

Centro Universitario de Occidente  
Ing. Mauricio Lopez  
Aux. Eriksson Hernandez  
Aux. David Martinez



Proyecto de Laboratorio Arquitectura de Computadores y Ensambladores 1  
Grupo #7

INTEGRANTES:

201832069 Marcos Andrés Aguare Bravo  
201930699 Erick Daniel Morales Xicará  
201931707 Luis Emilio Maldonado Rodríguez  
201931804 Marco Jose Munguia Alva

## **Introducción**

En el campo de la Arquitectura de Computadores y Ensambladores, es fundamental comprender cómo aplicar la teoría en proyectos prácticos y cómo diseñar soluciones efectivas para problemas específicos. En esta ocasión uno de esos problemas es la recopilación de datos de una pelota que rueda por una base con diferentes ángulos de inclinación. Esta tarea es crucial en numerosas aplicaciones, como la investigación de la física de movimiento, la optimización de procesos industriales y la robótica.

## **Solución propuesta a la práctica**

En este proyecto de Arquitectura de Computadores y Ensambladores 1 se va a utilizar la plataforma de desarrollo de código abierto Arduino para crear un dispositivo que permita medir el tiempo de una pelota que rueda por una base con diferentes ángulos de inclinación. Para llevar a cabo este experimento, se van a utilizar varios componentes, entre ellos, un microcontrolador ATMega328, una breadboard para crear el circuito sin necesidad de soldar, dos micro servo para controlar la inclinación de la base, dos sensor ultrasónico HC-SR04 para medir el tiempo que tarda en recorrer la pelota la rampa y un keypad para introducir los ángulos a calcular . Se seguirán varios pasos para construir el dispositivo, como graduar la inclinación de la base a diferentes ángulos de inclinación y fijar los sensores de proximidad en la base para medir la altura de la pelota en cada punto del experimento. Así mismo los datos que abstrae el sensor y son guardados en el código arduino, envían los datos a través de un módulo Bluetooth HC-06 a Nuestra aplicación en Android (KOTLIN) la cual se encargará de procesar los datos del tiempo y mostrará la tabla de información.

El proyecto combina la teoría de la arquitectura de computadoras y la programación de ensambladores con la práctica de la construcción de prototipos electrónicos y la robótica.

## Componentes

**Arduino:** Es una plataforma de desarrollo de código abierto que se utiliza para crear prototipos de dispositivos electrónicos interactivos, el arduino es ampliamente utilizado por diseñadores, ingenieros, artistas, aficionados y estudiantes en todo el mundo para crear una amplia gama de proyectos.

Arduino se compone de varias partes esenciales, cada una con su función específica:

- Microcontrolador: es el ‘cerebro’ del arduino, el cual procesa la información que se le proporciona a través de los diferentes sensores y dispositivos conectados. El microcontrolador más comúnmente utilizado en las placas del arduino es el ATMega328.
- Puertos de salida/entrada: los puertos O/I son los que permiten que el microcontrolador se comunique con el mundo exterior, donde los puertos están etiquetados como pines digitales (GPIO) y pines analógicos (ADC).
- Conectores de alimentación: Arduino se alimenta a través de una fuente de alimentación externa, como una batería o un adaptador de corriente. La mayoría de las placas de Arduino tiene un conector de alimentación que acepta una fuente de alimentación de 5V a 12V DC.
- Cristal oscilador: es el componente que proporciona la señal del reloj necesaria para sincronizar las operaciones del microcontrolador.
- Regulador de voltaje: Es el componente que se encarga de regular el voltaje de la fuente de alimentación a un nivel constante de 5V o 3.5V, que es el voltaje de operación de la mayoría de los componentes de Arduino.
- USB-to-serial converter: Es el componente que permite que la placa de Arduino se comunique con una computadora a través de un cable USB. Este componente convierte las señales de la placa Arduino en señales seriales que pueden ser interpretadas por una computadora.

**Breadboard:** Es una placa base o tablero que se utiliza para crear prototipos electrónicos sin necesidad de soldar los componentes. También se le conoce como una “placa de pruebas”, donde sus orificios de un breadboard están conectados electrónicamente a través de pistas conductoras ocultas debajo de su superficie. Esto permite que los componentes conectados a diferentes orificios se organizaran en filas y columnas, y están diseñados para aceptar componentes electrónicos con pastillas o pines que se insertan en los orificios.

**Micro Servo:** Es un tipo de motor pequeño y preciso utilizado en aplicaciones de modelismo y robótica para controlar el movimientos de pequeños objetos o partes mecánicas, consiste en un conjunto de engranajes y un circuito de control interno, todo ello contenido es una carcasa compacta y liviana. La carcasa suele tener un tamaño de alrededor de 20-30 mm de ancho, 10-15 mm de altura y 30-40 mm de longitud.

**Sensor ultrasónico HC-SR04:** Es un sensor de distancia muy común utilizado en proyectos electrónicos, robótica y domótica. El sensor funciona emitiendo ondas sonoras de alta frecuencia y midiendo el tiempo que tarda la onda en reflejarse en un objeto y volver al sensor. A partir de esta medición, el sensor puede calcular la distancia del objeto.

