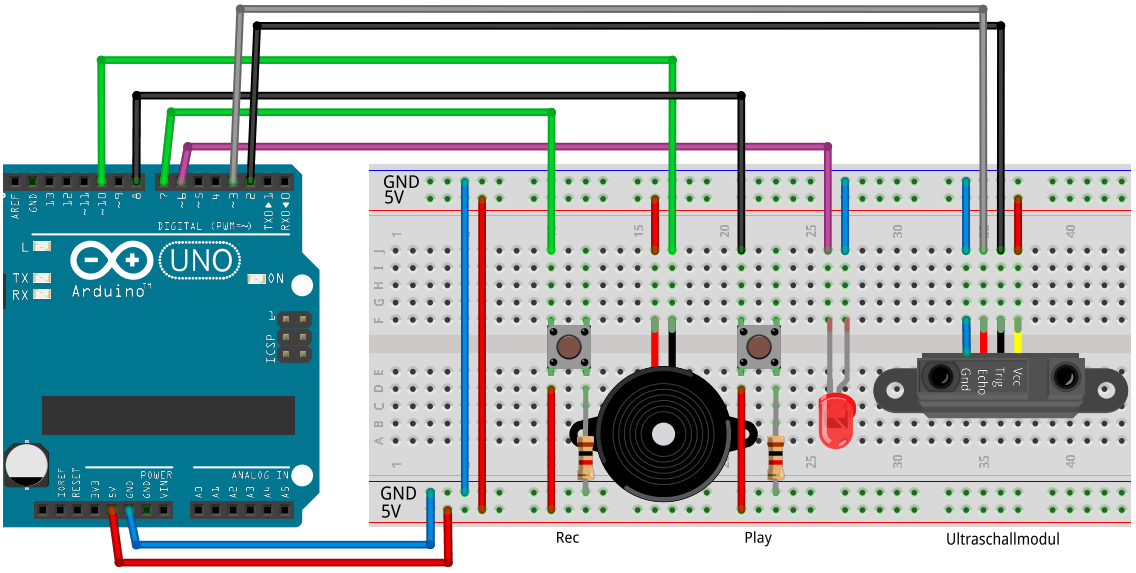
Abgabe

Dieses Übungsblatt ist bis Freitag, 13.06. um 12:00 Uhr per Email an den eigenen Tutoren abzugeben. Benennen Sie die Abgabe bitte eindeutig: "*Matrikelnummer*_Abgabe_*Blattnummer.Format*".

1 Theremin

Das folgende Bild zeigt den Aufbau für ein Theremin mit Piezolautsprecher, Ultraschallmodul, Status-LED, sowie Record- und Play-Taster. Die schnellen Pins 2 und 3 werden für das Ultraschallmodul genutzt.



Realisieren Sie folgende Software für die angegebene Schaltung:

a. Implementieren Sie ein Theremin. Je nach per Ultraschall gemessenem Abstand ändert sich die Frequenz des ausgegebenen Tons.

b. **Bonusaufgabe:** Interpolieren Sie stufenlos zwischen den ausgegebenen Tonhöhen.

c. **Bonusaufgabe:** Wenn der Record-Taster gedrückt wird, sollen alle gespielten Töne abgespeichert werden. Die Aufzeichnung kann durch erneutes Drücken des Record-Tasters oder Wechseln in den Play-Modus gestoppt werden. Solange die Aufzeichnung läuft, soll die Status-LED alle 300 ms blinken.

   Ein Druck auf den Play-Taster aktiviert den Play-Modus. Die aufgezeichnete Tonsequenz wird kontinuierlich zyklisch abgespielt. Solange der Play-Modus aktiviert ist, soll die Status-LED kontinuierlich leuchten. Ein erneuter Druck auf den Play-Taster oder das Wechseln in den Record-Modus stoppt das Abspielen.

d. **Bonusaufgabe:** Hinterlegen Sie eine interessante Melodie (möglicherweise algorithmisch erzeugt), die abgespielt wird, wenn auf Play gedrückt, aber bisher noch nichts aufgezeichnet wurde.

e. **Sonderpreis**: Für die schönste Video-Demonstration eines selbstgebauten Theremins in Aktion (mit Live-Musik) vergeben wir einen Sonderpreis. Die Videos sollten dabei maximal 60 Sekunden lang sein und auf YouTube hochgeladen werden. Der Link auf das YouTube-Video ist mit Betreff "Theremin-Video" bis zum **12.06. 12:00** an ping@lists.st.cs.uni-saarland.de zu übersenden.

