    state = 110;
```

```
            break;
        case 110 :    //avanzar sin seguir pared hasta llegar a la pared de enfrente
            PORTDbits.RD11 = 1;
            delay(100);
            PORTDbits.RD7 = 1;
            delay(100);
            while (1)
            {
               SeguirPared(3);
            }
            state = 999;
            break;


        default:
            state = 999;
            break;
      }
   }
}
/**/
void __ISR(_TIMER_1_VECTOR, ipl2) Timer1Handler(void)
{
   INTDisableInterrupts();        //Disable interrupts
   mT1ClearIntFlag();             //Limpiar la bandera de interrupcion

   INTEnableInterrupts();         //Enables interrupts again
}
/**/

void __ISR(_CORE_TIMER_VECTOR, ipl3) CoreTimerHandler(void)
{
   INTDisableInterrupts();        //Disable interrupts
   mCTClearIntFlag();             //Limpiar la bandera de interrupcion
   UpdateCoreTimer(CORE_TICK);    // Reinicia el CoreTimer
   contServo++;

   if (contServo == 20000){
      contServo = 0;
   }
   INTEnableInterrupts();         //Enables interrupts again
}
```

Código de Visión:

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <math.h>
#include <queue>
#include <time.h>
#include <iostream>
#include <exception>
#include <opencv2/GPIOClass.h>
#include <opencv2/GPIOClass.cpp>

using namespace std;
using namespace cv;

RNG rng(12345);
#define PI 3.14159265;
int cont = 1;

int Xcentrada, Xobjeto, Yobjeto;

bool victima, vr, va, d, i;

#define Trian1_min     0.170616
#define Trian1_max    0.217556
#define Trian2_min     0.000284645
#define Trian2_max    0.00782807

#define oTrian1_min  0.173794
#define oTrian1_max  0.195688
#define oTrian2_min  0.0010709
#define oTrian2_max  0.00787981

vector<Point> points;

int Px;
int Py;

//-------------------------------------------------------------------------------------------
//////////////////////////////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------------------
CvPoint detecta(Mat imagen)
{
        int x = rand() % imagen.cols;
```

```
	int y = rand() % imagen.rows;

	return cvPoint(x, y);
}

//--------------------------------------------------------------------------------------------
////////////////////////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------------------------------------------
void explora(CvPoint p, Mat img, Mat regiones, int regionesActual)
{
	CvPoint actual = p;
	regiones.at<uchar>(actual) = regionesActual;

	int x = actual.x;
	int y = actual.y;
	queue<CvPoint> lista;
	lista.push(p);

	double area = 0;

	//momentos estadisticos
	double m10 = 0;
	double m01 = 0;

	double m11 = 0;
	double m20 = 0;
	double m02 = 0;

	//momentos centralizados
	double u20 = 0;
	double u02 = 0;
	double u11 = 0;

	double up20 = 0;
	double up02 = 0;
	double up11 = 0;

	double thetaInc = 0;
	double phi1 = 0;
	double phi2 = 0;

	while (!lista.empty())
	{
	area++;
	actual = lista.front();
	lista.pop();
```

```
// Inicia expansion
x = actual.x;
y = actual.y;
//Momentos estadisticos primer orden
m10 += x;
m01 += y;
//momentos estadisticos segundo orden
m11 += x*y;
m20 += x*x;
m02 += y*y;
//arriba, izquierda, abajo, derecha
Point v[4] = { Point(x, y + 1), Point(x - 1, y), Point(x, y - 1), Point(x + 1, y) };

for (int i = 0; i < 4; i++)
{
if (v[i].x < 0)
v[i].x = 0;
if (v[i].x >= regiones.cols)
v[i].x = regiones.cols - 1;

if (v[i].y < 0)
v[i].y = 0;
if (v[i].y >= regiones.rows)
v[i].y = regiones.rows - 1;

int v1;

v1 = img.at<uchar>(v[i]);
if (v1 == 255 && regiones.at<uchar>(v[i]) == 0)
{
        regiones.at<uchar>(v[i]) = regionesActual;
        lista.push(v[i]);
}
}
}

//centroides
double xMedia = m10 / area;
double yMedia = m01 / area;
Point centro = Point(xMedia, yMedia);


//Momentos centralizados
u11 = m11 - yMedia*m10;
u20 = m20 - xMedia*m10;
u02 = m02 - yMedia*m01;
```

```
        //Momentos primo centralizados
        up11 = u11 / area;
        up20 = u20 / area;
        up02 = u02 / area;

    //Momentos invariantes
        double n20 = u20 / pow(area, 2);
        double n02 = u02 / pow(area, 2);
        double n11 = u11 / pow(area, 2);

        phi1 = n20 + n02;
        phi2 = pow((n20 - n02), 2) + (4 * pow(n11, 2));

        thetaInc = 0.5 * atan2((2.0*u11), (u20 - u02));
        double theta = thetaInc * (180 / 3.14159265);

        //----------------------Deteccion de Victimas---------------------------

        if(area < (61500) && (area <(35200) && area > (5000)))
        {
        cout << "Area Region " << regionesActual << ": " << area << endl;
        cout << "phi1: " << phi1 << endl;
        cout << "phi2: " << phi2 << endl;
        cout << "theta: " << theta << endl;
        cout << "_____________________________________________________________" << endl;

        double minMarg = 0.90;
