

Soft Skills und Technische Kompetenz

Programmierung I
Gruppe - Montag

Malte Grave, 30.11.2020

Agenda

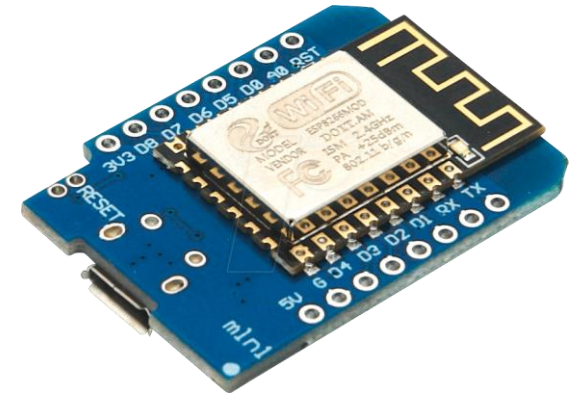
- Fragen zum letztem Aufgabenblatt?
- Wemos D1 Mini kennenlernen
- Einstieg ins Arduino Framework
- Pull-Up / Pull-Down Widerstände
- Übungen

Fragen zu letzten Übung?

- Hattet ihr Probleme bei den Schaltungen?
- Was habt ihr gelernt?
- Was könnt ihr besser machen?
- Gibt es Fragen?

ESP-12F Specs

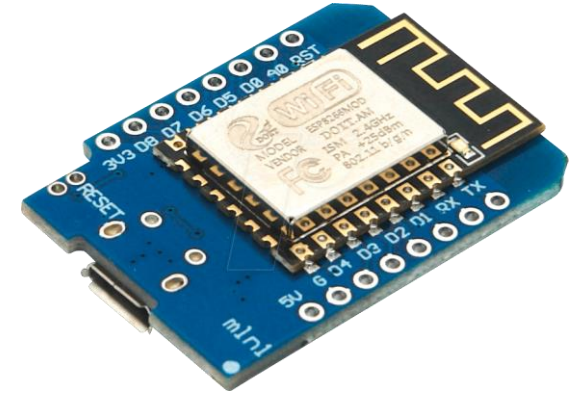
- Euer μ C (Microcontroller) ist der D1 Mini
- Base Clock Speed sind 80 MHz
- Bis zu 160MHz möglich



Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins	1(3.2V Max)
Clock Speed	80/160MHz
Flash	4M Bytes
Size	34.2*25.6mm
Weight	3g

ESP-12F Pins

- Wir haben verschiedene Pins
- Diese werden GPIO's genannt
- *GPIO* – *general purpose input/output*
- *A0* – A steht für Analog
- *D0-8* – D steht für Digital



Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

ESP-12F GPIO's

- Unterschied zwischen Analog und Digital?
- Analog liefert eine Zahl zwischen 0 und $2^{10}-1$
- Digital hingegen nur 0 oder 1
0 = LOW, 3.3 = HIGH bzw. 1
- Der ADC ist dafür verantwortlich
(10 Bit groß beim ESP-12F)

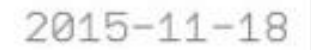
Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

ESP-12F GPIO's

- Was sind TX und RX?
- TX steht für Transmit
- RX steht für Receive
- Für die Serielle Kommunikation
(Zwischen z.B PC und μ C oder Sensoren)

Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.2V	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

