**Erasmus+ KA210-VET**

**Small-scale partnerships in vocational**

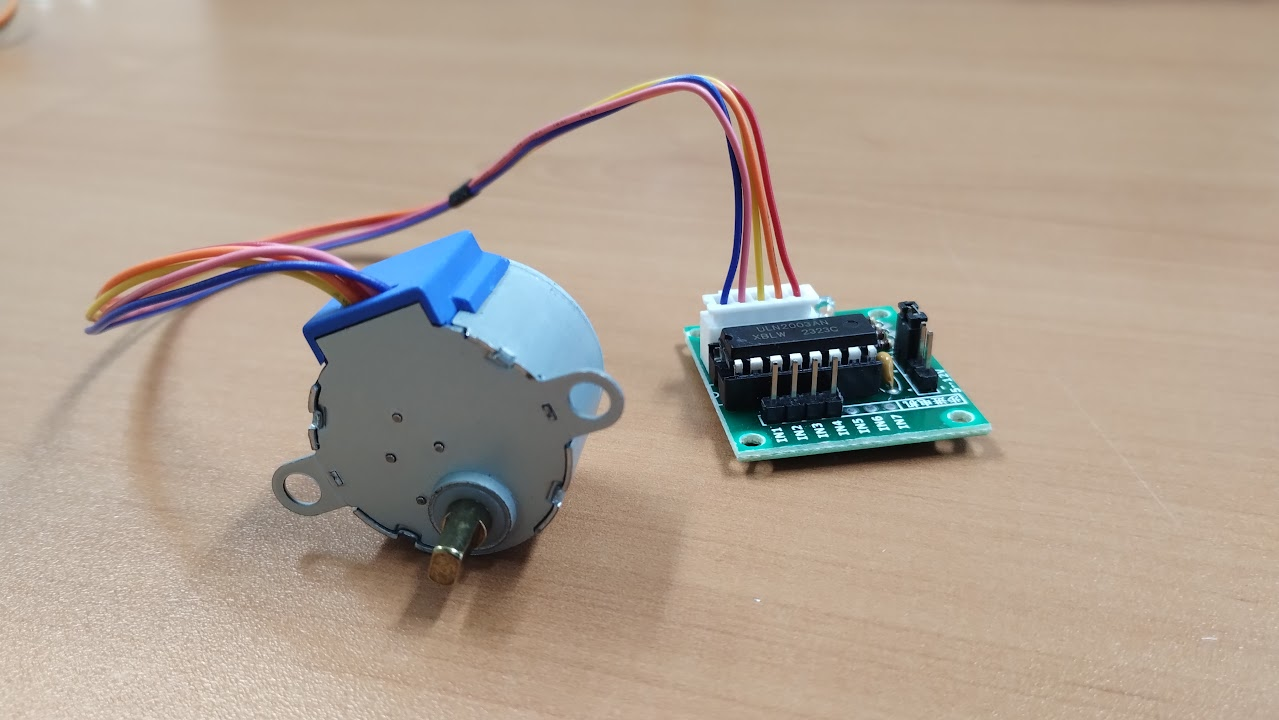
**education and training**

**Project Title: “Using Arduinos in Vocational Training”**

**Project Acronym: “UsingARDinVET”**

**Project No: “2023-1-RO01-KA210-VET-000156616”**

**Motors Module and Training Kit**

****

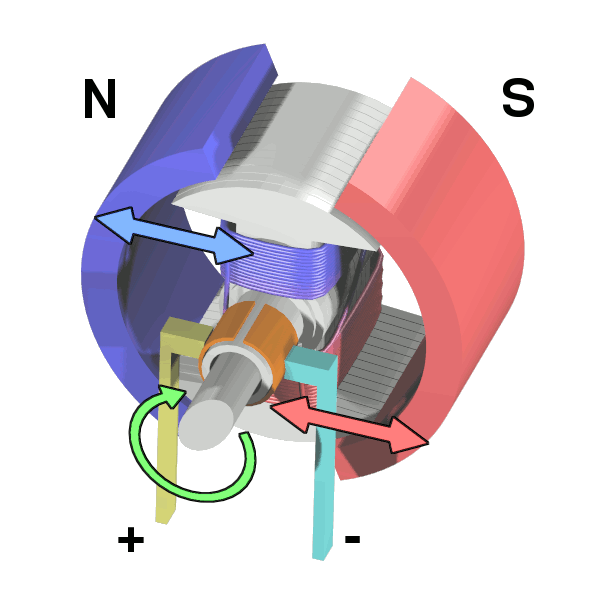
### DC MOTORS

### Overview

A DC (Direct Current) motor is a type of electric motor that runs on direct current electricity. It is one of the most widely used types of motors due to its simplicity, reliability, and ability to provide continuous rotation. DC motors are commonly found in appliances, toys, tools, and various other devices that require mechanical movement.

In this document, we will explain the basic working principle of DC motors, their components, and how they operate.

### Basic Principle of DC Motor Operation

A DC motor works on the principle of electromagnetic induction, which states that when a current-carrying conductor is placed in a magnetic field, it experiences a force that causes it to move. This force is known as the Lorentz force.

Key Principle:

A current flowing through a coil of wire generates a magnetic field around the coil.

The coil is placed in an external magnetic field (usually created by permanent magnets or electromagnets).

The interaction between the magnetic field of the coil and the external magnetic field generates a force, causing the coil to rotate.

This rotational motion is transferred to the motor’s shaft, providing mechanical movement.

