Universidad Rey Juan Carlos

Sistemas Empotrados y de Tiempo Real

[PROYECTO FINAL – BRAZO ROBÓTICO]

JOSÉ ANDRÉS VILLAGRÁN ROBLES

Tabla de contenido

Introducción 3

Diseño del Proyecto 4

Ensamblaje físico 5

Desarrollo del Código Arduino 6

- **Pines, constantes y variables globales** 6
- **Funciones auxiliares** 8
- **Funciones principales** 10

Problemas y soluciones encontradas 12

- **Servomotores demasiado rápidos** 12
- **Estancamiento en los límites** 13
- **Pobre respuesta en joysticks** 13
- **Temblores generales por el peso** 13
- **Movimiento restringido por los tornillos** 13

Casos de uso 14

Costes de los Materiales 15

Conclusión 16

Tabla de ilustraciones

Ilustración 1. Piezas del diseño encontrado 4
Ilustración 2. Garra o gripper 5
Ilustración 3. Parte izquierda móvil 5
Ilustración 4. Montaje listo 5
Ilustración 5. Función applyAngles() 8
Ilustración 6. Función pushJoy() 9
Ilustración 7. Función gotoSlow() 9
Ilustración 8. Función autoSequence() 10
Ilustración 9. Función setup() 11
Ilustración 10. Función loop() 12

Introducción

Para el proyecto final de esta asignatura, inicialmente se consideró la idea de desarrollar un invernadero inteligente. Tras echar un vistazo al blog de la escuela y notar que la idea era muy recurrente, se decidió optar por un proyecto más original e interesante: un brazo robótico.

Este brazo robótico está compuesto por cuatro servomotores que controlan la base, un brazo izquierdo (será considerado izquierdo según el punto de vista del propio robot), un brazo derecho (lo mismo para este) y una garra o gripper. Cada uno de estos servomotores se podrá controlar mediante dos joysticks, y además tendrá una funcionalidad extra con la que se moverá automáticamente hacia un lado o hacia otro. También usará una placa Arduino UNO R3 y una placa Shield para la conectividad del brazo, joysticks y demás elementos como el sensor de luz utilizado para la funcionalidad automática.

Los objetivos del proyecto son varios: por una parte, trabajar en la parte física del robot; por otro, lograr un control preciso mediante los joysticks, logrando un movimiento sólido y eliminando cualquier tipo de temblor; y, finalmente, trabajar en la parte autónoma del proyecto, logrando que el brazo se mueva por sí mismo.

En esta memoria y los documentos anexos se detallan los objetivos alcanzados y el desarrollo de cada una de las etapas.

Diseño del Proyecto

El primer paso en la realización del proyecto fue la gestión del diseño. En cuanto se obtuvo el visto bueno de la idea, se comenzaron a evaluar diferentes enfoques para conseguir un diseño robusto para el brazo, considerando que el proyecto sería llevado a cabo por un único miembro del equipo. Así, en un principio, se contempló un diseño de brazo robótico compatible con impresión 3D, ya que era una opción flexible en costos y personalización. Se contactó con un amigo personal del equipo, quien disponía de una impresora 3D. Sin embargo, debido a la alta demanda de la impresora por otras tareas, la preparación de algunas piezas se vio retrasada. Por este motivo, se descartó ese diseño y se buscó otra solución.

La alternativa seleccionada consistió en un diseño ya disponible públicamente en un archivo PDF, proporcionado por una empresa externa especializada en la fabricación de componentes para proyectos tecnológicos muy pequeños, de talleres de aprendizaje. Además, la propia empresa ofrecía un servicio de recorte de piezas en acrílico, lo que facilitó la obtención de estas.

Seguidamente, se procedió a la compra de los tornillos, tuercas y demás elementos necesarios para montar la estructura del brazo. Luego, se obtuvo siete servomotores, aunque solo se necesitasen cuatro, por si alguno llegaba a fallar; se compraron cables para unir los componentes; los joysticks; una fuente externa para que los servomotores tuviesen la potencia necesaria; y, por supuesto, la placa Arduino UNO R3 y la placa *shield*. Con todo esto, el siguiente paso fue proceder con el ensamblaje físico.