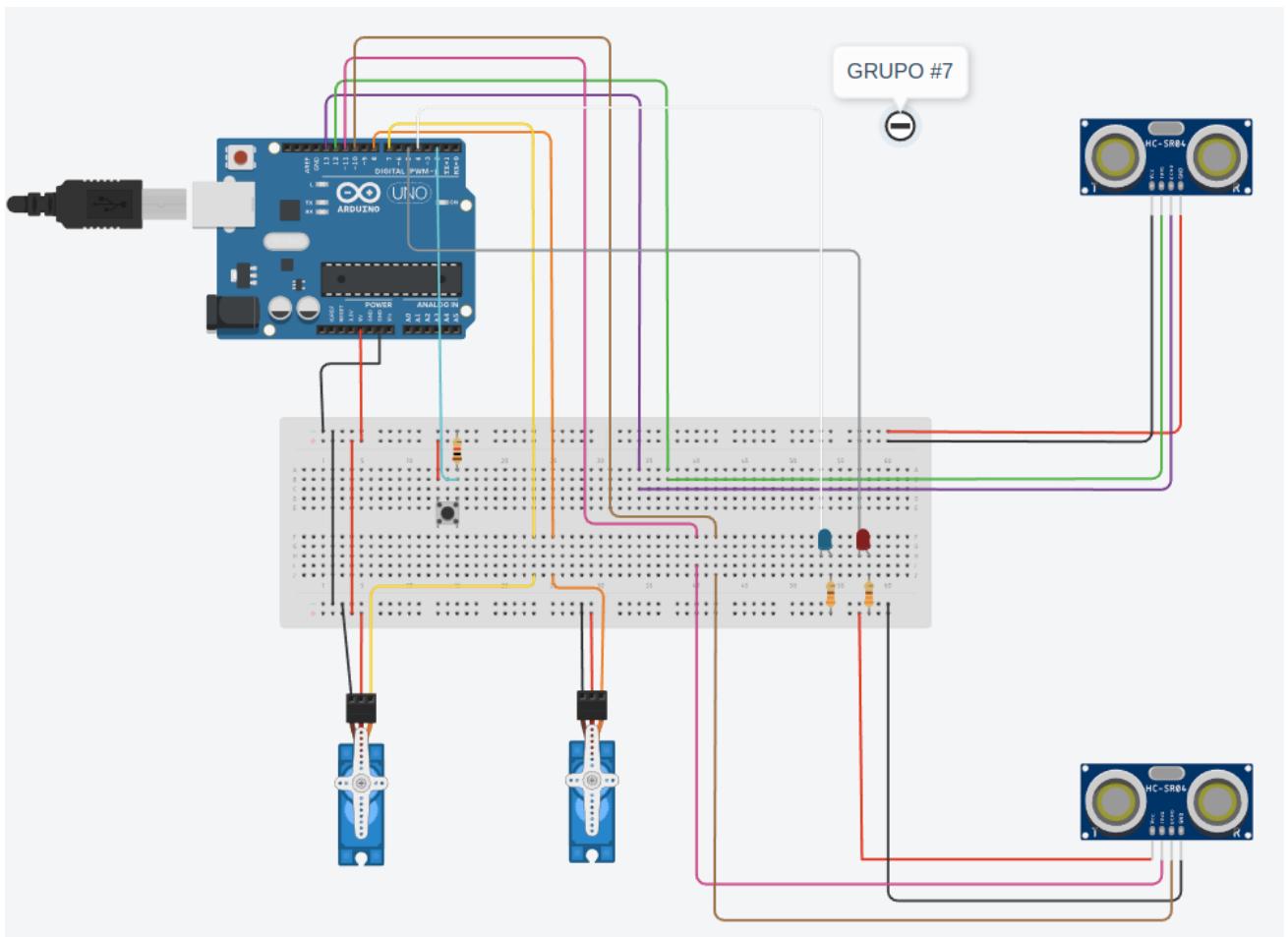
**YwRobot 3,3/5 V:** Módulo de fuente de alimentación para protoboard. Ideal para trabajos o prototipos en donde se requiera una fuente regulada. Esta tarjeta suministra directamente tu protoboard con alimentación de 5 o 3.3V. Compatible con las protoboards que tengan rieles o líneas de VCC y GND. La corriente máxima que suministra es de 700mA pero si estas tomando el USB de la computadora entonces es de aproximadamente 400mA.

**Bluetooth HC-06:** Es un módulo de comunicación inalámbrica Bluetooth serie que se puede utilizar con Arduino u otros microcontroladores. El HC-06 funciona como un dispositivo esclavo, lo que significa que solo puede conectarse a un dispositivo maestro, como un teléfono inteligente, una computadora o un microcontrolador.

## Prototipo #1 para el proyecto:

Link de prototipo realizado en tinkercad: <https://www.tinkercad.com/things/fd6vFNDPSkQ>

Imagen prototipo:



Código para la prueba de sensores y servos:

```
// C++ code
//
#include <Servo.h>
int serv_one=0;
int serv_two=0;
Servo servo_8; //servo_two
Servo servo_7; //servo_one
//codigo sensores ultrasónico
int TRIG_1 = 12;
int ECHO_1 = 13;
int TRIG_2 = 11;
```

```

int ECHO_2 = 10;
int LED = 5;
int LED_2 = 4;
int duracion;
int distancia=0;
int distancia2=0;
long tp_init(int triger_pin, int echo_pin);
long distance(int triger_pin, int echo_pin);

void setup()
{
    servo_8.attach(8,500,2500);
    servo_7.attach(7,500,2500);
    pinMode(TRIG_1, OUTPUT);
    pinMode(ECHO_1, INPUT);
    pinMode(LED,OUTPUT);
    pinMode(TRIG_2, OUTPUT);
    pinMode(ECHO_2, INPUT);
    pinMode(LED_2,OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    for(serv_one=0;serv_one<=100;serv_one+=1){
        servo_8.write(serv_one);
        delay(10);
    }
    for(serv_one=180;serv_one>=0;serv_one-=1){
        servo_8.write(serv_one);
        delay(10);
    }
    for(serv_two=0;serv_two<=100;serv_two+=1){
        servo_7.write(serv_two);
        delay(10);
    }
    for(serv_two=180;serv_two>=0;serv_two-=1){
        servo_7.write(serv_two);
        delay(10);
    }
    // PARA LOS SENSORES
    distancia= distance(TRIG_1, ECHO_1);
}