        double maxMarg = 1.10;

        if (phi1 > Trian1_min*minMarg && phi1 < Trian1_max*maxMarg && phi2 >
Trian2_min*minMarg && phi2 < Trian2_max*maxMarg)
        {
        cout << "¡¡VICTIMA!!" << endl;
        cout << "Centro de victima: " << centro << endl;
        victima = true;

        //X y Y del objeto
        Xobjeto = centro.x;
        Yobjeto = centro.y;

        // Derecha o Izquierda segun el objeto con respecto a el centro de la imagen
        if(Xcentrada < Xobjeto)
        {
           cout<<"Derecha"<<endl;
                d = true;
```

```
                i = false;
        }
        else
        if(Xcentrada > Xobjeto)
        {
                cout<<"Izquierda"<<endl;
                i = true;
                d = false;
        }

        else if (Xcentrada == Xobjeto)
        cout<<"Centro"<<endl;

        cout << "_____" << endl;
        }

        else victima = false;

        int val = sqrt(u20)/100;
        int val2 = sqrt(u02)/100;
        line(regiones, (centro - Point(val*cos(thetaInc), val*sin(thetaInc))),
        (centro + Point(val*cos(thetaInc), val*sin(thetaInc))), 127, 1);

        line(regiones, (centro - Point(val2*cos(1.570796327 + thetaInc),
val2*sin(1.570796327 + thetaInc))),
        (centro + Point(val2*cos(1.570796327 + thetaInc), val2*sin(1.570796327 +
thetaInc))), 127, 1);

        }
   //-------------------------------------------------------------------------------------------
        //-------------------Deteccion de obstaculos-------------------------------------
        if(area > (61500))
        {
        cout << "Area Region " << regionesActual << ": " << area << endl;
        cout << "phi1: " << phi1 << endl;
        cout << "phi2: " << phi2 << endl;
        cout << "theta: " << theta << endl;

        double minMarg = 0.90;
        double maxMarg = 1.10;

        if (phi1 > oTrian1_min*minMarg && phi1 < oTrian1_max*maxMarg && phi2 >
oTrian2_min*minMarg && phi2 < oTrian2_max*maxMarg)
        {
        cout << "¡¡OBSTACULO!!" << endl;
        cout << "Centro de obstaculo: " << centro << endl;
```

```
        victima = true;
                //X y Y del objeto
        Xobjeto = centro.x;
        Yobjeto = centro.y;

        // Derecha o Izquierda segun el objeto con respecto a el centro de la imagen
        if(Xcentrada < Xobjeto)
        {
           cout<<"Derecha"<<endl;
                d = true;
                i = false;
        }
        else
        if(Xcentrada > Xobjeto)
        {
           cout<<"Izquierda"<<endl;
                i = true;
          d = false;
        }

        else if (Xcentrada == Xobjeto)
        cout<<"Centro"<<endl;

        cout << "_____________________________________________________________" << endl;

        }

        int val = sqrt(u20)/100;
        int val2 = sqrt(u02)/100;
        line(regiones, (centro - Point(val*cos(thetaInc), val*sin(thetaInc))),
        (centro + Point(val*cos(thetaInc), val*sin(thetaInc))), 127, 1);

        line(regiones, (centro - Point(val2*cos(1.570796327 + thetaInc),
val2*sin(1.570796327 + thetaInc))),
        (centro + Point(val2*cos(1.570796327 + thetaInc), val2*sin(1.570796327 +
thetaInc))), 127, 1);

        }
//-------------------------------------------------------------------------------
   cont++;
}
//-------------------------------------------------------------------------------
////////////////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------
Mat segmenta(Mat imgOriginal)
{
```

```
	Mat regiones = Mat::zeros(imgOriginal.rows, imgOriginal.cols, CV_8UC1);


	if (imgOriginal.channels() != 1)
	{
	cout << "Error: La imagen no est· en blanco y negro" << endl;
	return regiones;
	}

	int regionesActual = 255;
	int pixelesImagen = imgOriginal.rows*imgOriginal.cols;

	//Calculo de centro de la imagen en X
	Xcentrada = imgOriginal.cols/2;


	for (int i = 0; i < pixelesImagen*0.5; i++)
	{

	if (regionesActual < 0)
	break;

	CvPoint p = detecta(imgOriginal);

	int valor = 255;

	valor = imgOriginal.at<uchar>(p);

	  if (valor == 255 && regiones.at<uchar>(p) == 0)
	{

	explora(p, imgOriginal, regiones, regionesActual);
	regionesActual -= 7;
	}
	}

	return regiones;
}
//-------------------------------------------------------------------------------------------
////////////////////////////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------------------

/*Mat eliminacion(Mat src){
```

```
//---------- ELIMINACION DE COLOES ESPECIFICOS ----------

	for (int i = 0; i < src.rows; i++)
	{
	for(int k=0; k<src.cols; k++)
	{
	CvPoint p = detecta(src);

	Vec3b intensity = src.at<Vec3b>(p);
	float blue = intensity.val[0];
	float green = intensity.val[1];
	float red = intensity.val[2];

	//eliminacion de verdes

	if(green > 160)
	{
	src.at<Vec3b> (p) [0] = 0;