Ilustración 1. Piezas del diseño encontrado

Ensamblaje físico

Esta parte del proyecto fue la más problemática. Tras seguir el diseño del brazo montándolo poco a poco, se comprobó que el diseño tenía algunos detalles implícitos a tener en cuenta que afectaban a su funcionamiento. El mayor problema fue identificar qué era lo que fallaba cuando ya estaba listo. Ni siquiera un código simple hacía que los servomotores funcionasen perfectos.

Tras mucha prueba y error, desmontando y montando de nuevo todas las partes del brazo, se pudo observar que había partes que no encajaban y necesitaban ir de manera distinta a la que se colocaron en un principio. Sin embargo, esto no quedaba ahí: la presión de los tornillos bloqueaba las posiciones de los servos, impidiéndoles el paso a pesar de otorgarles energía y potencia; y, el peso del *gripper* hacía que el movimiento del brazo fuera a trompicones. Fue tedioso pero se logró lidiar con cada uno de estos problemas hasta tener un resultado sólido y fluido que pudo vivir en armonía con el código Arduino desarrollado.

Ilustración 3. Parte izquierda móvil

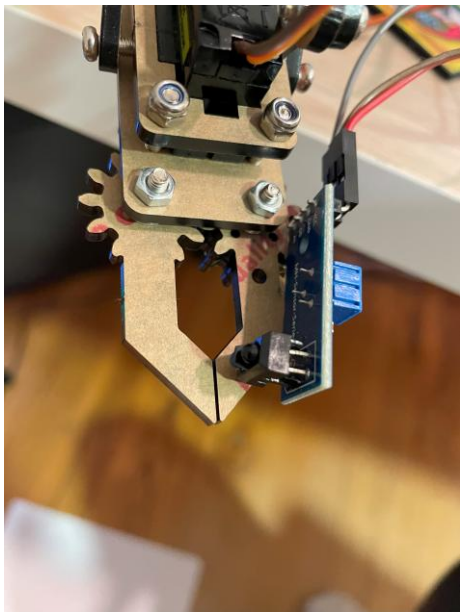

Ilustración 2. Garra o gripper

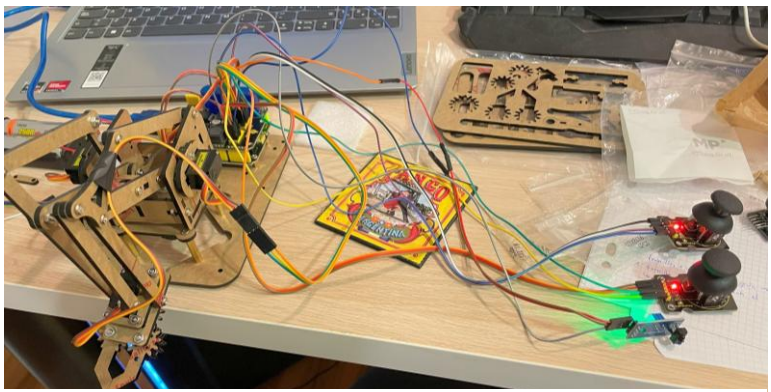

Ilustración 4. Montaje listo

Desarrollo del Código Arduino

Para el desarrollo del código en Arduino se utilizó la librería ***Servo.h***. Esta librería facilitaba mucho la tarea de mover los servomotores a los ángulos necesarios.

Pines, constantes y variables globales

Ahora se mostrarán brevemente los pines, constantes y variables globales del proyecto.

