Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Informe Final

Lanzador de Volleyball

Integrantes: Aníbal Fuentes Jara
Daniel Guzmán Pradel
Tomás Lara Aravena
Simón Sepúlveda Osses
Hans Starke Díaz
Profesores: Matías Mattamala Aravena
Miguel Patiño Sazo
Santiago, Chile

Índice de Contenidos

Lista de Figuras

Resumen

En el presente informe se detalla el trabajo realizado durante el curso de taller de diseño Beauchef Proyecta, en el cual se trabajó en el proyecto de lanzador de balones de volleyball.

El proyecto abordado se dividió en tres equipos durante el presente semestre, dos equipos de Ingeniería Mecánica, uno dedicado a la base del lanzador de balones (equipo de omni-wrist) y otro dedicado al lanzador en si (equipo lanzador); el tercer equipo corresponde al equipo eléctrico, cuyo trabajo se detalla en este informe. El trabajo realizado en este equipo esta orientado a la integración del sistema como un todo, centrándose principalmente en tres áreas: el equipo de visión computacional, orientado a la detección de personas dentro de la cancha con el fin de realizar un sistema inteligente que sea capaz de arrojar los balones a los jugadores, el equipo de motores y omni-wrist, encargado del diseño y construcción del omni-wrist, programación de motores, además del cálculo de trayectorias necesarias para disparar el balón en la posición entregada por el equipo de visión, y el equipo de comunicación, encargado de comunicar ambas componentes mediante Wi-Fi.

El resultado del trabajo del semestre corresponde a una serie de módulos que serán útiles para el lanzador final, además de un prototipo a escala, que integra la parte de visión computacional con el omni-wrist, mediante comunicación serial.

1. Introducción

En el presente informe se muestra el desarrollo de un lanzador de pelotas de Volleyball. Este corresponde a un proyecto multidisciplinario desarrollado en el curso taller de diseño Beauchef Proyecta.

Este proyecto nace en el contexto de que en la universidad existen entrenamientos de volleyball, tanto de la rama de volleyball como de los cursos DR de volleyball, estos entrenamientos se realizan 8 veces a la semana con una duración de aproximadamente una hora y media cada uno, dando un total de 12 horas de juego por semana. Además en este contexto se tiene que existen distintas rutinas de entrenamiento, las cuales son más básicas para el DR, mientras que más exigentes para el DR.

Así, se identifican las siguientes problemáticas:

- Acciones de juego disminuyen su velocidad al ser las mismas personas que participan del entrenamiento las que deben realizar los lanzamientos.
- Asistencia del entrenador en lanzamientos consecutivos del balón producen desgaste físico en él debido a las largas jornadas de entrenamiento.
- Máquinas de lanzamiento de balones mediante rodillos producen daño en estos, lo que implica un alto costo monetario para la institución.

Así, con el contexto y las problemáticas identificadas surge la idea de realizar un lanzador de balones de volleyball que pueda ser utilizado en las prácticas de la universidad.

El proyecto a lo largo del semestre fue abordado por tres equipos, dos de ellos de Ingeniería Mecánica, los cuales se encargaron de diseñar y construir la parte mecánica del lanzador, en particular, un equipo se encargó de la base del lanzador, en el que se desarrolla el mecanismo omni-wrist, un sistema mecánico que ofrece dos grados de libertad para apuntar el lanzador; mientras que el segundo equipo mecánico se encargó de diseñar y construir el brazo del lanzador, que irá montado al omni-wrist y que se encarga de tomar el balón y mediante un mecanismo con resortes similar a una honda disparar el balón.

Por otro lado, el tercer equipo corresponde al equipo de Ingeniería Eléctrica (detallado en este informe), este equipo se encarga de integrar distintas piezas por separado para realizar el sistema como un todo. El trabajo se divide en tres módulos: detección en cancha, movimiento de motores y comunicación entre los dos anteriores.

La detección en cancha se realiza con el uso de dos modelos CNN: `YOLOv3` y `Tiny YOLOv3`, el movimiento orientador del lanzador con Motores Stepper controlado por Arduino y la comunicación entre ambos sistemas con el microcontrolador ESP8266 mediante Wi-Fi.

Del trabajo realizado por el equipo eléctrico se obtienen una serie de resultados:

- Diseño de la base del lanzador (omni-wrist) en formato stl.
- Prototipo de la base: impresa en 3D, acoplada con servomotores para su movimiento y con un programa para su control.
- Sistema de visión computacional: implementado en python y con el uso de herramientas de Deep Learning; nos permite identificar a las personas en la cancha, realizar un tracking de las mismas y seleccionarlas para apuntar hacia ellas.

- Comunicación Wi-Fi: esquemático y códigos para comunicación Wi-Fi entre el sistema de visión computacional (mediante el lenguaje de programación python) y un arduino, esto con el uso del hardware ESP8266 conectado al arduino.
- Códigos para control de motores stepper y calibración mediante fines de carrera.
- Cálculos de trayectorias y ángulos para el lanzamiento del balón.
- Demo del sistema con prototipo de la base.
- Documentación en github.

Este trabajo se encuentra documentado en el github: https://github.com/anibalfuentesjara/EL5004-Volleyball

2. Solución propuesta

En base al problema planteado en la sección anterior se plantea como solución un sistema que utilice análisis de imágenes con el fin de a jugadores en la cancha para posteriormente, utilizando un mecanismo de lanzamiento, realizar un disparo de balones hacia el objetivo deseado, indicado a su vez a través de una interfaz amigable al usuario. La solución propuesta se presenta en la figura 1, cuyos diferentes módulos serán detallados a continuación.