### Factors Affecting DC Motor Operation

Several factors can influence the performance of a DC motor:

* **Voltage:** The speed and torque of a DC motor are directly related to the voltage applied to it. Higher voltage generally results in higher speed and torque.
* **Speed:** Increasing the applied voltage increases the motor’s speed because it increases the current through the rotor windings, producing a stronger magnetic field and greater force.
* **Torque:** The torque, or rotational force, is proportional to the current. Higher current means higher torque.
* **Current:** The current determines how much torque the motor can produce. Higher current increases the torque but may also lead to overheating if the motor is not properly cooled.
* **Resistance:** The resistance of the rotor windings affects the current flow. Higher resistance limits the amount of current that can flow, which reduces the motor's torque. On the other hand, lower resistance allows more current to flow, increasing torque and speed.
* **Magnetic Field Strength:** The strength of the stator’s magnetic field also affects the motor’s performance. Stronger magnetic fields result in greater interaction between the stator and rotor, producing more force and thus higher performance.

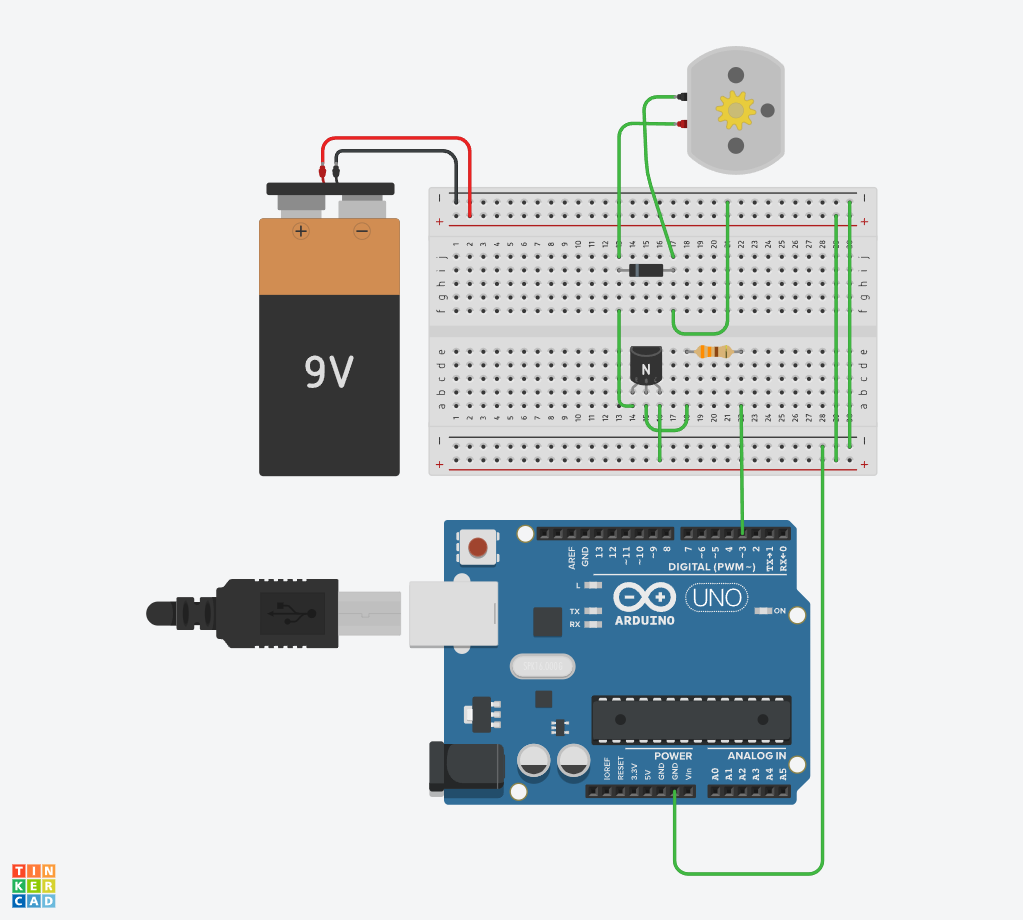
### Applications

Electric DC motors are used when a smooth continuous rotation is needed. Plain DC motors are voltage controlled so we can decide how much speed and torque the motor will produce. Plain DC motors can’t be used when precision control over speed or position of the motor is needed. In these cases, we usually choose step motors or servomotors instead.

### Power requirements

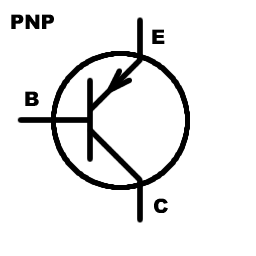
Most of the available DC motors consumptions are higher than the maximum current than Arduino can deliver and can cause damage to Arduino digital outputs, so we need external power supply to the motor and control this power supply using Arduino’s output pins. The most usual way to control DC motors in Arduino is using a motor controller or building our own controller using a transistor.

### Example project: Driving a DC motor with a transistor



This circuit uses a PNP 2N2222 transistor to control the 9V power supply using Arduino. To build this you will need:

* 1 DC motor
* 1 PNP 2N2222 transistor
* 1 330Ω resistor
* 1 Zener diode
* 1 9V battery



PNP Transistor can be used as a electronically controlled switch. The current will flow from emmiter pin (E) to colector pin (C) only when enought current flows from base pin (B) to colector (C). In other words we can control the current between E and C pins using B pin.

The circuit is very simple. The transistor will allow to pass 9V current from the batery to the motor only when pin 3 is active. As pin 3 is a PWM pin we can send different pulse witdh to control the speed of the motor using analogWrite method.

The diode is used to avoid the motor to send current to the circuit due to inertial movement of the motor. The resistance is used to adapt the output of the arduino pin to the correct input values of the transistor.

