Abgabe

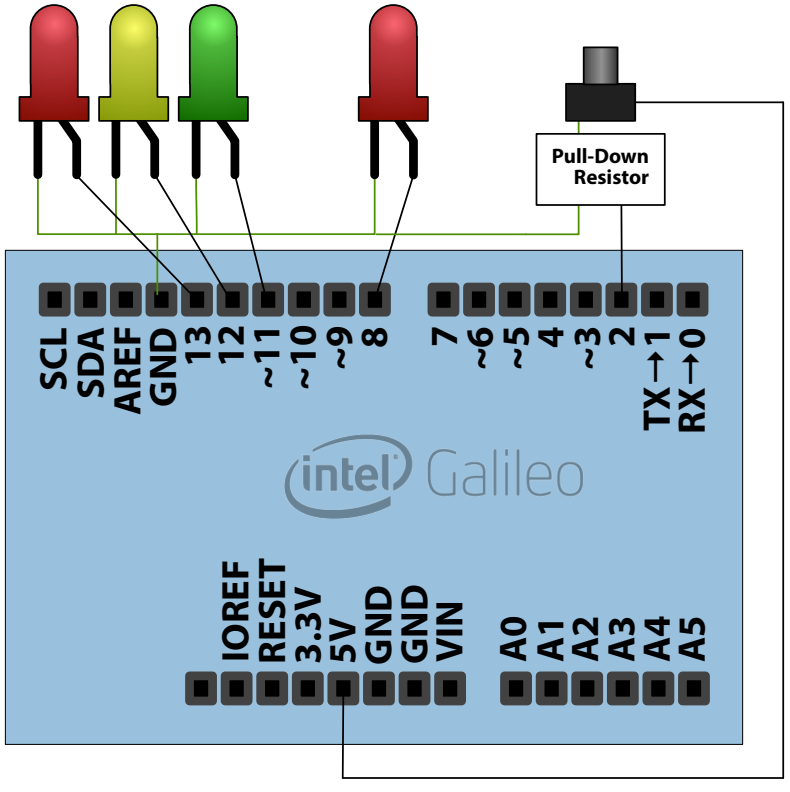
Dieses Übungsblatt ist bis Freitag, 09.05. um 12:00 per E-Mail an den eigenen Tutoren abzugeben.
Bennennung beispielsweise "$Matrikelnummer_Abgabe_$Blattnummer.$Format".

1 Automatische Ampelsteuerung

Auf diesem Übungsblatt geht es darum, den Einsatz von `delay(ms)`, `millis()` und `if () {}` an einem praktischen Projekt zu erproben.
Es geht um die Realisierung einer Ampel, die Automobilen anzeigt, ob sie fahren dürfen oder nicht. Zunächst soll die Ampel rein zeitgesteuert sein.
Ein Ampelübergang läuft dabei wie folgt ab:

1. Die Ampel beginnt mit grün. Sie bleibt für 30 Sekunden grün.
2. Danach soll die Ampel auf gelb schalten. Sie bleibt 3 Sekunden gelb.
3. Danach soll die Ampel auf rot schalten. Sie bleibt für 10 Sekunden rot.
4. Danach soll die Ampel auf gelb schalten. Sie bleibt wieder für 3 Sekunden gelb.
5. Danach soll die Ampel wieder auf grün schalten. Sie bleibt für 30 Sekunden grün.
6. Sie wird immer wieder gelb, rot, gelb, grün usw., wie oben schon beschrieben.

Zur Realisierung der Verzögerung sollen Sie dabei der Einfachheit wegen zunächst die Funktion `delay(ms)` einsetzen, die die in Millisekunden übergebene Zeit lang wartet.