Beispiellösung.

```
#include <Tone.h>
#include <Wire.h>

// Pin fuer den Lautsprecher
int speakerPin = 10;

// Pins zum Steuern des Abstandssensors
int trigPin = 2;
int echoPin = 3;

// Pins fuer die Buttons zum Aufnehmen und Abspielen
int wiedergabePin = 8;
int aufnahmePin = 7;

// Pin fuer die blinkende LED waehrend der Aufnahme
int ledPin = 6;

// speichert, ob wir eine Aufnahme am Abspielen sind
boolean spieleAufnahme = false;

// speichert, ob wir eine Aufnahme starten muessen
boolean starteAufnahme = false;

// Aufnahme bezogene Variablen
const int ANZAHL_TOENE = 2000;
int tonhoehe[ANZAHL_TOENE];
int anzahlGespeicherterToene;

// Anzahl der Schritte beim Interpolieren
const int INTERPOLIEREN_SCHRITTE = 5;

void setup() {
  // deklariere die Input und Output Pons
  pinMode(speakerPin,  OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT_FAST);
  pinMode(aufnahmePin, INPUT);
  pinMode(wiedergabePin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void sendTrigger() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
}
```

```
long fetchPulse() {
  unsigned long warten = micros();
  while (digitalRead(echoPin) == LOW) {
    if (micros() - warten > 30000) // 30 ms
      return -1;
  }
  unsigned long start = micros();
  while (digitalRead(echoPin) == HIGH) {
    if (micros() - start > 30000) // 30 ms
      return -1;
  }
  unsigned long ende = micros();
  return ende - start;
}

boolean tasterabfrage_play() {
  // kontrolliere ob die Wiedergabetaste gedrueckt ist
  // wenn ja, warte bis die Taste losgelassen wurde
  if (digitalRead(wiedergabePin) == HIGH) {
    while (digitalRead(wiedergabePin)) {}
    return true;
  }
  return false;
}

boolean tasterabfrage_record() {
  // kontrolliere ob die Aufnahmetaste gedrueckt ist
  // wenn ja, warte bis die Taste losgelassen wurde
  if (digitalRead(aufnahmePin) == HIGH) {
    while (digitalRead(aufnahmePin)) {}
    return true;
  }
  return false;
}

int generiereUndSpieleTon() {
  // ermittle die Tonhoehe
  sendTrigger();
  int duration = fetchPulse();
  int frequenz = map(duration,0,20000,100,2000);

  // spiele den Ton ab
  spieleTon(frequenz);

  // gibt die berechnete Frequenz zurueck
  return frequenz;
}

void spieleTon(long wert) {
  if (wert >= 0) {
    tone(speakerPin, wert);
  } else {
    noTone(speakerPin);
  }
}
```

```
void spieleTonInterpoliert(long wert1, long wert2) {
  // wenn beide Toene nicht existieren, spiele keinen Ton ab
  if (wert1 < 0 && wert2 < 0) {
    noTone(speakerPin);
  } else {
    // wenn ein Ton nicht existiert, normiere diesen auf 0
    if (wert1 < 0) { wert1 = 0; }
    if (wert2 < 0) { wert2 = 0; }

    // spiele INTERPOLIEREN_SCHRITTE viele Toene, die zwischen beiden Stufen
       liegen
    long differenz = (wert2 - wert1);
    long schritt = differenz/INTERPOLIEREN_SCHRITTE;
    for (int i=1; i <= INTERPOLIEREN_SCHRITTE; i++) {
      spieleTon(wert1*schritt);
    }
  }
}

void loop() {
  // pruefe ob eine Aufnahme gestartet werden soll
  if (tasterabfrage_record()) { starteAufnahme = true; }

  // Toene aufnehmen
  if (starteAufnahme) {
    int i = 0;
    while (true) {
      // generiere den Ton
      int frequenz = generiereUndSpieleTon();

      // speichere den Ton
      tonhoehe[i] = frequenz;
      if (i >= ANZAHL_TOENE - 1) {
        anzahlGespeicherterToene = i;
        starteAufnahme = false;
        break;
      } else { i++; }

      // pruefe, ob die Aufnahme abgebrochen werden soll
      if (tasterabfrage_record()) {
        anzahlGespeicherterToene = i;
        starteAufnahme = false;
        break;
      }

      if (tasterabfrage_play()) {
        anzahlGespeicherterToene = i;
        starteAufnahme = false;
        spieleAufnahme = true;
        break;
      }
    }
  }
```

```
  // Toene abspielen
  else if (spieleAufnahme) {
    int i = 0;

    while (true) {
      // spiele den Ton ab
      // ohne Bonus: spieleTon(tonhoehe[i]);
      spieleTonInterpoliert(tonhoehe[i-1], tonhoehe[i]);

      // bestimme den naechsten zu spielenden Ton
      if (i < ANZAHL_TOENE - 1) { i++; }
      else { i = 0; }

      if (tasterabfrage_record()) {
        starteAufnahme = true;
        spieleAufnahme = false;
        break;
      }

      if (tasterabfrage_play()) {
        spieleAufnahme = false;
        break;
      }
    }
  }
  // ansonsten gib einfach die aktuellen Toene aus
  else {
    generiereUndSpieleTon();
  }
}
```