The code to control the motor using pin 3 is:

Example program 1: Runing the motor at different speeds

**int motorPin = 3;** //The pin attached to the motor

**void setup()**

**{**

**pinMode(motorPin, OUTPUT);**

**}**

**void loop()**

**{**

**analogWrite(motorPin, 255);** //Full speed

**delay(2000);**

**analogWrite(motorPin, 100);** //Medium Speed

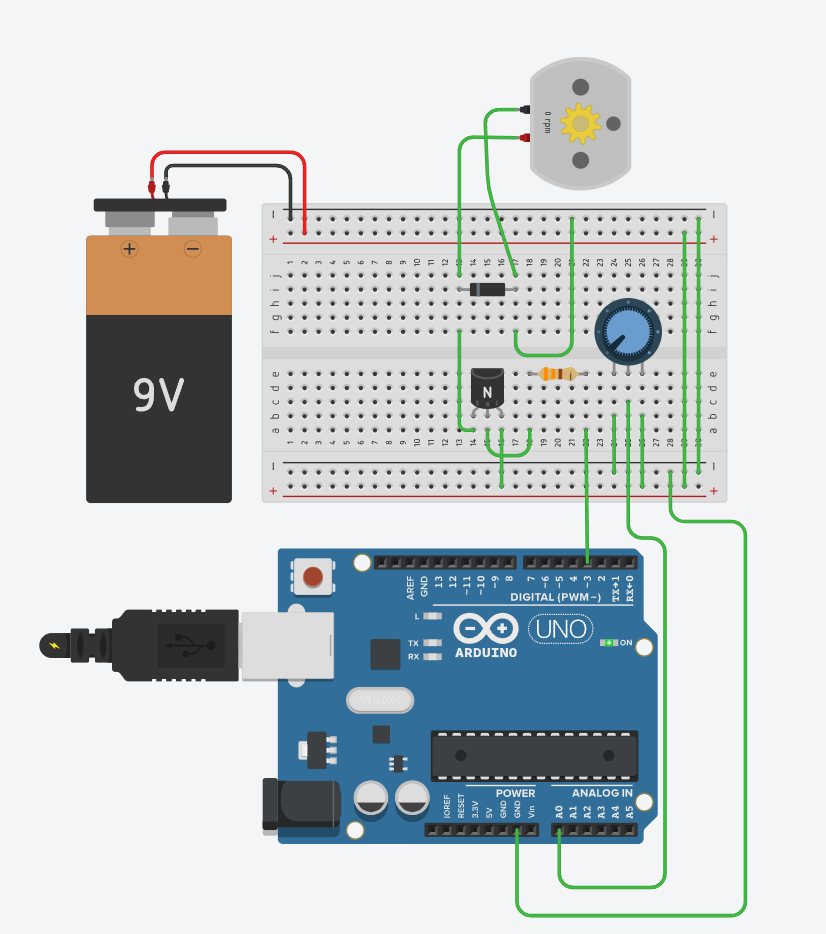
**delay(2000);**

**analogWrite(motorPin, 10);** //Low speed

**delay(2000);**

**}**

Lets modify out circuit to add a potetiometer to control motor’s speed.



Example program 2: Controlling the motor with a potentiometer

**int motorPin = 3; //The pin attached to the motor**

**int controlPin = 0; //The pin attached to the pot**

**int speed = 0; //The speed assingned to the motor**

**void setup()**

**{**

**pinMode(motorPin, OUTPUT); //Set up the motor pin as output pin**

**}**

**void loop()**

**{**

**speed = analogRead(controlPin); //Read the potentiometer value and store it on speed**

**analogWrite(motorPin, speed); //Send speed to the motor**

**delay(500);**

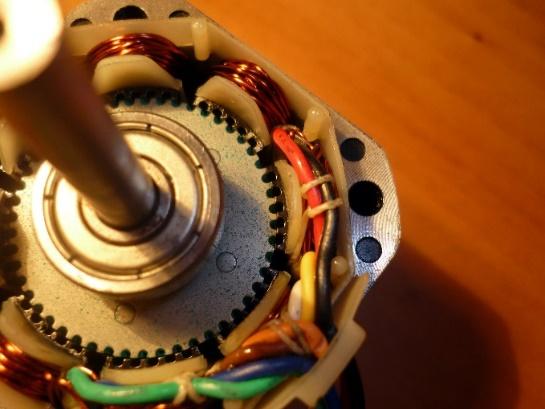
**}**

### STEPPER MOTORS

### Overview

A stepper motor is a type of electric motor that moves in discrete steps, making it ideal for precise control of position, speed, and direction. Unlike traditional DC motors, which continuously rotate, stepper motors divide a full rotation into multiple steps, each typically ranging from 0.9° to 1.8° per step, depending on the motor design. This unique feature allows them to achieve very accurate and repeatable movements without the need for encoders or other position-sensing devices.

### How Stepper Motors Work

The operation of a stepper motor is based on electromagnetism. Inside a stepper motor, there are multiple coils or windings arranged around a central rotor. When a current is passed through a specific coil, it creates a magnetic field that interacts with the rotor, causing it to rotate by a certain angle. By sequentially energizing different coils in a controlled pattern, the rotor is moved in precise increments (steps).

The key to controlling the movement of a stepper motor lies in the driver circuit, which regulates the sequence and timing of current pulses sent to the coils. The driver can operate in different modes, such as:

Full step: Energizes one coil at a time, giving a larger step size (e.g., 1.8° per step).

Half step: Energizes two coils alternately, allowing smaller steps and smoother motion.

Microstepping: Divides each full step into smaller steps, improving smoothness and resolution, and reducing vibration.

### Advantages of Stepper Motors

* **Precision and accuracy:** Stepper motors are highly accurate, making them ideal for applications that require exact positioning, such as 3D printers, CNC machines, and robotic arms.
* **No feedback required:** Stepper motors can operate without external feedback systems (like encoders) because the position of the rotor is determined by the number of steps moved.
* **Reliability:** These motors are simple in construction, leading to fewer parts that could fail.
* **Low cost:** Stepper motors are relatively inexpensive, making them a cost-effective solution for many applications.

