Deal AI: Robot Croupier

Emil Wilmer Carvajal Viteri

Resumen— Los robots están cada vez más presentes en el día a día para facilitar comodidad al usuario realizando tareas que permiten ahorrar tiempo y esfuerzo. Actualmente existen muchos tipos de robots para todo tipo de aplicaciones. La inteligencia artificial juega cada vez un papel más importante en la creación de nuevas aplicaciones.
En este trabajo se ha creado un robot que actúa como *Croupier* o *Dealer* en el juego del póquer, concretamente la modalidad de Texas Holdem. Deal AI (*Dealer with Artificial Intelligence*) es un robot que permite preparar el inicio de la partida repartiendo cartas y fichas a los jugadores. Después comunica el turno del jugador actual y sigue en todo momento el estado de la partida. Se le pueden hacer preguntas sobre el estado de la partida y respecto las reglas de juego. También ofrece cambios de fichas si los jugadores lo solicitan. Finalmente comunica el ganador cuando la ronda ha terminado y los jugadores han mostrado sus cartas.

Palabras clave— Robot, Deal AI, brazo mecánico, scara, póquer, fichas, cartas *croupier*, *dealer*, inteligencia artificial, single board, Computer Arduino Atmel, YOLO, speech recognition.

Abstract— Robots are increasingly present in everyday life to provide convenience to the user by performing tasks that save time and effort. Today there are many types of robots for all kinds of applications. Artificial intelligence plays an increasingly important role in the creation of new applications.
In this work we have created a robot that acts as a Croupier or Dealer in the game of poker, specifically the Texas Holdem modality. Deal AI (Dealer with Artificial Intelligence) is a robot that prepares the start of the game by dealing cards and chips to the players. It then communicates the current player's turn and follows the status of the game at all times. It can be asked questions about the state of the game and the rules of the game. He also offers token changes if the players request it. Finally it announces the winner when the round is over and the players have shown their cards.

Index Terms— Robot, Deal AI, mechanical arm, scara, poker, chips, croupier cards, dealer, artificial intelligence, single board, Computer Arduino Atmel, YOLO, speech recognition.

1 Introducción

EL PÓQUER es un juego de apuestas mediante el uso de cartas en el cual los jugadores trataran de conseguir la mejor combinación de cartas para ganar. Cada jugador empieza con 2 cartas y a medida que pasan las rondas se desvelan hasta 5 cartas. Las pujas iniciales están previamente determinadas. [1]

Deal AI es un robot que se encarga de las funciones que hace un *Dealer o Croupier.* Este es el encargado de gestionar la partida, así como de repartir fichas, cartas y administrar el bote común.

Este robot consiste básicamente en un robot SCARA de 4 grados de libertad y efector final rotacional. [2] Se ha elegido la estructura de este robot ya que abarca bastante área de trabajo y permite llegar a las posiciones de un tablero de póquer y permite modificar la orientación de los objetos. Con este brazo se puede manipular las fichas y cartas.

Para poder recoger objetos se han estudiado diferentes formas como el uso de una ventosa, motores de vacío y un solenoide. Para poder voltear las cartas se ha creado un objeto donde el robot suelta la carta y esta aparece invertida en una bandeja.

Este robot está diseñado para ser implementado en una mesa semicircular. Éste se ubica en el centro posterior de la mesa. También dispone de una cámara con la cual puede reconocer la posición de los objetos y su valor aplicando visión por computador. Esta cámara se ubica en una estructura externa al robot para tener una vista cenital. El hecho de conocer las cartas de los jugadores permite al robot determinar el ganador de la partida.

Funciones del Deal AI:

- Manipulación de objetos.
- Mostrar cartas.
- Quemar cartas.
- Girar cartas.
- Control del estado de la partida.
- Responder preguntas.
- Acatar órdenes.
- Dar cambio de fichas.
- Declarar ganador.

- *E-mail de contacto: emilcarvajal-2000@hotmail.com*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Carlos García Calvo (Departamento de Ciencias de la Computación)*
- *Curso 2021/22*

2 Objetivos

El principal objetivo es crear una escena de póquer donde se muestre que el robot cumple con sus principales funcionalidades.
Para ello se determinaron 8 objetivos de bajo nivel que debían de cumplirse de forma progresiva. Estos consisten en recopilar información, realizar un estudio previo, realizar diseños, crear un chasis, montar los componentes, desarrollar el software, realizar pruebas y por último aplicar modificaciones y mejoras.

Se puede encontrar una lista detallada de los objetivos en el apéndice A1. Objetivos.

4 Planificación

Para poder desarrollar este proyecto, se determinó una lista de actividades que se agruparon en diversas fases. Estas actividades se planificaron rigorosamente en función de su dificultad:

1. Inicio y planificación del proyecto.
2. Experimentación.
3. Diseño.
4. Ejecución.
5. Test i optimización.
6. Cierre del proyecto.

En el apéndice A.2 Planificación se detallan estas fases.

5 Estructura y Tecnologías

Este proyecto está estructurado en 6 fases que ya se han mencionado en el apartado de planificación. Las más importantes son la fase 2, fase de experimentación, y fase 4, fase de ejecución. En la fase de experimentación se estudió el uso de una Raspberry Pi ya que es una tecnología que permite el acople de una cámara, varios dispositivos más y permite también instalar software en ella. [3] También se realizaron las primeras pruebas mediante el uso de una placa Arduino y un shield de motores ya que es una tecnología más práctica.

En la fase de ejecución se desarrolló todo el software y se construyó el robot. Para ello, se utilizó *Google Collab*, *Jupyter Notebook* y *Anaconda Spyder* con el lenguaje de programación Python. Para el módulo de reconocimiento de voz se usó una API de Google (Speech API). Para el módulo de reconocimiento de objetos se utilizó el software YOLO con su red neuronal *Darknet*.

Los diagramas de *software* se realizaron con el programa Draw Io, una herramienta de creación y edición de diagramas libre que permite la integración con diversas plataformas. [4]
Los diseños de las piezas del robot se crearon en la plataforma online Tinkercad, un software gratuito de diseño y modelado 3D que permite diseñar cualquier objeto de forma intuitiva. [5]
Los diseños del circuito y los componentes se hicieron con el programa Fritzing, un programa libre de automatización de diseño electrónico. [6]
También se utilizaron algunas imágenes de una escena real del proyecto simulada mediante el programa Coppelia.
Las versiones del software tienen una copia en local y una copia online (*Github*).

Para la parte de documentación se utilizó Microsoft Office 365 que incluye Microsoft Word, PowerPoint, Excel y OneNote. El respaldo de la documentación se encuentra en Google Drive.

6 Desarrollo

A continuación, se muestra todo el progreso realizado en función de las fases planificadas.

6.1 Inicio i Planificación del Proyecto

En primer lugar, se determinó el enfoque y el ámbito del proyecto.

En segundo lugar, se realizó una búsqueda de documentación sobre los recursos que se disponían y se estudió la metodología del proyecto.