```

```

Serial.println(distancia);
if(distancia>45){
    digitalWrite (LED, HIGH);
}else{
    digitalWrite (LED, LOW);
}
distancia2= distance(TRIG_2, ECHO_2);
Serial.println(distancia2);
if(distancia2>45){
    digitalWrite (LED_2, HIGH);
}else{
    digitalWrite (LED_2, LOW);
}

}

long tp_init(int triger_pin, int echo_pin){
    digitalWrite(triger_pin,LOW);
    delayMicroseconds(2);
    digitalWrite(triger_pin,HIGH);
    delayMicroseconds(10);
    digitalWrite(triger_pin,LOW);
    long microseconds= pulseIn(echo_pin,HIGH);
    return microseconds;
}

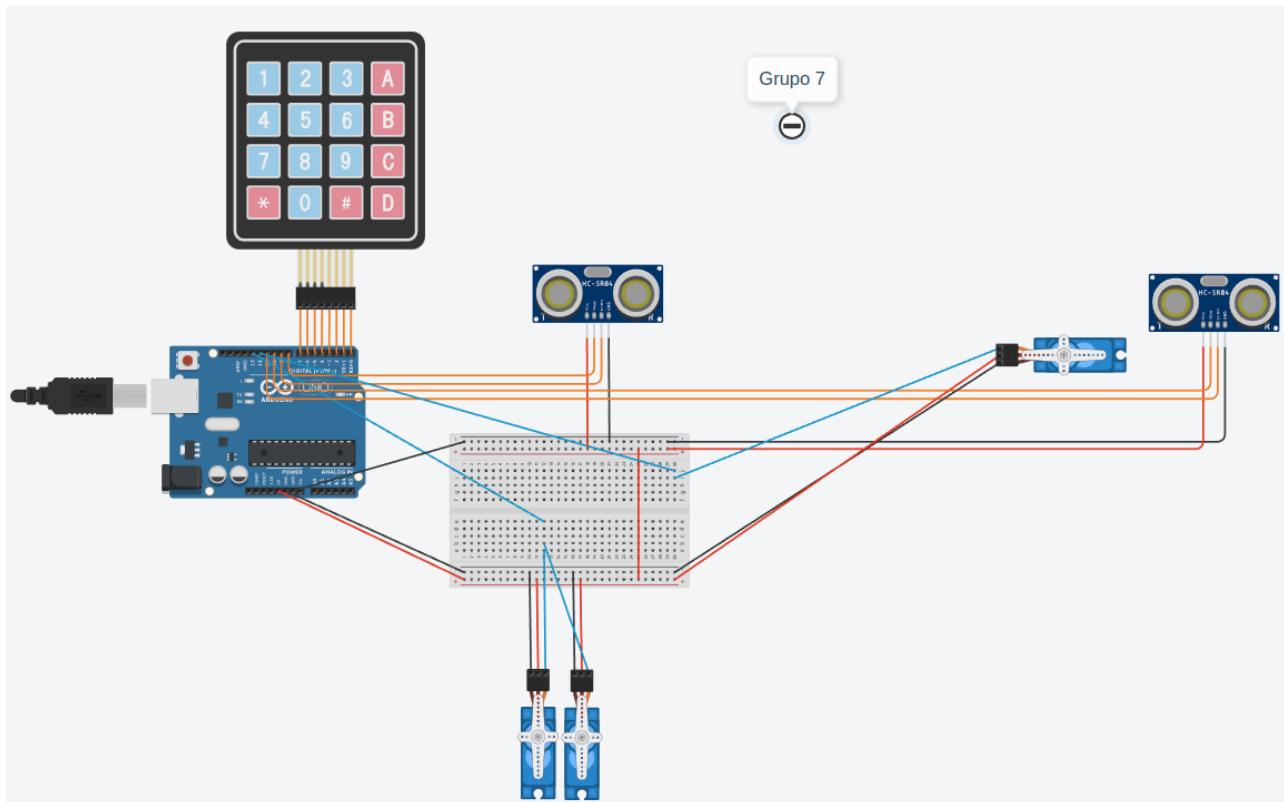
long distance(int triger_pin, int echo_pin){
    long microseconds= tp_init(triger_pin,echo_pin);
    long distance;
    distance= microseconds/29/2;
    if(distance==0){
        distance=999;
    }
    return distance;
}

```

## Prototipo #2 para el proyecto:

Link de prototipo realizado en tinkercad: <https://www.tinkercad.com/things/f6T0Q6kqqH1>

Imagen prototipo:



Código para la prueba de sensores y servos y distancia de angulos:

```
class NewPing {  
public:  
    NewPing(int TRIGGER_PIN, int ECHO_PIN, int MAX_DISTANCE) {  
        trigPin = TRIGGER_PIN;  
        echoPin = ECHO_PIN;  
        maxDistance = MAX_DISTANCE;  
  
        pinMode(trigPin, OUTPUT);  
        pinMode(echoPin, INPUT);  
    }  
  
    int ping_cm() {  
        // Clears the trigPin  
        digitalWrite(trigPin, LOW);  
        delayMicroseconds(2);  
        digitalWrite(trigPin, HIGH);  
        delayMicroseconds(10);  
        digitalWrite(trigPin, LOW);  
  
        // Reads the echoPin, returns the sound wave travel time in microseconds  
        unsigned long duration = pulseIn(echoPin, HIGH);  
  
        // Calculating the distance  
        float cm = (duration * 0.0343) / 2;  
        return cm;  
    }  
};
```

```

delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in
microsecond
long duration = pulseIn(echoPin, HIGH);
// Calculating the distance
int distance = duration * 0.034 / 2;
// Checks out of range
if (distance > maxDistance) {
    distance = 0;
}
return (distance);
}

private:
    int trigPin;
    int echoPin;
    int maxDistance;
};

#include <Keypad.h>
#include <Servo.h>

// SERVO MOTORS
#define SERVOPIN 13
#define SERVO_RAMP 12
int servo_angle = 0;
Servo servo_1;
Servo servo_2;

// KEYPAD
byte rows[4] = {7, 6, 5, 4};
byte cols[4] = {3, 2, 1, 0};
char options[4][4] = {{'1', '2', '3', 'A'},
                      {'4', '5', '6', 'B'},
                      {'7', '8', '9', 'C'},
                      {'*', '0', '#', 'D'}};
Keypad keyword = Keypad(makeKeymap(options), rows, cols, 4, 4);
char button;

