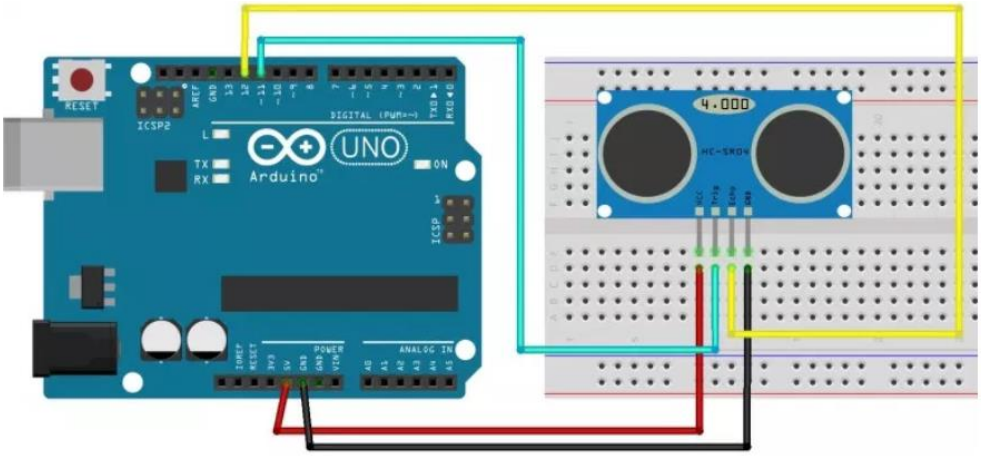
Ultrasonic sensor + stepper motor

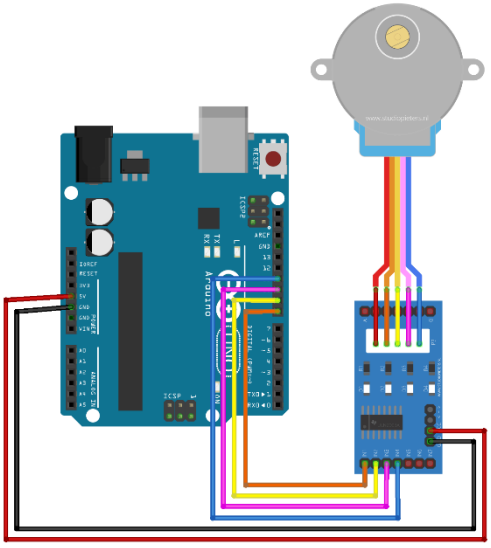
In this tutorial we are going to go over how-to setup and use a ultrasonic distance sensor on your Arduino and based on the value the distance measures the lock should close and then reopen with a button. You're going to need an Arduino with a breadboard and a couple of jumper cables and your ultrasonic distance sensor, a stepper motor and ULN2003 driver board which is necessary to connect the motor to the Arduino. First, we're going to go over the wiring of the jumper cables.

Connecting pins

First, we start with the pins of the sensor. The first pins we connect on the breadboard are the ground pin on your Arduino and ground pin on your sensor, the black cable as seen on the picture. Next, we connect the echo (Echo) and the trigger (Trig) pin with 2 free ports on the Arduino, in this case GPI/O pin 11 and 12 as indicated by the yellow and blue cable on the picture. At last we connect the 5V Arduino pin with the VCC pin on our sensor as indicated by the red cable in the picture.



The driver has 2 pins for power and 4 pins for input. Wire the negative pin to a GND pin on the Arduino and the positive pin to the 5V pin on the Arduino. The 4 other pins (IN1 IN2 IN3 IN4) on the driver can be wired to any of the GPIO pins on the Arduino (We connected IN1-4 on the driver to pin 8-11 on the Arduino)



The motor should be ready to work now, you should be able to move it by uploading some code onto the Arduino. We want to be able to control it by using the signals received from the ultrasonic sensor on the other Arduino though. This is explained later.

Connecting Arduino's

In order to make the firs Arduino talk to the second Arduino we must connect a GPI/O pin and connect the ground pins of each Arduino with each other. It is important that we put resistors between the connection of the GPI/O pins or else we break

them. We connect pin 7 on the Arduino that controls the motor to pin 13 on the Arduino that controls the sensor.

The code

Before you blatantly copy paste the code try to understand how I works, so if you want to change how the whole process behaves you have a better idea how to do this. Instead of explaining the code line by line I'm going to explain the general flow of things so you can understand it.

