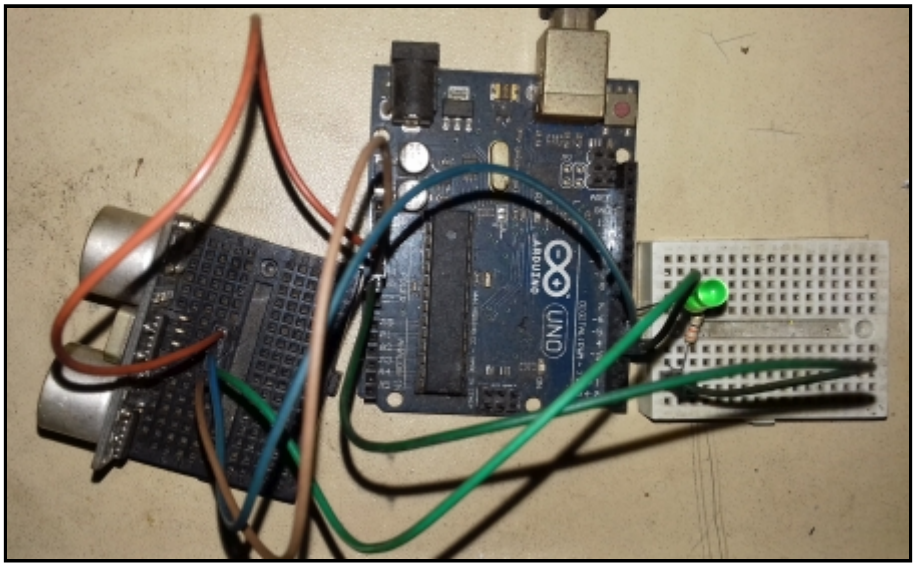
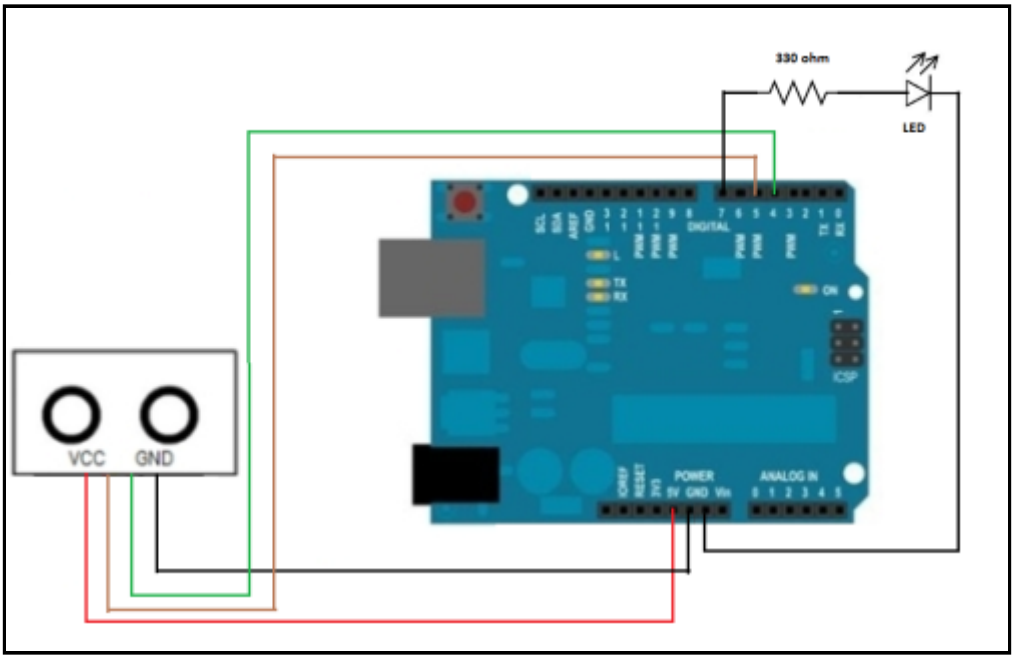
PC Based Automatic Push-Up Counter

No matter how busy we are, we have to spend time for exercise. In this small project, we will build a PC based push-up counter. With the help of Arduino and pySerial, a Python library to handle serial communication, the project can be finished quickly. No need to deal with 7-segment or other types of display. The number of components to build this kind of device can be minimized.



Here are the list of components used:
-PC (1)
-Arduino Uno (1)
-Ultrasonic Sensor (1)
-LED light (1)
-Resistor 330 ohm (1)
-Wire

Here is the circuit:



This simple project consists of two files, one for Arduino (*PushupCounter.ino*) and the other, which is written in Python, is used as user interface (*PushupCounter*).