Beispiellösung. Listing 1: Lösung: Einfache Ampelsteuerung

```
// uncomment following line to speed things up
#define DEBUG

// following 6 lines of code can be ignored for the solution
// and are just there to speed the delay up.
#ifdef DEBUG
void delayFast(long ms) { delay(ms / 10); }
#define delay(ms) delayFast(ms)
#endif


// pin definitions
int pinRed = 13, pinYellow = 12, pinGreen = 11;

void setup() {
  pinMode(pinRed, OUTPUT);
  pinMode(pinYellow, OUTPUT);
  pinMode(pinGreen, OUTPUT);

  // we want a serial connection for debugging
  Serial.begin(9600);
}

void loop() {
  // green phase
  // this will send a debug message over the serial connection
  Serial.println("green");
  // turn green LED on
  digitalWrite(pinGreen, HIGH);
  // wait for specified amount of time in milliseconds
  delay(30000);
  // turn off green LED
  digitalWrite(pinGreen, LOW);

  // first yellow phase
  Serial.println("yellow");
  digitalWrite(pinYellow, HIGH);
  delay(3000);
  digitalWrite(pinYellow, LOW);

  // red phase
  Serial.println("red");
  digitalWrite(pinRed, HIGH);
  delay(10000);
  digitalWrite(pinRed, LOW);

  // second yellow phase
  Serial.println("yellow");
  digitalWrite(pinYellow, HIGH);
  delay(3000);
  digitalWrite(pinYellow, LOW);
}
```

2 Interaktive Ampelsteuerung

1. Stellen Sie ihr Programm aus der vorherigen Aufgabe jetzt auf die Benutzung von `millis()` und `if () {}` um. Sie können sich dazu merken, wie lange sich die Ampel schon in der aktuellen Phase (rot, gelb, grün) befindet. Das Programm aus Listing 1 demonstriert dabei nochmal, wie man ein Blinklicht über `millis()` realisieren kann, und kann entsprechend erweitert als Basis für die Ampelsteuerung dienen.

2. Erweitern Sie ihre Ampelsteuerung aus der vorherigen Teilaufgabe:

   (a) An Pin 2 ist ein Taster mit Pull-Down-Widerstand angeschlossen (wie in der Vorlesung).

   (b) Die Ampel wird weiterhin Automobilen anzeigen, ob sie fahren können oder nicht.

   (c) Die Ampel bleibt jetzt dauerhaft grün, solange kein Fußgänger die Straße überqueren möchte.

   (d) Wenn der Taster gedrückt wird, wechselt die Ampel nach rot. Der Übergang läuft weiterhin ab, wie in Aufgabe 1 beschrieben. (3 Sekunden gelb, 10 Sekunden rot, 3 Sekunden gelb)

   (e) Die Ampel soll nach jeder Rotphase für 10 Sekunden grün sein. Wird in dieser Zeit der Taster gedrückt, wird die Ampel nach Ablauf der 10 Sekunden grün.

   (f) Tasterdrücke werden also nie ignoriert, aber unter Umständen erst zeitverzögert ausgeführt.

   (g) (Tipp: Es empfiehlt sich die Verwendung einer Variable, die speichert, wann die letzte Rotphase abgeschlossen war.)

   (h) (Tipp: Es empfiehlt sich außerdem die Verwendung einer Variable, die speichert, ob der Taster gedrückt und noch nicht verarbeitet wurde.)

Listing 2: Warten per millis()

```
// In dieser Variable merken wir uns, wann der Zustand wechseln soll.
int zeitNaechsterWechsel = 0;

// In dieser Variable merken wir uns, ob die LED gerade leuchtet oder nicht.
int ledLeuchtet = 0;

// ... setup() und die Definition von ledPin fehlen ...

void loop() {
  // Ist es jetzt schon an der Zeit fuer den naechsten Zustandswechsel?
  if (millis() > zeitNaechsterWechsel) {
    // Wenn die LED vorher leuchtete, leuchtet sie jetzt nicht mehr.
    ledLeuchtet = !ledLeuchtet;

    // LED-Zustand aktualisieren.
    digitalWrite(ledPin, ledLeuchtet);

    // Nach 5 Sekunden findet der naechste Zustandswechsel statt.
    zeitNaechsterWechsel = millis() + 5000;
  }
}
```