```

```

// ULTRASONIC SENSORS
#define TRIGGER2_PIN 8
#define ECHO2_PIN 9
#define TRIGGER_PIN 10
#define ECHO_PIN 11
#define MAX_DISTANCE 200
#define MIN_DISTANCE 5
NewPing sensor_ult1(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
NewPing sensor_ult2(TRIGGER2_PIN, ECHO2_PIN, MAX_DISTANCE);

void setup() {
    Serial.begin(9600);
    servo_1.attach(SERVOPIN);
    servo_2.attach(SERVO_RAMP);
    servo_1.write(0);
    servo_2.write(90);
}

void loop() {
    // Read the keypad
    button = keyword.getKey();
    if (button) {
        Serial.println(button);
        if (button == '1') {
            servo_angle = 0;
        } else if (button == '4') {
            servo_angle = 45;
        } else if (button == '7') {
            servo_angle = 90;
        } else if (button == '*') {
            calculate_distance();
        }
        move_servo(servo_angle);
    }
}

// Function to move the servos
void move_servo(int angle) { servo_1.write(angle); }

void calculate_distance() {

```

```

long time_start = 0;
long time_end = 0;
int distance = sensor_ult1.ping_cm();
int distance2 = sensor_ult2.ping_cm();
servo_2.write(0);

while (distance2 > MIN_DISTANCE) {
    if (time_start == 0) {
        // Get the time when the first sensor change the distance
        time_start = millis();
    }

    // Calculate distance with two sensors
    distance = sensor_ult1.ping_cm();
    distance2 = sensor_ult2.ping_cm();
}

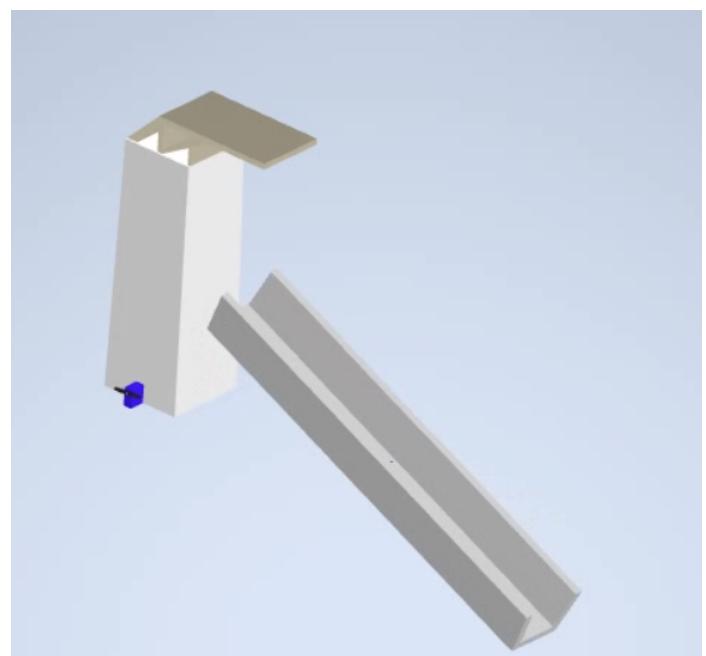
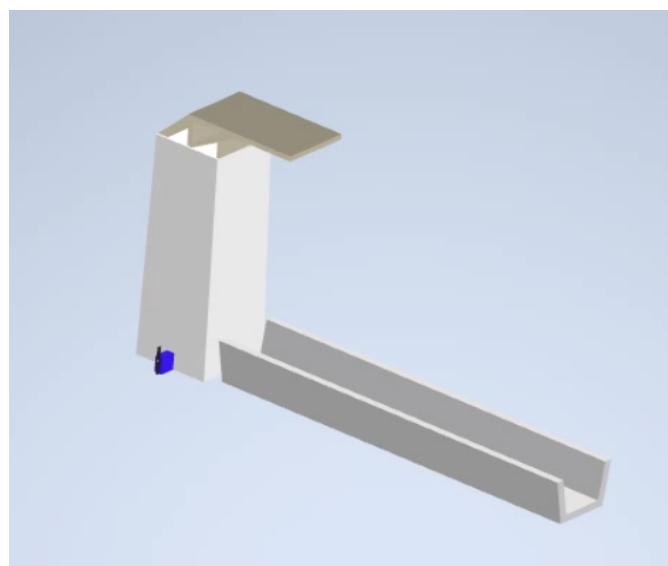
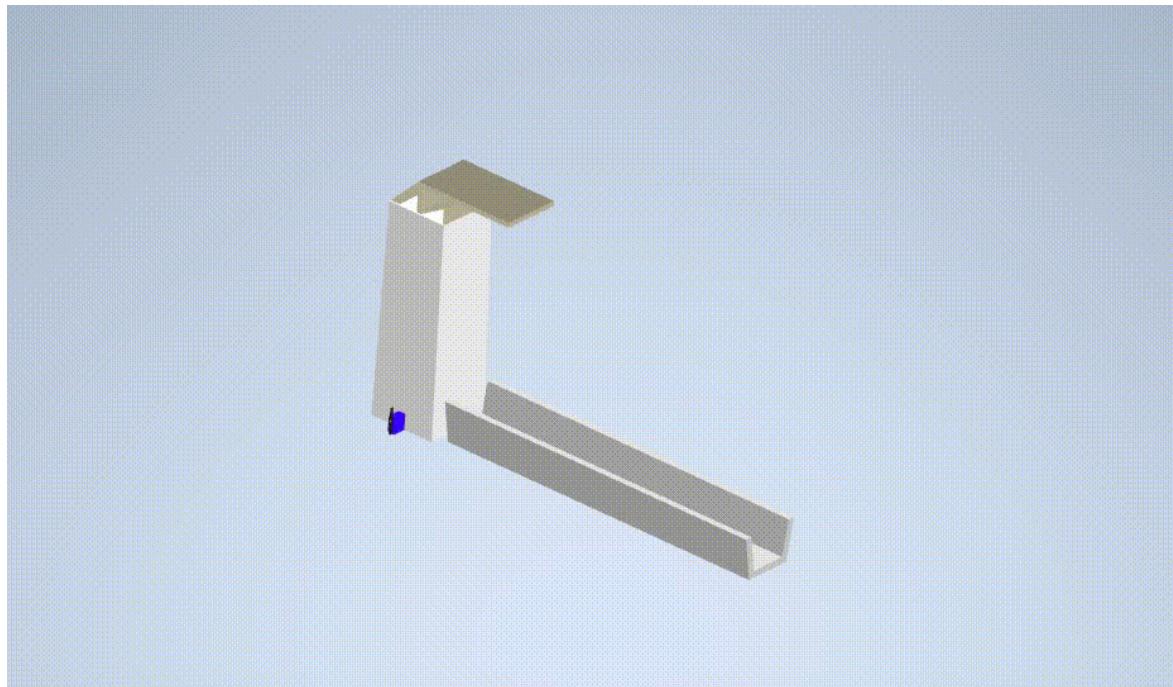
// if distance is less than MIN_DISTANCE, then the sensor
detected an object
if (distance2 <= MIN_DISTANCE) {
    time_end = millis();
    // Calculate time elapsed
    float time_elapsed = (time_end - time_start);

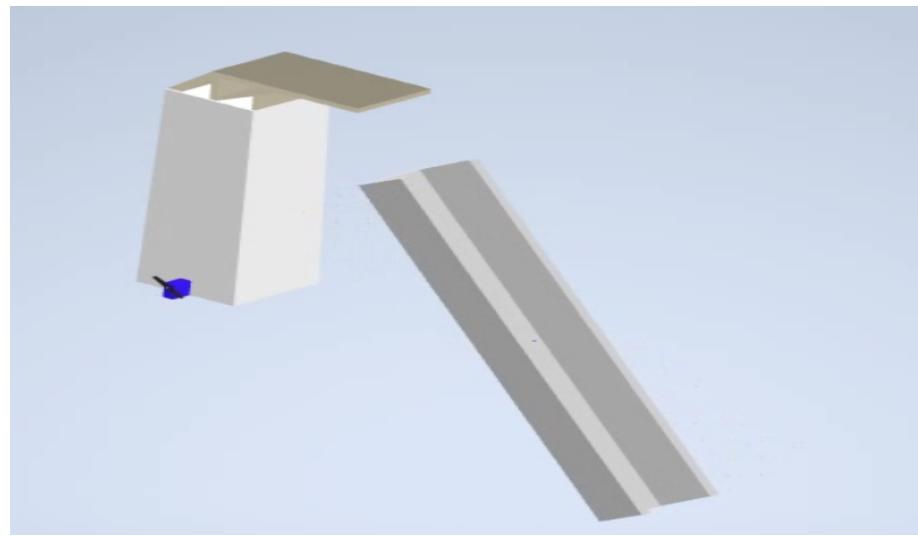
    Serial.print(
        "Tiempo transcurrido hasta que el primer sensor detectó el
mínimo: ");
    Serial.print(time_elapsed);
    Serial.println(" ms");
}
delay(100);
}