Por último, se realizó un análisis de riesgos y se elaboró un plan de contingencia para los posibles problemas que podían suceder en el trascurso del proyecto. Este análisis se encuentra detallado en el apéndice A3. Análisis de riesgos y plan de contingencia.

6.2 Experimentación

Primero se diseñó un mapa conceptual de módulos software y un esquema de componentes hardware con la idea de utilizar una Raspberry PI.
También se elaboró un primer diseño del chasis pensado para que los motores se concentren en la base. De esta forma el robot tiene más estabilidad.

Para poder realizar la fase de diseño y la fase de ejecución se realizó un estudio previo. Este estudio consistía en ver cómo respondería el robot en función de qué componentes se utilicen y de cómo ellos están conectados.

Para este estudio se tuvieron en cuenta **2 alternativas**:

1. Uso de Arduino UNO más ordenador más shield de motores.
2. Uso de Raspberry Pi con shield de motores.

Se simularon movimientos de motores conectados a una placa Arduino UNO y un *shield* de motores.
De esta forma se realizó un primer contacto con la placa y la conexión entre motores. Después se realizó el mismo procedimiento, pero con motores reales que se eligieron específicamente para este robot.

Se estudió el uso de una Raspberry Pi para poder crear un robot independiente de un ordenador. Para ello, se ejecutó el sistema YOLO (ya entrenado para detección de gatos) en una Raspberry Pi 4. La detección de objetos era correcta, pero se realizaba de forma lenta. También se ejecutó un sistema de reconocimiento de voz, que resultó 10 más lento que utilizando un ordenador.

Estas pruebas indicaron que el procesado con una Raspberry Pi es demasiado lento como para que sea viable una partida a tiempo real, sobre todo para los sistemas de reconocimiento de objetos y voz. Por este motivo, se descartó su uso y la idea de crear un robot independiente de un ordenador.

Existe una tabla con ventajas y desventajas de las 2 alternativas en el apéndice A4. Métodos estudio previo.

6.3 Diseño

Todo el diseño se ha pensado en base a una simulación de una escena real de una partida de póquer mediante el uso del programa *Coppelia*.

Fig. 1: Escena real de póquer simulada.

6.3.1 Mapa conceptual del proyecto

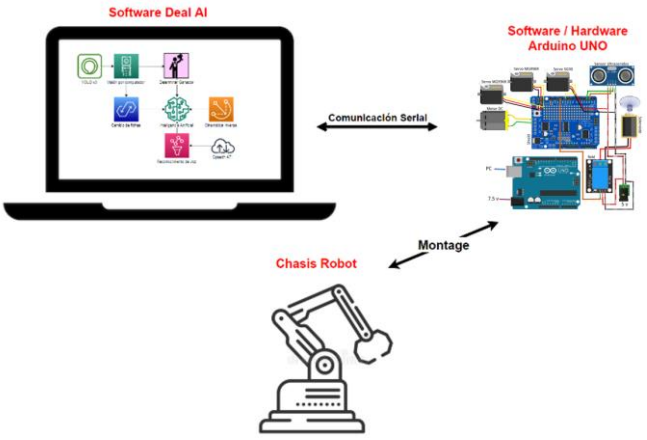


Fig.2: Representación gráfica del proyecto.

Uso de Arduino UNO con shield de motores + ordenador:
Este esquema representa la estructura final del proyecto. El software principal del robot se encuentra ubicado en un ordenador. Este se ha desarrollado en lenguaje Python mediante el uso del IDE Spyder. El software que controla los motores, el sensor de ultrasonido y el solenoide se ubica en la placa Arduino UNO desarrollado en lenguaje C++ mediante el IDE Arduino. El solenoide se usa para poder coger y soltar mediante el uso de una ventosa. Cuando está cerrado permite coger un objeto y cuando se activa deja pasar aire a la ventosa expulsando el objeto acoplado.

Se ha escogido el uso de una placa **Arduino UNO** por el hecho de que permite el uso de interrupciones que permiten variar la velocidad de los servos. Esto ayuda a realizar movimientos menos bruscos, evitando inercias. También porque Arduino UNO permite el uso de una placa ***shield*** que se acopla directamente encima de la placa Arduino. Esta *shield* permite poder conectar motores del tipo Servo, *Stepper* y DC, aumentando la potencia y precisión del controlador. También simplifica conexiones y reduce el cableado.

6.3.2 Diseño conceptual Software Deal-AI

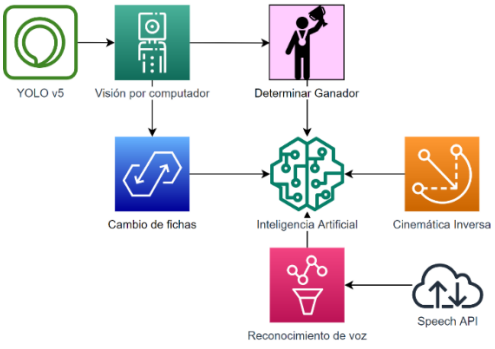


Fig. 3: Esquema conceptual del software del Deal AI.

Este diseño representa la estructura de organización de cómo están conectados los diversos módulos.

6.3.3 Software Deal-AI

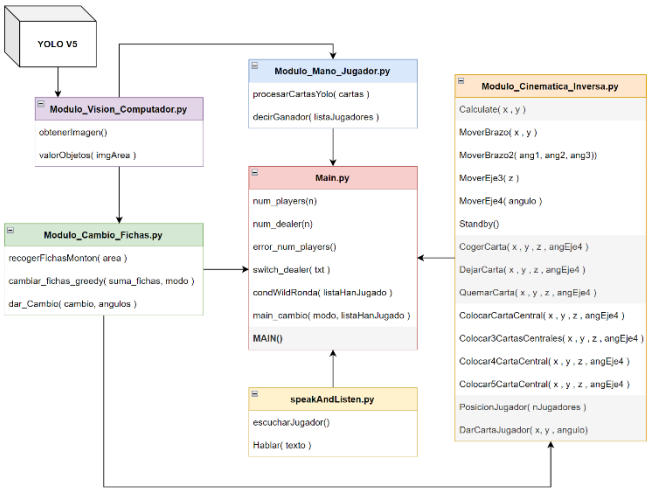


Fig. 4: Esquema de los módulos software.

Diseño software que representa los archivos phyton '.py' junto a sus principales funciones de alto nivel.

Módulo_Cinemática_Inversa: Módulo que controla los movimientos del brazo mecánico.
Módulo_Cambio_Fichas: Módulo que controla el sistema de cambio de fichas.
Módulo_Visión_Computador: Módulo que se encarga de gestionar las funciones de visión por computador.
Módulo_Mano_Jugador: Módulo que se encarga de decir quién es el ganador de la "mano".
speakAndListen: Módulo que se encarga de las funciones de interacción robot-humano.
Main: Sistema principal que controla el juego y los movimientos del robot.
YOLOv5: Repositorio local de YOLO.