En la declaración de pines se tiene que:

Nombre de la variable	Componente asignado	Pin utilizado
PIN_BASE	Servomotor en la base	A1
PIN_RIGHT	Servomotor del brazo derecho	A0
PIN_LEFT	Servomotor del brazo izquierdo	6
PIN_GRIPPER	Servomotor de la garra	9
BTN_IZQ	Pulsación del joystick izquierdo	8
PIN_SENSOR	Sensor de luz	7
JOY_LEFT_X	Eje X del joystick izquierdo	A3
JOY_LEFT_Y	Eje Y del joystick izquierdo	A4
JOY_RIGHT_X	Eje X del joystick derecho	A2
JOY_RIGHT_Y	Eje Y del joystick derecho	A5

Por otro lado, se crearon algunas constantes para afianzar los límites de los servomotores:

Nombre de la variable	Componente al que afecta	Valor
MIN_ANG	Valor mínimo general	5
MAX_BASE		130

MIN_BASE	Servomotor de la base	30
MIN_RIGHT	Servomotor del brazo derecho	75
MAX_RIGHT		150
MAX_LEFT	Servomotor del brazo izquierdo	120
MAX_GRIP	Servomotor del gripper	50
MIN_GRIP		0

Sin estos límites, los servomotores podrían quedar atascados.

Por último, se definieron algunas otras constantes y variables globales:

Nombre de la variable	**Utilidad**	**Valor**
ANG_VELOCITY	Velocidad a la que los servomotores giran	1
THRESHOLD	Umbral para asegurar movimiento por joysticks	100
WAIT	Mini *cooldown* para el loop	30
COOLDOWN	*Cooldown* para el modo automático	1500

base	Servomotor de la base	Se hará **attach** de PIN_BASE
right	Servomotor del brazo derecho	Se hará **attach** de PIN_RIGHT
left	Servomotor del brazo izquierdo	Se hará **attach** de PIN_LEFT
gripper	Servomotor de la garra	Se hará **attach** de PIN_GRIPPER

baseAng	Ángulo actual de la base	Inicialmente 80

rightAng	Ángulo actual del brazo derecho	Inicialmente 150
leftAng	Ángulo actual del brazo izquierdo	Inicialmente 80
gripperAng	Ángulo actual del gripper	Inicialmente 5

mode	Enum para saber qué modo está activado { MANUAL, AUTOMATIC }	Inicialmente MANUAL
timeNext	Comprobación del *lopp cooldown*	Inicialmente 0, se irá sumando la constante WAIT
lastAuto	Comprobación del *automatic cooldown*	Inicialmente 0, se irá asignando el tiempo actual cada vez
cxL, cyL, cxR, cyR	Guardarán los centros (los valores iniciales) de los joysticks	Alrededor de 300, depende del joystick

Funciones auxiliares

Ahora se mostrarán las funciones auxiliares del proyecto.

applyAngles() se encargará de escribir en los servos los ángulos actuales de cada parte (guardados en variables globales como se ha visto anteriormente). Por ello, esta función será llamada solo cuando se modifiquen sus valores o cuando se necesite el movimiento de los servos.

```
// Function to move the servos to their actual angles
void applyAngles() {
  base.write(baseAng);
  right.write(rightAng);
  left.write(leftAng);
  gripper.write(gripperAng);
}
```

Ilustración 5. Función applyAngles()

pushJoy() se encarga de identificar si un joystick ha sido movido lo suficiente como para superar el umbral utilizando las variables de los centros de los joysticks, y devolver posteriormente el valor del ángulo atualizado. El valor devuelto siempre será el ángulo actual más uno (o menos uno si se mueve el joystick al lado contrario), de esta manera siempre se asegura que el brazo se mueve de manera controlada por los joysticks, y no de manera brusca y sin control. Esto, sumado al cooldown de 30 milisegundos (constante WAIT), hace que el brazo se mueva de manera controlada y calmada, pero no excesivamente lenta.

```
// Joysticks function, that uses the joystick centers of the beginning to know if they have been pushed
int pushJoy(int push, int center, int actualAng, bool inverse = false) {
  int d = push - center;
  if (abs(d) < THRESHOLD)
    return actualAng;

  int dir = (d > 0) ? +1 : -1;
  if (inverse)
    dir = -dir;

  return actualAng + dir * ANG_VELOCITY;
}
```

