EVENT-CAM
Prototipo de cámara aérea

Garcia D.; Valdez I.; Herrera A.; Romero L. E.; Carrillo A. A.; Jüergens K. N.

Departamento de Ingeniería en Electrónica
Tecnológico de Monterrey

Abstracto

El documento presenta el desarrollo de un sistema de control bajo la plataforma de Raspberry PI, compuesto por una cámara suspendida en cuatro tensores automatizados que te permiten llegar de un punto (coordenada) a otro de manera precisa y con un movimiento estable. El sistema cuenta con transmisión livestream en una página web completamente funcional diseñada para el mismo. Como extra, el sistema se puede adaptar a cualquier tipo de estructura haciendo de este un prototipo factible y comerciable. Por supuesto, resuelve la problemática planteada a continuación.

Palabras clave: **Raspberry PI, Control, PWM, I²C, Spider contoller, Livestream, encoder, WIFI, motores DC, python.**

1. Introducción

Actualmente existen diferentes métodos para grabar eventos tanto culturales como deportivos, la desventaja que presentan algunos de estos es la necesidad de una estructura especifica que no les permite adaptarse a todo tipo de lugar, otro punto a mencionar es el angulo de grabación el cual es contante, evitando tener una experiencia interactiva y panorámica del evento. Por lo que se propone un sistema controlable y programable que se adecue a la necesidad de cualquier evento y/o cliente.

2. Propuesta inicial del modelo

2.1 Elementos a medir

Funcionamiento del protocolo I²C: protocolo de comunicación que permite mover los motores.

Funcionamiento del protocolo WIFI: protocolo que permite mandar desde la interfaz las instrucciones hasta la tarjeta.

Movimiento simultaneo de los motores: para llegar a un punto deseado es necesario mover más de un motor y así tener mayor control en los cuadrantes donde se mueve la cámara.

Precisión del movimiento: para conocer la distancia que deberá moverse se hace uso del *encoder* y una conversión matemática a esto se le

integro un controlador PID para obtener un movimiento suave y preciso.

Movimiento de la cámara: con la finalidad de tener un panorama completo de grabación existen dos grados de libertad unicamente para mover la cámara.

Transmisión vía livestream: Haciendo uso de la dirección IP de la cámara es posible transmitir en tiempo real.

Manejo de errores y excepciones: Se pensaron todos los escenarios probables de error y para evitar el fallo del sistema en cualquiera de estos casos se hizo uso de manejo de excepciones en el código y se establecieron medidas de seguridad.

2.2 Diagrama a bloques

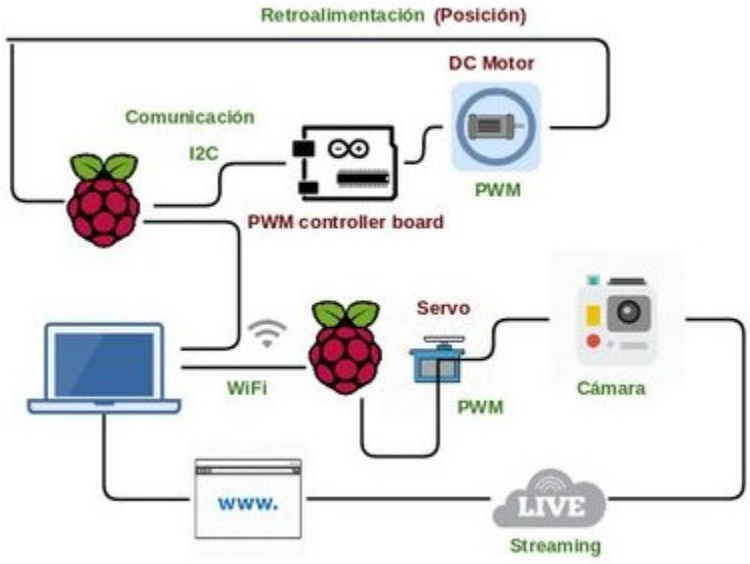

Figura 1. Diagrama de Bloques

En el diagrama de bloques se presta el flujo del funcionamiento del sistema, este se encuentra dividido en dos partes fundamentales la comunicación y la retroalimentación, puesto que desde las tarjetas se le indica que distancia moverse y el PID retroalimenta la entrada para llegar a una posición deseada, de la misma manera el Livestream es la manera de comprobar que la cámara esta funcionando de manera correcta.