2 Algorithmen

Realisieren Sie die folgenden Algorithmen. Achten Sie dabei auf die korrekte Behandlung von Fehlern und Randfällen.

a. Entwickeln Sie einen Algorithmus `int arrayMax(int a[], int size)`, der das größte Element aus dem übergebenen Zahlen-Array `a` der Größe `size` zurückgibt. Zum Beispiel:

```
int a[] = {3, 1, 2, 5, 8, 4};
Serial.println(arrayMax(a, 6)); // => 8
Serial.println(arrayMax(a, 4)); // => 5
Serial.println(arrayMax(a, 2)); // => 3
```

Beispiellösung.

```
int arrayMax(int a[], int size) {
  // kontrolliere, ob das Array mindestens size viele Elemente hat
  // wenn nicht, setze size auf diese Anzahl Elemente
  int arraysize = sizeof(a)/sizeof(int);
  if (arraysize < size) { size = arraysize; }

  // ermittle das groesste Element im Array
  int maxValue = a[0];
  for (int i = 1; i < size; i++) {
    if (maxValue < a[i]) {
      maxValue = a[i];
    }
  }
  return maxValue;
}
```

b. Entwickeln Sie einen Algorithmus `int elementCount(int a[], int size, int elem)`, der zurückgibt, wie oft das Element `elem` im Zahlen-Array `a` der Größe `size` vorkommt. Zum Beispiel:

```
int a[] = {3, 1, 3, 1, 3, 8, 8, 4};
Serial.println(elementCount(a, 8, 0)); // => 0
Serial.println(elementCount(a, 8, 1)); // => 2
Serial.println(elementCount(a, 8, 3)); // => 3
```

Beispiellösung.

```
int elementCount(int a[], int size, int elem) {
  // kontrolliere, ob das Array mindestens size viele Elemente hat
  // wenn nicht, setze size auf diese Anzahl Elemente
  int arraysize = sizeof(a)/sizeof(int);
  if (arraysize < size) { size = arraysize; }

  // zaehle die Element in dem Array
  int count = 0;
  for (int i = 0; i < size; i++) {
    if (elem == a[i]) { count++; }
  }
  return count;
}
```

c. Entwickeln Sie eine Funktion `void printAry(int a[], int size)`, die alle Elemente des Zahlen-Arrays `a` der Größe `size` hübsch formatiert über `Serial.print()` ausgibt und danach eine neue Zeile per `Serial.println()` beginnt. Zum Beispiel:

```
int a[] = {3, 1, 3, 1, 3, 8, 8, 4};
printAry(a, 0); // => []
printAry(a, 8); // => [3, 1, 3, 1, 3, 8, 8, 4]
```