ESP-12F Schaltplan

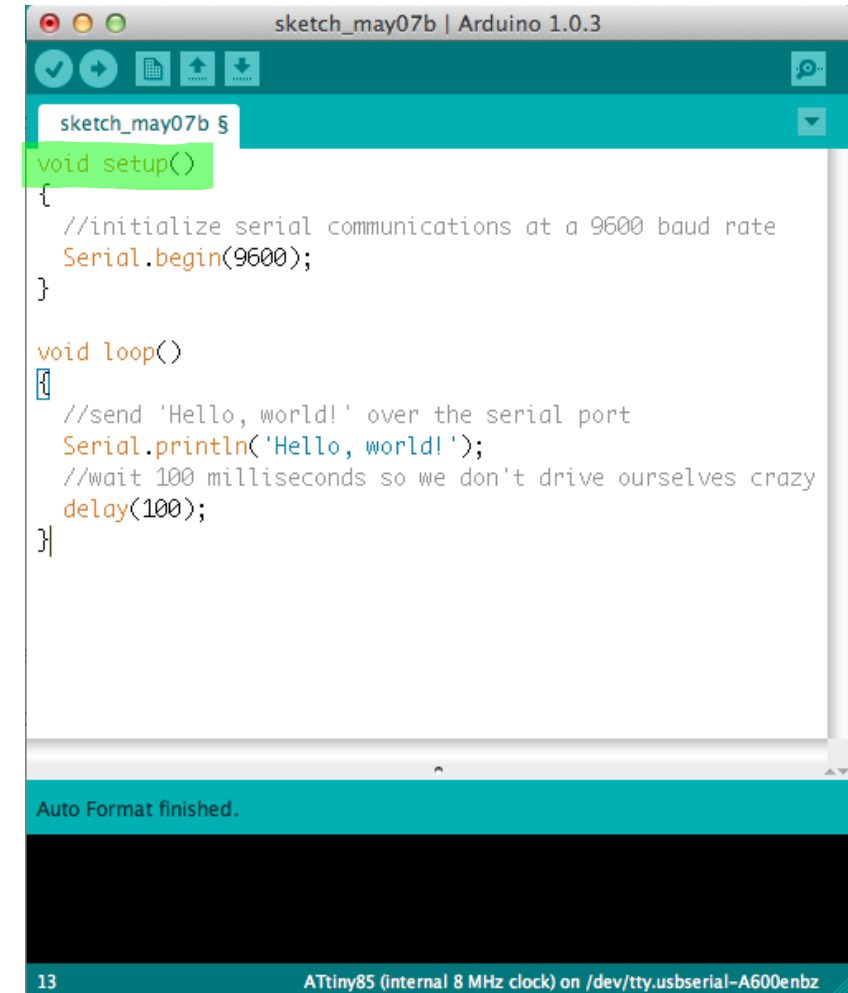


Programmierung

- Wir benutzen das Arduino Framework
- Die Sprache ist ein C/C++ Dialekt
- Verschiedene Bibliotheken stehen uns zur Verfügung

Das Framework

- Jedes Programm hat eine „void setup()“ Methode
- Zuständig für Initialisierung
- Wird nur 1x aufgerufen



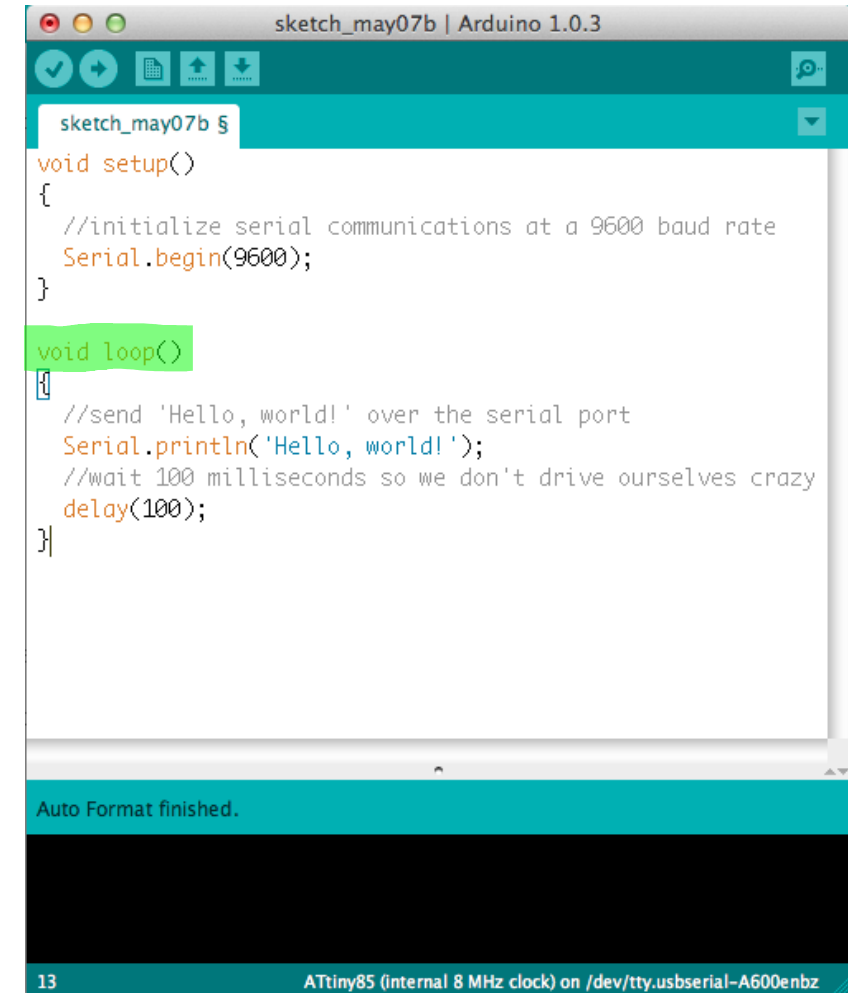
```
sketch_may07b | Arduino 1.0.3  
sketch_may07b $  
void setup()  
{  
  //initialize serial communications at a 9600 baud rate  
  Serial.begin(9600);  
}  
  
void loop()  
{  
  //send 'Hello, world!' over the serial port  
  Serial.println('Hello, world!');  
  //wait 100 milliseconds so we don't drive ourselves crazy  
  delay(100);  
}
```