Todos estos módulos se detallan en el apartado 4.12 Desarrollo del software.

Como alternativa a este esquema se podía desarrollar la función "Calculate(x, y)" en el software del Arduino UNO (c++). Esta función realiza los cálculos de la cinemática inversa. Se descartó esta alternativa porque la ejecución de la función es más lenta en la placa que en un ordenador.

6.3.4 Componentes Hardware

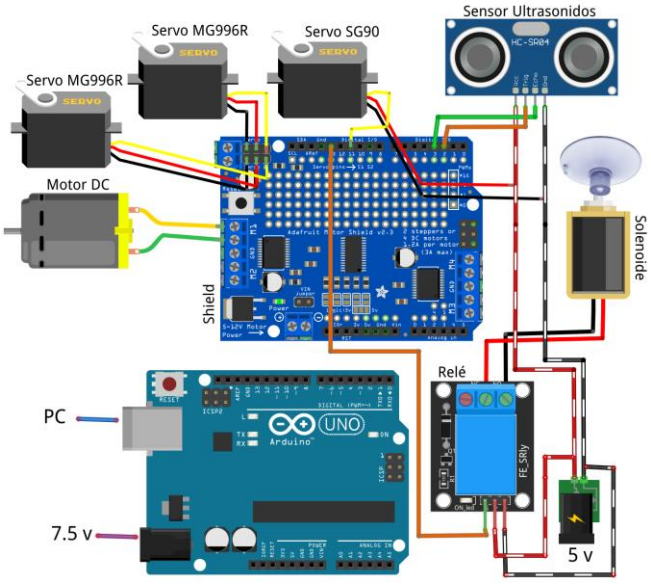


Fig. 5: Esquema de componentes.

Diseño hardware con el software principal del robot ubicado en el ordenador. Esto significa que el robot está siempre conectado al ordenador, el cual envía las ordenes al Arduino UNO que son procesadas por un software instalado en la placa. Se han separado las fuentes de alimentación de los motores y la de los servos y Arduino. A los servos MG996R y al motor DC se les proporciona alimentación externa de 7.5 voltios. El resto recibe energía del ordenador de 5 voltios.

La conexión entre el software del robot y el software del Arduino UNO se produce mediante una comunicación serial por medio de un puerto. Esta comunicación se realiza a través del intercambio de una secuencia de bits. [3]

Conexiones (→ Serie , // Paralelo)

Motor Shield	→	*Arduino UNO*
SERVOS MG996R (pin , v , gnd)	→	*Motor Shield ((9,10) , +, -)*
SERVO SG90 (pin , v , gnd)	→	*Motor Shield (11 , 5v, GND)*
MOTOR DC (pin1, pin2)	→	*Motor Shield (M1A, M1B)*
HC - SR04 (Vcc , GND , Trig,Echo)	→	*Motor Shield (5v, GND, 2, 3)*
Relé KY-019 (pin ,V, GND)	→	*Motor Shield (13, 5v, GND)*
Solenoide (+ , -)	→	*Relé KY-019 (Common, NO)*

6.3.5 Chasis

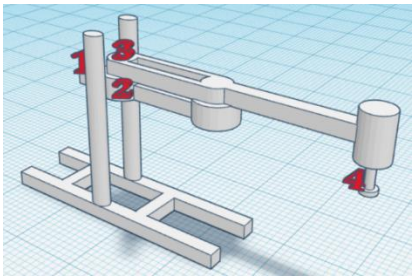


Fig. 6: Diseño de la estructura del chasis.

Este diseño está pensado para que los motores se concentren en la base. De esta forma el robot tiene más estabilidad.
Los números representan la ubicación de los motores.

TABLA 1
Motores y descripción

N.	Motor	Justificación
1	VEX: 2 Wire Motor 293 100 rpm 1,67 N-m	Motor grande de corriente continua de 7'2 V. Se ha elegido este motor porque es lo suficientemente potente para poder subir y bajar todo el brazo de forma lineal-vertical.
2	Servomotor MG996R	Servomotores medianos de 6V. Permiten mover el brazo y el antebrazo de 0 a 180º, lo necesario para este proyecto. Son de buen tamaño y peso. 11 kg/cm , 4'8V – 7'2V.
3	Servomotor MG996R	
4	Servomotor SG90	Servomotor liviano y pequeño de 5V. Este servomotor permite orientar las cartas. 1'8 kg/cm , 4'8V – 6V.

Estructura de los ejes
Para crear el chasis se ha utilizado material Vex en aluminio y acero ya que es un material muy resistente y permite resultados de movimientos muy precisos. La composición de los ejes se realiza mediante engranajes de plástico.

6.4 Ejecución

6.4.1 Creación de un chasis

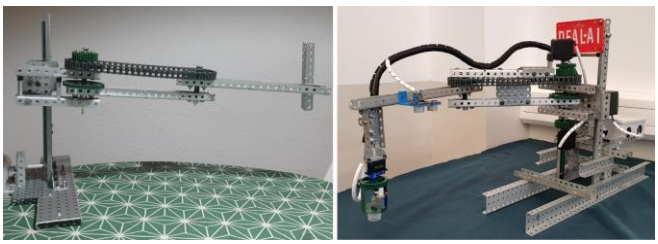


Fig. 8: Prototipo y chasis final.

Como se puede observar en la imagen de la izquierda de la Fig.8, el principal problema era el elevado peso del brazo que provocaba una inclinación respecto la base. Por este motivo se implementó otro diseño donde se añadió otra columna central, lo cual distribuía el peso de la base del brazo en 2 columnas y corrigió el problema en gran mesura.

6.3.6 Objeto volteador de cartas

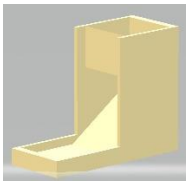


Fig. 7: Diseño del objeto que gira una carta.

Esta pieza tiene 1 entrada superior y una salida inferior. Cuando se introduce una carta por la entrada, esta se gira por la estructura del objeto y termina en una bandeja.

6.4.2 Desarrollo del software

6.4.2.1 Visión por computador

Reconocimiento de valor y posición de cartas

YOLO
Para esta actividad se ha utilizado YOLO v5, un sistema de detección de objetos que funciona sobre *Darknet*, una red neuronal convolucional escrita en C que funciona como Framework. *Darknet* procesar una imagen, pasándola a escala de grises y detectando los bordes de los objetos con un filtro. Después reduce la imagen a un problema de clasificación de clases y enmarcar el objeto identificado en una caja. [7]

Se ha elegido este sistema porque es un detector de objetos rápido y preciso, lo que lo hace ideal para aplicaciones de visión artificial. También ofrece buenas prestaciones procesando imágenes en tiempo real.

Introducción entrenamiento
Los sistemas con inteligencia artificial requieren de grandes bases de datos para poder ejecutar tareas muy complejas como lo es detectar y clasificar un objeto.

