analogWrite am esp32

Datum: 2023-11-18

Drivysteuerung: Christoph kann auf einer Installation das alte Programm funktionierend erstellen und der neu Code wird kompiliert, aber die Reaktionszeit auf Bluetooth-Eingabe ist sehr hoch und die Motoren fahren ... seltsam.

Wo kommt die Funktion her?

`#include <analogwrite.h>` und `#include <arduino.h>` schlagen fehl.

Es gibt weder in .arduino15 noch .arduinoIDE analogwrite.h.

arduino.h gibt es in:

```
.arduino15/packages/arduino/hardware/avr/1.8.6/firmwares/wifishield/wifiHD/src/SOFTWAR
.arduino15/packages/arduino/hardware/avr/1.8.6/firmwares/wifishield/wifi_dnld/src/SOFT
```

Suchen wo die Funktion definiert ist:

```
grep -i "void *analogwrite" -r .arduino15
.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal-ledc.c:void anal
.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal-ledc.c:void anal
.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal-ledc.c:void anal
.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal.h:void analogWri
.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal.h:void analogWri
.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal.h:void analogWri
.arduino15/packages/arduino/hardware/avr/1.8.6/cores/arduino/Arduino.h:void analogWrit
.arduino15/packages/arduino/hardware/avr/1.8.6/cores/arduino/wiring_analog.c:void anal
.arduino15/libraries/Firmata/examples/ServoFirmata/ServoFirmata.ino:void analogWriteCa
.arduino15/libraries/Firmata/examples/SimpleAnalogFirmata/SimpleAnalogFirmata.ino:void
.arduino15/libraries/Firmata/examples/StandardFirmataBLE/StandardFirmataBLE.ino:void a
.arduino15/libraries/Firmata/examples/StandardFirmataChipKIT/StandardFirmataChipKIT.in
.arduino15/libraries/Firmata/examples/AnalogFirmata/AnalogFirmata.ino:void analogWrite
.arduino15/libraries/Firmata/examples/StandardFirmataWiFi/StandardFirmataWiFi.ino:void
.arduino15/libraries/Firmata/examples/StandardFirmata/StandardFirmata.ino:void analogW
.arduino15/libraries/Firmata/examples/OldStandardFirmata/OldStandardFirmata.ino:void a
.arduino15/libraries/Firmata/examples/StandardFirmataPlus/StandardFirmataPlus.ino:void
.arduino15/libraries/Firmata/examples/StandardFirmataEthernet/StandardFirmataEthernet.
```

in .arduinoIDE ist nichts.

esp32-hal-ledc.c

Pfad: packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal-ledc.c

Der Prototyp für die include-Anweisung (die schon jemand macht) steht in :

.arduino15/packages/esp32/hardware/esp32/2.0.11/cores/esp32/esp32-hal.h

```
static int8_t pin_to_channel[SOC_GPIO_PIN_COUNT] = { 0 };
static int cnt_channel = LEDC_CHANNELS;
static uint8_t analog_resolution = 8;
static int analog_frequency = 1000;
void analogWrite(uint8_t pin, int value) {
    // Use ledc hardware for internal pins
    if (pin < SOC_GPIO_PIN_COUNT) {
        int8_t channel = -1;
```

```
        if (pin_to_channel[pin] == 0) {
            if (!cnt_channel) {
                log_e("No more analogWrite channels available! You can have maximum %u
                return;
            }
            cnt_channel--;
            channel = cnt_channel;
        } else {
            channel = analogGetChannel(pin);
        }
        log_v("GPIO %d - Using Channel %d, Value = %d", pin, channel, value);
        if(ledcSetup(channel, analog_frequency, analog_resolution) == 0){
            log_e("analogWrite setup failed (freq = %u, resolution = %u). Try setting
            return;
        }
        ledcAttachPin(pin, channel);
        pin_to_channel[pin] = channel;
        ledcWrite(channel, value);
    }
}

int8_t analogGetChannel(uint8_t pin) {
    return pin_to_channel[pin];
}

void analogWriteFrequency(uint32_t freq) {
    if (cnt_channel != LEDC_CHANNELS) {
        for (int channel = LEDC_CHANNELS - 1; channel >= cnt_channel; channel--) {
            if (ledcChangeFrequency(channel, freq, analog_resolution) == 0){
                log_e("analogWrite frequency cant be set due to selected resolution! T
                return;
            }
        }
    }
  }
  analog_frequency = freq;
}

void analogWriteResolution(uint8_t bits) {
    if(bits > LEDC_MAX_BIT_WIDTH) {
        log_w("analogWrite resolution width too big! Setting to maximum %u bits)", LED
        bits = LEDC_MAX_BIT_WIDTH;
    }
    if (cnt_channel != LEDC_CHANNELS) {
        for (int channel = LEDC_CHANNELS - 1; channel >= cnt_channel; channel--) {
            if (ledcChangeFrequency(channel, analog_frequency, bits) == 0){
                log_e("analogWrite resolution cant be set due to selected frequency! T
                return;
            }
        }
    }
    analog_resolution = bits;
}
```