Beispiellösung. Listing 3: Lösung: Ampelsteuerung mit millis()

```
/* Nur Ergaenzungen zum vorher abgedruckten Code */

int t0;

void setup() {
  ... /* Verbleibender Code wie zuvor */
  t0 = millis();  // Reference point for time measurements
}

void loop() {
  // we measure the current time in milliseconds with millis()
  // and substract the reference point from setup(). Then we scale
  // it to seconds and divide by rest by the total time of one
  // traffic light cycle (46 seconds).
  int t = (millis() - t0) / 1000 % 46;

  // There are 4 different phases. Which one we are in is selected
  // by the if, depending on our current time.
  if (t < 30) {
    // First 30 seconds are red
    Serial.println("red");

    // turn off LED of previous phase
    digitalWrite(pinYellow, LOW);
    // turn on LED of current phase
    digitalWrite(pinRed, HIGH);
  } else if (t < 33) {
    // Notice this is an else if, therefore we can exclude that
    // we're in the previous case (t < 30). Thus we select here on
    // 30 <= t < 33, which means yellow (first one)
    Serial.println("yellow");
    digitalWrite(pinRed, LOW);
    digitalWrite(pinYellow, HIGH);
  } else if (t < 43) {
    // 33 <= t < 43 - green
    Serial.println("green");
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinGreen, HIGH);
  } else {
    // If t is not less than 43 seconds, we must be in the
    // last phase.
    // 43 <= t < 46 - second yellow
    Serial.println("yellow");
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, HIGH);
  }
}
```

Beispiellösung. Listing 4: Lösung: Erweiterte Ampelsteuerung

```
// pin definitions
int pinRed = 13;
int pinYellow = 12;
int pinGreen = 11;
int pinButton = 2;

int t0;

// time of end of last red phase
// initialize to -20, so an immediate button press will trigger
// switching the traffic light
int tLastRed = -20;

// time reference for switching to red
// we use 0 as indicator, that this variable isn't set yet.
int tPhaseBegin = 0;

// if the button was pressed
boolean buttonPressed = false;


void setup() {
  // configure output pins for LEDs as OUTPUT
  pinMode(pinRed, OUTPUT);
  pinMode(pinYellow, OUTPUT);
  pinMode(pinGreen, OUTPUT);
  // configure button pin as INPUT
  pinMode(pinButton, INPUT);

  // we want a serial connection for debugging
  Serial.begin(9600);

  // reference point for time measurements
  t0 = millis();

  // at start the green LED should be on
  digitalWrite(pinGreen, HIGH);
}


void loop() {
  // measure current time. As before, we substract our start time
  // and scale to seconds.
  long t = (millis() - t0) / 1000;

  // check if button was pressed
  if (digitalRead(pinButton) && !buttonPressed) {
    Serial.println("button␣pressed");
    buttonPressed = true;
  }
```

```
  // if the last red phase was more than 10 seconds ago and the
  // button was pressed, we turn the traffic light red
  if (t - tLastRed > 10 && buttonPressed) {
    if (tPhaseBegin == 0) {
      // remember start time of phase change
      tPhaseBegin = t;
    }

    // tC: time in current change phase
    int tC = t - tPhaseBegin;

    // similar to exercise 1
    if (tC < 3) {
      Serial.println("yellow");
      digitalWrite(pinGreen, LOW);
      digitalWrite(pinYellow, HIGH);
    }
    else if (tC < 13) {
      Serial.println("red");
      digitalWrite(pinYellow, LOW);
      digitalWrite(pinRed, HIGH);
    }
    else if (tC < 16) {
      Serial.println("yellow");
      digitalWrite(pinRed, LOW);
      digitalWrite(pinYellow, HIGH);
    }
    else {
      // traffic light turns green again
      Serial.println("green");
      digitalWrite(pinYellow, LOW);
      digitalWrite(pinGreen, HIGH);

      // also remember time of last red phase
      tLastRed = t;
      // reset button press
      buttonPressed = false;
      // and reset phase begin time
      tPhaseBegin = 0;
    }
  }
}
```