Para poder entrenar a YOLO se ha creado una base de dados desde cero. Como YOLO requiere de unas 300 imágenes para poder detectar y clasificar correctamente un objeto y se tienen 52 cartas por clasificar (52 objetos), se ha creado una base de datos con 15.600 imágenes de cartas.

Base de datos
Primero se realizaron 100 fotografías por cada objeto a clasificar. Después se realizó un proceso de etiquetado donde se determinaron los marcos de coordenadas (x,y,z,w) de

las regiones donde se encontraban las cartas en las fotografías. El proceso de etiquetado se realizó con el programa "labelImg.py". Esto consumió muchas horas de trabajo. Para no repetir el proceso 2 veces más (solo se tenían 100 imágenes por categoría), se realizó un proceso de *data augmentation* (mediante la plataforma online *Roboflow*) que permitió obtener hasta 300 imágenes por categoría, conservando el etiquetado de las 100 imágenes originales.
Este proceso editaba y creaba nuevas imágenes aplicando un pre-procesado, aumentando color, resaltando brillo, contraste, etc.

Entrenamiento
Una vez creada la base de datos, se dividió el *dataset* en 75/15/10: 75% conjunto de entrenamiento, 15% conjunto de validación y 10% conjunto de test. Después se configuró YOLO en *Google Collab* y se entrenó el sistema con 120 iteraciones. En ver que la precisión del sistema obtenía casis resultados perfectos a partir de 60 iteraciones, se configuró el sistema con este nombre en la siguiente ejecución.

Resultados

Fig. 9: Reconocimiento de cartas con YOLO.

Para esta ejecución de 60 iteraciones observamos una *accuracy* máxima de 99% en el 15% de conjunto de test. La *accuracy* es la cantidad de predicciones positivas que fueron correctas respecto el total de casos examinados.

Reconocimiento de valor y posición de fichas

Para la ejecución de esta actividad se ha realizado el mismo proceso que se utilizó para el reconocimiento de cartas mediante el uso de YOLO, ya comentado en el apartado anterior.

El proceso de *data augmentation* se hizo en conjunto al *dataset* creado para las cartas (sin aumento), aplicando una nueva configuración de mosaico, para así poder variar el tamaño de las cartas en las imágenes y también poder mezclarlas.
Este proceso no hubiera sido necesario para las fichas ya que se tomaron las fotografías aplicando diversas distancias a los objetos.

Entrenamiento
Una vez creada la base de datos, se dividió el *dataset* final (con fichas y cartas) en 75% conjunto de entrenamiento, 15% conjunto de validación y 10% conjunto de test. Después se configuró YOLO en *Google Collab*, se importó el *dataset* final de *Roboflow* y finalmente se entrenó el sistema con 60 iteraciones. Se configuró el sistema con este nombre ya que con este se obtuvieron resultados muy precisos cuando solo se entrenaron las cartas.

El resultado con 60 iteraciones fue de tan solo 55% de *accuracy* respecto el conjunto de test. Esto fue debido a la configuración de mosaicos que se agregó en el proceso de *data augmentation* ya que, con esta configuración, el algoritmo entrenaba con cartas de diferentes tamaños, hecho que dificultaba su aprendizaje.

Finalmente se aumentaron hasta 150 iteraciones, consiguiendo así una *accuracy* del 99% respecto el conjunto de test.

Resultados

Fig. 10: Reconocimiento de fichas con YOLO.

La Fig.10 representa imágenes no vistas por el algoritmo, donde se observa una accuracy de 95% de media.

6.4.2.2 Módulo de cinemática inversa

Módulo que controla los movimientos del brazo mecánico. Se realiza cinemática inversa ya que se programa al robot para que se coloque en posiciones concretas.
Básicamente, tras haber establecido una posición o una rotación de un motor, se envían ordenes al Arduino mediante comunicación serial. Después el controlador ordena la cinemática a realizar a los componentes hardware mediante el *shield* de motores.

De esta manera todo el software del robot se encuentra en el ordenador y el Arduino solo se encarga de ejecutar las ordenes que se envían des del ordenador.

Cálculo de la cinemática inversa

Se han desarrollado **2 alternativas** para obtener los valores que se han de aplicar a los servos:

1. Tabla de D-H: Cálculo de las ecuaciones de movimiento del robot mediante una tabla de Denavit Hartenberg. [2]
2. Geometría 2D: Cálculo de los ángulos de los servos en un espacio de 2 dimensiones mediante el uso de ecuaciones trigonométricas.

La alternativa 1 se descartó porque para conseguir una solución óptima de forma rápida es mejor un método geométrico.
Para más información consultar el apéndice A5. Alternativas cálculo cinemática inversa.

Simulación cinemática inversa

Para poder crear este módulo, se hizo previamente una simulación de los servos de 180º conectados al Arduino mediante el uso de la plataforma online Tinkercad.

De esta forma se desarrolló la función **calculate(x , y)** (escrita en lenguaje C++) que determina los ángulos en que los motores se tienen que orientar para colocar el robot en una posición específica. Esta función representa la segunda alternativa de obtener la cinemática inversa "**Geometría 2D**".

Finalmente se implementó todo este software en el módulo de cinemática inversa programado en lenguaje Python.

Las **funciones principales** de alto nivel que forman parte de este módulo se encuentran en el apéndice A5. Funciones cinemática inversa.

6.4.2.3 Módulo de cambio de fichas

Módulo que controla el sistema de cambio de fichas y consiste en 3 pasos:

1. Recoger las fichas que se desean cambiar y llevarlas a la banca.
2. Tratamiento del valor total de las fichas para establecer cuantas monedas y de qué tipo ha de devolver, es decir, determinar qué fichas se tienen que devolver al jugador en función si quiere obtener fichas con más o menos valor.
3. Recoger las fichas de cambio de la banca y colocarlas en la posición de las fichas de cambio del jugador.

Para esto se han desarrollado 3 **funciones**:

1. **recogerFichasMonton(area):** Por medio de un área establecida controla el módulo de visión por computador de las fichas (YOLO) para poder reconocer las fichas y obtener su posición. Luego a través de las funciones de cinemática inversa coloca esas fichas en su determinada posición de la banca.

2. **cambiar_fichas_greedy(suma_fichas, modo):** Algoritmo *greedy* (obtenido de "Análisis y diseño de algoritmos", asignatura de la mención de Computación) implementado para saber qué fichas ha de devolver el robot. Incorpora 2 modos para devolver fichas con valores bajos o máximos. Recibe la suma total del valor de las fichas (valor de retorno de la función anterior).

3. **dar_Cambio(cambio, ángulos):** Recibe qué fichas se han de devolver (valor de retorno de la función anterior) y los ángulos correspondientes a la posición donde el jugador dejó sus fichas de cambio (valor de retorno de la función "recogerFichasMonton()"). Luego, a través de las funciones de cinemática inversa, coloca esas fichas en las posiciones de cambio del jugador con los ángulos recibidos.

