Abgabe

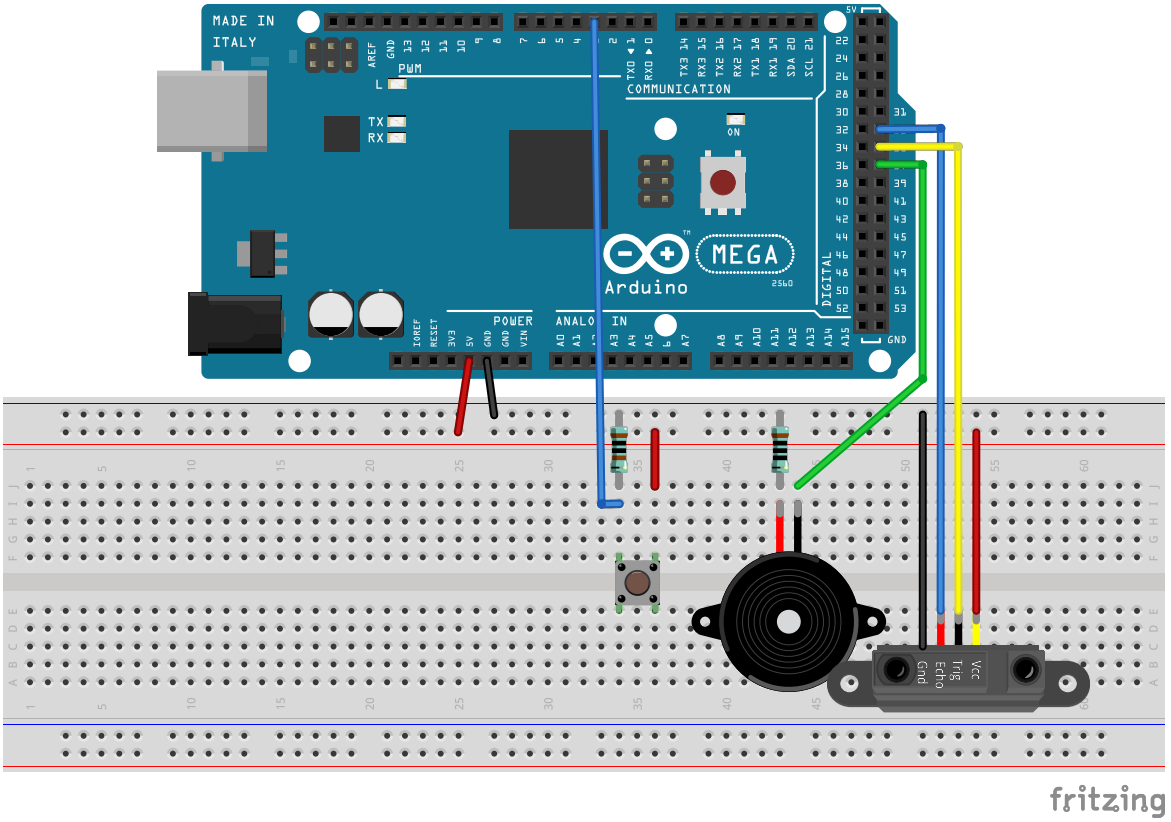
Geben Sie die Lösung bis **Dienstag, 9. Juni um 14:00** als PDF-Dokument per E-Mail an Ihren Tutor ab. Die Abgabeformalitäten sind die Gleichen wie auf Blatt 1. Bitte achten Sie darauf! Wir werden Abgaben mit inkorrekten Dateiformat o.Ä. nicht bewerten.

Bitte geben Sie auch zusätzlich zum PDF die Ino-Dateien zu den Aufgabenteilen ab! Dies erleichtert erheblich die Korrektur. Benennen Sie dazu die Dateien folgendermaßen: `Ping-MATRIKELNUMMER-Loesung-BLATTNUMMER-Aufgabe-AUFGABE.ino`. Also z.B.: `Ping-1234567-Loesung-5-Aufgabe-1.ino`.

1 Schaltung

In dieser Aufgabe sollen Sie den Ultraschall-Entfernungssensor einsetzen, um Entfernungen zu messen und in Töne auf dem Piezo-Lautsprecher zu wandeln.

Bauen Sie die Schaltung wie in der Abbildung dargestellt auf. Die Schaltung wird für Aufgabe 2 und 3 benötigt. Sie müssen für diese Aufgabe jedoch keinen Code abgeben.



Verwenden Sie folgenden Code, um die Signallaufzeit des Ultraschallsensors zu messen:

```
int sonarTimeout = 1000;

/**
 * Sendet einen Ultraschall-Puls aus und wartet auf das Echo.
 * Rueckgabewert: Verzoegerung des Echos in Mikrosekunden.
 */
unsigned long triggerAndFetchSonar() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  return pulseIn(echoPin, HIGH, sonarTimeout);
}
```

Dabei ist `trigPin = 33` als `OUTPUT` und `echoPin = 35` als `INPUT` konfiguriert.

