Montage und Programmierung eines Roboters für ROBOCUP JUNIOR RESCUE mit Arduino Nano

Teil 2.3: Interrupt

INTERRUPT

„Paralleler“ Programmablauf

Wir definieren zusätzliche Ausdrücke:
Bisher haben wir LED gleichgesetzt mit 13.
Jetzt kommt noch
- **Trigger_US**, ist das Gleiche wie die Zahl 7
- **Echo**, wird zu 2
Immer wenn wir im Programm den Ausdruck Echo benutzen, ersetzt der Compiler ihn durch die Zahl 2.

```
34  #define LED 13 //LED of the Arduino board is mounted at Pin 13
35  #define Trigger_US     7     //Ultraschalltriggerpin
36  #define Echo           2     //Ultraschallechopin
37
38  unsigned long US_Takt; //alle 40 Millisekunden US neu initialisieren
39  unsigned long Echo_Time; //Laufzeit des Echos
40  unsigned long Echo_Time_begin; //Anfang der US-Messung
41  unsigned long Laufzeit;
42  unsigned long LED_Takt; //alle 500ms toggelt die LED
```

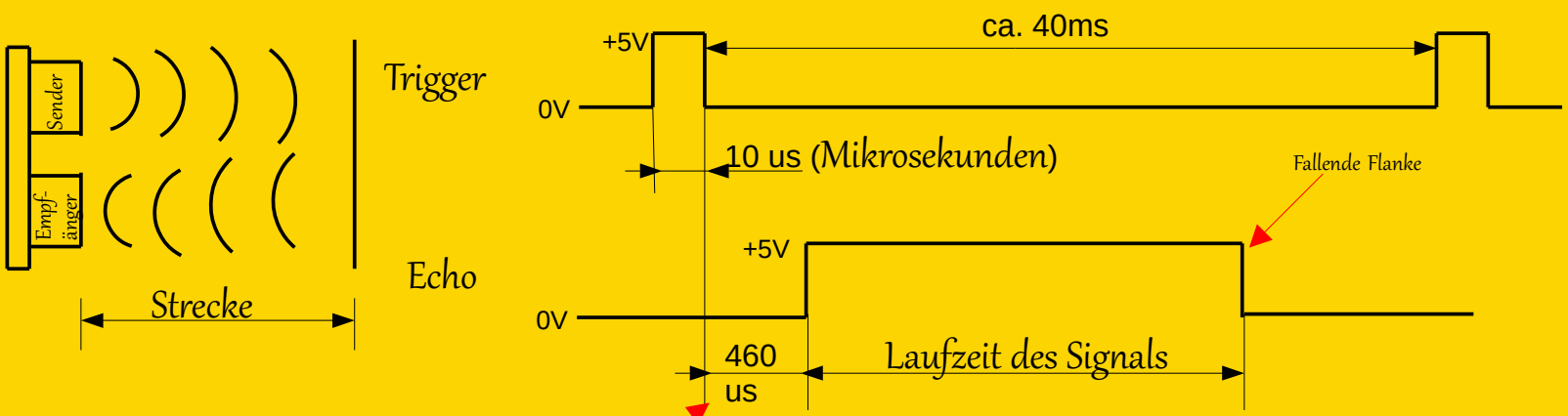
Außerdem brauchen wir sehr große Variablen:
unsigned long: Das sind 4 Bytes.
Mit 4 Bytes = 32 Bits lassen sich Dezimalwerte von 0 bis 4.294.967.295 darstellen.
unsigned bedeutet, dass man darin keine negativen Zahlen abspeichern kann.
Brauchen wir negative Zahlen müssen wir sie als **signed** deklarieren.