Debido a la falta de pines con PWM que poseía la Raspberry PI, se integro un tarjeta controladora de PWM como se puede observar en el diagrama anterior.

2.3 Metodología

-Se diseño un modelo semejante a una cancha de football al igual que algunas piezas mecánicas y se probó la eficiencia de los motores.
-Se desarrollo un algoritmo capaz de interpretar coordenadas y convertirlas en señales PWM para mover cada motor.
-Mediante el protocolo I2C se comunicó la Raspberry Pi con la tarjeta controladora de PWM.
-Se implementó un controlador PID para hacer más eficiente el movimiento.
-Se desarrollo un sistema visión capaz de rastrear un objeto y seguirlo.
-Se Integró una página web para transmitir el contenido.

3. Desarrollo

La primera etapa para realizar este proyecto consistió en ver el funcionamiento de los motores, por lo que se escogieron motores DC que ya traían incorporado un encoder con el cual se pudo contar el numero de *rising edges* y así tener un control sobre la posición del motor.

Se usó este modelo que ya traía integrado el encoder y optocoplador, lo que hizo más sencillo su manejo.

Figura 2: Motor DC con encoder.

Se estableció una comunicación tipo I2C entre la Spider Controller (tarjeta controladora de PWM) y la Raspberry PI, de esta manera es posible indicarle el ancho de pulso y el numero de *steps* necesarios para mover los motores. Donde la RPi funge como *master* y la *Spider controller* como *Slave*, por lo que solo fue necesario establecer la comunicación en un sentido.

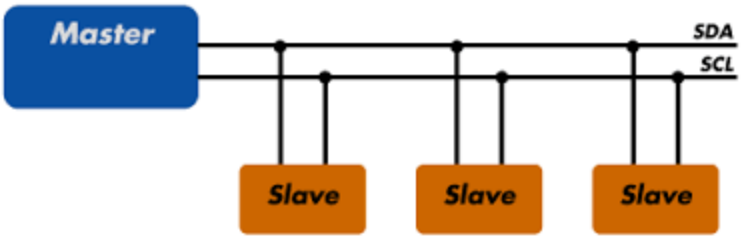

Figura 3: Diagrama I2C.

Con esta comunicación se logró mover los cuatro motores pero solo en una dirección, a lo que se recurrió al uso de un par de puentes H para modificar la dirección y sentido de cada uno de los motores, siendo la conexión de este como se presenta en la figura no. 3 donde

vc → 7,5 volts (proporcionado por una fuente de voltaje)
Vcc1 → 5 volts (proporcionado por la RPi)

y las señales de *enable* se conectaron a pines de las Raspberry Pi, para así controlar los motores.

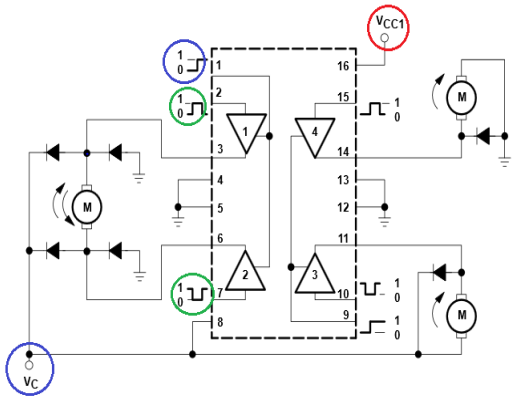

Figura 4: Conexión de un puente H.

Raspberry Pi B+ J8 Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Figura 5: Mapa GPIO de la Raspberry Pi

Una de las principales problemáticas con las que nos encontramos fue la estabilidad de la cámara, porque al mover esta también se movía la imagen capturada por la cámara. La solución, se imprimió y adapto un modelo estabilizador en 3D diseñado específicamente para sujetar una cámara del tamaño de una GoPro.

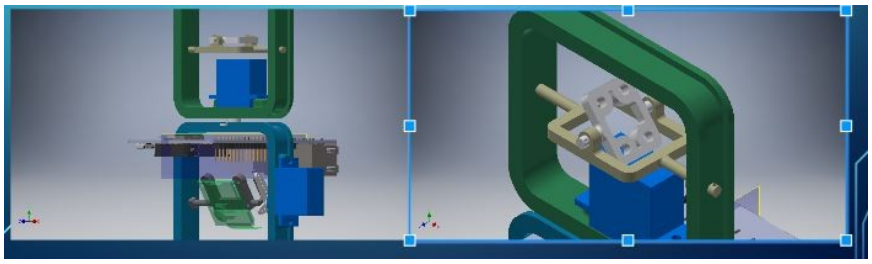

Figura 6: Modelo estabilizador.