Auto Format finished.

13 ATtiny85 (internal 8 MHz clock) on /dev/tty.usbserial-A600enbz

Das Framework

- Jedes Programm hat eine „void loop()“ Methode
- Die eigentliche Logik wird dort hineingeschrieben
- Wird nur „dauert“ aufgerufen
- Messungen können durchgeführt werden ...
- Wir benutzen 115200 Baud



```
sketch_may07b | Arduino 1.0.3  
sketch_may07b $  
void setup()  
{  
  //initialize serial communications at a 9600 baud rate  
  Serial.begin(9600);  
}  
void loop()  
{  
  //send 'Hello, world!' over the serial port  
  Serial.println('Hello, world!');  
  //wait 100 milliseconds so we don't drive ourselves crazy  
  delay(100);  
}
```

Auto Format finished.

13 ATTiny85 (internal 8 MHz clock) on /dev/tty.usbserial-A600enbz

Das Framework

- **Wichtige Konstanten**
- HIGH: High Pegel = 3.3V
- LOW: Low Pegel = 0V
- INPUT: Pin wird als Eingang genutzt
- OUTPUT: Pin wird als Ausgang genutzt

Das Framework

– Analog Pins

– `int analogRead(pin)`

- Analog liefert eine Zahl zwischen 0 und $2^{10}-1$

– Angenommen `analogRead` gibt 500 zurück, dann beträgt die Spannung am Pin etwa 1.6V