3. **Bonusaufgabe 1:** Wie zuvor, jedoch:

   Während der Fußgänger auf das Signal wartet (wenn der Taster gedrückt wird, während die Ampel grün sein soll) blinkt eine weitere rote LED 4x pro Sekunde, um zu zeigen, dass das Signal kommt.

Beispiellösung. Listing 5: Lösung: Mit Signal-Licht

```
// pin definitions
int pinRed = 13, pinYellow = 12, pinGreen = 11, pinSignal = 10;
int pinButton = 2;

int t0;

// time of end of last red phase
// initialize to -20, so an immediate button press will trigger
// switching the traffic light
int tLastRed = -20;

// time reference for switching to red
// we use 0 as indicator, that this variable isn't set yet.
int tPhaseBegin = 0;

// if the button was pressed
boolean buttonPressed = false;


void setup() {
  // configure output pins for LEDs as OUTPUT
  pinMode(pinRed, OUTPUT);
  pinMode(pinYellow, OUTPUT);
  pinMode(pinGreen, OUTPUT);
  // configure button pin as INPUT
  pinMode(pinButton, INPUT);
  // configure signal LED as OUTPUT
  pinMode(pinSignal, OUTPUT);

  // we want a serial connection for debugging
  Serial.begin(9600);

  // reference point for time measurements
  t0 = millis();

  // at start the green LED should be on
  digitalWrite(pinGreen, HIGH);
}

void loop() {
  // measure current time. As before, we substract our start time
  // and scale to seconds.
  long t = (millis() - t0) / 1000;

  // check if button was pressed
  if (digitalRead(pinButton) && !buttonPressed) {
    Serial.println("button␣pressed");
    buttonPressed = true;
  }
```

```
  // if the last red phase was more than 10 seconds ago and the
  // button was pressed, we turn the traffic light red
  if (buttonPressed) {
    if (t - tLastRed <= 10) {
      // bonus exercise: button was pressed, but pedestrians still
      // have to wait.  since we scale our time to seconds here,
      // we will just use millis() and scale to 1/8 of a second
      int tS = millis() * 8 / 1000;
      digitalWrite(pinSignal, tS % 2 == 0);
    } else {
      // 10 seconds passed, traffic light will turn red now
      // also make sure signal LED is off.
      digitalWrite(pinSignal, LOW);

      if (tPhaseBegin == 0) {
        // remember start time of phase change
        tPhaseBegin = t;
      }

      // tC: time in current change phase
      int tC = t - tPhaseBegin;

      // similar to exercise 1
      if (tC < 3) {
        Serial.println("yellow");
        digitalWrite(pinGreen, LOW);
        digitalWrite(pinYellow, HIGH);
      } else if (tC < 13) {
        Serial.println("red");
        digitalWrite(pinYellow, LOW);
        digitalWrite(pinRed, HIGH);
      } else if (tC < 16) {
        Serial.println("yellow");
        digitalWrite(pinRed, LOW);
        digitalWrite(pinYellow, HIGH);
      } else {
        // traffic light turns green again
        Serial.println("green");
        digitalWrite(pinYellow, LOW);
        digitalWrite(pinGreen, HIGH);

        // also remember time of last red phase
        tLastRed = t;
        // reset button press
        buttonPressed = false;
        // and reset phase begin time
        tPhaseBegin = 0;
      }
    }
  }
}
```

4. **Bonusaufgabe 2:** Realisieren Sie das Fußgänger-Gegenstück zur Auto-Ampel aus den vorherigen Aufgaben. Betreiben Sie beide Ampeln gleichzeitig und aufeinander abgestimmt. Benutzen Sie Ports 7 bis 5 für die LEDs rot bis grün.