Ilustración 6. Función pushJoy()

gotoSlow(), por otro lado, controlará que el brazo también se mueva de manera controlada en el contexto del modo automático. Este modo requiere movimientos que no están asignados a los joysticks, por eso se necesita otra función que asegure el movimiento controlado directamente del robot. Los parámetros referentes al servo y a la variable global que guarda el ángulo de este se pasarán como punteros, para que se puedan actualizar en la propia función.

```
// Function to move the servos not that fast
void gotoSlow(Servo &s, int &varAng, int target, uint8_t pausa = 15) {
  while (varAng != target) {
    varAng += (varAng < target) ? ANG_VELOCITY : -ANG_VELOCITY;
    s.write(varAng);
    delay(pausa);
  }
}
```

Ilustración 7. Función gotoSlow()

autoSequence() es la función que definirá los movimientos dados en el modo automático. Se guiará por una variable local que establecerá a qué lado debe dirigirse el brazo, dependiendo de si el objeto a recoger está en la derecha o en la izquierda (siempre irá al lado contrario). Los pasos definidos son: asegurar el objeto cerrando la garra; mover los brazos para subir la estructura y evitar golpear algún objeto del entorno; girar la base al lado correspondiente; soltar el objeto abriendo la garra; girar la base a una posición inicial; y volver a cerrar la garra.

```
// Automatic "leave the piece" function
void autoSequence() {
  int baseTargetAng;

  if (baseAng < 90)
    baseTargetAng = MAX_BASE;
  else
    baseTargetAng = MIN_BASE;

  // Close gripper
  gotoSlow(gripper, gripperAng, MIN_GRIP);
  delay(15);

  // Look at the right place
  gotoSlow(right, rightAng, 120);
  delay(15);
  gotoSlow(left, leftAng, 100);
  gotoSlow(base, baseAng, baseTargetAng);
  delay(15);

  delay(1500);

  // Leave the piece, therefore open the gripper
  gotoSlow(gripper, gripperAng, MAX_GRIP);
  delay(15);

  // Back to "home" position
  gotoSlow(base, baseAng, 80);

  // Close gripper again
  gotoSlow(gripper, gripperAng, MIN_GRIP);
  delay(15);
}
```

Ilustración 8. Función autoSequence()

Funciones principales

Ahora, por último, se mostrarán las funciones de ***setup()*** y ***loop()***.

En ***setup()*** se preparan todos los componentes necesarios para el funcionamiento del brazo. Se prepara el Serial, las variables de los servos haciendo ***attach()*** y se aplican enseguida los ángulos definidos de cada uno,

para que el brazo empiece en una posición inicial. Luego, se configuran los pines del sensor y del botón del joystick con ***pinMode()***; y, finalmente, se guardan los valores de los centros de los ejes de los joysticks.

```
void setup() {
  Serial.begin(9600);

  base.attach(PIN_BASE);
  right.attach(PIN_RIGHT);
  left.attach(PIN_LEFT);
  gripper.attach(PIN_GRIPPER);

  applyAngles();

  pinMode(PIN_SENSOR, INPUT);
  pinMode(BTN_IZQ, INPUT_PULLUP);

  delay(300);

  cxL = analogRead(JOY_LEFT_X);
  cyL = analogRead(JOY_LEFT_Y);
  cxR = analogRead(JOY_RIGHT_X);
  cyR = analogRead(JOY_RIGHT_Y);

  delay(1000);
}
```

Ilustración 9. Función setup()

Finalmente, en ***loop()***, lo primero que se hace es recoger el tiempo actual en la variable *now* con la función ***millis()***. Posteriormente, se distinguirá entre el funcionamiento manual y el automático. En el modo automático, se llamará a ***autoSequence()***, se asignará en *lastAuto* el tiempo actual y el modo se pondrá de nuevo en manual.