```

## **Modelo en 3D de la maqueta a realizar**

- Modelo en Acción

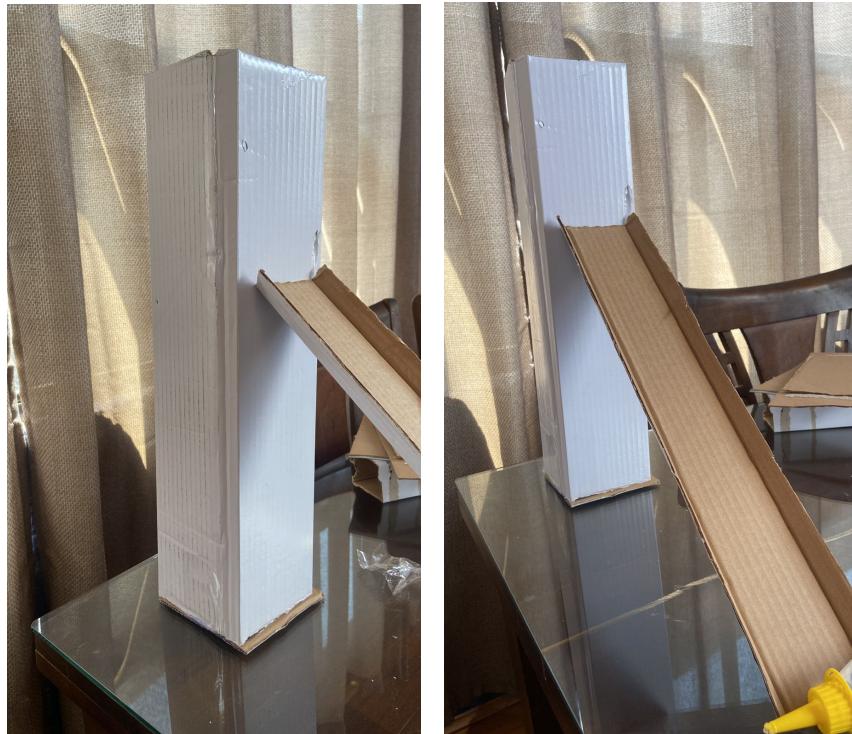




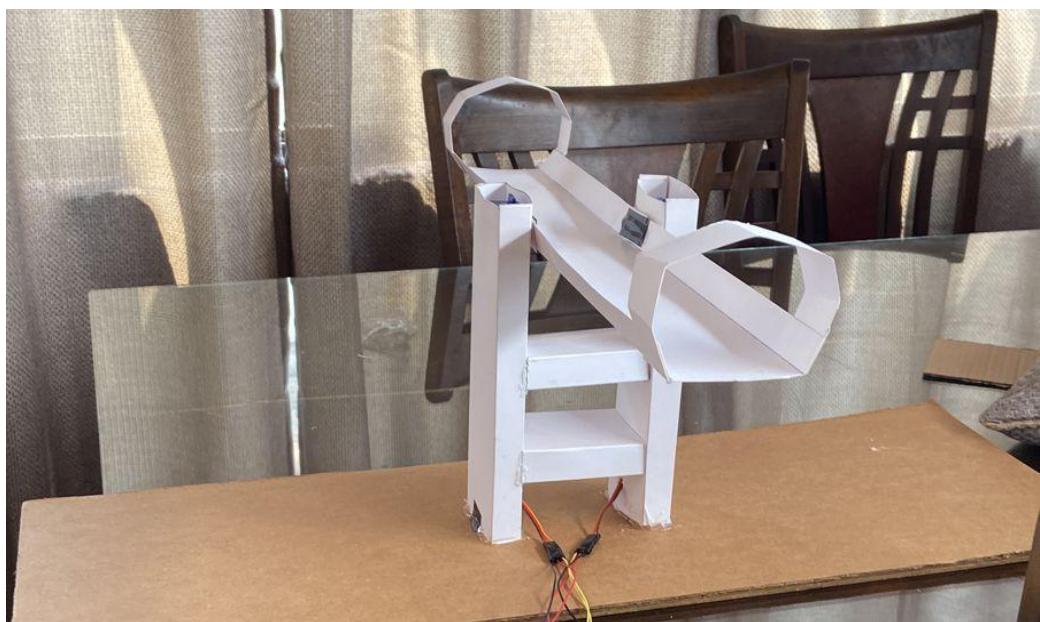
