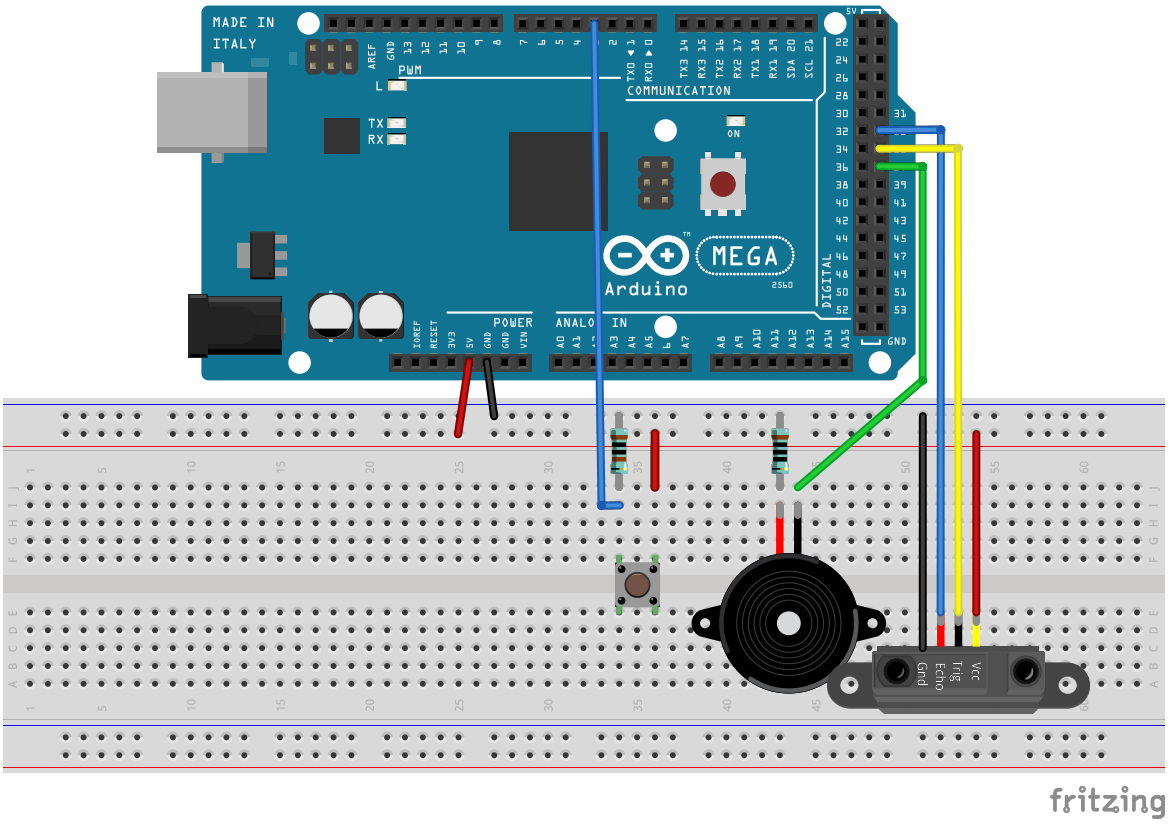
Deadline	**12th of January 12 p.m.**
	Send as email to your tutor
Mail Subject	$[PFE15][Assignment5]\$matriculationnumber$
Accepted Formats	PDF, txt
File Name Convention	$PFE15\_Assignment5\_\$matriculationnumber.\$format$
Remark	You are supposed to include all code examples into your document

0 Prerequisites [0 Points]

On this exercise sheet, you should use an ultrasonic distance meter to measure distances and turn them into sounds using the piezo speaker.

Assemble the circuit as depicted in the figure below. It will be used in the exercises 2 and 3.