Por otro lado, si el modo está en manual, y además ha pasado el *mini cooldown*, se leerá la entrada de cada joystick y se llamará a ***pushJoy()*** por cada servomotor para conseguir el valor del nuevo ángulo al que se llega moviendo los joysticks. Este valor se contendrá entre los límites haciendo ***constrain()***. Cuando se tengan todos los ángulos, se actualizarán en las variables globales y se llamará a ***applyAngles()*** para mover los servos. Después, se leerán los valores del sensor y del botón del joystick izquierdo. Si se ha superado el *cooldown* del modo automático, el sensor detecta un objeto y el botón es pulsado al mismo tiempo, entonces el modo se cambia a AUTOMATIC, y se avisa por consola.

```
void loop() {
  unsigned long now = millis();

  if (mode == MANUAL) {
    if (now >= timeNext) {
      timeNext += WAIT;

      baseAng = constrain(pushJoy(analogRead(JOY_LEFT_X), cxL, baseAng), MIN_BASE, MAX_BASE);

      rightAng = constrain(pushJoy(analogRead(JOY_LEFT_Y), cyL, rightAng), MIN_RIGHT, MAX_RIGHT);

      leftAng = constrain(pushJoy(analogRead(JOY_RIGHT_X), cxR, leftAng), MIN_ANG, MAX_LEFT);

      gripperAng = constrain(pushJoy(analogRead(JOY_RIGHT_Y), cyR, gripperAng), MIN_GRIP, MAX_GRIP);

      applyAngles();
    }

    bool sensorHit = digitalRead(PIN_SENSOR) == LOW;
    bool btnHit = digitalRead(BTN_IZQ) == HIGH;

    if (
      sensorHit && btnHit && (now - lastAuto > COOLDOWN)) {
      mode = AUTO;
      Serial.print("ACTIVANDO MODO AUTOMATICO");
    }
  }

  if (mode == AUTO) {
    autoSequence();
    lastAuto = millis();
    mode = MANUAL;
  }
}
```

Ilustración 10. Función loop()

Problemas y soluciones encontradas

Durante el desarrollo del proyecto, se encontraron muchos problemas aparentemente difíciles de explicar. Sin embargo, se pudo encontrar una solución a cada uno de ellos, con mucha prueba y error.

Servomotores demasiado rápidos

Uno de los problemas que se encontraron fue que los servomotores iban demasiado rápido cuando se utilizaban los joysticks, o cuando se enviaban señales con la librería *Servo.h*. Por ello, se tuvo que crear la función ***gotoSlow()***, que fuerza la pausa entre movimientos, y así el brazo se mueve de manera controlada y no bruscamente.

Estancamiento en los límites

Otro problema muy común que ocurría y no se daba fácil explicación era el estancamiento que se producía cuando se movía un servomotor a un ángulo cerca de sus límites **físicos** (0º - 180º). Este error fue difícil de solucionar, porque cuando se producía, parecía ser un error de los propios joysticks; y, a partir de cuando se producía, los servomotores se descontrolaban por completo. Para solucionarlo, se tuvieron que crear las **constantes** explicadas anteriormente. Muchas de estas constantes están en un intervalo más cerrado que el propio físico de los servos, debido a que la propia estructura muchas veces limitaba el paso de algunas piezas.

Pobre respuesta en joysticks

Este problema surgió cuando se detectó una pésima calidad en la respuesta al movimiento de los joysticks. Fue necesario implementar ***cooldowns*** en la lectura de los joysticks, por eso se creó la constante WAIT, que se aplica en el bucle ***loop()***.

Temblores generales por el peso

El peso de la propia estructura limitaba el movimiento. Este fue el segundo problema más **grave** y difícil de dar explicación. La causa es **sutil**: uno no identifica fácilmente por qué el brazo tenía temblores (todo lo demás estaba correcto). Se tardó bastante en encontrar qué es lo que estaba pasando, hasta que se pudo confirmar que era el propio peso de las piezas, sobre todo de la garra, lo que a veces limitaba el movimiento y daba temblores. Esto se consiguió **desmontando** y montando repetidas veces la estructura, hasta poder identificar el problema. Una vez identificado, se aligeró la garra y se implementó el alcance de apertura suficiente para que los brazos derechos e izquierdos pudieran **recogerse** lo suficiente como para que el peso se distribuyera por la base.

