Documentación e instalación

Marco Ney Rojas Jiménez

e-mail: xecklet@gmail.com

# **Desarrollo del proyecto**

# **Construcción**

El brazo robótico que se utilizó en la construcción corresponde con el siguiente modelo:



Para poder imprimir el brazo robótico se utilizó la impresora Cubify Pro de segunda generación, que se encuentra ubicada en el laboratorio de diseño

La impresión de las partes del robot se realizó en la impresora y tardó aproximadamente tres semanas, debido a que la impresión era bastante lenta y la superficie de impresión muy reducida.

Una parte clave de la construcción del brazo robótico, corresponde con el movimiento de las articulaciones, las cuales son realizadas utilizando servomotores. En la siguiente tabla, se muestra el costo de total de cada servomotor:

Tabla 2: Costo de los servos motores

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Servo Tower Pro MG995** | **Servo Futaba S3003** | **MicroServo MG90S** |
| Imagen | Product Details | Product Details | Standard MG90S Metal Geared Micro Tower Pro Servo For Plane Helicopter Boat Car |
| Cantidad | 1 | 3 | 4 |
| Costo | $10,99 | $38,64 | $18,59 |

El costo total de la compra de los servomotores 68.22 dólares. Al comprar los servomotores por Amazon, estos duraron aproximadamente cuatro semanas en llegar, lo que generó un importante retraso.

Los últimos materiales por conseguir, fueron los tornillos. Esto fue un problema, debido a que los tornillos que se necesitaba principalmente eran de 2x15 mm, los cuales no se pueden conseguir en ferreterías. Al hablar con varias personas, se llegó a la conclusión de que los tornillos debían están en la Casa del Tornillo, donde se especializan en la construcción de los mismos. Al final el brazo ocupo aproximadamente 18 tornillos de 2x6 mm, 8 tornillos de 3x25 mm y 36 tornillos de 2x15 mm.

Una vez con todos los materiales, la construcción del brazo robótico pudo dar inicio. El brazo fue completado al cabo de dos semanas. Para comprobar que todas las articulaciones se movían correctamente, se programó una sub rutina para el movimiento de las articulaciones.

Uno de los problemas que se presentó con las pruebas, fue que la pinza no cerraba correctamente debido a la calidad de la impresión.

# **Micro controlador**

El micro controlador escogido fue el Arduino, que aunque cuenta con capacidades básicas de recursos, su nivel de procesamiento es el ideal y cuenta con la cantidad necesario de salidas digitales. Otra de los puntos por el cual se escogió, fue el hecho de cuanta con una gran cantidad de ejemplos y no era necesario adquirir uno, pues ya se contaba con uno. Por otro lado, no era la primera vez que el estudiante trabaja con un Arduino.

# **Alimentación y control de los servomotores**

El Arduino por sí solo, no es capaz de poner a funcionar ocho servomotores al mismo tiempo, su máxima capacidad es de dos o tres servos. En resumen, la potencia del Arduino era un problema, principalmente por el hecho de que su máxima corriente es de 500 mA, que es la corriente que provee el puerto USB.

La solución al problema de potencia fue el uso de una fuente de alimentación externa, que consistió en una apartador de corriente alterna, que convierte el voltaje de un tomacorriente a 5V, con una corriente máxima de 1.5 A. Al realizar esto, el problema de la potencia fue resuelto.

Por otro lado, para controlar cada servo motor de manera independiente, se utilizaron 8 pines digitales donde se define el ángulo de cada servo motor de manera independiente. En la siguiente imagen se muestran las conexiones:

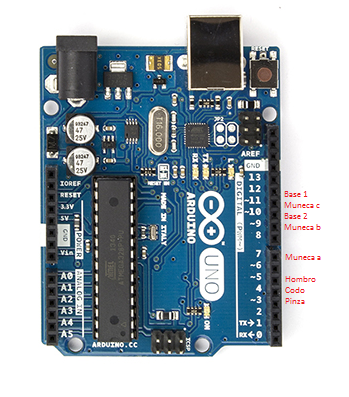


Figura 6.Conexión de los pines del Arduino con partes específicas del brazo robótico.