### Disadvantages of Stepper Motors

* **Low efficiency:** Stepper motors can be less energy-efficient compared to other types of motors, especially at higher speeds.
* **Torque drop at high speeds:** At high speeds, stepper motors tend to lose torque, which can affect performance in certain applications.
* **Vibration and noise:** Stepper motors can produce vibrations and noise, especially when operating at lower speeds or with low current.

### Applications for Stepper Motors

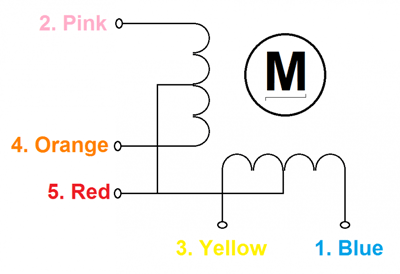
Stepper motors are widely used in applications that require precise control of position and rotation, including:

* **3D printers:** To accurately move the print head and build up the material layer by layer.
* **Robotics**: For precise control of robotic arms and other moving parts.
* **CNC machines**: To move the tool or workpiece in controlled increments.
* **Camera platforms**: For precise pan-and-tilt control in photography and videography.
* **Medical equipment**: For applications such as pumps, prosthetics, and diagnostic machinery.

Stepper motors are an essential component in many applications where precision and control are critical. Their ability to divide full rotation into small, discrete steps allows for accurate positioning, and their simplicity makes them a popular choice for both hobbyists and industrial engineers. Although they may not be the best option for high-speed, high-efficiency applications, their advantages in control and reliability make them indispensable in fields such as robotics, manufacturing, and automation. Usually stepper motors designs use sensors to detect when the moving objects reach the beginning or end of the path, these sensors are called limit switch sensors and are essential determining the initial position of a stepper motor.

### Example project: Moving the second hand of a clock

In this example we are going to use a small stepper motor called 28BYJ-48



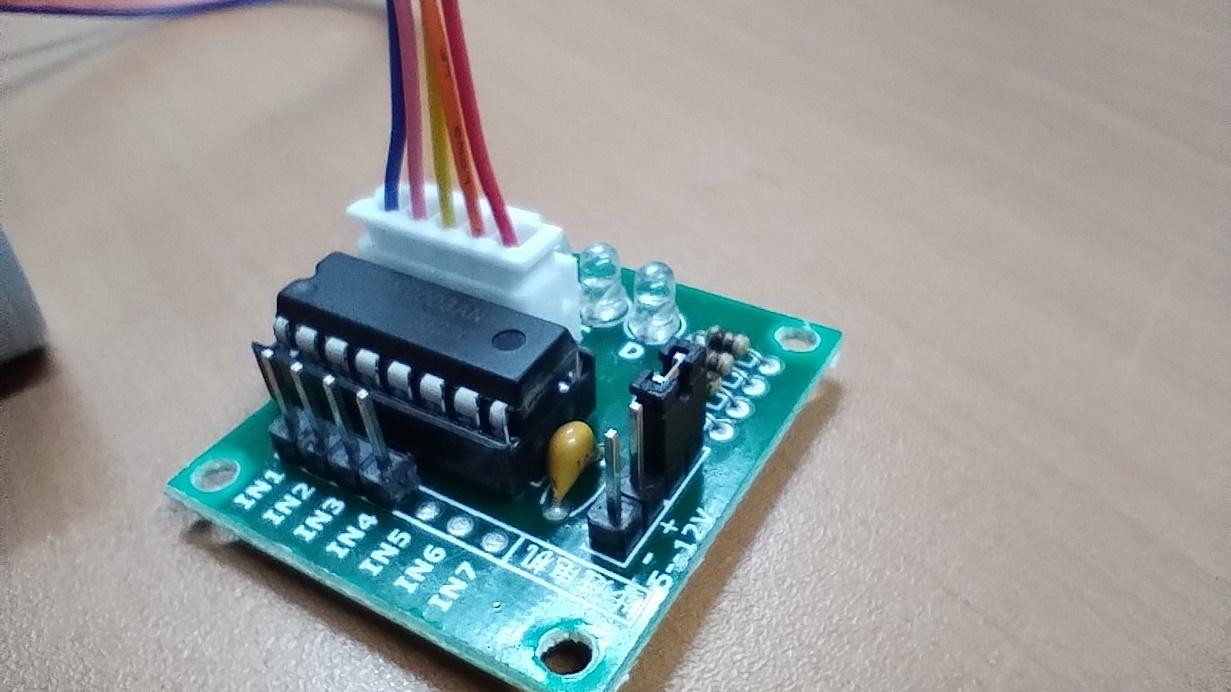
This motor has four coils with a unipolar scheme cabling and is usually controlled by a ULN2003 based circuit that provides current to each coil to avoid damaging the I/O pins of our Arduino.

The 28BYJ-48 motor has 64 steps per revolution when use in half-step mode and has a internal 1:64 gear ratio so 64 \* 64 = 4096 total steps per revolution.