Posteriormente se calculo toda la matemática para obtener las distancias necesarias de los cuatro postes al punto actual. La energía cinemática y potencial con respecto a la masa de un sistema como este es el siguiente:

$$E_k = 1/2 \cdot m \cdot \dot{x}^2 + 1/2 \cdot m \cdot \dot{y}^2 + 1/2 \cdot m \cdot \dot{z}^2$$

$$E_p = m \cdot g \cdot z$$

para obtener el modelo dinámico, es decir como se comportaran los objetos en movimiento dentro de este sistema, es necesario definir la velocidad externa del cambio de coordenadas $\dot{p} = [\dot{x} \quad \dot{y} \quad \dot{z}]^T$ y la velocidad interna del cambio de coordenadas $\dot{\phi} = [\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3]^T$ El jacobiano quedaría de la siguiente manera:

$$\dot{\phi} = J_* \cdot \dot{p}$$

$$J_* = \begin{bmatrix} J_{*11} & J_{*12} & J_{*13} \\ J_{*21} & J_{*22} & J_{*23} \\ J_{*31} & J_{*32} & J_{*33} \end{bmatrix}$$

Finalmente el modelo modelo matemático para un sistema de una cámara suspendida por un cable paralelo es:

$$u = G_v \cdot \ddot{\phi} + L_v \cdot \dot{\phi} + S_v \cdot (1 + \mu) \cdot M_*$$

Donde se obtuvo el modelo matemático y cinemática directa, primero estableciendo 3 grados de libertad para cada uno de los motores, lo equivalente a mover el motor sobre los ejes x, y, z. Siendo las variables para el movimiento de los motores θ1, θ2, θ3 y θ4. En base a esto solamente falto calcular la distancia

de cada uno de los vectores generados al punto deseado, haciendo uso del teorema de pitagoras.

```
#calculos distancias punto actual
ai =  math.sqrt(((xi-xa)**2)+((yi-ya)**2)+((zi-za)**2))
bi =  math.sqrt(((xi-xb)**2)+((yi-yb)**2)+((zi-zb)**2))
ci =  math.sqrt(((xi-xc)**2)+((yi-yc)**2)+((zi-zc)**2))
di =  math.sqrt(((xi-xd)**2)+((yi-yd)**2)+((zi-zd)**2))
#Calculos distancias para llegar al punto deseado
af =  math.sqrt(((xf-xa)**2)+((yf-ya)**2)+((zf-za)**2))
bf =  math.sqrt(((xf-xb)**2)+((yf-yb)**2)+((zf-zb)**2))
cf =  math.sqrt(((xf-xc)**2)+((yf-yc)**2)+((zf-zc)**2))
df =  math.sqrt(((xf-xd)**2)+((yf-yd)**2)+((zf-zd)**2))
#Distancias para llegar al punto deseado
adif=af-ai
bdif=bf-bi
cdif=cf-ci
ddif=df-di
a=int(adif*222)
b=int(bdif*222)
c=int(cdif*222)
d=int(ddif*222)
```

Una vez obtenidos los valores correspondientes a cada una de las distancias necesarias en los cuatro postes, se realizó una conversión de cm a *steps* de esta manera de logró mover un número de vueltas adecuados para llegar hasta el punto deseado.