6.4.2.4 Módulo de reconocimiento de voz

Módulo que se encarga de las funciones de interacción robot-humano. Para desarrollar este módulo se han utilizado las siguientes **librerías**:
- **Speech_recognition**: para el reconocimiento de voz y así poder generar ordenes al robot.
- **GTTS**: para leer con voz sintética (text-to-speech) las respuestas dadas por el robot.
- **Playsound**: para reproducir sonidos y poder leer los resultados obtenidos por el gTTS.
- **Os**: para eliminar el archivo mp3 generado.

Funciones:
- **escucharJugador():** activa el micrófono, escucha al jugador y transforma el audio a texto mediante el uso de la API de reconocimiento de voz de *Google.*
- **hablar(texto):** trasforma el texto pasado por parámetro, genera y ejecuta un archivo de voz .mp3 y por último elimina el archivo.

6.4.2.5 Módulo de visión por computador

Módulo que se encarga de gestionar las funciones de visión por computador y conecta con el sistema YOLO.

Funciones:
- **obtenerImagen():** enciende la cámara, obtiene una imagen y la procesa.
- **valorObjetos(imageArea):** función que obtiene los valores y las posiciones de los objetos de la mesa (cartas y fichas).

Para ello se utiliza la librería 'torch.hub' que permite pasar una imagen al modelo entrenado de YOLO sin necesidad de importar sus archivos al *software* del Deal AI. De esta manera YOLO devuelve los objetos detectados y sus posiciones. Finalmente, estos resultados son procesados para que se acoplen correctamente al software del robot.

6.4.2.6 Módulo de mano de jugador

Módulo que se encarga de decir quién es el ganador de la mano. **Librerías** utilizadas**:**

- **Pokereval:** para calcular la puntación obtenida de 2,5,6 o 7 cartas (2 es el mínimo de cartas con las que puede combinar un jugador y 7 es el máximo.
- **Eval17:** para determinar el combo de una combinación de cartas (carta alta, pareja, trio, póker…)

Funciones:

- **procesarCartasYolo(cartas):** esta función recibe los resultados de cartas que detecta YOLO. Estos los procesa en nuevas clases (clase Carta) de las librerías 'pokerval' y 'eval 17' para así poder determinar las puntuaciones y los combos.
- **decirGanador(listaJugadores):** función que declara quien es el ganador. Recibe los jugadores que han llegado a la última ronda. Obtiene los valores de las cartas de la mesa y de los jugadores con la función 'valorObjetos(imageArea)'. Determina las puntuaciones con una función de 'Pokereval' y los combos con una función de 'Eval17'.
  Finalmente devuelve el número del jugador con más puntación y su combo ganador.

6.4.2.7 Inteligencia Artificial

Main

Este es el núcleo del robot, donde todos los inputs son transmitidos a este módulo el cual se encarga de la gestión de la partida y el control del brazo mecánico mediante el uso de órdenes. Estas órdenes pasan por el controlador en concreto y realiza una acción: comunicar un mensaje por el altavoz, realizar una secuencia de movimientos o determinar el ganador.

Este main representa el sistema del juego Póker en la modalidad Holdem Texas, pero con ciertos cambios que se adaptan a una partida cotidiana entre amigos.
Para ello, cada jugador empieza con una cantidad de fichas determinadas.

Manos y rondas

El sistema está organizado por "Manos" y "Rondas". Se denominan **Manos** a 4 Rondas completas donde en la última se decide quién es el jugador que ha ganado la Mano y se lleva todo el dinero invertido. Una **Ronda** se termina cuando todos los jugadores activos de la Mano han apostado la misma cantidad de fichas. En la Ronda 0 se reparten las cartas, en la Ronda 1 se colocan 3 cartas centrales en la mesa y en la Ronda 2 y 3 se "quema" una carta y se coloca otra junto a las centrales. "Quemar" una carta no es más que coger una carta y descartar-la colocándola en una posición de la mesa.

Control

El sistema controla la secuencia de juego y mantiene un estado de los jugadores que no se han retirado para pasar a la siguiente ronda.
Controla en todo momento los jugadores que están activos y los que han abandonado la mesa (no se contemplan para las siguientes Manos y no se les reparte cartas).
También controla los jugadores que han "pasado". Estos jugadores simplemente no han hecho una puja inicial. En caso de que algún jugador no "pase" y puje, volverá a ser el turno de los jugadores que habían "pasado".

Cuando es el turno de un jugador se le pregunta, por **comandos de voz**, que acción desea realizar. Solo se ejecutan las acciones que ordena si se encuentran en el sistema. Estas **acciones** son típicas del juego como:

- "Paso": el jugador no desea subir la apuesta.
- "Igualo, Voy": el jugador sigue la apuesta máxima y coloca sus fichas en la mesa.
- "Me retiro": Se retira de la Mano y no se le preguntará nada hasta la próxima Mano.
- "Abandono": Abandona la mesa y ya no se le tendrá más cuenta en el juego.
- "Cambio a la alta / baja": El jugador coloca sus fichas en su área de cambio según si desea cambio a valores altos o bajos. Implementado por el **Modulo de Cambio.**
- Preguntas de estado: Se le puede preguntar al sistema información sobre el estado del juego:
  - Ejemplos: "Dealer actual?", "Ronda actual?, "Quién es la ciega grande?", "Apuesta máxima?"…

Cuando se ha llegado a la **última Ronda** de una Mano el sistema comunica a los jugadores que han permanecido al final de la Mano que muestren sus cartas.
El sistema reconoce todas las cartas que hay en la mesa gracias a los módulos "mano del jugador" y "visión por computador" y determina el jugador que ha ganado y la combinación de cartas con lo que lo ha hecho.

Para las siguientes Manos el sistema de juego es el mismo. Se termina el juego cuando solo queda un jugador en la mesa o alguien ha ordenado "Finalizar póquer".

6.4.2.8 Software *Arduino UNO*

Software en lenguaje C++ que controla los servos, el sensor de ultrasonido y el solenoide para controlar la ventosa.

El programa principal consiste en un bucle que espera a recibir una orden a través de una comunicación serial con el software del Deal AI desarrollado con *Python.*
Esta orden es una cadena de tipo (dispositivo, número del dispositivo, valor a cambiar). Por ejemplo, para que el servo 1 se oriente en el ángulo 180 seria: "s,1,180 ".
Para procesar esta orden está la función **ejecutarOrden(orden)** que fragmenta la cadena y envía la orden al dispositivo especificado.

Cabe también destacar la función **distObjeto()** que devuelve la distancia entre el sensor de ultrasonido y la mesa u objeto a detectar. Esta función se utiliza sobre todo para poder parar el motor DC cuando se le pide al robot colocarse en una posición Z determinada.

6.5 Test y Optimización

6.5.1 Software Deal AI

Una vez desarrollado todo el software se realizaron varias pruebas de unidad para comprobar que todas las funciones se realicen correctamente. Estas pruebas se hicieron des del archivo main.py. De esta forma se aseguró que todos los módulos se acoplaran de manera correcta.