Beispiellösung.

Listing 6: Lösung: Mit Fußgänger-Ampel

```
// pin definitions
int pinRed = 13;
int pinYellow = 12;
int pinGreen = 11;
int pin2Red = 7;
int pin2Yellow = 6;
int pin2Green = 5;
int pinButton = 2;
int pinSignal = 10;

int t0;

// time of end of last red phase
// initialize to -20, so an immediate button press will trigger
// switching the traffic light
int tLastRed = -20;

// time reference for switching to red
// we use 0 as indicator, that this variable isn't set yet.
int tPhaseBegin = 0;

// if the button was pressed
boolean buttonPressed = false;


void setup() {
  // configure output pins for LEDs as OUTPUT
  pinMode(pinRed, OUTPUT);
  pinMode(pinYellow, OUTPUT);
  pinMode(pinGreen, OUTPUT);
  // configure button pin as INPUT
  pinMode(pinButton, INPUT);
  // configure signal LED as OUTPUT
  pinMode(pinSignal, OUTPUT);

  // we want a serial connection for debugging
  Serial.begin(9600);

  // reference point for time measurements
  t0 = millis();

  // at start the green LED should be on
  digitalWrite(pinGreen, HIGH);
  digitalWrite(pin2Red, HIGH);
}
```

```
void loop() {
  // measure current time. As before, we substract our start time
  // and scale to seconds.
  long t = (millis() - t0) / 1000;

  // check if button was pressed
  if (digitalRead(pinButton) && !buttonPressed) {
    Serial.println("button␣pressed");
    buttonPressed = true;
  }

  // if the last red phase was more than 10 seconds ago and the
  // button was pressed, we turn the traffic light red
  if (buttonPressed) {
    if (t - tLastRed <= 10) {
      // bonus exercise: button was pressed, but pedestrians still
      // have to wait. since we scale our time to seconds here,
      // we will just use millis() and scale to 1/8 of a second
      int tS = millis() * 8 / 1000;
      digitalWrite(pinSignal, tS % 2 == 0);
    } else {
      // 10 seconds passed, traffic light will turn red now
      // also make sure signal LED is off.
      digitalWrite(pinSignal, LOW);

      if (tPhaseBegin == 0) {
        // remember start time of phase change
        tPhaseBegin = t;
      }

      // tC: time in current change phase
      int tC = t - tPhaseBegin;

      // similar to exercise 1
      if (tC < 3) {
        Serial.println("yellow");
        digitalWrite(pinGreen, LOW);
        digitalWrite(pinYellow, HIGH);
        // bonus exercise 2: also switch pedestrian traffic lights
        digitalWrite(pin2Red, LOW);
        digitalWrite(pin2Yellow, HIGH);
      } else if (tC < 13) {
        Serial.println("red");
        digitalWrite(pinYellow, LOW);
        digitalWrite(pinRed, HIGH);
        digitalWrite(pin2Yellow, LOW);
        digitalWrite(pin2Green, HIGH);
      } else if (tC < 16) {
        Serial.println("yellow");
        digitalWrite(pinRed, LOW);
        digitalWrite(pinYellow, HIGH);
        digitalWrite(pin2Green, LOW);
        digitalWrite(pin2Yellow, HIGH);
      }
```

```
      else {
        // traffic light turns green again
        Serial.println("green");
        digitalWrite(pinYellow, LOW);
        digitalWrite(pinGreen, HIGH);
        digitalWrite(pin2Yellow, LOW);
        digitalWrite(pin2Red, HIGH);

        // also remember time of last red phase
        tLastRed = t;
        // reset button press
        buttonPressed = false;
        // and reset phase begin time
        tPhaseBegin = 0;
      }
    }
  }
}
```