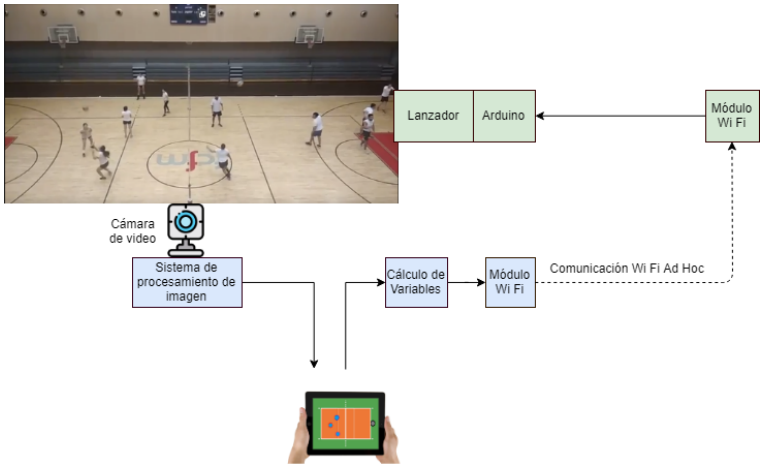
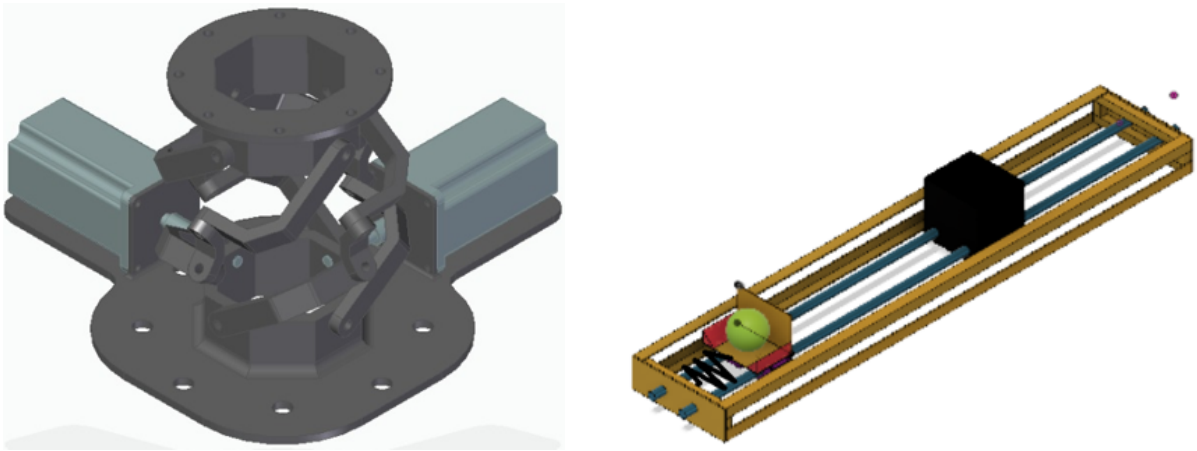


Figura 1: Esquema de la solución propuesta

Los módulos utilizados de acuerdo a la secuencia en que se utilizan corresponden a los siguientes:

- **Cámara:** Se utiliza una camára de gran ángulo de apertura con el fin de poder capturar la cancha en su totalidad. El sistema plantea el posicionamiento de la cámara en un punto elevado sobre el medio de la cancha, tal como se presenta en la figura 1
- **Sistema de procesamiento de imagen:** El sistema de procesamiento de imagen utiliza las imágenes capturadas por la cámara para detectar a las personas y delimitarlas mediante *bounding boxes*, adicional a la detección de personas se debe realizar la determinación de sus coordenadas, con el fin de que esta pueda ser utilizada para el cálculo de variables de disparo.
- **Interfaz con usuario:** En la interfaz de usuario, el usuario, en este caso quien guíe el entrenamiento, debe seleccionar el objetivo a enviar el balón.
- **Cálculo de variables de disparo:** En este módulo se debe realizar el cálculo de los ángulos de giro para los motores en el lanzador de modo que el cañón presente la orientación necesaria y el lanzamiento se realice a una velocidad tal que permita alcanzar el objetivo.
- **Módulo Wi Fi:** El módulo Wi Fi debe enviar la información de las variables de disparo al módulo Arduino que se encuentra en el lanzador.
- **Arduino:** La placa Arduino que se encuentra en el lanzador se encarga del manejo de los motores con el fin de realizar el lanzamiento.

- **Lanzador:** El lanzador corresponde a la estructura desarrollada por el equipo mecánico que permita el lanzamiento de balones de Voleyball. Imágenes del mecanismo se presentan en las figura 2a y 2b , para el mecanismo Omniwrits que permita apuntar y el lanzador, respectivamente.



(a) Omniwrist (b) Lanzador

Figura 2: Mecanismo de lanzador

3. Metodología de trabajo

Para llevar a cabo el trabajo se dividen las tareas en equipos funcionales, cada uno encargado de un área del proyecto. Estos equipos se conforman como Equipo de omni-wrist y motores, Equipo de visión, y Equipo de comunicaciones. A continuación se presenta la metodología seguida por cada uno de estos equipos presentando los elementos más importantes desarrollados en cada área.

3.1. Equipo de Visión

El equipo de visión fue el encargado de entregar al lanzador las coordenadas de los jugadores en la cancha. Esta tarea se divide en tres súbmodulos: detección de personas, *tracking* y transformación de homografía. Si bien los avances del equipo de visión fueron anexados al github del proyecto estos se encuentran por separado en el github https://github.com/Jackerz312/Person_Volley.

En la Figura 3 se presenta un esquema del sistema de visión computacional, en el se observa la cancha de volleyball, donde sobre ella se ubica una cámara, la cual estará conectada a un computador o un sistema embebido (que será el punto central de procesamiento del lanzador de balones de volleyball), encargado de procesar las imágenes y enviar la información al lanzador mediante Wi-Fi.