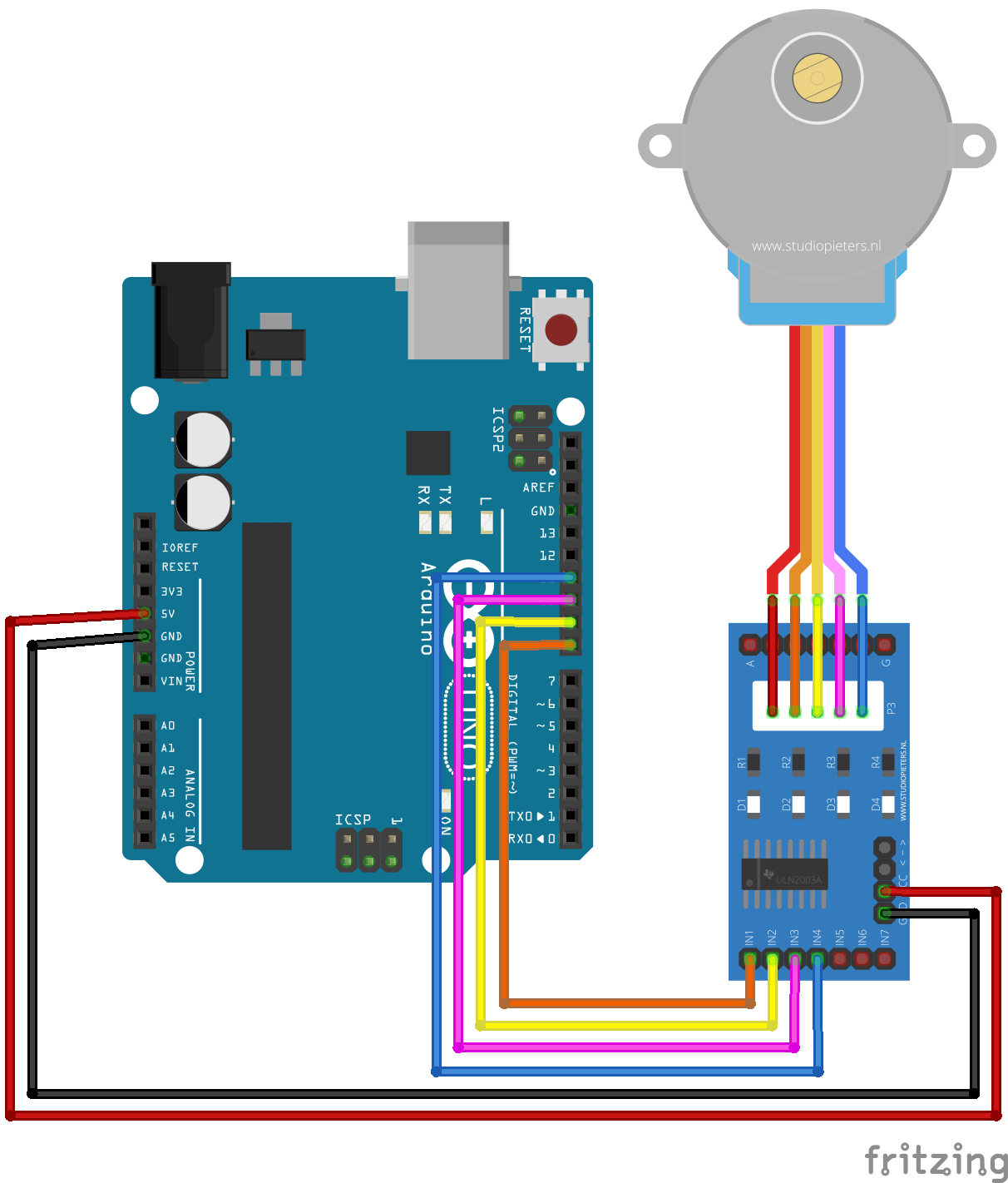
In this project we are going to use:

* 1 28BYJ-48 Stepper motor
* 1 ULN2003 based controller
* Cables
* Arduino Uno board

Connect the stepper motor to the controller using the appropriate plug:



Now connect the controller to the Arduino board following this schema:



To control the stepper motor we are going to use the Stepper.h library. The most useful methods of Stepper library are:

| stepper(steps, pin1, pin2, pin3, pin4) | Creater a new stepper with the following parameters:  Steps: Number of steps per revolution  Pin1 and Pin 2: Pins attached to the motor  Pin 3 and Pin 4: (Optional): Pins attached to the motor if the motor have for cables |
| --- | --- |
| step(steps) | Move the stepper a specific number of steps |
| setSpeed(rpm) | Set the motor speed to a specific revolution per minute. Please consider the maximum speed of your motor. |

Example program: Move the motor one revolution forward and one revolution backwards

**#include <Stepper.h>** // Include the steper control library

**Stepper stepper(4096, 8, 10, 9, 11);** // Create a stepper with 4096 steps per revolution

**void setup() {**

**}**

**void loop {**

**stepper.step(4096);** //One complete revolution

**delay(5000);** //wait for 5 seconds

**stepper.step(-4096);** //One complete revolution backwards (negative)

**delay(5000);** //wait for 5 seconds

**}**

Example program: move the stepper motor 1/60th turn every second.

**#include <Stepper.h>** // Include the steper control library

**Stepper stepper(4096, 8, 10, 9, 11);** // Create a stepper with 4096 steps per revolution

**void setup() {**

**stepper.setSpeed(240)** // 240 rpm speed for quick seconds hand movement

**}**

**void loop {**

**stepper.step(68);** //60 divided into 4096 equals aprox 68 steps per second

**delay(1000);** //each second we move the motor

**}**

### SERVO MOTORS

### Overview

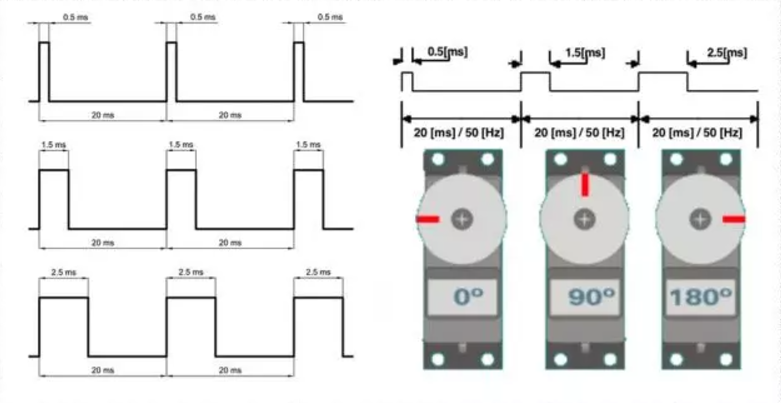
Servo motors are widely used in robotics, automation, and other fields where precise control of angular position is required. They differ from regular motors in that they can rotate to a specific position, rather than continuously rotating. This makes them ideal for tasks like moving robotic arms, controlling the position of a camera, or adjusting the angle of solar panels.