Beispiellösung.

```
void printAry(int a[], int size) {
  // kontrolliere, ob das Array mindestens size viele Elemente hat
  // wenn nicht, setze size auf diese Anzahl Elemente
  int arraysize = sizeof(a)/sizeof(int);
  if (arraysize < size) { size = arraysize;  }

  // gib das Array huebsch aufbereitet aus
  Serial.print("[");
  for (int i = 0; i < size; i++) {
    if (i != 0) { Serial.print(",␣"); }
    Serial.print(a[i]);
  }
  Serial.println("]");
}
```

d. Entwickeln Sie einen Algorithmus `void elementReplace(int a[], int size, int replaceElem, int replaceWith)`, der alle Vorkommen des Elements `replaceElem` im Zahlen-Array `a` der Größe `size` durch die Zahl `replaceWith` ersetzt.

```
int a[] = {3, 1, 3, 1, 3};
elementReplace(a, 5,  3, 30); printAry(a, 5); // => [30, 1, 30, 1, 30]
elementReplace(a, 4, 30, 35); printAry(a, 5); // => [35, 1, 35, 1, 30]
elementReplace(a, 5,  1,  0); printAry(a, 5); // => [35, 0, 35, 0, 30]
```

Beispiellösung.

```
void elementReplace(int a[], int size, int replaceElem, int replaceWith) {
  // kontrolliere, ob das Array mindestens size viele Elemente hat
  // wenn nicht, setze size auf diese Anzahl Elemente
  int arraysize = sizeof(a)/sizeof(int);
  if (arraysize < size) { size = arraysize; }

  // ersetze Elemente in in dem Array
  for (int i = 0; i < size; i++) {
    if (a[i] == replaceElem) { a[i] = replaceWith; }
  }
}
```

e. **Bonusaufgabe:** Können Sie einen Algorithmus `int distinctCount(int a[], int size)` entwickeln, der angibt wie viele verschiedene Zahlen im übergebenen Zahlen-Array `a` der Größe `size` existieren. Hilft es Ihnen, wenn Sie das Array vorher sortieren? Können Sie einen Algorithmus implementieren, der das Ziel erreicht ohne das Array zu sortieren?

```
int a[] = {3, 1, 3, 1, 3, 8, 8, 4};
Serial.println(distinctCount(a, 3)); // => 2
Serial.println(distinctCount(a, 4)); // => 2
Serial.println(distinctCount(a, 7)); // => 3
Serial.println(distinctCount(a, 8)); // => 4
```

Beispiellösung.

In einem sortierten Array muessen wir nur noch die Anzahl der Wertänderungen zählen:

```
int distinctCount(int a[], int size) {
  // kontrolliere, ob das Array mindestens size viele Elemente hat
  // wenn nicht, setze size auf diese Anzahl Elemente
  int arraysize = sizeof(a)/sizeof(int);
  if (arraysize < size) { size = arraysize; }

  // zaehle die Anzahl verschiedener Elemente
  int counter = 1;
  for (int i=1; i<size; i++) {
    if (a[i-1] != a[i]) { counter++; }
  }
  return counter;
}
```

Wenn das Array unsortiert ist, müssen wir mithilfe einer zweiten Datenstruktur die verschiedenen Elemente mitzählen. Eine mögliche Implementierung kann wie folgt aussehen:

```
int distinctCount(int a[], int size) {
  // kontrolliere, ob das Array mindestens size viele Elemente hat
  // wenn nicht, setze size auf diese Anzahl Elemente
  int arraysize = sizeof(a)/sizeof(int);
  if (arraysize < size) { size = arraysize; }

  // speichert die Anzahl der verschiedenen Elemente
  int counter = 0;

  // speichert die verschiedenen gefundenen Zahlen.
  // Da es maximal size-viele Elemente geben kann,
  // benoetigen wir entsprechend viele Arrayslots.
  int gefundeneWerte[size];

  for (int i = 0; i < size; i++) {
    // kontrolliere, ob die aktuelle Zahl bislang schon gefunden wurde
    boolean gleichenWertGefunden = false;
    for (int j = 0; !gleichenWertGefunden && j < counter; j++) {
      if (gefundeneWerte[j] == a[i]) {
        gleichenWertGefunden = true;
      }
    }

    // wenn die Zahl noch nicht gefunden wurde,
    // trage sie ein und aendere den Counter
    if (!gleichenWertGefunden) {
      gefundeneWerte[counter] = a[i];
      counter++;
    }
  }

  return counter;
}
```