2 Sortieren

Implementieren Sie ein Programm, das wie auf Blatt 3 eine Messung von bis zu 30 Sekunden durchführt. Die Messpunkte sollen im Takt von 100ms vom Ultraschall-Sensor erfasst werden. Achten Sie darauf, dass Sie gegebenenfalls `sonarTimeout` anpassen müssen. Wie auf Blatt 3 wird die Messung durch einen Tastendruck gestartet und beendet.

Am Ende der Messung soll diesmal der *Median* ermittelt und per serieller Schnittstelle ausgegeben werden. Sortieren Sie dazu das Feld mit einem der in der Vorlesung besprochenen Algorithmen und wählen Sie dann das mittlere Element aus (gegebenenfalls abrunden). Sie können entweder Sortieren per Einfügen oder Sortieren durch Mischen verwenden.

3 Laufzeitanalyse

Wir wollen feststellen, wie schnell verschiedene Sortieralgorithmen sind. Erweitern Sie den Sketch folgendermaßen:

1. Befüllen Sie zunächst das Array `test_data` mit zufällig erzeugten Zahlen zwischen 0 und (einschließlich) 10000. Benutzen Sie dazu die Funktion `random()`[1].

2. Der Code, um die Tests auszuführen, ist bereits vorgegeben. Erweitern Sie die Schleife so, dass nach Aufrufen der Tests die Ergebnisse (`time_is` und `time_ms`) wie folgt auf der seriellen Schnittstelle ausgegeben werden:

   ```
   Anzahl Elemente: 600
   Geschwindigkeit:
   InsertionSort: 432 ms
   MergeSort: 234 ms
   ```

3. Erweitern Sie `test_insertion_sort()` und `test_merge_sort()` mit Hilfe von zwei `millis()`-Aufrufen so, dass die Zeit des Sortierens in Millisekunden gemessen und zurückgegeben wird.

[1] `http://www.arduino.cc/en/Reference/Random`

4. Fügen Sie zuletzt noch die Definition von `insertion_sort()` und `merge_sort()` in den Code ein.
5. **Bonusaufgabe:** Betrachten Sie die folgenden Sonderfälle. Welcher Algorithmus ist schneller?
   (a) Das zu sortierende Feld *ist bereits sortiert.*
   (b) Das zu sortierende Feld ist *bis auf ein Element* bereits sortiert.

   Nutzen Sie Variationen von `generate_test_data()`, um diese Fragen zu lösen, und begründen Sie Ihre Beobachtungen. Treten die obigen Sonderfälle in der Praxis auf?

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

void generate_test_data(int n) {
  // Fuellen Sie das Array test_data mit n zufaellig generierten Daten
}

long test_insertion_sort(int n) {
  // Hier muss noch Zeit gemessen werden
  insertion_sort(test_data, n);

  return 0; // Zeit in ms zurueck geben
}

long test_merge_sort(int n) {
  // Hier muss noch Zeit gemessen werden
  merge_sort(test_data, buf, n);

  return 0; // Zeit in ms zurueck geben
}

void setup() {
  Serial.begin(9600);
  Serial.println("Laufzeitanalyse");

  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);

    // Geben Sie an dieser Stelle die Ergebnisse aus

    Serial.prinln("");
  }
}

void loop() {
  // Hier geschieht nichts
}

// Hier die Definitionen von InsertionSort und MergeSort aus der Vorlesung
    einfuegen.
```

4 Theremin

Programmieren Sie einen Theremin[2]. Dabei soll der gemessene Abstand vom Ultraschallsensor verwendet werden, um Töne zu erzeugen. Um einen schönen Klang zu erzeugen, beachten Sie bitte folgende Punkte:

1. Ist der Piezo-Lautsprecher zu laut, kann er das Signal des Ultraschallsensors übertönen. Wählen Sie einen geeigneten Widerstand, um die Lautstärke zu begrenzen.
2. Die gemessene Signallaufzeit sollte auf das Interval $[150, 1000]$ μs begrenzt werden.
3. Dieses Interval wird dann auf den Frequenzbereich $[1, 4000]$ Hz linear abgebildet. Hierfür stellt die Arduino-Bibliothek eine Funktion namens `map()` zur Verfügung.
4. Verwenden Sie diesen Wert, um einen *gleitenden Durchschnitt* zu bilden. Sei y die abgebildete Frequenz; dann berechnet sich der gleitende Durchschnitt x_{t+1} aus dem letzten Durchschnitt x_t wie folgt:
$$x_{t+1} = \frac{4 \cdot x_t + y}{5}$$
5. Achten Sie beim Implementieren des gleitenden Durchschnitts auf Rundungsfehler!

Bonusaufgabe: Erstellen Sie ein Musikvideo mit Hilfe Ihres Theremins. Laden Sie es auf YouTube und senden Sie die Webadresse an Ihren Tutor. Das beste Video wird prämiert.

[2] `http://de.wikipedia.org/wiki/Theremin`