We first start by defining the pins as input or output pins in the setup, after this the loop begins. We have a boolean which sets the state of the lock to closed or open and based on this value the sensor behaves differently. If the lock is indeed in a locked state it constantly waits for a button to be pressed in order to unlock it. The locking mechanism is explained in the motor code. If the lock is in an open state, the sensors starts scanning. The way the sensor calculates the code is based on how long the wave has to travel back and forth in microseconds. One's the sensor detects an object closer than 10 cm it executes a loop to check if that object is indeed still in range. It does this by checking if the distance is still less than 10 every second for 5 seconds, if this is not the case it just goes back to a normal scanning mode. However, if the object is indeed closer than 10 cm for 5 seconds the lock is engaged.

Now we explain how the locking works by looking at the motor code. In here we define the pin which is connected with the other Arduino and set it to check for a high voltage. We also define a boolean of the locked state like with the other Arduino. Before the loop is started the locked boolean is set to false so the Arduino checks if it the pin is set to a high voltage if so, it starts the motor to rotate for about 90 degrees and sets the lock on true. Then it waits for the pin to be set on a low voltage by the other Arduino. This is done by pressing the button on the sensor Arduino which sets the voltage to low, then the motor is rotated back for about 90 degrees and the locked state is set to false. This process is repeated over and over again.

In order to test this out we must put each piece of code onto the correct Arduino. We do this by plugging each Arduino in separately and uploading each piece of code onto it. After we plug in each Arduino, we check the serial output of each Arduino and we should be good to go.

Sensor code

```
const int button = 0;
const int trigPin = 4;
const int echoPin = 5;
const int lockedPin = 13;

boolean locked = false;
int buttonState = 0;
long duration;
int distance;
int counter;

void setup() {
  pinMode(trigPin, OUTPUT);          // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);           // Sets the echoPin as an Input
  pinMode(lockedPin, OUTPUT);        // Sets the lockedPin as an Output
  digitalWrite(lockedPin, LOW);      // Clear the lockePin
  Serial.begin(115200);
}

void loop() {
    if (locked){
        Serial.println("Lock is locked");
        buttonState = digitalRead(button);
        if (buttonState == LOW) {
            locked = false;
            digitalWrite(lockedPin, LOW);
            Serial.print("Pin value: ");
            delay(7000);
        }
    } else {
      // Clears the trigPin
      digitalWrite(trigPin, LOW);
      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
      microseconds
      duration = pulseIn(echoPin, HIGH);

      // Calculating the distance
      distance= duration*0.034/2;
        if (distance > 184) {
            Serial.println("Out of range");
        } else {
            Serial.print("Distance: ");
            Serial.print(distance);
            Serial.println("cm");
        }

        if (distance < 10) {
            counter = 0;
            for (int i = 0; i < 5; i++) {
                digitalWrite(trigPin, LOW);
                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
        microseconds

                duration = pulseIn(echoPin, HIGH);

                // Calculating the distance
                distance= duration*0.034/2;
                if (distance < 10) {
                    ++counter;
                    Serial.println(counter);
                }
                if (distance > 184) {
                    Serial.println("Out of range");
                } else {
                    Serial.print("Distance: ");
                    Serial.print(distance);
                    Serial.println("cm");
                }
                delay(1000);
            }
            if (distance < 10 && counter == 5) {
                locked = true;
                digitalWrite(lockedPin, LOW);
                digitalWrite(lockedPin, HIGH);
                Serial.print("Pin value: ");
                Serial.println(digitalRead(lockedPin));
                delay(500);
            }
        }
    }
    delay(1000);
}
```

Motor code

```
#include <Stepper.h>
#define STEPS 2038 // the number of steps in one revolution of your motor
(28BYJ-48)
Stepper stepper(STEPS, 8, 10, 9, 11);
const int lockedPin = 7;
boolean locked = false;

void setup() {
    stepper.setSpeed(6);
    pinMode(lockedPin, INPUT);
    Serial.begin(115200);
}

void loop() {
    Serial.print("Pin value: ");
    Serial.println(digitalRead(lockedPin));
    if(!locked){
        if(digitalRead(lockedPin)==HIGH){
            stepper.step(-510);
            locked = true;
        }
    }
    if(locked){
        if(digitalRead(lockedPin)==LOW){
            stepper.step(510);
            locked=false;
        }
    }
   delay(1000);
}
```