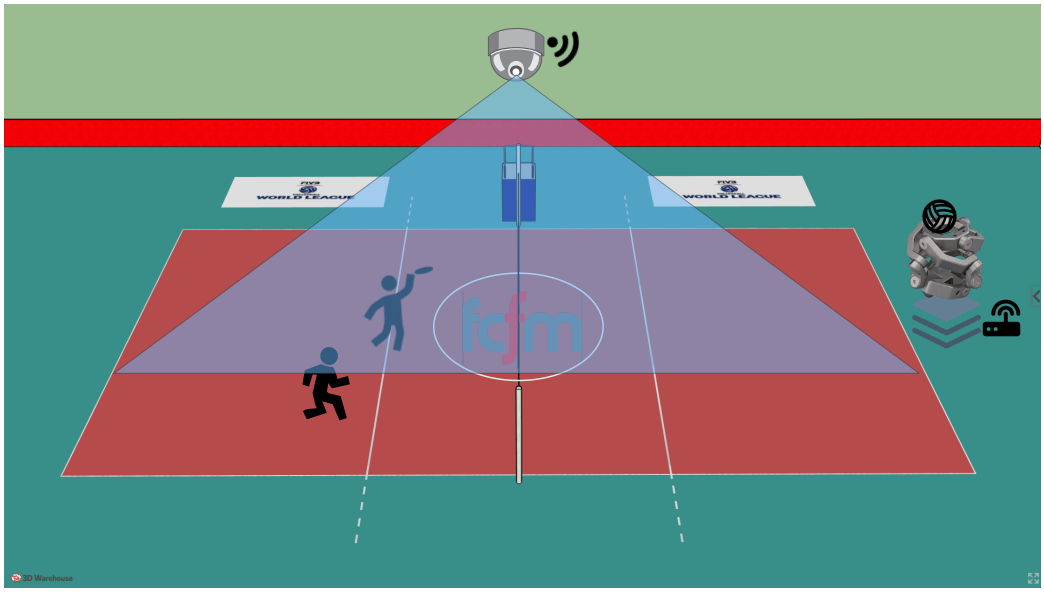
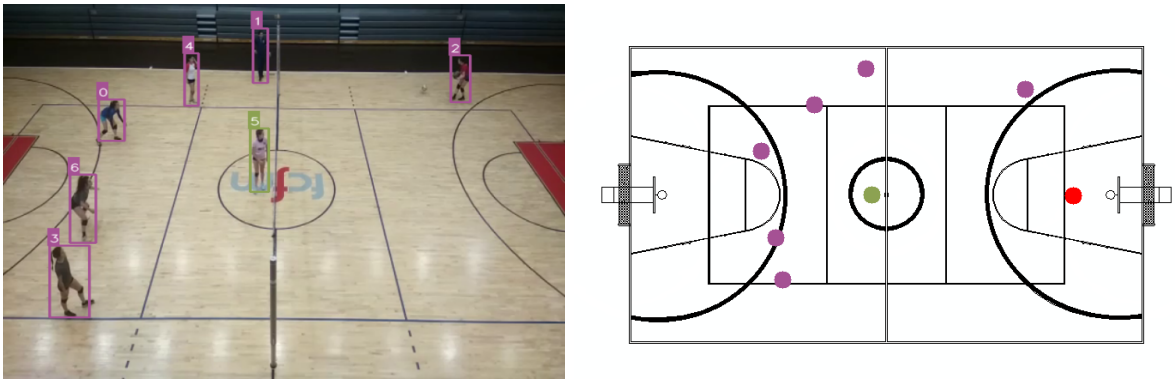


Figura 3: Esquema sistema de visión computacional.

El software de visión computacional presentará un funcionamiento como el mostrado en la Figura 4, a la izquierda se muestra la imagen capturada por la cámara, sobre esta imagen se realiza la detección de las personas mediante uso de *Deep Learning* (detección), además, para cada persona detectada se realiza un seguimiento en el tiempo (*tracking*) para identificar a cada persona de manera separada; finalmente, con las detecciones realizadas se realiza una transformación de homografía que nos permite proyectar a estas personas a sus coordenadas dentro de la cancha, tal como se muestra en la figura 4b, este mapeo nos facilita además el cálculo del lanzamiento del balón. Junto a esto se tiene que, tal como se muestra en la figura, es posible seleccionar a una

persona para lanzarle el balón.



(a) Detecciones. (b) Proyección a la cancha.

Figura 4: Diagrama del software de visión computacional.

3.1.1. Detección

Uno de los aspectos fundamentales para el proyecto, era la necesidad de contar con la posición de las personas dentro de la cancha. Para esto, se lleva a cabo un análisis de distintas tecnologías que se pueden utilizar, tales como LADAR, RTLS y visión computacional, las cuales se comparan en la Tabla 1, donde se observan las grandes ventajas del sistema de visión por computador frente al resto de opciones, por lo cual es finalmente escogido.

Tabla 1: Ventajas y desventajas de sistemas de detección de personas.

Sistema	Ventajas	Desventajas
RTLS	Alta precisión	- Necesidad de tags - Poca escalabilidad
LIDAR	-	- Rango acotado - Desconocimiento de objetos - Poca escalabilidad - Baja velocidad de escaneo
Vision Computacional	- **Conocimiento de objetos que se observan** - **Alta escalabilidad** - **Capacidad de realizar seguimiento**	- Posibilidad de fallo con baja iluminación.

El modelo de visión computacional escogido corresponde a YOLOv3 [1], debido a alta precisión y velocidad capaz de ser utilizado en tiempo real. Para su uso se entrena en COCO dataset [2] únicamente en datos de personas. Además se entrena una versión liviana llamada Tiny YOLOv3, para su uso en sistemas embebidos. La precisión de los modelos entrenados se muestra en la Tabla 2 utilizando como métrica el *mean average precision*.