PushupCounter.ino:

```
const int echo = 4;
const int trigger = 5;
const int LED = 7;

double duration;
double distance;
double maxdistance;

String prevposition;
char inChar;

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

  maxdistance = 50;
  prevposition = 'up';

  pinMode(trigger,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED,OUTPUT);

  digitalWrite(trigger,LOW);
  digitalWrite(echo,LOW);
  digitalWrite(LED,LOW);
}

void loop() {
```

```
  serialEvent();

  digitalWrite(trigger,HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigger,LOW);

  duration=pulseIn(echo,HIGH);
  distance=(duration/2)/29.1;

  if((distance >= maxdistance)&&(prevposition == 'down')){
    Serial.println(1);
    Serial.write(255);    //end of data transfer
    prevposition = 'up';
  }

  if((distance < maxdistance)&&(prevposition == 'up')){
    prevposition = 'down';
  }

  delay(100);
}

void serialEvent() {
  if (Serial.available()) {
    inChar = (char)Serial.read();
    if (inChar == '1'){
      digitalWrite(LED, HIGH);
    }
    else{
      digitalWrite(LED, LOW);
    }
  }
}
```

Explanation

```
String prevposition;
```

As you know, push-up consists of up and down movement. The counting process will be conducted by the program everytime the user has completed one cycle (one up and one down). The variable *prevposition* holds the previous user's position.

```
double duration;
double distance;
double maxdistance;
```

The ultrasonic sensor works by sending and receiving inaudible sound waves. To determine the distance to an object, it calculates the time interval between sending the signal and receiving the echo. The variable *duration* is used to store that time interval. The value is then used to calculate distance.

```
digitalWrite(trigger,HIGH);
delayMicroseconds(1000);
digitalWrite(trigger,LOW);

duration=pulseIn(echo,HIGH);
distance=(duration/2)/29.1;
```

Initially, the user is in the 'up' position. Signal from the sensor is unblocked. The variable *maxdistance* holds the predetermined value of the distance between the sensor and a certain object in which the signal can be considered unblocked. When the user is in the 'down' position, signal from the sensor is blocked. The distance from the sensor to the user is calculated. The value is stored in the variable *distance*. The two variables, *distance* and *maxdistance* will be compared to determine whether the counting process needs to be performed.

```
if((distance >= maxdistance)%26;%26;(prevposition == 'down')){
    Serial.println(1);
    Serial.write(255);    //end of data transfer
```

```
    prevposition = 'up';
  }

  if((distance < maxdistance)%26;%26;(prevposition == 'up')){
    prevposition = 'down';
  }
```

If *distance* is equal or greater than *maxdistance,* and the user's previous position is 'down', the program sends '1' to the python script and also resets the variable *prevposition*. The python script adds the counter by an integer 1. Put it simply, the function of the group of code is to avoid unwanted calculation. If the user holds his body in the 'down' position, the counting process will not be performed.

```
if((distance < maxdistance)%26;%26;(prevposition == 'up')){
    prevposition = 'down';
  }
```

The above line of code runs everytime the user moves from 'up' to 'down' position.

```
void serialEvent() {
  if (Serial.available()) {
    inChar = (char)Serial.read();
    if (inChar == '1'){
      digitalWrite(LED, HIGH);
    }
    else{
      digitalWrite(LED, LOW);
    }
  }
}
```

The above is just an extra. It alerts the user when he has completed his exercise. The 'end' signal is sent by the python script. Feel free to change the LED to a speaker or other kinds of output devices.

PushupCounter.py

```
from time import sleep
import serial


counter = 0

console = serial.Serial('COM4',9600)

console.write(b'0')                              #0 = LED OFF

if console.isOpen():
    print('STARTING')
    while counter<10:
        data = console.readline()
        if data == b'ÿ1 ':
            counter+=1
            print(counter)
        sleep(1)
    console.write(b'1')                 #1 = LED ON
console.close()
```

Explanation

```
counter = 0
```

In the variable *counter*, you determine how many 'up' and 'down' movements (cycles) need to exercised by the user.

```
console = serial.Serial('COM4',9600)
```

The above can be considered as the core of the program. Communication with serial port is opened. The

COMM port used to connect your PC to Arduino may be different than mine.

```
console.write(b'0')
```

Initially, the LED alert light is turned off by sending '0' to the Arduino.

```
if console.isOpen():
    print('STARTING')
    while counter<10:
        data = console.readline()
        if data: == b'ÿ1 '
            counter+=1
            print(counter)
        sleep(1)
    console.write(b'1')                #1 = LED ON
console.close()
```

If communication can be performed, the data from Arduino can be processed. The variable *counter* is added by 1 everytime 1 push-up cycle has been conducted.