Las pruebas no consistían en funciones de *testing,* sino que simplemente eran llamadas a las diversas funciones de alto nivel de todos los módulos.

Cabe destacar el buen acople que hubo entre el repositorio local de YOLO y el módulo de visión por computador donde se obtuvieron resultados como estos:

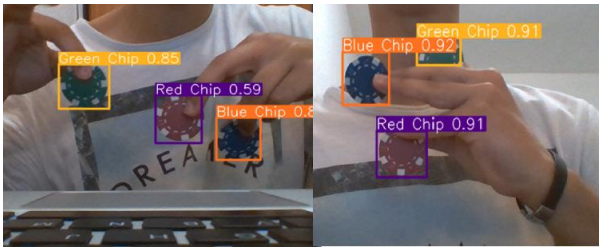


Fig. 11: Reconocimiento de fichas en tiempo real.

Mediante el uso de la webcam del ordenador, observamos buenos resultados en tiempo real sin haber mostrado todo el contorno de las fichas y en condiciones de baja luminosidad.

6.5.2 Hardware / Software Arduino

Una vez montado todo el hardware se realizaron pruebas de funcionamiento mediante el software desarrollado para el Arduino UNO. Estas pruebas consistían en enviar órdenes al Arduino UNO directamente desde el terminal del IDE Arduino.
Se comprobó el correcto funcionamiento de todas las ordenes.

6.5.3 Comunicación ordenador-Arduino

Se comprobó el correcto funcionamiento de todas las ordenes enviadas a través del IDE Spyder.
Para esto se requiere de programa emisor y un programa receptor. En ese caso el software principal del robot, escrito en phyton, corresponde al programa emisor, y el software del Arduino corresponde al programa receptor.

6.5.4 Software Deal AI – Arduino - Chasis

Para realizar las pruebas con el chasis se colocaron todos los componentes *hardware* en su posición.
Se resolvieron todos los problemas de mecánica ajustando todos los ejes. Uno de los ejes más críticos era el prismático, el cual permite subir y bajar todo el brazo.

EL motor DC de este eje no era capaz de cumplir su función y se buscó una relación de engranajes más pequeña para solucionarlo. Se añadió un segundo motor por si fuera necesario coger objetos más pesados y además conseguir una simetría.

7 Resultados

Fig. 12: Escena real de póquer con Deal AI.

Como resultado se obtiene un robot funcional que cumple con todos los requisitos para ser *Croupier,* el cual tiene que estar siempre conectado a un ordenador.

Todo el software se encuentra en un archivo programado en Python, a excepción del *software* del Arduino escrito en C++ que controla a los motores y sensores.

El robot no tiene fallos en el sistema principal del juego. Anuncia en todo momento el turno del jugador actual e informa del estado de la partida. Se comunica con los jugadores de forma correcta. Realiza movimientos no bruscos

evitando inercias. Sube y baja utilizando un sensor de ultrasonidos que determina su parada ante un objeto. Procesa órdenes de jugadores en un tiempo no superior a 2 segundos. Clasifica fichas y cartas en menos de 2s con una precisión entre 80 y 99%.

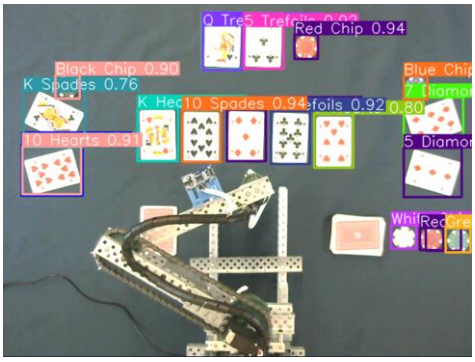


Fig. 13: Reconocimiento de objetos en una escena real.

Aunque determina correctamente las posiciones de los objetos en la mesa, cuando se coloca el manipulador encima de ellos, no es del todo preciso por tener holguras en el chasis y por la precisión real de los motores. Aun así, esto no impide la recolección de objetos. También realiza cambios de fichas entre jugador y banca entre 20 y 30 segundos en función del nombre de fichas por cambiar. Gira las cartas soltándolas en un objeto con una estructura que realiza el giro. Por último, en la última ronda de la mano, determina quién es el ganador, reconociendo las cartas en juego y utilizando un sistema de puntuaciones.

8 Conclusión y Propuesta de mejora

Se han conseguido unos módulos *software* funcionales. La cinemática inversa se calcula con error máximo de 2cm. Esto afecta a veces a la hora de coger las fichas. También se realizan movimientos de forma no brusca. El sistema YOLO puede reconocer fichas y cartas de forma óptima. Al módulo de reconocimiento de voz le cuesta a veces reconocer sonidos, pero, aun así, es muy eficiente para permitir hacer jugadas completas. Los dispositivos conectados al Arduino y shield de motores y colocados en el chasis funcionan correctamente. La ventosa succiona las cartas y las fichas perfectamente.

Se ha escogido el uso de una placa Arduino UNO más el shield de motores por la conectividad y la posibilidad de usar interrupciones para variar la velocidad de los servos.

Se ha desestimado el uso de una Raspberry Pi 4 ya que el procesado es demasiado lento como para que sea viable una partida a tiempo real, sobre todo para los sistemas de reconocimiento de objetos y voz.

Se ha cumplido el propósito de este proyecto con recursos accesibles al alumno. Para evitar el error de los 2cm sería necesario un presupuesto mayor mejorando la mecánica y los motores. Se podría también implementar el software desarrollado en un robot Scara industrial.

Fig. 14: Robot Scara TS2 de Stäubli.

De esta forma no habría ningún problema en el apartado técnico. También se podría mejorar el procesado de voz donde, en lugar de reconocer solo frases claves, se podría reconocer todo tipo de palabras mediante procesamiento de lenguaje natural.

Agradecimientos

Quiero agradecer a mi tutor, Carlos García Calvo, por haberme guiado en todo momento durante el desarrollo de este proyecto. Por su gran ayuda e implicación, siguiendo el proyecto muy de cerca para una mejora continua. También por resolver todas mis dudas, realizar reuniones semanales y aportar todo el material necesario. Sus conocimientos de robótica han sido claves, sobre todo, en todo lo relacionado con el hardware.

Bibliografía

[1] **Gonzalez, Y. F**. (2022, 15 febrero). ¿Qué es el poker y cómo se juega? Kelbet. [En línea] https://kelbet.es/que-es-el-poker-y-como-se-juega.html

[2] **Craig, J. J**. (2006). Robotica. Pearson Educación.