Im setup() kommen ein paar Pins hinzu:
Trigger_US (Pin 7) wird ein Output Pin
Echo (Pin 2) wird ein Input Pin
Außerdem sagen wir dem Compiler, dass der Echo Pin einen **Interrupt** auslösen soll. Wenn an ihm eine fallende Flanke festgestellt wird, soll er die Funktion US aufrufen.

```
44  void setup() {
45    pinMode(Trigger_US, OUTPUT); //Pin 4 als Ausgang festlegen
46    pinMode(Echo, INPUT); //Pin 2 als Eingang festlegen
47    attachInterrupt(digitalPinToInterrupt(Echo), US, FALLING); //Bekanntmachung:
48    //Interrupt wird ausgelöst durch fallende Flanke an Pin "Echo" (Pin 2), es
49    //soll dann die Funktion "US()" ausgeführt werden.
50    pinMode(LED, OUTPUT);
51    Serial.begin(74880); //Baud Rate
```

Output bedeutet, dass wir den Pin entweder auf **0V** oder **+5V** legen können. Null Volt ist gleichbedeutend mit einer Null im **Binärsystem**, die +5Volt sind gleichbedeutend mit einer Eins.
Input bedutet, dass wir an dem Pin Signale von 0V (=0) und +5V (=1), entgegen nehmen können.

Ultraschall-Sensor



Der Ultraschallsensor gibt von sich aus keine Werte aus.
Wir müssen ihn dazu immer wieder auffordern.
Etwa alle 40ms setzen wir für 10us (Mikrosekunden) den Trigger Pin auf +5V.
Dadurch wird er veranlasst ein kodiertes Signal auszusenden.
Das dauert **460us**. Dann zieht er den Echo Pin von 0V auf +5V. Wenn das Signal zurück kommt, wird der Echo Pin von **+5V wieder auf 0V** gezogen. **Fallende Flanke**.
Das kann der Mikrocontroller registrieren.

Wir haben dem Mikrocontroller mitgeteilt, dass er bei einer **fallenden Flanke** am Echo Pin (Pin 2) einen Interrupt auslösen soll.

```
47    attachInterrupt(digitalPinToInterrupt(Echo), US, FALLING); //Bekanntmachung:
```

Das bedeutet, dass er das laufende Programm **sofort** unterbrechen soll. Dabei werden alle Daten, die gerade verarbeitet werden „gerettet“.
Dann soll er die Funktion **US()** ausführen.
Nach Beendigung der Funktion, springt der Mikrocontroller wieder zurück an die Stelle, an der er das Programm unterbrochen hat, holt sich die geretteten Daten und arbeitet dort weiter.

Die Funktion **US()** beinhaltet nur zwei Zeilen.
Die (unsigned long) Variable **Echo_Time** erhält den aktuellen Wert der **internen Uhr** in Mikrosekunden (**micros()**)
Dann wird noch die Strecke vom US-Sensor zum Hindernis berechnet.

```
86  //Interrupt: Fallende Flanke an Pin 2
87  //Hierher springt der Mikrocontroller, wenn die Flanke am Echo-Pin von +5V auf 0V(GND) fällt
88  void US (void) {
89    Echo_Time = micros(); //aktuelle Zeit in us abspeichern
90    Laufzeit = ((Echo_Time - Echo_Time_begin) - 460) /58;
91  }
```

Von der Differenz aus **Echo_Time** und **Echo_Time_begin** werden die 460us abgezogen. Das Ergebnis wird dann noch durch 58 geteilt.
Für **einen cm braucht der Schall ca. 29us**.
Da der Schall den Weg zweimal zurücklegen muss, teilen wir durch 58.

Zum Hauptprogramm:
In Zeile 70 speichern wir in der Variablen **US_Takt** die **aktuelle „Zeit“** des Mikrocontrollers in ms (Millisekunden) ab.
Wenn wir die Endlosschleife erneut durchlaufen, wird in Zeile 63 geprüft, ob die aktuelle Zeit mehr als 40ms größer geworden ist als US_Takt. Es sind 40ms vergangen.

Wenn die **Aussage wahr** ist, wird der US_Trigger Pin für **10us** auf **+5V** (**HIGH**) gezogen, bevor er wieder auf **0V** (**LOW**) fällt.
Damit wird eine neue Messung initialisiert.
Danach wird die **aktuelle Zeit** in Mikrosekunden gespeichert, die zur Berechnung der Entfernung gebraucht wird.
Am Ende des Hauptprogramms wird mit dem gleichen Schema die LED getoggelt. (Zustand gewechselt)
Das Ergebnis von digitalRead wird negiert (^ = xor).
Die LED blinkt mit der Taktfrequenz 1s.