3.1.2. Tracking

Para el *tracking* de personas se utiliza una modificación a implementación de un *tracker* por centroides [3]. Este *tracker* consiste en relacionar las detecciones entre cuadros mediante las distan-

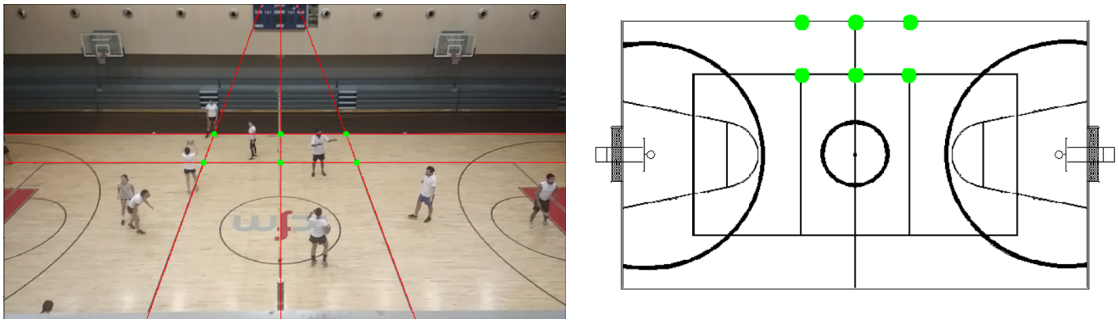
Tabla 2: *Average precision* de modelos entrenados.

Modelo	*Mean Average Precision (%)*
YOLOv3	73.78
Tiny YOLOv3	52.52

cias entre los *bounding boxes*, donde se asocia un ID al objeto el cual tenga menor distancia entre sus centroides.

3.1.3. Transformación de homografía

Para realizar el mapeo de las personas detectadas a la cancha se utiliza el método de transformación de homografía [4]. Este método consiste en encontrar una transformación que sea capaz de relacionar y transformar dos imágenes, a través de ciertos puntos coincidentes. Para ello se consideran los puntos verdes señalados en la figura 5.



(a) `Imagen capturada` (b) `Figura de la cancha`

Figura 5: Puntos seleccionados para realizar la transformación de homografía.

Se seleccionan estos puntos debido a que son más fáciles de estimar calculando las intersecciones entre líneas (figura 5(a)), sin embargo, si se tiene la cámara fija es posible seleccionar otros puntos para calcular la transformación de homografía y calcular la transformación solo una vez.

Ya con estos puntos es posible encontrar la transformación mediante la función de openCV findHomography. Posteriormente, con esta transformación se pueden proyectar puntos de una imagen a otra, tal como se muestra en la figura 4.

3.2. Equipo de Omni-Wrist y Motores

El trabajo del equipo de Omni-Wrist y Motores se centró principalmente en el desarrollo de un mecanismo de control del movimiento para el dispositivo "orientador" del lanzador de balones de volleyball, es decir, el *Omniwrist III* (cuya única referencia es el video en [5]), esto puesto que se tuvo un mayor contacto con este equipo mecánico, además de que para llevar a cabo avances en el módulo lanzador, se debía contar con especificaciones del mecanismo desarrollado por el equipo mecánico correspondiente. El omniwrist III corresponde a un sistema mecánico, que a través de dos motores logra apuntar en los ejes de libertad Pitch y Roll.

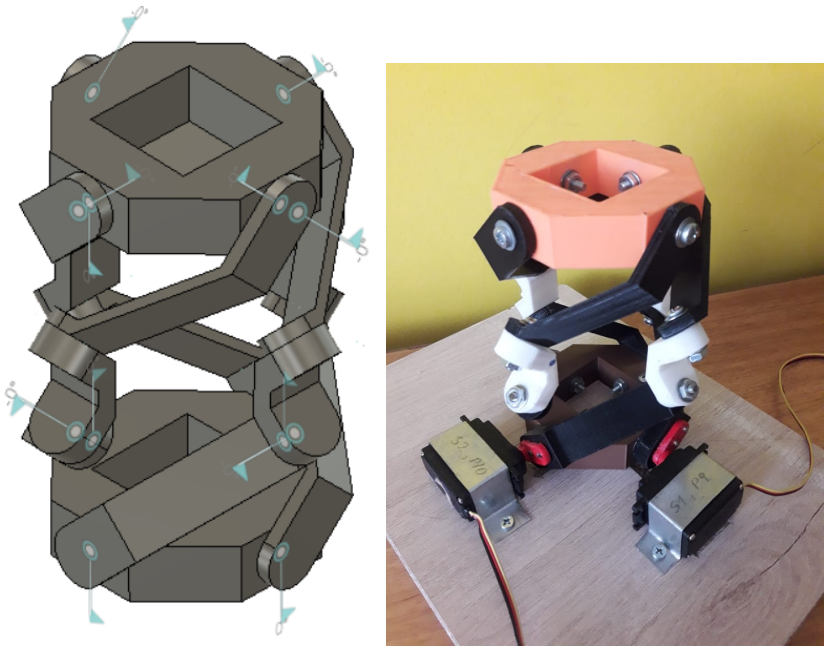
Adicionalmente, este equipo desarrolló los cálculos de trayectoria necesarios para determinar los movimientos a realizar por los mecanismos del lanzador. En las siguientes sub secciones se presentan los principales desarrollos realizados por el equipo de motores, los que concluyen con la construcción del prototipo presentado al cuerpo docente.

La documentación de estas secciones se encuentra en el github del proyecto.

3.2.1. Construcción de Omniwrist

Con el fin de poder realizar un prototipo que permita probar los principales aspectos del proyecto, como la capacidad de seguimiento de objetivos y la comunicación a través de Wi-Fi, se realiza un modelo de Omniwrist en el programa Fusion360, para posteriormente imprimir este en las impresoras 3D disponibles en Fablab.