El algoritmo que cambia el ángulo es bastante sencillo, este utiliza dos arreglos: el primero de ellos posee el ángulo actual de cada servo motor y el segundo contiene la posición a la cual debe de ir. Para llegar hasta la posición deseada, se compara la posición final con la inicial y dependiendo del resultado se suma o se resta un grado al servo motor en cada ciclo de reloj.

# **Comunicación por puerto serial**

El método inicial lee en cada ciclo la entrada serial, si encontraba algún bit, comenzaba a leerlo bit por bit, hasta recuperar de forma exitosa la cadena de caracteres. Luego tomada la cadena, la decodificaba y actualizaba la posición a la cual los servo motores debían ir.

La lectura del puerto serial siempre era exitosa, al igual que la decodificación y la actualización de las posiciones, el problema radicaba en la consulta del puerto serial. Cuando el método preguntaba si existía algo en dicho puerto, se retrasaba en aproximadamente 500 ms, lo cual hacia que el brazo robótico se moviera de forma pausada y lenta.

La solución a este problema fue utilizar un método que funcione con eventos. A diferencia del método anterior, este comienza a leer el puerto serial cada vez que llegaba un nuevo mensaje, por lo que no se generaba ningún retraso en el ciclo principal del Arduino.

# **Programación de la aplicación**

La decisión final fue programar en Unity 4.6f, utilizando C#. La razón principal, es debido a que posee una librería muy poderosa. Dicha librería cuenta con la capacidad de *renderizar* el modelo de las manos en tiempo real.

Entre las características que tiene la aplicación se encontra: transformar los datos obtenidos en ángulos, enviar los datos al puerto serial, mostrar el modelo de las manos y la posición de los servo motores en tiempo real.

La parte más complicada de la aplicación, consistió en la transformación de los datos de posición y rotación en ángulos. La solución implementada fue tomar dos puntos fijos en el espacio e implementar una función lineal con base en esos puntos.

Además, la aplicación limita el movimiento de los servomotores, esto para evitar que las partes del robot colisionen con la placa de pruebas, que se encuentra detrás del brazo robótico, estropeando el circuito controlador. Las articulaciones que fueron limitadas fueron: la base, el hombro y el codo, por lo que sus rangos de rotaciones son de 45 a 135 grados, 60 a 180 grados y 60 a 180 grados, respectivamente. Además, debido a la rotación requerida por la pinza, el servo se limitó de 100 a 150 grados.

Por otro lado, el movimiento principal de del brazo utilizado la posición de las manos, limita la cantidad de articulaciones que se rotan, a solamente cuatro. Esto se debe, a que cuando se estaban realizando las pruebas de movimiento, el Leap Motion, muchas veces perdía la posición exacta de las manos; y al realizar dos rotaciones en la mano izquierda, se afectaban mutuamente, como resultado se generaban movimientos inesperados. La solución final fue mover dos articulaciones por mano:

La mano derecha mueve la base con la posición X y se encarga de abrir o cerrar la pinza cuando cierra o abre la mano respectivamente.

La mano izquierda mueve el hombro con la posición Z y controla la inclinación de la muñeca del brazo robótico, con la inclinación de la mano

Al controlar el brazo robótico de esta forma, fue más fácil su movimiento. Además, como ciertas articulaciones fueron limitadas a una posición estática, se puede dar la posibilidad que esa articulación no esté funcionando o simplemente esta estático en el código.