	src.at<Vec3b> (p) [1] = 0;
	src.at<Vec3b> (p) [2] = 0;
	}

	}
	}

	return src;
}*/
//-------------------------------------------------------------------------------------------
////////////////////////////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------------------

void identificaColor(Mat vrojo){

	for (int i = 0; i < vrojo.rows; i++)
	{
	for(int k=0; k<vrojo.cols; k++)
	{
	CvPoint p = detecta(vrojo);

	if((p.x == Xobjeto)&&(p.y == Yobjeto)&&(Xobjeto != 0)&&(Yobjeto!=0))
	{
		cout<<"Mismo punto! "<< p.x <<" "<<p.y <<" | "<<Xobjeto<<"
"<<Yobjeto<<endl;

		Vec3b intensity = vrojo.at<Vec3b>(p);
```

```
                    float blue = intensity.val[0];
                   float green = intensity.val[1];
                    float red = intensity.val[2];
                    cout<<"B : "<<blue<<endl;
                    cout<<"G : "<<green<<endl;
                    cout<<"R : "<<red<<endl;

                    //Deteccion de rojo, amarillo o azul
-------------------------------------------------------------

                    if(blue < 50 && green < 50)
                    {
                    if((red > 75 && red < 230) && victima)
                    {
                     cout<<"Color de Victima: ROJO" << endl;
                    vr = true;
                    va = false;
                    }
                    }

                    else
                    if(blue < 45 && green > 50)
                    {
                    if((red < 195 && red > 100) && victima)
                    {
                    cout<<"Color de Victima: AMARILLO" << endl;
                    va = true;
                    vr = false;
                       }

                    }

          }
          }
          }

}
//-------------------------------------------------------------------------------------------------
////////////////////////////////////////////////////////////////////////////////////////////////
//-------------------------------------------------------------------------------------------------

int main(int argc, char *argv[])
{
          Mat src, vrojo, imagenThreshold, imagenHSV, sat, hue, val;
          vector<Mat> channels;
```

```
        string inputstate;

        GPIOClass* gpio17 = new GPIOClass("17"); //create new GPIO object to be attached
to  GPIO17
        GPIOClass* gpio4 = new GPIOClass("4"); //create new GPIO object to be attached to
GPIO4
        GPIOClass* gpio27 = new GPIOClass("21"); //create new GPIO object to be attached
to  GPIO21
        GPIOClass* gpio22 = new GPIOClass("22"); //create new GPIO object to be attached
to  GPIO22
        GPIOClass* gpio23 = new GPIOClass("23"); //create new GPIO object to be attached
to  GPIO23

        gpio17->export_gpio(); //export GPIO17
        gpio4->export_gpio(); //export GPIO4
        gpio27->export_gpio(); //export GPIO21
        gpio22->export_gpio(); //export GPIO22
        gpio23->export_gpio(); //export GPIO23

        gpio17->setdir_gpio("out"); //GPIO17 set to output
        gpio4->setdir_gpio("in"); // GPIO4 set to input
        gpio27->setdir_gpio("out"); // GPIO21 set to output
        gpio22->setdir_gpio("out"); // GPIO22 set to output
        gpio23->setdir_gpio("out"); // GPIO23 set to output


        gpio4->getval_gpio(inputstate);

        //int v=1;

   while (inputstate == "1") //v==1
        {
        //v= 0;
        usleep(50000);
        gpio4->setval_gpio("0");
        usleep(50000);

        VideoCapture camera = VideoCapture(0);
        camera.read(src);

        vrojo = src;

        int pixelesImagen = src.rows*src.cols;
        cout<<"Pixeles totales: "<<pixelesImagen<<endl;
        cout<<"COLS: "<<src.cols<<endl;
        cout<<"ROWS: "<<src.rows<<endl;
```

```
        cout<<"-----------------------------------"<<endl;

        //eliminacion(src);

           cvtColor(src, imagenHSV, CV_BGR2HSV);

           split(imagenHSV, channels);

           hue = channels[0];
           sat = channels[1];
           val = channels[2];

           threshold(sat, sat, 150, 255, THRESH_BINARY);   // 150 -> th

           bitwise_not(segmenta(sat),sat);

           identificaColor(vrojo);
           identificaColor(vrojo);

           if(d)
           {
              gpio22->setval_gpio("1");
          gpio23->setval_gpio("0");
           }
           if(i)
           {
              gpio22->setval_gpio("0");
          gpio23->setval_gpio("1");
           }
           if(vr)
           {
          gpio17->setval_gpio("1");
          gpio27->setval_gpio("0");
           }
           if(va)
           {
          gpio17->setval_gpio("0");
          gpio27->setval_gpio("1");
           }
        //cin >> v ;
           gpio4->getval_gpio(inputstate);
           usleep(50000);

           }
}
```