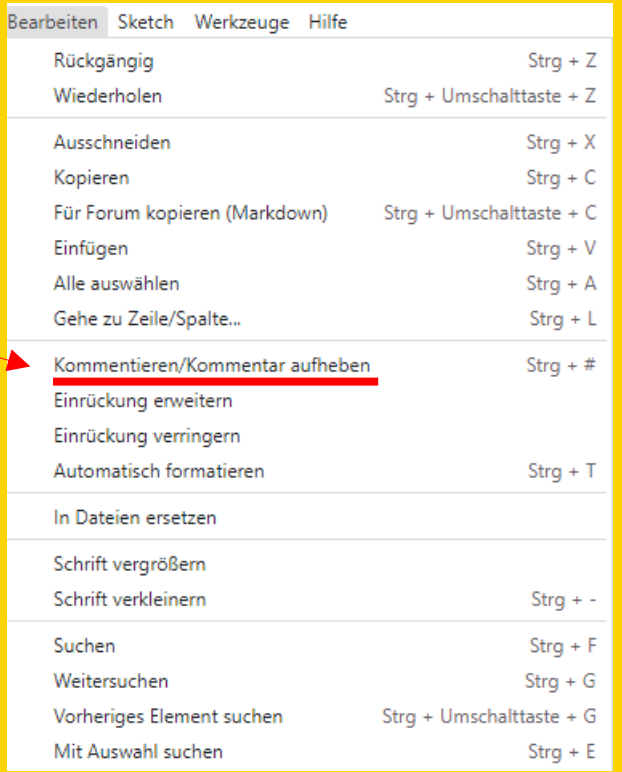
```
61  //millis() liefert die Zeit die seit dem Einschalten des...
62  //...Arduino Boards vergangen ist, in MILLIsekunden
63    if(millis()- US_Takt > 40) { //40 ms vorbei ?
64      digitalWrite (Trigger_US, HIGH); //Pin 4 für...
65      delayMicroseconds(10); //...10us (Microsekunden)...
66      digitalWrite (Trigger_US, LOW);//...HI triggern (von 0V auf 5V)
67      Echo_Time_begin = micros(); //Start der Zeitmessung des US-Signals
68      //micros() liefert die Zeit die seit dem Einschalten des...
69      //...Arduino Boards vergangen ist, in MIKROsekunden
70      US_Takt = millis(); //aktuelle Zeit in ms abspeichern
71    }
72    Serial.println(Laufzeit); //Laufzeit des Signals anzeigen
73    //Die Zeile 32 unbedingt beim Wettlauf auskommentieren, braucht viel Zeit!
74    //Aufgabe: Motor vorwärts drehen lassen, wenn ein Hindernis <15cm vor dem
75    //US-Sensor ist, Motor stoppen
76
77  /***LED blinken "nebenbei"********/
78    if(millis()- LED_Takt > 500) { //500 ms vorbei ?
79      digitalWrite(LED, digitalRead(LED) ^ 1); //Toggle LED
80      LED_Takt = millis(); //aktuelle Zeit in ms abspeichern
81    }
82  }
```

Aufgabe:

Kommentiere den Teil der blinkenden LED aus.

```
76
77  /***LED blinken "nebenbei"********/
78  //   if(millis()- LED_Takt > 500) { //500 ms vorbei ?
79  //     digitalWrite(LED, digitalRead(LED) ^ 1); //Toggle LED
80  //     LED_Takt = millis(); //aktuelle Zeit in ms abspeichern
81  //   }
82  // }
83
```

Zeilen markieren
Auskommentieren
Jede Zeile hat jetzt am Anfang zwei Schrägstriche //



Schreibe ein Programm:
Wenn ein Hindernis vor dem US-Sensor ist, dann soll die LED aus gehen.
Sonst soll sie an sein.

Hierfür brauchst du den Befehl if/else.

```
if(Bedingung erfüllt) {
   Mache etwas;
}
else {       [Bedingung ist nicht erfüllt]
   Mache etwas anderes;
}
```

Warum holen wir den Abstandswert nicht einfach mit pulseIn()?
Weil wir dann das Programm so lange unterbrechen, bis der Wert endlich angekommen ist.
Das kann bis zu 40ms dauern.
Zeit genug um gegen die Wand zu knallen.

Montage und Programmierung eines Roboters für ROBOCUP JUNIOR RESCUE mit Arduino Nano

Teil 2.4: Black and White