El modelo desarrollado en Fusion360 se presenta en la figura 6(a), mientras que el modelo impreso en 3D se presenta en la figura 6(b). Una vez impreso este modelo fue acoplado a una base y se utilizaron dos servomotores para su control, para fines de una demo.



(a) Modelo diseñado. (b) Modelo construido.

Figura 6: Modelo de omni-wrist III diseñado en fusion 360 e impreso en 3D.

3.2.2. Movimiento con Motores Stepper

El movimiento de los mecanismos de *Omniwrist* y lanzador fue definido a lo largo del semestre a través del trabajo junto al cuerpo docente y se decidió la utilización de motores stepper, fundamentado principalmente en tres razones. En primer lugar, el torque que presentan estos motores es superior al presentado a otros tipos de motores analizados, lo que se consideró apropiado principalmente para el mecanismo de lanzador, donde este motor, sería forzado a moverse por un resorte estirado una distancia considerable. En segundo lugar, este tipo de motores alcanza una

gran precisión en su giro, especialmente al considerar medios pasos en su programación, lo que es útil principalmente en el movimiento que realiza el *Omniwrist*, destinado a apuntar a jugadores(as) objetivo; Una desventaja de este tipo de motores con respecto a otros corresponde a su falta de retroalimentación, lo que llevaría a complicaciones en caso de que este vea forzado su movimiento, sin embargo, esto se considera improbable debido al torque de los motores y en caso de presentarse sería superado a través del uso de un encoder. Como última razón, la disponibilidad de tres de estos motores en la universidad facilitó el proceso de pruebas en comparación con la necesidad de realizar la cotización y compra de otros motores.

Para el movimiento de los motores stepper se utilizó la librería de Arduino AccelStepper [6]. Esta librería permite mover motores Stepper de distintas configuraciones, permitiendo indicar, por ejemplo, el ángulo objetivo, y la velocidad máxima a alcanzar para llegar a ese ángulo. Estas correspondieron a las variables manipuladas para manejar el movimiento de los motores Stepper. El equemático a implementar para el movimiento de motores stepper utilizando la librería AccelStepper se presenta en la figura 7, donde es posible observar los distintos componentes necesarios para el funcionamiento del motor, correspondientes a la fuente de poder, el controlador (driver) del motor y la placa Arduino que entregue las señales de control al driver.

Utilizando esta configuración se realizan pruebas de movimiento de motores Stepper, utilizando diferentes sentidos de giro, ángulos objetivo y velocidades máximas de movimiento, con el fin de analizar el comportamiento de estos en relación con el torque y precisión de movimiento.

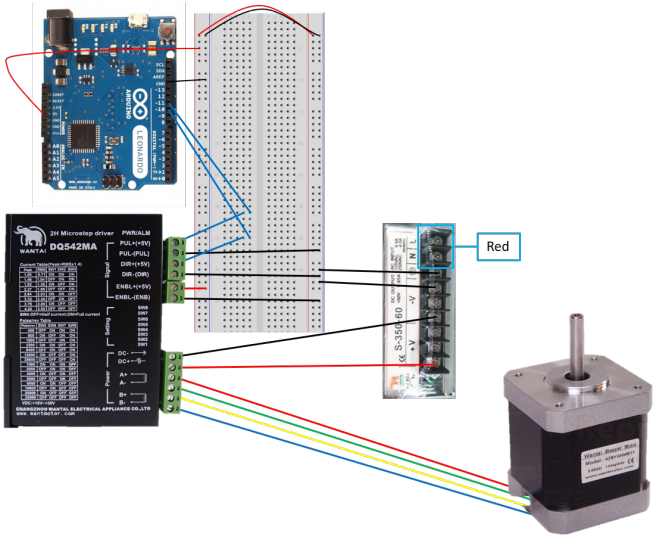


Figura 7: Esquemático de conexión de motores Stepper.

3.2.3. Trabajo con fines de carrera

Como se mencionó en la sección anterior, los motores Stepper a utilizar no poseen realimentación, por esta razón se desarrolla una solución para el movimiento inicial de los motores que incluya el uso de fines de carrera, dispositivos que una vez accionados cambian su estado. En el caso de las pruebas realizadas se utilizan fines de carrera de tipo Makerbot, los cuales al ser conectados a la placa Arduino pasan del estado HIGH a LOW al accionarse, y entregan esta información a la placa.

Se desarrolla un programa en Arduino que permita, a través del uso de interrupciones de tipo FALLING, realizar la calibración de las posiciones iniciales del motor, de esta manera, antes de inicial los movimientos de disparo del motor, se realiza una calibración, correspondiente a conocer la posición inicial de los motores. Para esto, el programa desarrollado realiza un movimiento inicial lento hasta que se accione uno de dos fines de carrera, permitiendo determinar si el motor se encuentra en su posición 0 o 180 grados, para luego conocer las coordenadas bajo las que debe realizar el movimiento. El esquemático de conexiones utilizando dos fines de carrera se presenta en la figura 8.

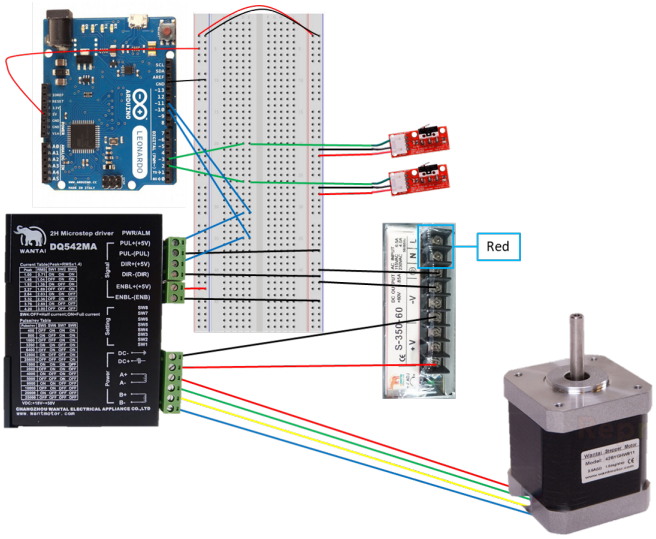


Figura 8: Esquemático de conexión de motores Stepper con fines de carrera.