### How do servo motors work?

A servo motor is a small motor equipped with a feedback system that allows it to achieve precise angular movement. Unlike standard DC motors, which rotate continuously in either direction, a servo motor can only rotate through a limited range (typically 0° to 180°), and its position can be controlled very precisely.

The servo motor consists of three key components:

* Motor: Drives the rotation.
* Feedback potentiometer: Monitors the motor's position.
* Control circuit: Receives the control signal and adjusts the motor's position accordingly.

Servo motors operate using Pulse Width Modulation (PWM) signals. The Arduino sends a PWM signal to the servo, which adjusts the motor's position based on the duration of the signal. 

A short pulse (1 ms) might set the servo to 0°.

A long pulse (2 ms) might set the servo to 180°.

A pulse of around 1.5 ms generally places the servo at the 90° position.

By varying the pulse width, the servo motor can be positioned anywhere within its operational range.

### Applications of Servo Motors

* **Robotics:** Servo motors are widely used in robots to control joints, wheels, or actuators.
* **Camera Gimbals:** To stabilize a camera, servo motors can control the orientation of the camera to keep it steady.
* **Antenna Positioning:** Servo motors are used in radio and satellite dishes to adjust antenna direction.
* **RC Vehicles:** Remote-controlled cars, planes, and boats often use servos to steer and control other mechanical components.

Servo motors are essential components for projects that require precise control of angles. With Arduino, controlling a servo is a simple task, thanks to the Servo library, which abstracts away the complexity of generating PWM signals. One of the biggest advantages of servos over stepper motors is that as the servo can detect it’s own position they usually don’t need limit sensors like stepper or DC motors.

### Advantages of servo motors:

* Great precision without needing limit sensors.
* Can maintain their position even with external disturbances.

### Disadvantages of servo motors:

* Limited range of movement not suitable for continuous movement
* Limited maximum speed

### Example Project: Move a Servo motor:

The SG90 is a small and convenient servo motor that can be powered using Arduino pins. Bigger servos should have an external power supply.

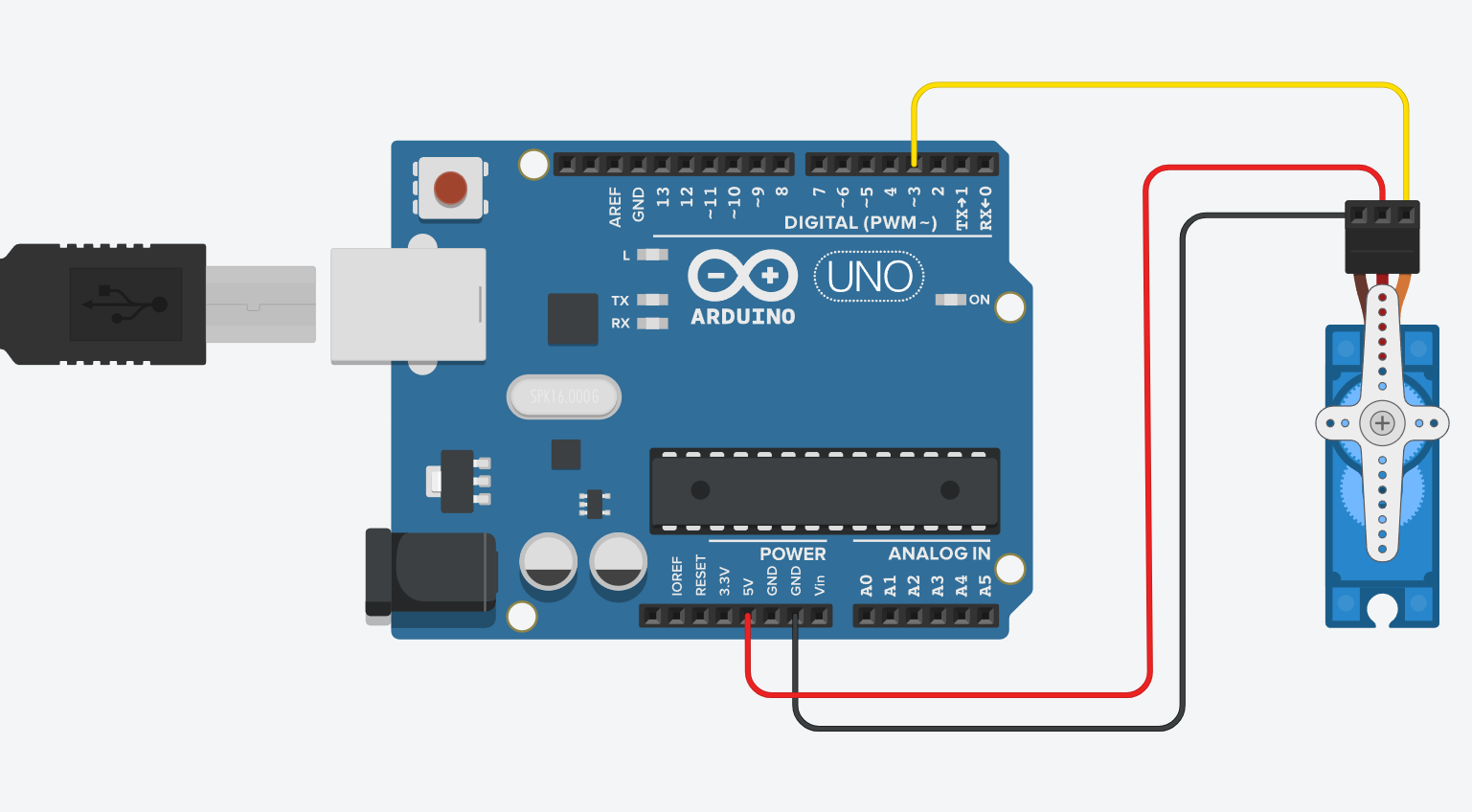
For this project we are going to use:

* 1 Servo motor SG90
* Cables
* Arduino Uno board

The servo control pin (yellow) needs to be connected to a PWM digital pin, we are going to use ~3 pin.

We are going to use Servo.h library to help us to manage the servo. The most importat Servo.h methods are:

| attach(pin, min, max) | Attach the servo variable to a pin. The pin must be a PWM digital pin.  The optional parameters min and max set the pulse witdh, in microseconds, that the servo expect to set the angle to minimun and maximun angle. If no min or max values are provided the default values 544 and 2400 miliseconds are used. |
| --- | --- |
| write(angle) | Set the servo angle at desired value. The servo will move to this position. |



**#include <Servo.h>; //The servo library**

**Servo myServo; //We need to create a servo object**

**void setup()**

**{**

**myServo.attach(3); //Cofigure the servo pin**

**myServo.write(0); //Set servo at 0 degrees angle**

**}**

**void loop()**

**{**

**myServo.write(90); //set servo at 90 angle**

**delay(1000); //wait for 1 second**

**myServo.write(180); //set servo at 180 angle (max)**

**delay(1000); //wait for 1 second**

**myServo.write(0); //set servo at 0 angle (min)**

**delay(1000); //wait for 1 second**

**}**

### Example Project: Car barrier with LASER car detection:

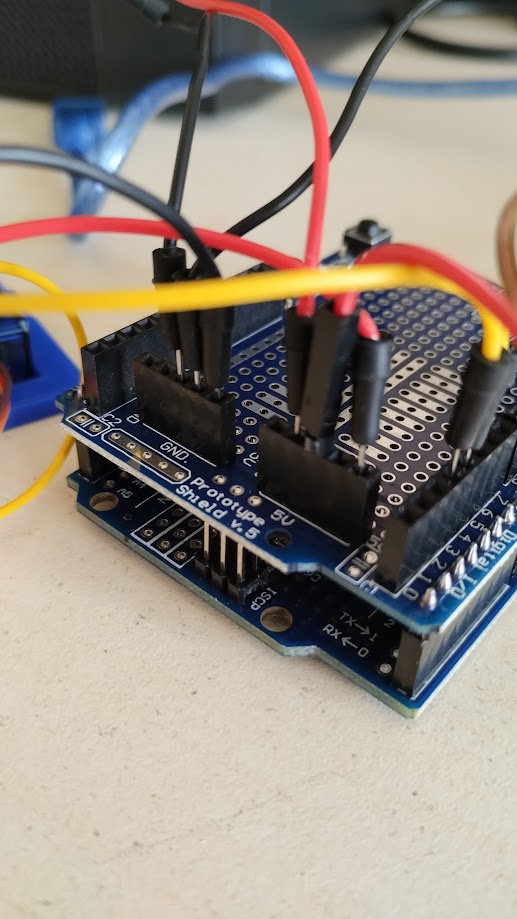
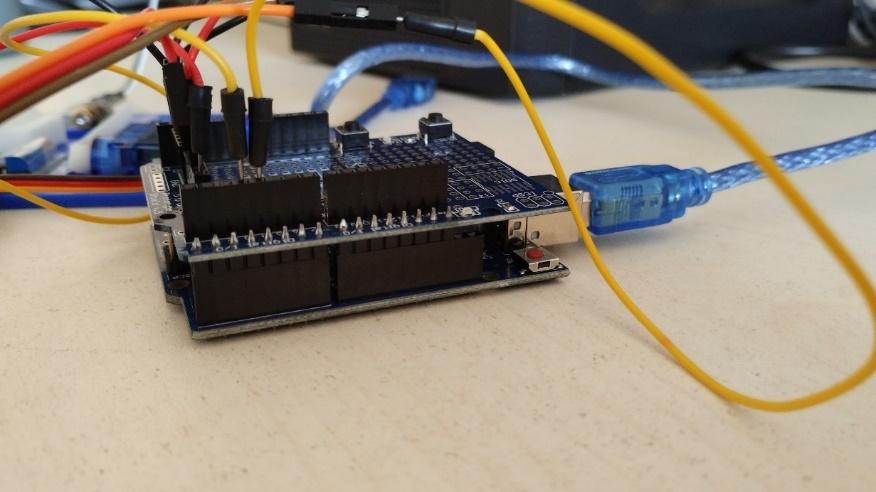
For this project we are going to use:

* 1 Laser emitter module
* 1 Light detector module
* 1 Servo motor SG90
* 1 Proto shield
* Cables
* Support board
* Barrier
* Car
* Arduino Uno board

Imagen que contiene circuito, electrónica

Descripción generada automáticamente

Connect the proto shield over the Arduino board. Please note that there are two headers for 5V and GND that are very convenient to connect the power cables of the sensors and the servo motor.