[3] **C, S.** (2020, 19 diciembre). Comunicación Serial con Arduino. Control Automático Educación. [En línea] https://controlautomaticoeducacion.com/arduino/comunicacion-serial-con-arduino/

[4] **Na8**. (2022, 7 mayo). Detección de Objetos con Python. Aprende Machine Learning. [En línea] https://www.aprendemachinelearning.com/deteccion-de-objetos-con-python-yolo-keras-tutorial/

[5] **GEEKNETIC**. ¿Qué es Raspberry Pi y para qué sirve? - Definición. (2020). [En línea] https://www.geeknetic.es/Raspberry-Pi/que-es-y-para-que-sirve

[6] **Mancomún**. (1970, 1 enero). ¿Qué es DrawIo? [En línea] https://www.mancomun.gal/es/solucion-tic/draw-io/

[7] **Bustamante, E. G**. (2021, 29 septiembre). ¿Qué es Tinkercad y para qué sirve? Spacetechies. [En línea] https://www.space-techies.com/que-es-tinkercad-y-para-que-sirve/

[8] **Hardware Hacking Mx**. (2014, 4 enero). Fritzing ¿Qué es y cómo se usa? [En línea] https://hardwarehackingmx.wordpress.com/2014/01/04/fritzing-que-es-y-como-se-usa/

[9] **330ohms**. ¿Cómo detectar objetos con YOLO? (2020, 17 noviembre. [En línea] https://blog.330ohms.com/2020/11/17/deteccion-de-objetos-con-yolo/

APÉNDICE

A1. OBJETIVOS

Num.	Descripción
[o1]	Recopilación de información: - Estudio de componentes y tecnologías. - Análisis de requisitos. - Determinación del alcance del proyecto y su viabilidad. - Análisis de riesgos y plan de contingencia.
[o2]	Estudio previo. Estudio de diversos métodos y tecnologías que se podrían aplicar en este proyecto.
[o3]	Realizar diseños de software y hardware del robot.
[o4]	Creación de un chasis. Con este chasis se hará el estudio previo y se desarrollará el proyecto.
[o5]	Montaje de componentes. - Creación de algún componente si es necesario.
[o6]	Desarrollo del software: - Preparación del sistema base - Cinemática inversa. - Reconocimiento de objetos y localización. - Manipulación de objetos. - Reconocimiento de voz. - Inteligencia artificial.
[o7]	Construcción del robot. Instalación del software. Realizar pruebas del resultado obtenido.
[o8]	Modificaciones y optimizaciones.

A2. PLANIFICACIÓN

Fase	Descripción
1. Inicio y planificación del proyecto	- Fase de definición del proyecto. - Se concreta cómo será el producto a desarrollar (tanto HW como SW y documentación) y se planifica y administran las actividades a desarrollar. - Se determina el alcance.
2. Experimentación	- Se hace un estudio previo para poder realizar la fase de diseño y la fase de ejecución de forma progresiva. - El estudio consiste en ver cómo responde el robot en función de qué componentes se utilizan y de cómo ellos están conectados. - Para esta fase se tienen en cuenta 3 alternativas: 1. Uso de Arduino UNO. 2. Uso de Raspberry Pi conectada al ordenador. 3. Uso de Raspberry Pi con el sistema instalado.
3. Diseño	- Fase en que se realizan diagramas de diseños HW y diseños SW.
4. Ejecución	- Fase de implementación del software final y construcción del robot.
5. Test i optimización	- Fase de mantenimiento final.
6. Cierre del proyecto	- Fase de informes finales y presentación final.

A3. ANÁLISIS DE RIESGOS Y PLAN DE CONTINGENCIA

Riesgo (R)
Probabilidad (P) [Alta/ Media/Baja] <-> [A/M/B]
Impacto (I) [Alto/medio/bajo] <-> [A/M/B]

R	Descripción	P	I	Plan de contingencia
1	La estructura que forma el esqueleto produce movimientos no controlados.	M	A	- Cambiar el tipo de material de la estructura por uno más preciso.
2	El peso de los componentes no están bien distribuidos: robot inestable.	M	A	- Modificar la estructura para mejorar el balance de cargas.
3	Los motores no permiten mover el brazo.	M	A	- Utilizar motores más grandes.
4	El robot colisiona con objetos de la mesa cuando está en movimiento.	M	A	- Modificar la estructura. - Aumentar la altura en que ubicará el brazo.
5	El sistema de recolección de objetos con ventosa no recoge objetos de manera óptima.	B	A	- Cambiar la válvula de succión. - Cambiar de sistema de recolección: utilizar un imán e imantar las cartas y fichas.
6	Giros bruscos hacen caer las fichas y / o cartas.	M	M	- Realizar movimientos más lentos - Reducir el movimiento al acercar-se al punto de destino.
5	Las cartas no entran en el objeto que gira las cartas.	B	A	- Modificar diseño del objeto que gira las cartas.

6	La cámara no obtiene imágenes suficientemente claras.	M	A	- Modificar posición de la cámara. - Cambiar de cámara.
7	Las fichas de la banca y las cartas de la baraja pueden mover-se durante la partida.	B	M	- Crear 2 objetos donde se encuentren las fichas de la banca y las cartas de la baraja.
8	Algunos componentes de la partida pueden obstaculizar la zona de juego de algún jugador.	B	M	- Reducir nombre de jugadores.
9	El robot colisiona con algún jugador cuando no está en movimiento.	B	B	- Poner siempre el robot en posición de *standby*.
10	El robot colisiona con algún jugador cuando está en movimiento.	B	B	- Avisos de robot en movimiento.
11	La placa madre no tiene suficientes recursos para ejecutar el software correctamente.	A	A	- Conectar la placa madre a un ordenador.
12	El robot no reconoce los comandos de los jugadores.	B	M	- Cambiar de micrófono.

A4. Métodos estudio previo

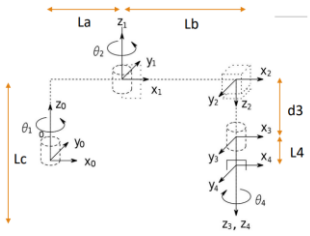
M1: Arduino UNO **M2:** Raspberry Pi

M	Ventajas	Desventajas
1.	- Permite trabajar con interrupciones. De esta forma se puede controlar la de los servomotores y atender sensores de manera inmediata. - Rápido para código simple.	- No dispone de entradas para periféricos indispensables como una cámara. - Siempre conectado al PC para ejecutar un sistema complejo, en este caso sería todo el software del robot Deal AI.
2.	- Sistema operativo integrado. - No requiere estar conectado al PC por lo que generaría un producto independiente. - Permite acople de dispositivos externos como una cámara, micrófono…	- Siempre conectado al PC. - Si el software está integrado, su ejecución es más lenta por no disponer de los recursos de un PC.

A5. Alternativas cálculo cinemática inversa

Se han desarrollado **2 alternativas** para obtener los valores que se han de aplicar a los servos:

1. Tabla de D-H: Calculo de las ecuaciones de movimiento del robot mediante una tabla de Denavit Hartenberg. [2]



	ϴ	d	a	α
1	ϴ1	Lc	La	0
2	ϴ2	0	Lb	180
3	0	d3	0	0
4	ϴ4	L4	0	180

Fig. 15 Tabla D-H Robot Scara 4 ejes.