3.2.4. Cálculo de trayectorias

Para la realización del cálculo de trayectorias se trabaja directamente con el equipo de visión, con el fin de utilizar los puntos (jugadores) determinados mediante mecanismos de visión computacional para calcular las condiciones del disparo del lanzador, correspondiente al posicionamiento del *cañon* y la velocidad de salida que debiese tener la pelota en su trayectoria hacia el objetivo.

En relación al cálculo de trayectorias, en primer lugar se realiza la determinación de la inclinación que debe tener el *cañon* al apuntar hacia el objetivo, esto se realiza de acuerdo al modelo plano presentado en la figura 9, donde el punto (x_0, y_0) representa la posición del lanzador considerada fija, y el punto (x_f, y_f) corresponde a la posición del objetivo seleccionado. De acuerdo a este esquema el ángulo a encontrar corresponde a α.

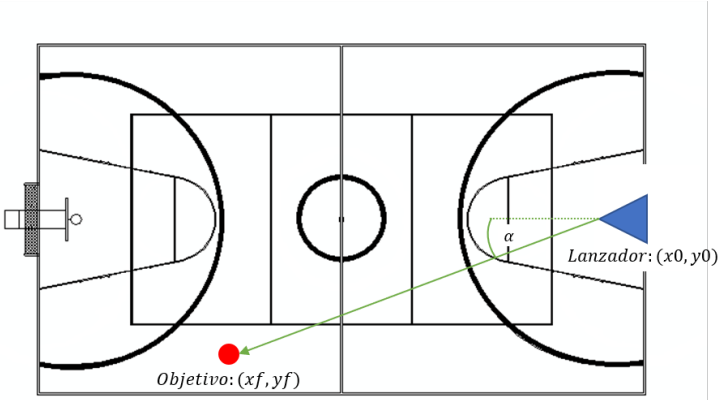


Figura 9: Esquema para cálculo de orientación.

De esta manera el cálculo de α se realiza mediante la ecuación (1).

$$\alpha = arctan(\frac{y_f - y_0}{x_f - x_0}) \tag{1}$$

Posterior a esto, es posible reducir el problema de tres a dos dimensiones, donde la distancia que debe horizontal que debe recorrer el balón corresponde a la flecha de color verde en la figura 9. A partir del tratamiento bi dimensional del problema los datos a encontrar corresponden a la elevación del cañon y la velocidad de lanzamiento. Para esto, se define una parábola utilizando tres puntos, en primer lugar, se utiliza como origen el lanzador y como destino la posición del objetivo, como tercer punto se utiliza una posición sobre la malla de mitad de cancha, que posee una altura oficial de aproximadamente 2.6 metros; De esta forma, utilizando la función *polyfit* de grado dos del módulo *numpy* de Python es posible obtener los coeficientes de la parábola definida por estos puntos. Utilizando la definición de la parábola es posible obtener la elevación necesaria derivando la ecuación de esta y calculando la tangente, lo que se traduce en evaluar la expresión (2) en el punto donde se encuentra el lanzador donde a y b corresponden al parámetro de segundo y primer grado de la parábola y el ángulo de elevación β se presenta en la figura 10

$$\beta = tan(2ax_0 + b) \tag{2}$$

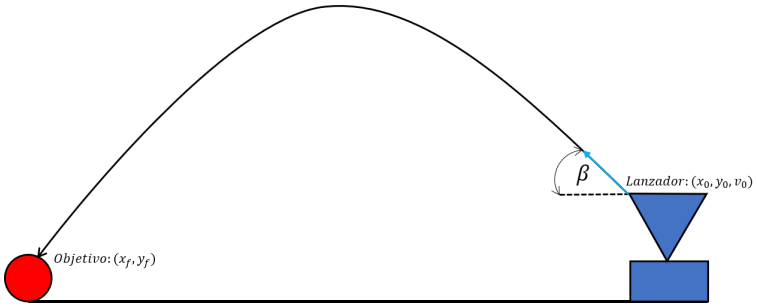


Figura 10: Esquema para cálculo de elevación y velocidad.

Para el caso de la velocidad es necesario utilizar las ecuaciones cinemáticas del balón en su

trayectoria parabólica. A partir de esto es posible encontrar la velocidad que se debe imprimir al balón en la salida para que este llegue al objetivo deseado, la que se presenta en la ecuación (3), donde g corresponde al valor de la aceleración de gravedad, utilizando en este caso un valor de 9.81.

$$v_0 = \sqrt{\frac{(0.5 * gx_f)^2}{cos(\beta)^2(y_f - y_0 - tan(\beta)x_f)}} \tag{3}$$

3.2.5. Ángulos de Giro de motores

Con el fin de realizar una demostración final del prototipo desarrollado se utiliza el modelo impreso del Omniwrist junto a motores servo; se utiliza este tipo de motores por la facilidad de controlar su posición angular, además de que se contaba con dos de estos a disposición del equipo de trabajo.

Para encontrar el ángulo de los motores es necesario realizar una transformación desde las coordenadas de orientación (α) y elevación (β) presentadas en la sección anterior a ángulos de orientación de ambos motores servo, los cuales se encuentran acoplados al Omniwrist de la manera presentada en la figura 6(b).