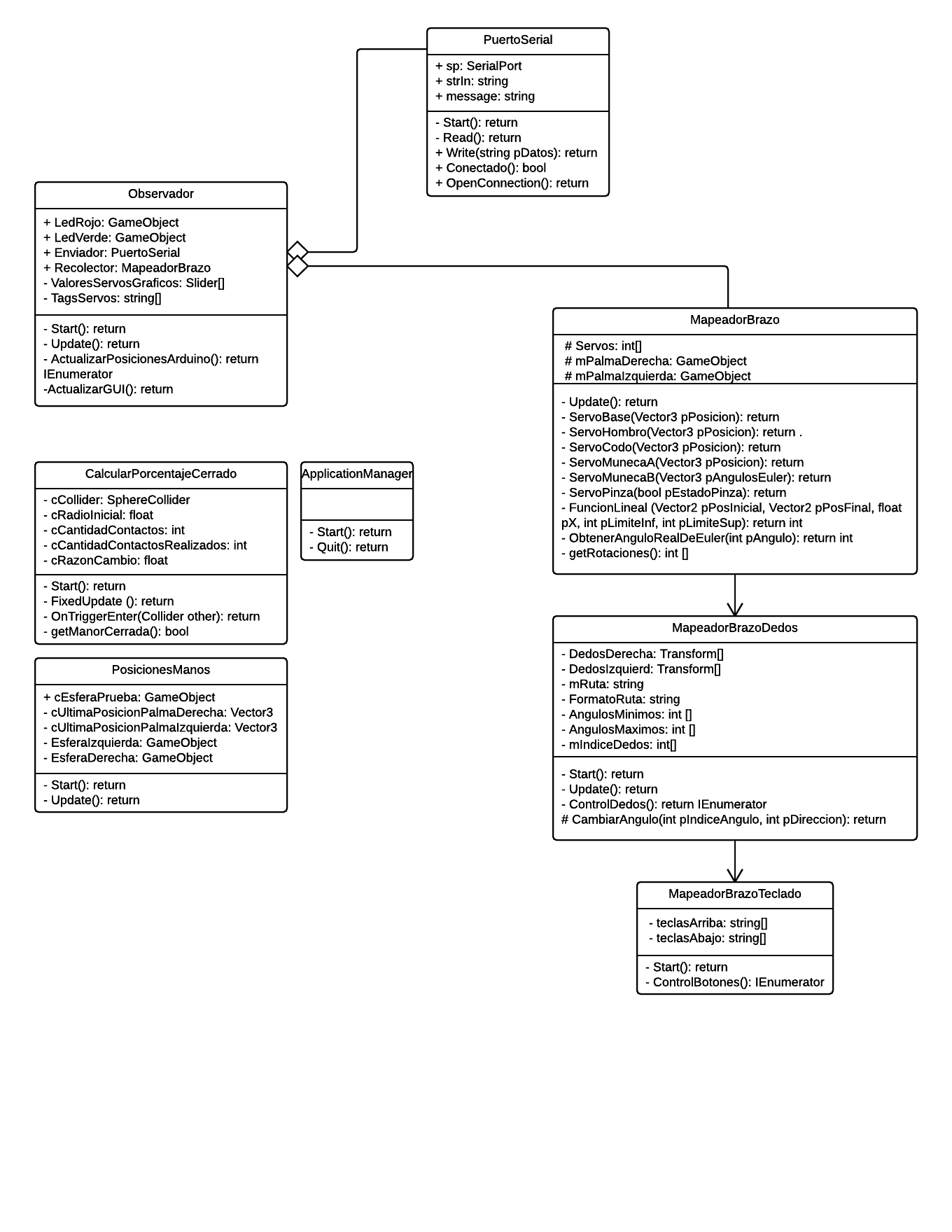
En la aplicación se implementó, un segundo método, el cual utiliza las flechas del teclado para mover cada articulación por separado. Su funcionamiento consiste aumentar o disminuir el ángulo utilizando las siguientes teclas:

1. Cadera: Aumentar "a" y disminuir "z"
2. Hombro: Aumentar "s" y disminuir "x"
3. Codo: Aumentar "d" y disminuir "c"
4. Rotación hombro: Aumentar "f" y disminuir "v"
5. Inclinación Muñeca: Aumentar "g" y disminuir "b"
6. Rotación Muñeca: Aumentar "h" y disminuir "n"
7. Pinza: Aumentar "m" y disminuir "j"