Frequenz und Auflösung sind umstellbar und per default auf 1kHz 8 bit.

ABER: Christoph kann auf einer Installation das alte Programm funktionierend erstellen und der neu Code wird kompiliert, aber die Reaktionszeit auf Bluetooth-Eingabe ist sehr hoch und die Motoren fahren ... seltsam.

ACHTUNG ... : in `analogWrite` werden diese Zeilen jedesmal ausgeführt.

```
    ledcAttachPin(pin, channel);
    pin_to_channel[pin] = channel;
```

```
    ledcWrite(channel, value);
```

das scheint mir gefährlich, weil (womöglich, es hängt von der Hardware dahinter ab) jedesmal der Puls neu beginnt.

Warum ist das wichtig

Wenn ich mit `analogWrite` einen Servo-Motor steuere und die PWM startet irgendwann (unvorhersehbar) dann kann ich jemand verletzen.

Es gibt dazu eine Application Note AN8020 - AVR136: Low-jitter Multi-channel Software PWM.

Versuche

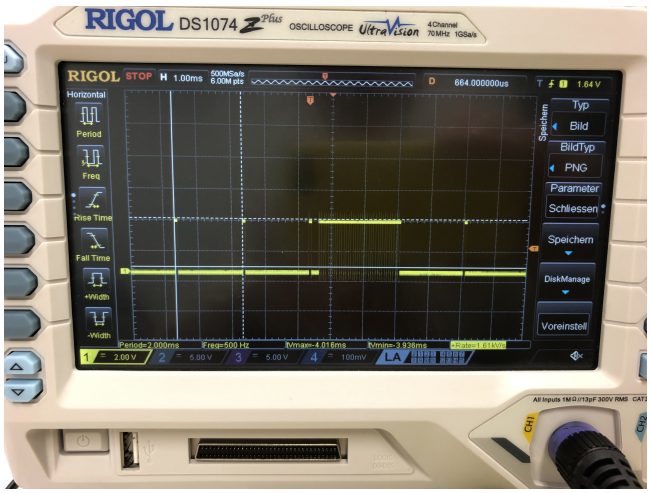
```
void setup() {
  Serial.begin(115200);
}

int loopCnt = 0;
void loop() {
  if (loopCnt > 11) {
    loopCnt = 10;
  }
  Serial.println(loopCnt);
  analogWrite(17, loopCnt);
  //delay(4);
  loopCnt += 10;
}
```

Am Oszilloskop 1kHz mit Pulsweite 38µs.

Abwechselnd mit mehr als 2,5ms höherfrequenten Pulsen.

15,7 High 0,64 Low 22,3 High 32,4 Low [µs]



ändern der Frequenz : 500Hz

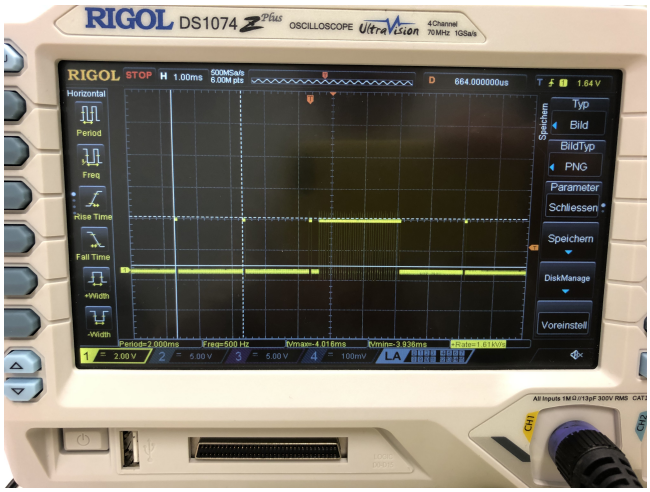
```
void setup() {
  Serial.begin(115200);
  analogWriteFrequency(500);
}

int loopCnt = 0;
void loop() {
  if (loopCnt > 11) {
    loopCnt = 10;
```

```
    }
    Serial.println(loopCnt);
    analogWrite(17, loopCnt);
    //delay(4);
    loopCnt += 10;
  }
```