Para realizar la transformación se realiza una calibración de los motores, tal que estos posean inicialmente un ángulo de calibración correspondiente a la posición vertical del Omniwrist, posteriormente se debe realizar un transformación desde el vector normal a la superficie del Omniwrist, correspondiente al vector que define la posición del lanzador hasta los ángulos de los motores, considerando que el movimiento en un ángulo arbitrario θ desde el reposo de los motores, influye en la posición del lanzador en un factor de 2θ, tal como se muestra en la figura 11.

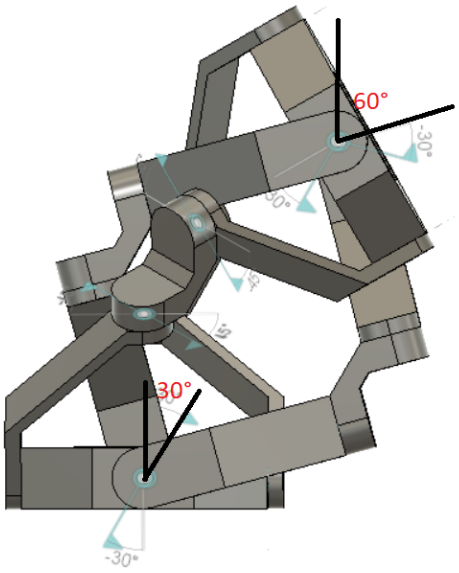


Figura 11: Relación entre el ángulo de un motor respecto al reposo, y el ángulo del lanzador.

El ángulo de giro para los motores 1 y 2 indicados en la figura 6(b) (siendo el motor 1 el motor derecho de la figura y el motor 2 el motor izquierdo), se presenta en las ecuaciones (4) y

(5), respectivamente. Donde el ángulo θ corresponde al ángulo en coordenadas esféricas, es decir, $\theta = 90^o - \beta$, de la misma forma que $\phi = -1 * \alpha$

$$\theta_1 = -arcsin\left(\frac{sin(\theta * \pi/180) * cos(phi * \pi/180)}{2}\right) * 180/\pi \tag{4}$$

$$\theta_2 = arcsin\left(\frac{sin(\theta * \pi/180) * sin(phi * \pi/180)}{2}\right) * 180/\pi \tag{5}$$

Las ecuaciones anteriores representan los ángulos de movimiento respecto al ángulo de reposo. Cabe destacar que los cálculos anteriormente presentados se realizan en el mismo sistema de visión computacional, utilizando código en Python, de esta manera es posible reducir el costo computacional asociado a la placa de Arduino, pues se envían directamente a este los ángulos de movimiento de los motores servo.

3.3. Equipo de Comunicación

El equipo de comunicación es el encargado de obtener las coordenadas en la cancha enviadas por el equipo de visión y comunicarlas al Arduino que controla los motores con el uso del módulo ESP8266. Este tiene una antena Wi-Fi con la cual puede recibir datos y escribirlos en serie a través de sus pines, lo cual se aprovecha para leer con Arduino y controlar los motores como se describe en la sección 3.2.2. Se elige este debido su bajo costo de \$3000 [1] y compatibilidad con otros microcontroladores.

La comunicación se da entonces entre un servidor local en Python y Arduino, teniendo como intermediario el ESP8266, cual recibe y envía entre ambas a través de la red Wi-Fi a la que se conectan los dispositivos usados.

3.3.1. ESP8266

El ESP8266 es un sistema en chip (SoC) desarrollado por la compañía China *Espressif* [7], cual contiene un microcontrolador y una antena transceptora de Wi-Fi. Este es el intermediario entre el computador que procesa la información visual de la cancha y el controlador de los motores, cual comunica a ambos mediante Wi-Fi.

El módulo se alimenta con 3.3V, cual es de suma importancia respetar puesto que un voltaje superior quema el módulo. Además, requiere corrientes de hasta $300mA$, por lo cual es requisito utilizar una fuente de alimentación para él o para el Arduino a cual va conectado. Para realizar las pruebas de este trabajo, se utiliza una fuente de 12VDC y 1A conectada al Arduino. En la Figura 12 se muestra el módulo y su mapeo de pines.

[1] https://amgkits.com/home/32-modulo-wifi-esp8266.html?search_query=esp8266&results=10

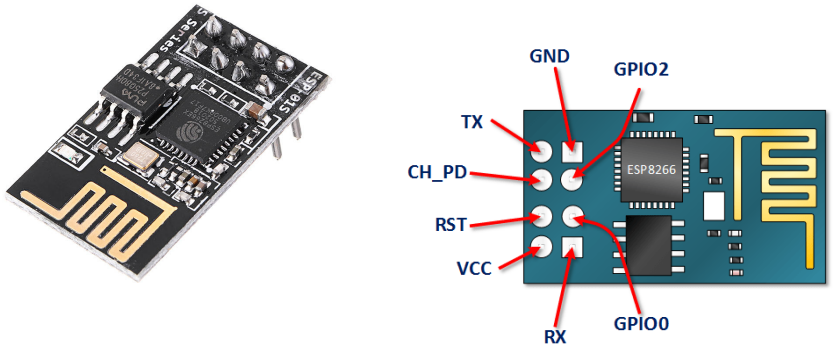


Figura 12: Módulo ESP8266

3.3.2. Comunicación con Python

La comunicación con el sistema de visión se realiza por un servidor local hecho en Python con el uso de la librería *socket*, cual envía las coordenadas de la cancha a los clientes, en este caso, el módulo ESP8266 cual posteriomente las envía a Arduino. Se realizan la comunicación con Python, puesto que en en este lenguaje es donde se implementa el procesamiento de imágenes.

Las coordenas de orientación y velocidad se reciben como un *string* separado por # de la forma $c_1\#c_2\#vel$ desde Python y se procesan en Arduino para obtener los tres valores como *float*.