Connect the – and S pin of the LASER module to the GND and 5V headers of the Proto Shield:

| LASER Emitter Module | |
| --- | --- |
| S | 5V |
| Middle pin | Not Connected |
| - | GND |



Connect the VCC, GND and D0 pins of the light detector module to the corresponding pins of the proto shield:



| Light Detector Module | |
| --- | --- |
| D0 | Digital Pin 4 |
| GND | GND |
| VCC | 5V |

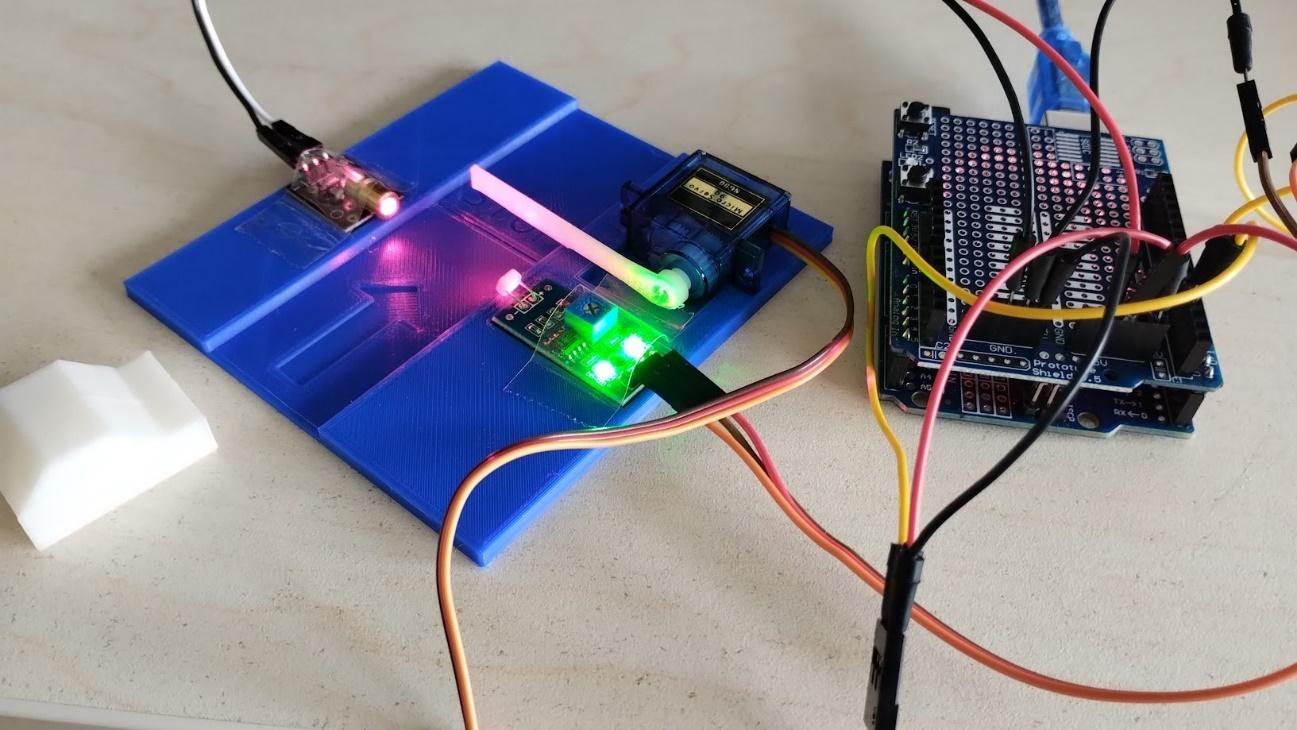
Turn on Arduino, point the LASER to the light sensor and adjust the sensibility of the light using a screwdriver on the blue potentiometer of the light sensor module. The left light should only turn on when the LASER hits on the sensor as in the image above.

Connect the Servo pins to power and data pin of the proto shield:

| Servo motor | |
| --- | --- |
| Brown | GND |
| Red | 5V |
| Yellow | Digital Pin ~3 |

Attach the barrier to the servo motor. You may need to adjust the angle of the barrier if the servo motor is not on the correct position.

Put all the elements in the support board and fix them with adhesive tape or blue tack to make sure they stay in place.



Controlling a Servo Motor with Arduino

To control a servo motor with an Arduino, we can use the Servo library, which makes the process easy and convenient. The library provides functions for attaching the servo to a pin, setting the position, and moving the servo smoothly to the desired angle.

To avoid false object detection the program will read the sensor each 250ms. The program will consider that there is a car waiting only when 4 readings in a row detect an obstacle between the LASER and the light sensor.

**#include <Servo.h>**

**Servo myServo;**

**int detections;**

**int sensorPin = 4;** //Light sensor pin

**int servoPin = 3;** //Servo Motor pin. Must be a PWM pin

**int open = 0;** //Position of servo on opened state

**int close = 128;** //Position of servo on closed state

**int threshold = 4;** //Number of row reads to be considered as detection

**int waitToPass = 4000;** //Time to wait for the car to pass

**void setup() {**

**myServo.attach(servoPin);** //Initialize the servo motor

**myServo.write(close);** //Set barrier on close position

**pinMode(sensorPin , INPUT);** //Set sensor pin as input pin

**detections = 0;** //Initialize the number of object detection

**}**

**void loop() {**

**if (digitalRead(sensorPin)==1){** //If there is something detected

**detections = min ( threshold, detections +1);** //Increment number of detections by one

//until reaching threshold

**} else {** //If no object is detected

**detections = max ( 0, detections -1);** //Decrement number of detections by one

//until reaching 0

**}**

**delay (250);** //Four reads by second

**if (detections == threshold){** //If threshold level of detection is reached

**myServo.write(open);** //Open barrier

**delay (waitToPass);** //Wait for the car to pass

**}**

**if (detections == 0){** //If no object is detected for threshold reads

**myServo.write(close);** //Close the barrier

**}**

**}**