Movimiento restringido por los tornillos

El problema más complicado de resolver sin duda fue la **presión de los tornillos**, que presionaban tanto la estructura, que muchos servomotores simplemente dejaban de moverse. Al principio, se pensó que algunos motores se habían roto, pero al sustituirlos por unos nuevos, se comprobó que no era ese el problema. Para resolver este problema, no basaba con saber cuál era la causa del problema, sino encontrar un equilibrio perfecto en la presión de los tornillos para que no estuvieran ni muy sueltos, ni muy firmes. De esa manera, el brazo podría moverse, sin que la estructura se desmoronase.

Casos de uso

Esta tabla muestra los distintos casos de uso del proyecto:

Descripción	Actor	Precondiciones	Flujo básico
El usuario puede controlar el servomotor de la base moviendo el eje X del joystick izquierdo	Usuario	Brazo encendido, servomotores en posición inicial	1.- El usuario mueve el joystick hacia la izquierda o derecha 2.- El brazo se mueve a su izquierda o derecha, hasta que se deje de mover el joystick o se haya alcanzado el límite
El usuario puede controlar el servomotor derecho (según la perspectiva del robot) moviendo el eje Y del joystick izquierdo	Usuario	Brazo encendido, servomotores en posición inicial	1.- El usuario mueve el joystick hacia arriba o hacia abajo 2.- La parte derecha se inclina más o menos, alejando o acercando la garra
El usuario puede controlar el servomotor izquierdo (según la perspectiva del robot) moviendo el eje X del joystick derecho	Usuario	Brazo encendido, servomotores en posición inicial	1.- El usuario mueve el joystick hacia la izquierda o derecha 2.- La parte izquierda sube o baja, acercando más o menos la garra al nivel del suelo
El usuario puede controlar el servomotor de la garra moviendo el eje Y del joystick derecho	Usuario	Brazo encendido, servomotores en posición inicial	1.- El usuario mueve el joystick hacia arriba o hacia abajo 2.- La garra se abre o se cierra, hasta que se deje de mover el joystick o se haya alcanzado el límite

Modo automático. El brazo puede moverse de manera autónoma una vez recibida la señal del sensor de luz en HIGH y presionándose el joystick izquierdo	Sistema (brazo robótico) y Usuario	Brazo encendido, servomotores en posición inicial, sensor en señal HIGH y joystick izquierdo presionado	1.- El usuario abre la garra 2.- El usuario coloca un objeto dentro de la garra 3.- El usuario aprieta el joystick izquierdo 4.- El brazo lleva el objeto hacia la derecha, si inicialmente el brazo estaba orientado hacia la izquierda, o viceversa en caso contrario.

La demostración real de los casos de uso se expresa en el vídeo anexo.

Costes de los Materiales

En el desarrollo de este proyecto, fue necesario utilizar materiales que implican un costo económico, tales como servomotores, placas electrónicas, cables, y componentes mecánicos. Estos materiales son esenciales para garantizar la funcionalidad y robustez del brazo robótico, permitiendo que el sistema cumpla con los requisitos de precisión y control necesarios.

Material	**Precio**
Arduino UNO R3	0€ (incluida en el kit de la asignatura)
Placa Shield	9,67€
Servomotores	11,90€
Piezas	39,90€
Tornillos, tuercas, cables	≈5€
Sensor de luz	4,50€
Joystick	4,67€

-	**Total = 75,64€**

Conclusión

El desarrollo de este proyecto ha sido una experiencia enriquecedora, en la que se ha logrado integrar exitosamente tanto el control manual como autónomo mediante la programación en Arduino y la configuración de componentes electrónicos. Aunque se presentaron algunos desafíos técnicos, el proyecto ha cumplido con los objetivos establecidos. Este trabajo ha fortalecido mis habilidades en robótica y electrónica, y me ha abierto nuevas oportunidades para seguir explorando este campo.