link video modelo rampa: [Prototipo](#)

## **Maqueta Física**

### **Prototipo #1**

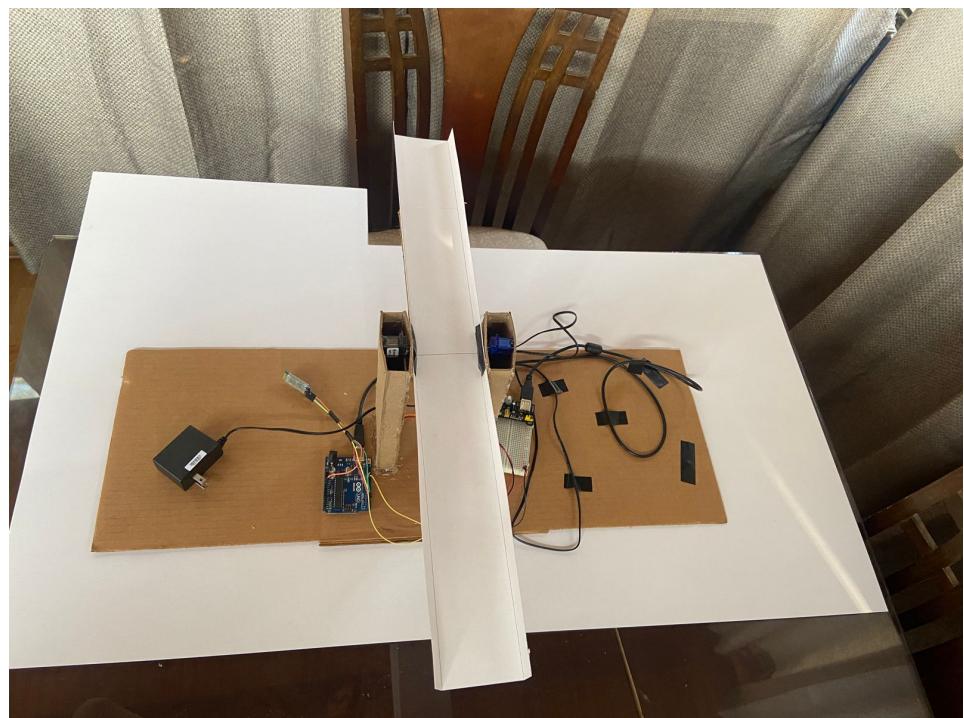
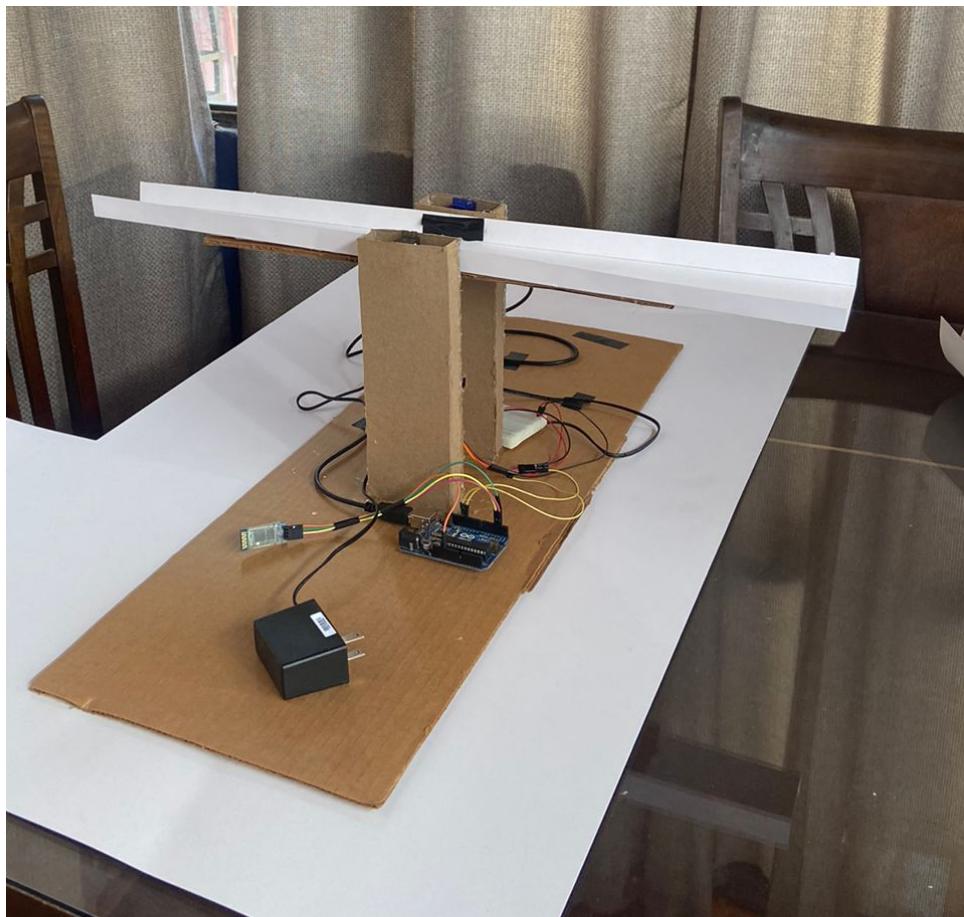


