# Informe Final Proyecto

Mario Alpízar/Andrey Herrera

***Resumen*—Implementar un proyecto que controla una plataforma que se balancea logrando estabilizar una bola en un solo punto.**

## I. Descripción del Sistema

El proyecto consiste en la utilización de una bola y una tabla de madera. Al poner la bola en esta tabla, esta bola por más que se desplace hacia cualquier lugar, la tabla se inclinará de manera que se la bola se vuelva a posicionar en el centro.

Además, la tabla se podrá inclinar mediante la utilización de un guante, respondiendo así, a la forma en que la mano se incline.

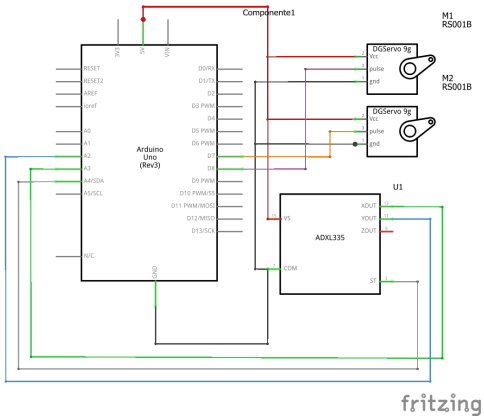
El video del proyecto a implementar en el cual nos basamos se encuentra en el siguiente link https://www.youtube.com/watch?v=p65XPP53rLo.

La función adicional que le deseamos implementar, es el uso de un dispositivo que mide los ejes 'x', y 'y', ya sea conectado a la muñeca de una persona o por medio de un guante, con el cual se pueda controlar la plataforma y sus diferentes movimientos.

## II. Requerimientos

- Control de motores.
- Reproducir experimento previamente creado.
- Crear una interfaz para observar la camara.
- Construir aplicación OpenCV.
- Comunicar aplicación OpenCV con arduino.
- Leer datos del guante.
- Correta comunicación del entre el guante y la plataforma.

## III. Esquememático




## IV. Descripción de la Solución

Se desarrolló una interfaz mediante Visual Studio 2017 en la cual esta interactúa entre la Webcam y el puerto serial del Arduino. Este código toma los frames de la cámara en un instante de la siguiente manera:

```
VideoCapture cap(0);
```

Este frame será llevado a través de un proceso de filtrado por color. Para este caso, el color a detectar será el rojo. Adicionalmente, el programa, mediante la librería OpenCV, abrirá dos cuadros para la visualización del objeto en tiempo real sin filtro y con filtro. Para realizar el filtrado antes mencionado, se filtra la imagen y esta se guarda en una variable tipo "Mat" la cual permite poseer los valores del frame filtrado.

```
erode(imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE,
Size(5, 5)));
dilate(imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE,
Size(5, 5)));
```

Una vez la imagen filtrada, se procede a realizar un seguimiento del objeto para determinar su posición en tiempo real.

```
Moments oMoments = moments(imgThres
holded);
```

Este comando ejecuta una función encargada de determinar la posición del objeto y además, imprimir su posición tanto en “X” como en “Y”. La siguiente parte de la etapa se basó en la creación de una interfaz. El problema de este, se debe a Visual Studio no soporta “Windows Form Application” por lo cual este se creó desde cero. Este diseño se basa en el link 1. Además, mediante la misma “Windows Form Application” se realizó la conexión al puerto USB. Una vez integrados los dos códigos, se procedió a crear un “string” para que el Arduino pudiera leer y diferenciar el valor de “X” del de “Y”. Para esto, el “string” se creó de la siguiente manera:

```
System::String^ str = posX.ToString()
+ ":" + posY.ToString()+"$";
```

Así, el string sería delimitado por el Arduino debido a los signos ”:ƨ ”S”. Una vez hecho esto, debido a que el “string” creado es diferente al “string” de sistema, se realiza una conversión.Además, se realizará un retraso de 20 milisegundos y luego el “string” será enviado por el puerto serie de la siguiente manera:

```
Sleep(20);
System::Convert::ToString(str);
if (this->serialPort1->IsOpen  ) {
this->serialPort1->WriteLine(str);
}
```

En el código de Arduino, se utilizará la librería “PID” la cual realizará los cálculos para controlar la posición de la bola y eliminar cualquier perturbación en el sistema. El Arduino realizará la lectura del “string” creado anteriormente en Visual Studio. Debido a que la lectura es “string”, se necesitan realizar dos pasos. El primero consiste en dividir el “string” para sacar los valores necesitados y el segundo, pasarlos a “Int”. Para esto, se realizó el siguiente fragmento:

```
inData = Serial.readStringUntil(':');
Input = inData.toInt();
inData2 = Serial.readStringUntil('$');
Input2 = inData2.toInt();
```

De esta manera, se obtiene los datos como valores numéricos independientes y utilizables. A partir de esta etapa, el código consiste en utilizar un mapeo, debido a que los valores de “X” y “Y”, superan el valor máximo de ángulo de los Servomotores. Luego, se procede a someter estos valores mapeados, bajo cálculos para realizar el control de la posición de la bola. Estas funciones, calcularán el ángulo de inclinación y determinarán su posición a medida que la bola se desplaza de la siguiente manera:

```
myPID.Compute();  //action control
X compute
myPID2.Compute(); //   action control
Y compute
myservo.write(Input);
myservo2.write(Input2);
```

## V. Análisis de Resultados

- La utilización de un retraso en el código también genera un retraso en la cámara en tiempo real.
- El algoritmo de filtrado funcionó de manera efectiva detectando el color del objeto y determinando su posición.
- El Windows Forms Application realizó un trabajo similar al LabView, teniendo una interfaz gráfica y una interfaz para modificar su código.
- La utilización de Winows Forms Application permite al usuario un uso más eficiente del código debido a que mediante este, no es necesario editar el puerto serial configurado directamente en el código.
- La creación y división de los valores de posición funcionó de manera efectiva permitiendo la legibilidad correcta de los datos entrantes.
- El PID, a pesar de que se comporta de manera que busca estabilizar la bola, no logra reducir su velocidad lo cual, a medida que la bola se aleje de su “SetPoint”, la plataforma comenzará a aumentar la velocidad de la bola.

## VI. Conclusiones

- Visual Studio presenta una amplia gama de aplicaciones a la hora de desarrollo de software.

- La librería OpenCV permite una manejo sencillo de archivos multimedia tanto en su lectura y escritura como a la hora de editarlos.
- Windows Forms Application facilita la creación y enlace entre una interfaz gráfica y el código desarrollado.
- La solución del PID, aunque no fue concluida, se acercó debido al comportamiento presentado cuando el sistema es perturbado.

## VII. Proximos Pasos

- Utilizar motores de mayor precisión para mejorar estabilidad del sistema.
- Alimentar independientemente los servos para mejorar funcionalidad del guante.
- Estabilizar y ubicar correctamente la cámara para no sufrir problemas de estabilización de la misma.
- Implementar mejoras en el código para un funcionamiento más estable de los servos.
- Aprender el correcto funcionamiento de los sistemas P.I.D.