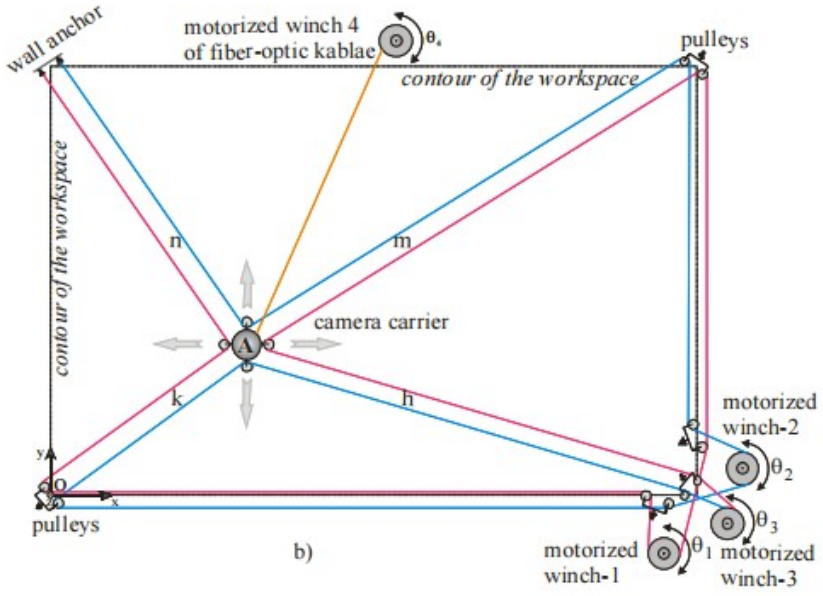

Figura 7: Mapa de los motores.

Después de saber a que punto se desea mover, se encontró la problemática de que el robot no tendría idea de como saber en que posición se encontraba, a lo que recurrimos a guardar las posiciones anteriores en un archivo .txt
La manera en la que establecemos las coordenadas sobre la maqueta consiste en lo siguiente, se establece un *home* un punto de referencia equivalente a un (0,0) del cual partimos y establecemos el resto de los valores positivos o negativos dependiendo de la ubicación de estos en el cuadrantes.

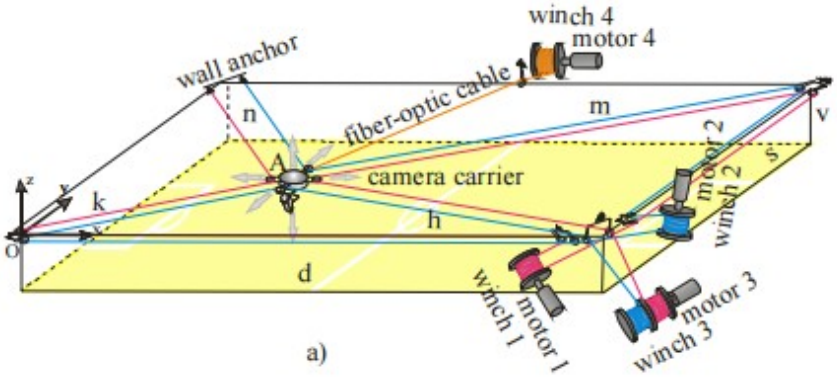

Figura 8: Mapa de los frames.

Una vez logrado el objetivo de llegar al punto deseado, nos aseguramos de que el movimiento fuera preciso y suave para esto se implementó un control PID el cual iba revisando la posición actual con respecto a la final, de esta manera la velocidad del motor se iba regulando en cuanto más se acercaba al objetivo, esta iba disminuyendo.

```
#Funcion de controlador PID
def pid_controller(y, yc, h=1, Ti=0.5, Td=0.01, Kp=100, u0=0, e0=0):
    # Step variable
    k = 0
    # Initialization
    ui_prev = u0
    e_prev = e0
    # Error between the desired and actual output
    e = yc - y
    # Integration Input
    ui = ui_prev + 1/Ti * h*e
    # Derivation Input
    ud = 1/Td * (e - e_prev)/h
    # Adjust previous values
    e_prev = e
    ui_prev = ui
    # Calculate input for the system
    u = Kp * (e + ui + ud)
    k += 1

    return u
```

Como parte extra del proyecto se probó un sistema de visión, implementado en la segunda Raspberry del prototipo, en donde la cámara es capaz de ver la pelota y seguirla haciendo uso de OpenCV. Este código es capaz de medir la distancia de la cámara a la pelota, esto gracias a que el programa arroja el valor del radio del objeto.

Figure 9: Sistema de visión activo.

Una vez que se logro esta etapa de movimiento, se transmitió en una página web diseñada específicamente para mostrar la imagen capturada por la webcam. La página cuenta con las funciones de crear tu cuenta (la cual se almacena en una base de datos), ingresar a el evento que se este transmitiendo y/o contratar el servicio.