### **Prototipo #2**



## Prototipo #3

### Definitivo



## **Tecnologías usadas**

### **Arduino - IDE**

Arduino es una plataforma de hardware y software libre diseñada para la creación de dispositivos electrónicos interactivos. El IDE de Arduino es una herramienta de programación gráfica que permite la programación de microcontroladores mediante la escritura de código en lenguaje C/C++. Además, cuenta con una amplia variedad de bibliotecas y ejemplos que facilitan la programación de diferentes componentes, como sensores y actuadores. Esta plataforma es muy utilizada en proyectos de robótica, electrónica y automatización.

### **Kotlin - Android Studio**

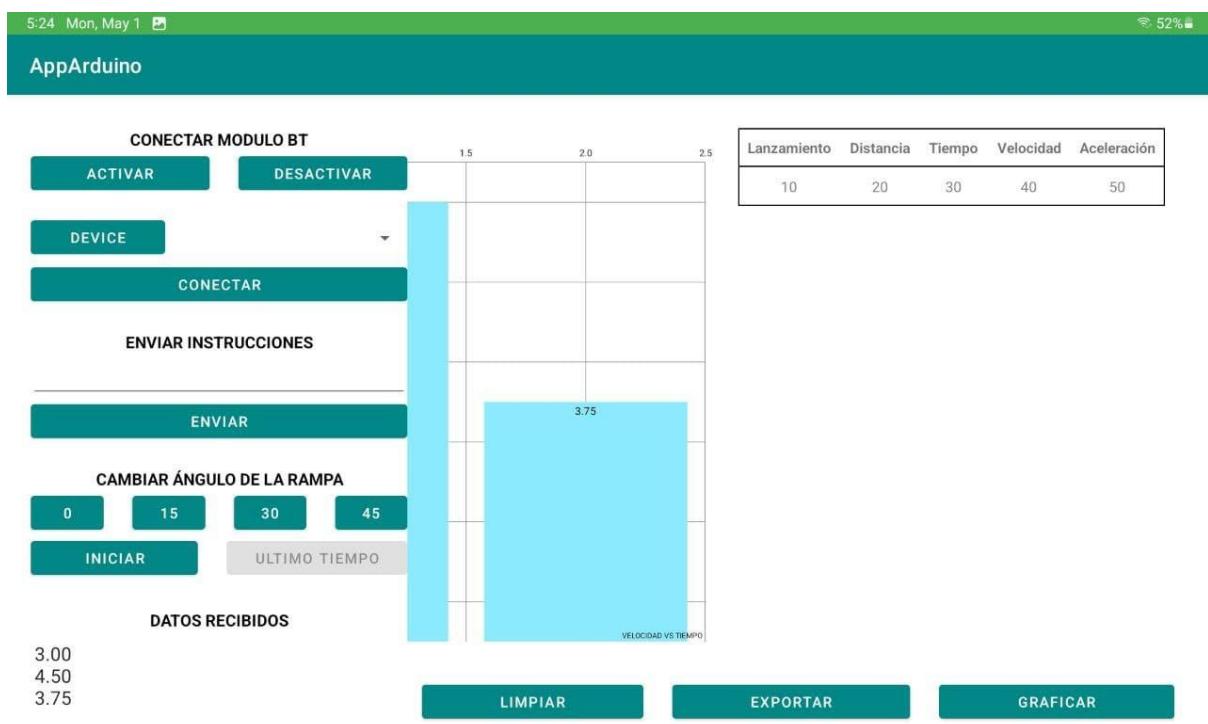
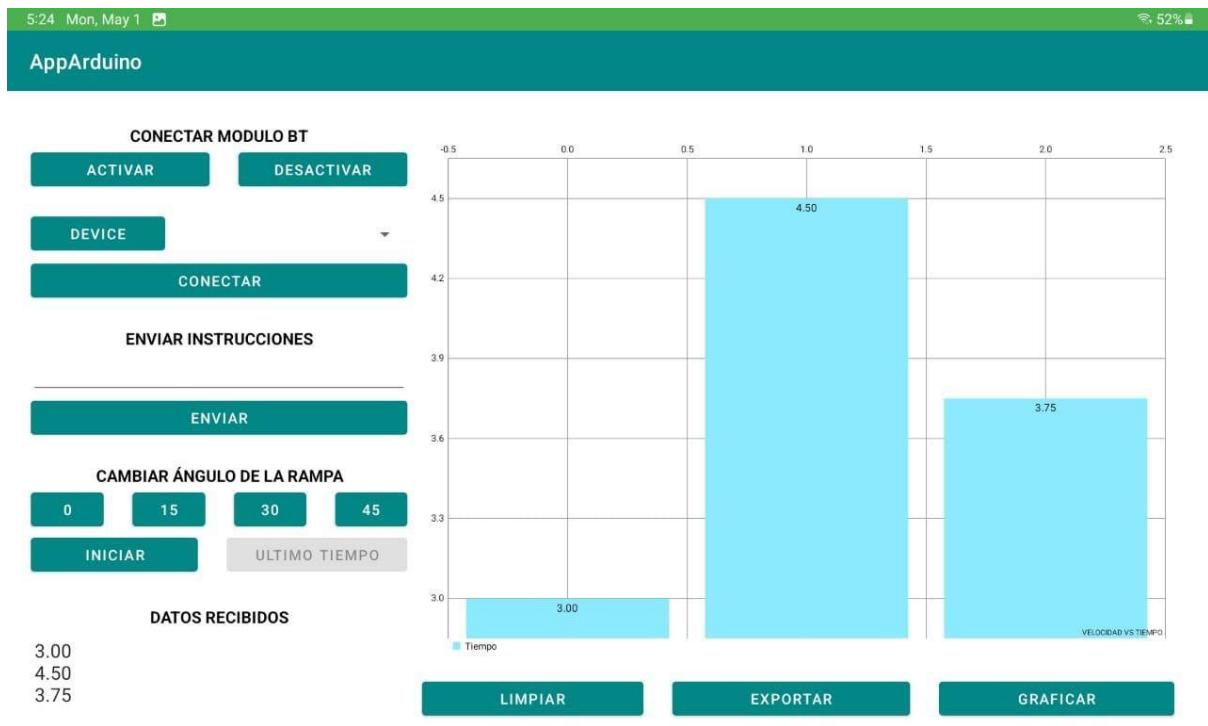
Kotlin es un lenguaje de programación multiplataforma que se ejecuta sobre la máquina virtual de Java. Este lenguaje ha ganado popularidad en el desarrollo de aplicaciones móviles para Android, ya que ofrece una sintaxis más sencilla y legible que Java, así como mejores características de seguridad y eficiencia en la ejecución de código. Además, cuenta con una amplia gama de bibliotecas y herramientas que facilitan el desarrollo de aplicaciones móviles.

### **Uso de ambas tecnologías**

El uso de Arduino y su IDE permitió la configuración y calibración de los componentes del sistema, mientras que Kotlin en Android Studio permitió el desarrollo de la aplicación móvil para la visualización de los resultados. Ambas tecnologías fueron fundamentales para el éxito del proyecto y destacan por su versatilidad y eficiencia en el desarrollo de soluciones tecnológicas.