Para ello, se realizan cálculos en función de la diferencia entre las distancias y orientaciones de un servo-motor y otro. Después de una resolución de fórmulas, se obtiene la siguiente matriz resultante para la cinemática inversa de un robot SCARA de 4 ejes:

$$\begin{bmatrix} 1.0\cos(\theta_1+\theta_2-\theta_4) & -1.0\sin(\theta_1+\theta_2-\theta_4) & 0 & la\cos(\theta_1)+lb\cos(\theta_1+\theta_2) \\ 1.0\sin(\theta_1+\theta_2-\theta_4) & 1.0\cos(\theta_1+\theta_2-\theta_4) & 0 & la\sin(\theta_1)+lb\sin(\theta_1+\theta_2) \\ 0 & 0 & 1.0 & -1.0d_3-1.0l_4+1.0lc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 16 Matriz Euler

En esta ecuación conocemos los valores de *la*, *lb*, *lc* y *l4* que equivalen a la distancia de cada brazo o *link*. Los datos que desconocemos son ϑ1, ϑ2, ϑ4 y d3, que equivaldrían a los ángulos de rotación de los servos 1, 2 y 4 (los cuales son rotacionales) y la altura a la que está situada el manipulador.
Para resolver las ecuaciones se utilizó la función "nsolve".

Problema: Esta manera no tiene en cuenta que solo se utilizan servos de 180º por lo que la función "nsolve" ofrece múltiples soluciones con grados superiores a 180, las cuales se podrían limitar. Finalmente, se descartó porque para conseguir una solución óptima de forma rápida es mejor un método geométrico cómo el que se explica a continuación.

2. Geometría 2D: Cálculo de los ángulos de los servos en un espacio de 2 dimensiones.

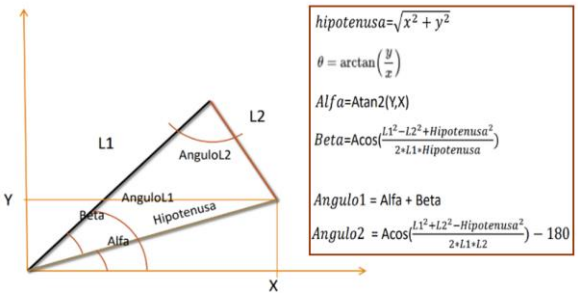


Fig. 9: Gráfico y fórmulas de obtención de ángulos.

L1 y L2 corresponden a la longitud de los brazos.

Para calcular el ángulo del manipulador (servo que gira las cartas) se suman o restan los ángulos iniciales y finales de los servos del brazo y el antebrazo. Se calcula de forma que siempre que esté orientado en el eje 'Y'.

A6. FUNCIONES CINEMÁTICA INVERSA

A continuación, se describen las **funciones principales** de alto nivel que forman parte de este módulo:

1. Movimientos del brazo:

- **Calculate(x, y):** Calcula los ángulos de rotación para los servos 1 y 2 en función de una posición (x,y) mediante el uso de las fórmulas mencionadas anteriormente.

  También calcula el ángulo del manipulador para que siempre esté orientado en el eje 'Y'. Si el brazo se encuentra en la región derecha de la mesa (+X,+Y), el manipulador siempre se orienta en +Y con ángulo 0. En cambio, si el brazo se encuentra en la región izquierda de la mesa (-X,+Y), el manipulador siempre se orienta en –Y con ángulo 180. Esto se hace así ya que es un servo de 180 grados y no puede colocarse siempre orientado a +Y.

- **MoverBrazo(x , y):** Obtiene los ángulos que tiene que mover el robot mediante la función "Calculate(x,y)" y envía las órdenes correspondientes al Arduino UNO para mover los servos. Esto mediante una comunicación serial.

- **MoverBrazo2(ángulo1 , ángulo2, ángulo3):** Envía las órdenes correspondientes al Arduino UNO para mover los servos con los 3 ángulos pasados por parámetro.

- **MoverEje3(posZ):** Envía al Arduino UNO la posición Z en la que se ha de colocar el robot mediante el motor DC.

- **MoverEje4(ángulo):** Envía al Arduino UNO la orientación en que se ha de colocar el manipulador. Tiene en cuenta la suma de la orientación más el ángulo del manipulador. Si la suma no se encuentra en el rango de 0 a 180 entonces se modifica la suma.

- **Standby():** Envía al Arduino UNO la orden de colocarse en posición de 'Standby', posición para no obstaculizar el juego.

2. Gestión de 1 carta:

- **CogerCarta(x , y , z , angEje4):** Secuencia de movimientos para recoger una carta. Recibe la posición de la carta y su orientación.

- **DejarCarta(x , y , z , angEje4):** Secuencia de movimientos para dejar una carta. Recibe la posición dónde se coloca la carta y su orientación.

- **QuemarCarta(x , y , z , angEje4):** Secuencia de movimientos para quemar una carta. Recibe la posición dónde se colocan las cartas quemadas y su orientación.

3. Gestión de las cartas centrales:

- **ColocarCartaCentral(x , y , z , angEje4):** Secuencia de movimientos para colocar una carta en el centro de la mesa orientada en posición +Y. Recibe la posición central de la carta y su orientación.

  La secuencia consiste en:
  1. Coger una carta de la baraja.
  2. Dejar carta en el objeto que la gira.
  3. Recoger carta de la bandeja del objeto.
  4. Mover carta a posición pasada por parámetro.
  5. Orientar y dejar la carta.

- **Colocar3CartasCentrales(x , y , z , angEje4):** Secuencia de movimientos para colocar 3 cartas en centro de la mesa orientadas en posición +Y. Se utiliza la función 'ColocarCartaCentral'. En el póker a este proceso se le denomina *flop*.

- **Colocar4CartaCentral(x , y , z , angEje4):** Secuencia de movimientos para colocar la cuarta carta central. Se utiliza la función 'ColocarCartaCentral()'. En el póker, a este proceso, se le denomina *turn*.

- **Colocar5CartasCentral(x , y , z , angEje4):** Secuencia de movimientos para colocar la quinta carta central. Se utiliza la función 'ColocarCartaCentral()'. En el póker a este proceso se le denomina *river*.

4. Jugadores:

- **PosicionJugador(nJugadores)**: Se pasa como parámetro el número de participantes. La función calcula y devuelve las posiciones geométricas (X,Y) de cada jugador.

- **DarCartaJugador(x, y, ángulo):** Secuencia de movimientos para dar las 2 carta a un jugador sin haber de girarlas. Recibe la posición de colocación de las cartas y su orientación.

5. Recolección de objetos:
   - **DejarItem()**: Envía al Arduino UNO una orden que activa el solenoide y deja pasar aire a la ventosa. Esto significa que, si hay algún objeto acoplado a la ventosa, este es expulsado.