# **Diseño de la aplicación**

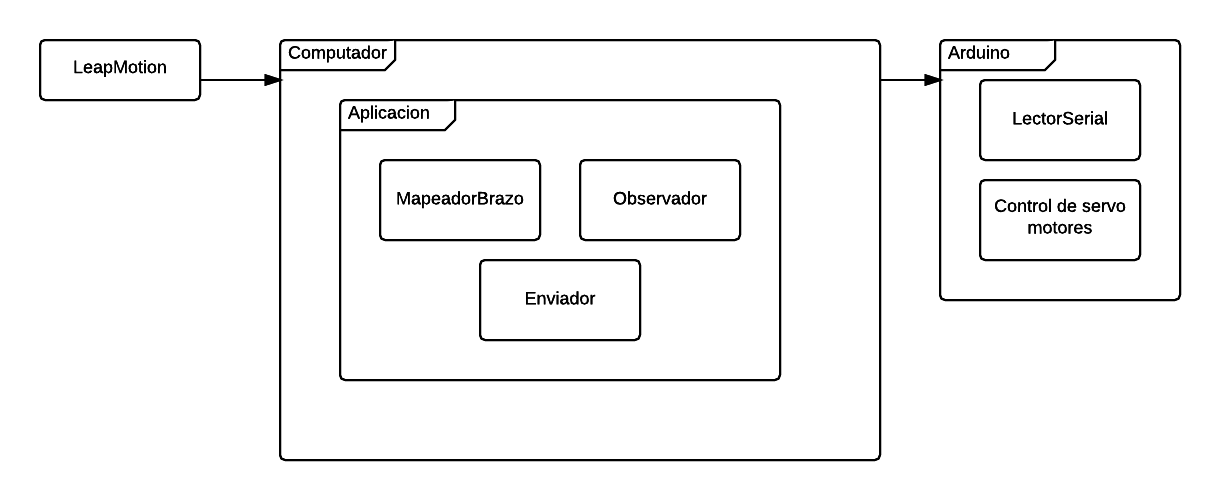
# **Diagrama de clases de la aplicación de escritorio**

En la siguiente imagen se presenta el diagrama de clases de la aplicación:



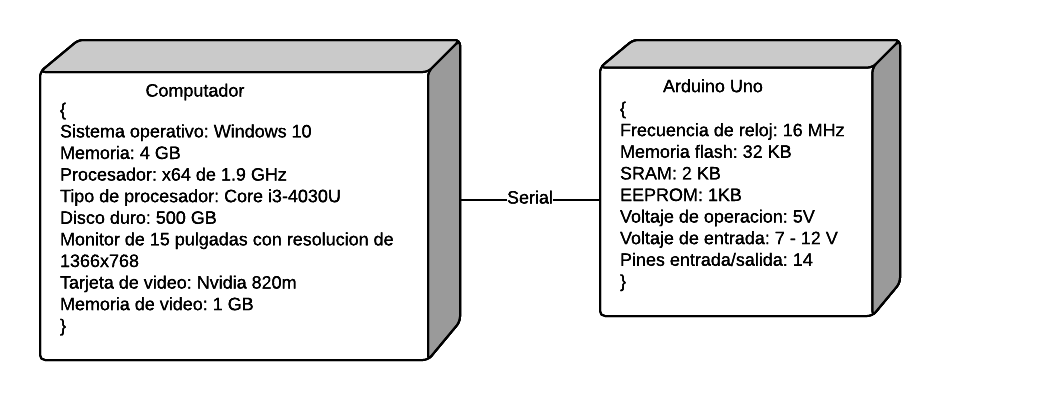
# **Diagrama de arquitectura**

En la siguiente imagen se muestra el diagrama de arquitectura implementado en el desarrollo del proyecto.



# **Diagrama de despliegue**

En ambiente en el cual el proyecto de diseño fue implementado es presentado en el siguiente diagrama:



# **Instalación**

Para poder utilizar la aplicación de escritorio es necesario realizar los siguientes pasos:

1. Descargar e instalar los drives del Arduino. Los cuales pueden ser descargados del siguiente link: <http://www.arduino.cc/en/Main/Software>
2. Ir al administrador de dispositivos (click derecho en el botón de inicio > administrador de dispositivos)
3. Ubicar el puerto COM del Arduino, ingresar en las propiedades del puerto y cambiar la velocidad a 115200 y utilizar el puerto COM3.
4. Descargar e instalar los drivers del Leap Motion. Los cuales pueden ser descargados del siguiente link: <https://developer.leapmotion.com/>
5. Luego en la carpeta del ejecutable de la aplicación se tienen los siguientes archivos:
   1. Aplicación\_Data
   2. Aplicacion.exe
   3. Leap.dll
   4. Leap3dInteractDll.dll
   5. LeapCSharp.dll
6. Los últimos tres archivos, deben ser copiados en la siguiente ruta:
   1. Para 32 bits (x86): C:\Windows\System32
   2. Para 64 bits (x64): C:\Windows\SysWOW64
7. Ejecutar el archivo “Apliacion.exe”.