**Código : [GitHub](#)**

## Modelo de la aplicación móvil



## Conclusión

El proyecto ha logrado funcionar de manera exitosa gracias al uso de tecnologías como el IDE de Arduino y Kotlin en Android Studio. El IDE de Arduino permitió la configuración y calibración de los componentes, lo que permitió una mejor precisión en la captura de datos, mientras que Kotlin en Android Studio permitió el desarrollo de una aplicación móvil eficiente para la visualización y comparación de los resultados. El uso de estas tecnologías fue clave para el desarrollo de una solución innovadora y eficiente, lo que permitió alcanzar los objetivos del proyecto de manera satisfactoria.

Además de permitir el correcto funcionamiento del proyecto, el uso de estas tecnologías también trajo consigo numerosas ventajas. Por un lado, el uso del IDE de Arduino facilitó la programación de los componentes del sistema, lo que redujo significativamente el tiempo y los recursos necesarios para el desarrollo del proyecto. Además, la amplia variedad de bibliotecas y ejemplos disponibles en el IDE de Arduino permitió una mayor flexibilidad y adaptabilidad del sistema a diferentes situaciones.

Por otro lado, el uso de Kotlin en Android Studio permitió el desarrollo de una aplicación móvil altamente eficiente, lo que permitió una mejor visualización y comparación de los datos obtenidos por el sistema. La sintaxis simple y legible de Kotlin facilitó la codificación de la aplicación, lo que permitió una mayor rapidez y calidad en el desarrollo. Además, la amplia variedad de herramientas y bibliotecas disponibles en Kotlin en Android Studio permitió una mayor adaptabilidad de la aplicación a diferentes situaciones y necesidades.

En general, el uso de estas tecnologías permitió el desarrollo de una solución tecnológica eficiente e innovadora, que logró cumplir con los objetivos del proyecto de manera satisfactoria. Asimismo, su uso permitió reducir significativamente los tiempos y costos de desarrollo, lo que podría llevar a su implementación en una gran variedad de aplicaciones y proyectos futuros.