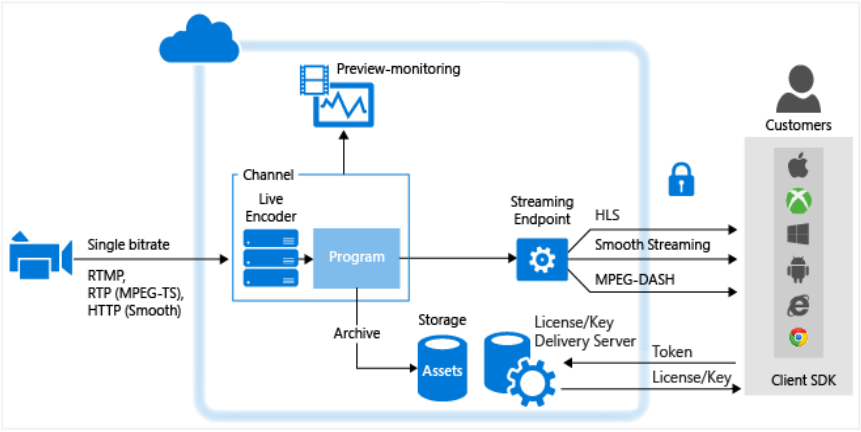

Figura 10: Funcionamiento de LiveStream.

Finalmente se establecieron dos métodos de control, una con posiciones predefinidas (tiro de esquina, penal, etc.) y la segunda “Manual” en la cual se puede controlar la cámara con el uso de las flechas del teclado, ambos códigos se adjuntan en la carpeta.

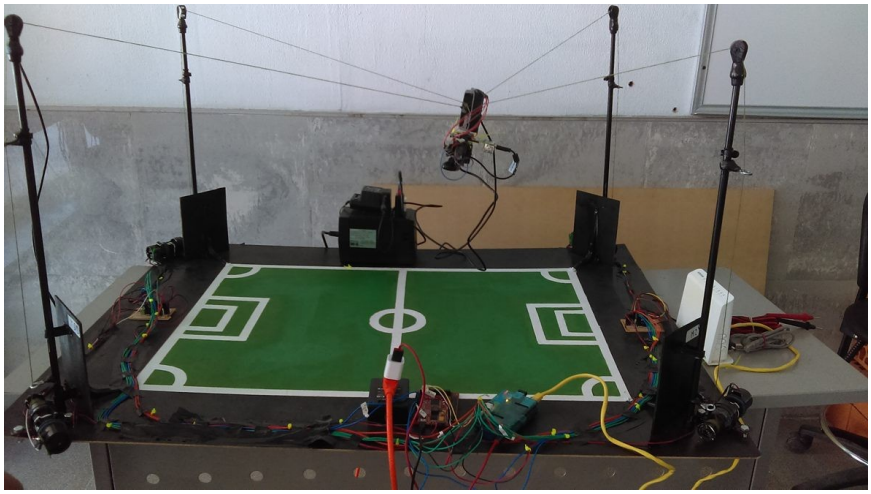

Figura 11: Prototipo final del proyecto.

4. Resultados y discusión

La cámara es capaz de llegar de un punto establecido a otro de manera precisa, transmitir el contenido, almacenar la posición actual y visualizar un objeto, es este caso una pelota. Alcanzando todos los objetivos planteados al inicio del proyecto. A pesar que durante la elaboración del proyecto nos encontramos con muchos obstáculos, tanto por que la Raspberry es relativamente nueva y en los foros no había mucha información, así como que la mecánica del proyecto era compleja.

5. Conclusiones

Siendo este un proyecto integrador se implementó el contenido de diferentes cursos como Robótica Aplicada, Sistemas Embebidos, Diseño de Aplicaciones Web, etc. Dando como resultado un prototipo totalmente funcional, comercial y viable, puesto que ocupamos componentes fáciles de adquirir. Finalmente este prototipo ofrece una solución que se ajusta a las necesidades del cliente.

6. Referencias

[1] SPONG , M. “ ROBOT MODELING AND CONTROL ”

[2] Beyeler, M. (2015). OpenCV with Python Blueprints. Birmingham, UK.

[3] AREF,M.M., GHOLAMI,P., TAGHIRAD,H.D.: Dynamic and Sensitivity Analysis of KNTU CDRPM: A Cable Driven Redundant Parallel Manipulator , 4th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications.

[4] Raspberry Pi Documentation by the Raspberry Pi Fundation