Abgabe

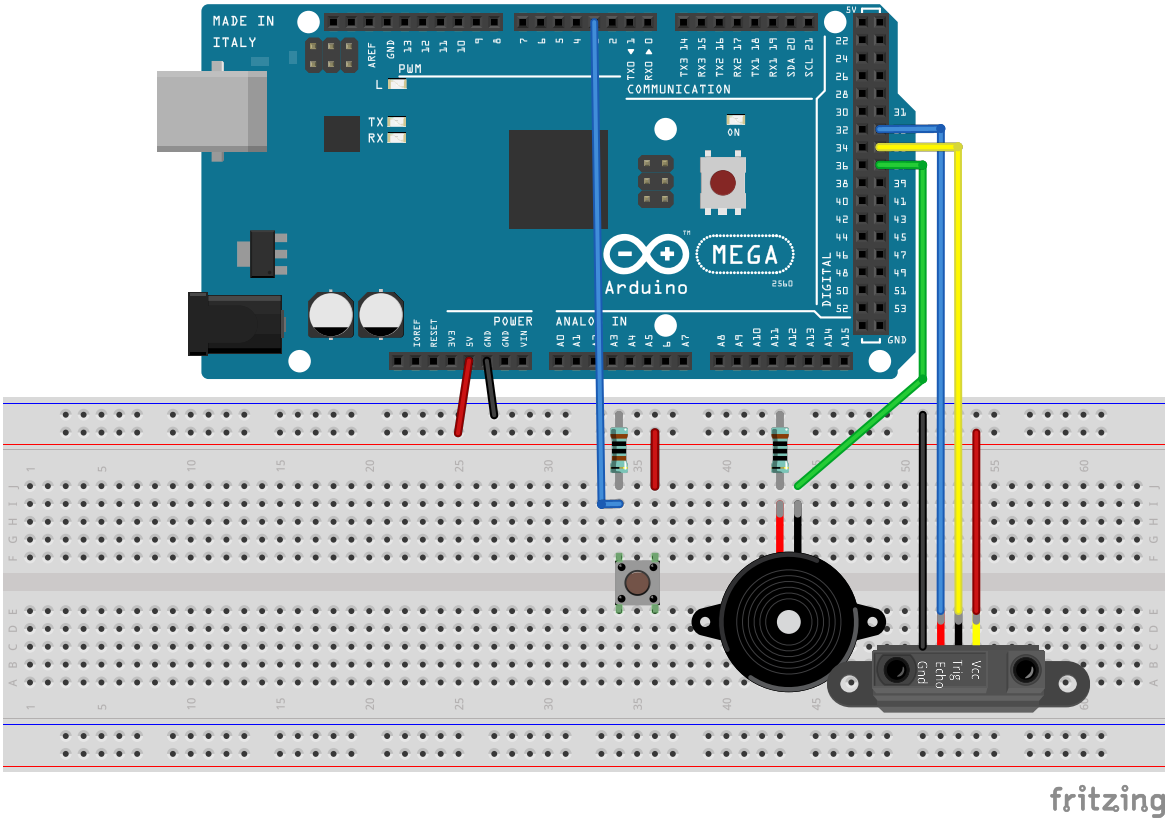
Geben Sie die Lösung bis **Montag, 19. Juni** um 23:59 als PDF-Dokument in einem ZIP, inklusive aller Quelldateien, per E-Mail an Ihren Tutor ab. Die Abgabeformalitäten sind die Gleichen wie auf Blatt 1.

1 Ultraschallsensor (7 Punkte + 2 Bonus)

1.1 Schaltung

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

Ihr Ultraschallsensor sieht anders aus, als der hier abgebildete. Es handelt sich dabei um das Bauteil, mit den beiden runden mit Netz bedeckten Zylindern.



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 = 35` als `OUTPUT` und `echoPin = 33` als `INPUT` konfiguriert.

1.2 Sortieren (3 Punkte)

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*[1] 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 Element an der mittleren Position aus (gegebenenfalls abrunden). Sie können entweder Sortieren per Einfügen oder Sortieren durch Mischen verwenden.

1.3 Bonus: Durchschnittsvergleich (2 Bonus)

Bilden Sie nun den normalen Durchschnitt des Arrays und vergleichen Sie diesen Wert mit dem vorher bestimmten Median. Sind die Werte gleich oder unterschiedlich? Erklären Sie das Verhalten kurz.

1.4 Theremin (4 Punkte)

Programmieren Sie ein 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()`[3] 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!

[1] `https://de.wikipedia.org/wiki/Median`
[2] `http://de.wikipedia.org/wiki/Theremin`
[3] `https://www.arduino.cc/en/Reference/Map`

2 Laufzeitanalyse (5 Punkte)

Wir wollen feststellen, wie schnell verschiedene Sortieralgorithmen sind. Erweitern Sie das 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()`[4].

2. Erweitern Sie den Sketch auf der nächsten Seite 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.

4. Fügen Sie zuletzt noch die Definition von `insertion_sort()` und `merge_sort()` in den Code ein.

5. 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?

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

```
void generate_test_data(int test_array[], int size) {
  // Fuellen Sie das Array test_data mit zufaellig generierten Daten
}

unsigned long test_insertion_sort(int test_array[], int size) {
  insertion_sort(test_array, n);

  return ???; // Laufzeit in ms zurueck geben
}

unsigned long test_merge_sort(int test_array[], int size) {
  int buf[size];
  merge_sort(test_array, buf, n);

  return ???; // Laufzeit in ms zurueck geben
}

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

  for (int i = 100; i < 1000; i += 100) {
    int test_data_1[i]; //Array fuer Zufallszahlen
    generate_test_data(test_data, i);
    unsigned long time_is = test_insertion_sort(test_data, i);

    int test_data_2[i]; //Array fuer Zufallszahlen
    generate_test_data(test_data, i);
    unsigned long time_ms = test_merge_sort(test_data, i);

    // Geben Sie an dieser Stelle die Ergebnisse aus
    Serial.prinln("");
  }
}

void loop() {
  // Hier geschieht nichts, Programm laueft nur in setup()
}
```