Die Frequenz sinkt auf 500Hz die Pulsweite ist 80µs. Das wäre 2000µs / 255 * 10 Zähler

und dann gibt es 2,5ms lang 67,5us High 500ns Low.



Aber beide Programm haben einen Fehler

Die Pulsweite ist immer 10.

Die reduzierte Version

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.println(10);
  analogWrite(17, 10);
}
```

Zeigt das selbe Bild.

Ohne Serial.println wird keine PWM ausgegeben.

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  analogWrite(17, 10);
  delay(2);
}
```

gibt korrekte Pulse aus

wenn 500Hz Frequenz eingestellt ist, werden

- bei 1ms Delay korrekte Pulse aber mit 1kHz ausgegeben.
- bei 2ms Delay korrekte Pulse aber mit 500Hz ausgegeben.
- bei 3ms Delay korrekte Pulse abwechselnd 1ms und 2ms low ausgegeben.

esp32 PWM - ledc Access

esp32 hat andere Timer ... wenn man die verwendet geht es

```
// vom Example LEDCSoftwareFade abgeleitet

// use first channel of 16 channels (started from zero)
const int LEDC_CHANNEL =     0;

// use 8 bit precission for LEDC timer
const int LEDC_TIMER_BITS =   8;

// use 500 Hz as a LEDC base frequency
const int LEDC_BASE_FREQ =   500;

const int LED_PIN =           17;

void setup() {
  ledcSetup(LEDC_CHANNEL, LEDC_BASE_FREQ, LEDC_TIMER_BITS);
  ledcAttachPin(LED_PIN, LEDC_CHANNEL);
}

int loopCnt = 1;

void loop() {
  ledcWrite(LEDC_CHANNEL, 50*loopCnt);
  loopCnt++;
  if (loopCnt > 2)
    loopCnt = 1;
}
```

Ergebnis

Der ledc Access muss verwendet werden.

ledcSetup sollte nicht mehrfach für einen Pin/Channel aufgerufen werden.

Am github ist das schon geändert.

https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-ledc.c

Hier wird in `analogWrite` nur `ledcWrite` jedes Mal aufgerufen, und (*HOFFENTLICH*) prüft `perimanGetPinBus` ob der Pin schon konfiguruert ist.

TODO: Man kann in espressif github den Code anschauen und vielleicht mit git blame herausfinden warum die Änderung gemacht wurde und was alles damit in Verbindung steht.

UND in ESP32Servo gibt es ein ESP32PWM ... das die Channels auch verwendet ... ACHTUNG.

arduino uno

Am arduino uno probieren, was passiert, wenn man während einer PWM-Periodendauer (2ms) die Pulsweite umstellt.

```
// arduino UNO analogWrite Versuch
void setup() {
}

int width = 0;
void loop() {
  analogWrite(6, width);
  width++;
  if (width > 255)
    width = 0;
```

```
}
```

Nachdem

- die Periodendauer bei 500Hz 2ms ist
- der Aufruf von `analogWrite` weniger als 10µs dauert

müsste sich die Pulsweite sprunghaft ändern.

ACHTUNG: schaut nihct gut aus der Ausgang ändert sich innerhalb 2ms mehrfach, die Periodendauer ist zufällig ... Pulsweite auch.

KORREKTUR: der arduino UNO liefert ein 1kHz PWM-Signal.

Wenn nach dem `analogWrite` 100µs delay gemacht werden ... schaut das Signal gut aus.

Aufeindanderfolgende Pulse:

High [µs]	Low [µs]	Delta(t-HIGH) [µs]
615	408	
652	372	37
692	332	40
732	290	40
772	252	40

Ergebnis

Periodendauer 1022µs / 255 = 4µs das bedeutet nur jeder 10-te Wert wird wirklich ausgeführt.

Manchmal beim Neustart bei 0 nicht ganz.