Use the following code to measure the signal propagation delay of the ultrasonic sensor:

```
int sonarTimeout = 1000;

/**
 * Sends an ultrasonic ping and expects the reply.
 * Return value: Delay between a ping and the corresponding reply in
 *               micro seconds.
 */
unsigned long triggerAndFetchSonar() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  return pulseIn(echoPin, HIGH, sonarTimeout);
}
```

Pin `trigPin = 33` should be configured as `OUTPUT` and `echoPin = 35` as `INPUT`.

1 Sorting [15 Points]

As on exercise sheet 3, you are asked to implement a tool that performs a measurement using the ultrasonic sensor of up to 30 seconds. The measure points should be collected in a 100ms interval[1] by the ultrasonic sensor. As on exercise sheet 3, the measurement should be started or stopped by a press of the pushbutton.

After the measurement is completed, the *median* should be calculated and printed using the serial port. You can use the algorithms presented in the lecture to sort the array and choose the middle element (round down if necessary). Possible search algorithms include insertion sort and merge sort.

2 Runtime Analysis [15 + 5* Points]

We want to compare different sorting algorithms against each other. Extend the provided sketch as described below:

1. At first, populate the array `test_data` with randomly generated values between 0 and (including) 10000. You can accomplish this using the function `random()`[2].
2. The test code is already provided. Extend the test execution loop by printing the test results (`time_is` and `time_ms`) that the results are printed on the serial console as shown below:

```
Number of elements: 600
Speed:
InsertionSort: 432 ms
MergeSort: 234 ms
```

3. Extend the functions `test_insertion_sort()` and `test_merge_sort()` with two `millis()`-calls to measure and return the time in milliseconds spent sorting.

[1] Hint: Think of lost replies! This could require an adjustment of *sonarTimeout*.
[2] `http://www.arduino.cc/en/Reference/Random`

4. Include the definitions of `insertion_sort()` and `merge_sort()`, as shown in the lecture slides, in your code.
5. **Bonus exercise:** Consider the following corner cases. Which algorithm is faster?
   (a) The array to be sorted *is already sorted.*
   (b) The array to be sorted is already sorted *expect for one element.*

   Use variations of `generate_test_data()` to solve these questions and explain your observations. Are these corner cases common in practice?

```
int test_data[1000];
int buf[1000];

void generate_test_data(int n) {
  // Fill the array test_data with randomly generated data.
}

long test_insertion_sort(int n) {
  // Measure time here
  insertion_sort(test_data, n);

  return 0; // return the time in ms
}

long test_merge_sort(int n) {
  // Measure time here
  merge_sort(test_data, buf, n);

  return 0; //return the time in ms
}

void setup() {
  Serial.begin(9600);
  Serial.println("Runtime analysis:");

  for (int i = 100; i < 1000; i += 100) {
    generate_test_data(i);
    long time_is = test_insertion_sort(i);

    generate_test_data(i);
    long time_ms = test_merge_sort(i);

    // Print out results

    Serial.prinln("");
  }
}

void loop() {
  // Nothing to do here. Party!
}

// Insert the definitions of insertion and merge sort as shown in the lecture
   here.
```

3 Theremin [20 + 10* Points]

Implement a Theremin[3]. Here, the distance measured by the ultrasonic sensor should be used to generate different sounds. To generate a nice sound, please consider the following hints:

1. In case the volume of the piezo speaker is too high, the signal generated by the ultrasonic sensor can be disturbed. Use an appropriate resistor to limit the volume.
2. The signal propagation delay should be limited to the interval $[150, 1000]$ μs.
3. This interval should be linearly mapped to a frequency range of $[1, 4000]$ Hz. For this purpose, the Arduino library provides a function called `map()`.
4. Use this previously generated value to compute a *floating average*. Let y be the mapped frequency. The floating average for x_{t+1} is computed from the previous average x_t as follows:

$$x_{t+1} = \frac{4 \cdot x_t + y}{5}$$

5. Beware of rounding errors in your floating average implementation.

Bonus exercise: Create a music video using your Theremin. Upload it on youtube and send the web address to your tutor. The best video will be awarded.

[3] `http://en.wikipedia.org/wiki/Theremin`