3.3.3. Control de motores

Para comunicar el Arduino que controla los motores con el módulo ESP8266 se utilizan las librerías `SoftwareSerial`, nativa de Arduino y WeeESP8266 disponible en https://github.com/itead/ITEADLIB_Arduino_WeeESP8266. El esquemático de la conexión se muestra en 13.

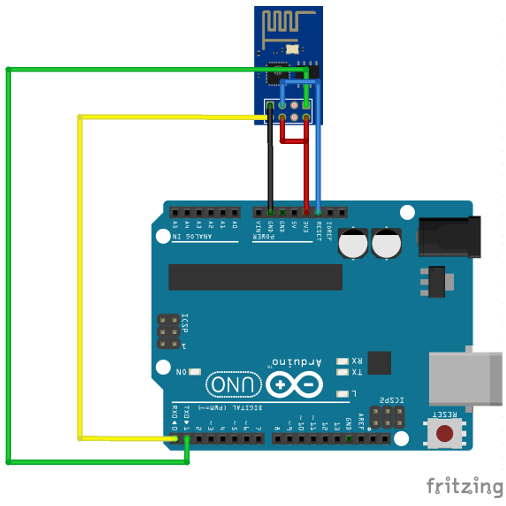


Figura 13: Conexión entre módulo ESP8266 y Arduino

El objetivo es recibir la información del módulo de visión y entregarlas al *script* controlador

que da la orientación al lanzador.

La librería SoftwareSerial se utiliza para realizar la comunicación por otros pines que son los dedicados para la comunicación de la placa Arduino (0 RX y 1 TX), con el fin de usar estos para el *debugging* del programa y ver información del estado de la conexión e información recibida, puesto que por defecto estos son los usados para la comunicación con el computador, con lo que si se usan con otro dispositivo, no es posible monitorear la información de salida de la placa. Respecto a esto, se cambia la conexión hecha en los pines 0 y 1 que se ven en la Figura 13 a los pines 2 y 3.

Por su parte, la librería `WeeESP` ofrece una API de fácil uso, con la cual consultamos la IP de nuestro dispositivo, el estado de la conexión TCP al *script* de Python y la información recibida para verificar el correcto funcionamiento de las partes.

La comunicación resulta exitosa entre ambas partes, sin embargo presenta una cantidad considerable de pérdida de datos, por lo que muchas veces se entregan coordenadas con dígitos cambiados o sin los separadores, imposibilitando mover correctamente el OmniWrist. Respecto a esto, se requieren métodos de corrección de error si no se desea cambiar la antena WiFi usada.

3.3.4. Comunicación Serial

Para confirmar la correcta coordinación entre el sistema de visión y el control de motores, se usa la librería `serial` de Python, cual permite el envío de datos en serie por puerto USB para recibir las coordenadas sin error. Para usarla, basta conectar el Arduino por USB al computador cual ejecuta el trabajo de visión y especificar el puerto USB al que se conecta. El archivo de ejemplo de uso de este se encuentra en https://github.com/anibalfuentesjara/EL5004-Volleyball/blob/master/Person_Volley/main_tracking.py. Por el lado de Arduino, solo es necesario usar la librería nativa `Serial` para leer repetidamente. Es requisito que ambos sistemas se comuniquen a la misma tasa de símbolos por segundo (*baud rate*); en este caso se elige una tasa de 9600 baud.

4. Conclusiones y trabajo propuesto

Durante el curso se pudieron implementar varios aspectos del lanzador de balones de volleyball, los cuales están asociados principalmente a la detección de personas dentro de la cancha y al control de la base del lanzador, para disparar el balón hacia los jugadores. Todo el desarrollo del trabajo quedó documentado en github.

Respecto al desarrollo del curso se recalca que el trabajo con los equipos mecánicos se vio dificultado por la falta de comunicación y contacto con estos equipos, y por la falta de un espacio en común que permita interactuar directamente con ellos.

Con respecto al trabajo propuesto existen diversos puntos a abordar:

- Integración con equipo mecánico: se debe integrar el desarrollo realizado con los motores stepper al equipo mecánico, de modo de utilizar los motores stepper en conjunto con los fines de carrera en el modelo del omni-wrist realizado por el equipo mecánico.
- Utilización de sistema embebido Nvidia Jetson Nano [8]. Este sistema permite realizar la detección a una tasa de 25 cuadros por segundo en Tiny YOLOv3, además de tener un tamaño compacto ideal para su conexión directa con la cámara para ser instalado en el recinto de manera poco invasiva. Además la adaptación de los códigos actuales es automática debido a que permite utilizar el lenguaje Python junto con las librerías ya utilizadas.
- Programa en android para control del lanzador e interfaz de usuario. Esto con la idea de que el lanzador pueda ser controlado de manera remota por el usuario, se pueden interpretar distintas modalidades, entre ellas, lanzamiento a ciertos jugadores seleccionados, lanzamiento a un punto de la cancha o lanzamiento aleatorio.
- Control del lanzador y estimación de velocidad de salida. Esto en conjunto con el equipo mecánico del lanzador.

Referencias

[1] J. Redmon and A. Farhadi, *"YOLOv3: An incremental improvement,"*CoRR, vol. abs/1804.02767, 2018.

[2] Lin TY. et al. (2014) Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham

[3] A. Rosebrock, *"Simple object tracking with OpenCV"*, Object Tracking, Tutorials, 2018. url: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

[4] S. Mallick, *"Image Alignment (Feature Based) using OpenCV"*, 2018. url: https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/.

[5] https://www.youtube.com/watch?v=BlKM2ayaBGA

[6] https://github.com/waspinator/AccelStepper

[7] Espressif Systems. *ESP8266EX Datasheet* https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Nov. 2018.

[8] Jetson Nano Brings AI Computing to Everyone, url: https://devblogs.nvidia.com/jetson-nano-ai-computing/