$$3.3V/1024 = 0.0032V = 32mV$$

$$0.0032V * 500 = 1.6V$$

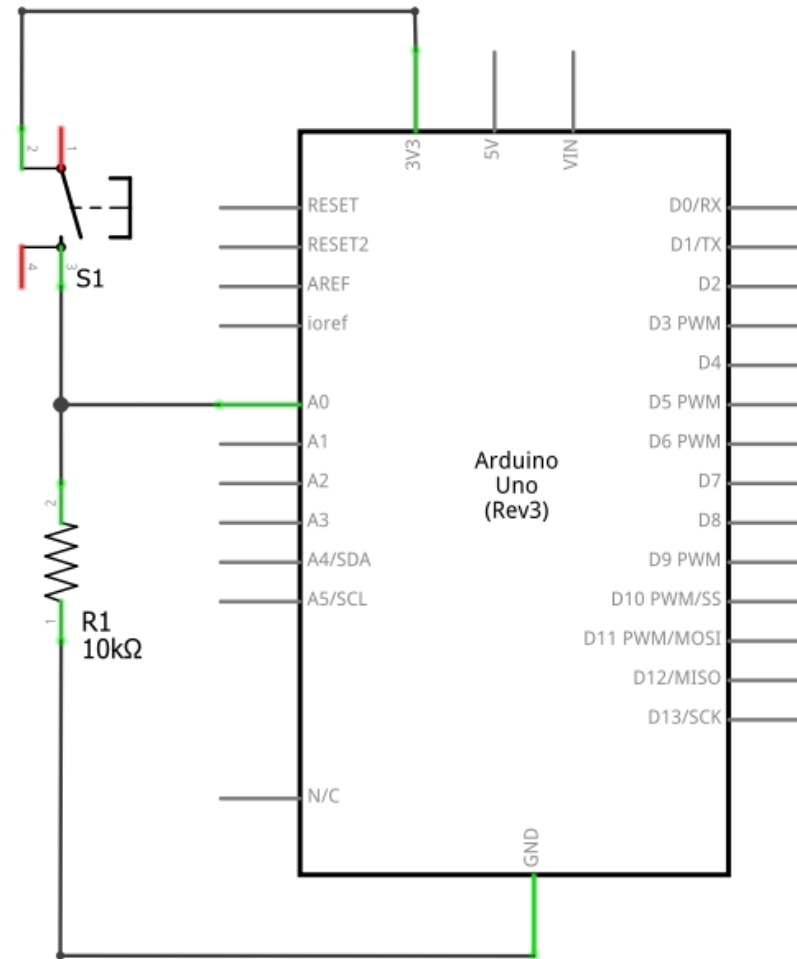
Das Framework

- **Pin Mode's und Digital Pins**
- pinMode(**pin** / [INPUT, OUTPUT])
- digitalWrite(**pin** / [HIGH, LOW])
- int digitalRead(pin)
 - 0 bis 0.825V LOW
 - 2.475V bis 3.3V HIGH
- Was passiert wenn nichts am Pin anliegt?

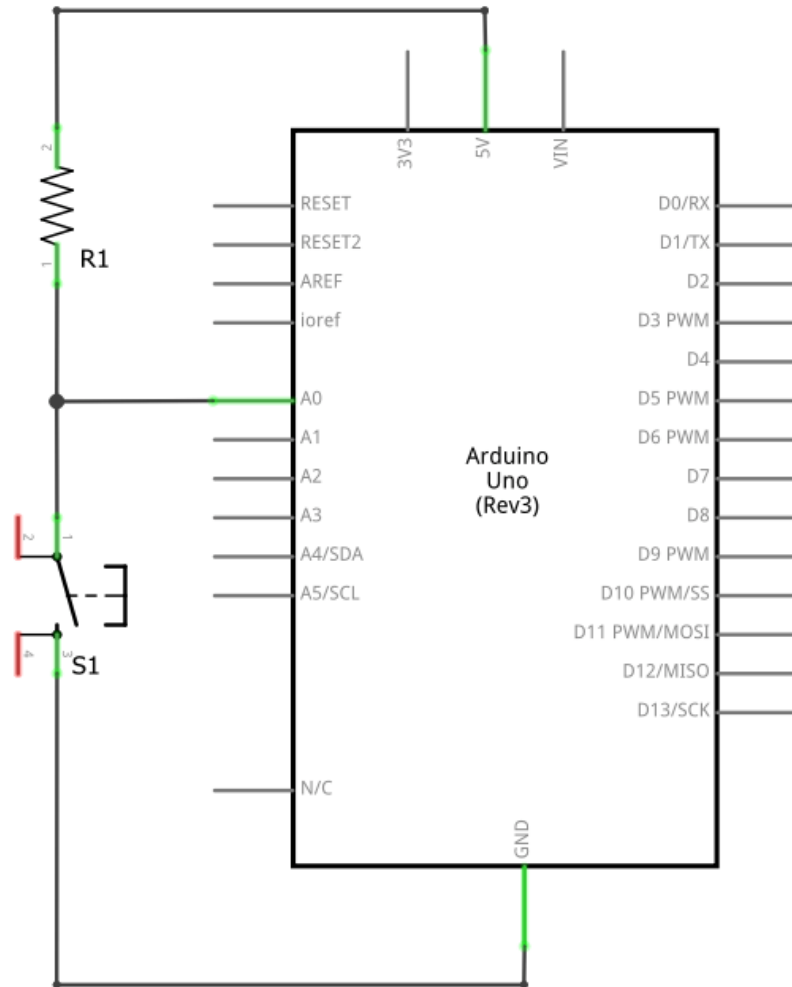
Pull-Down / Pull-Up Widerstände

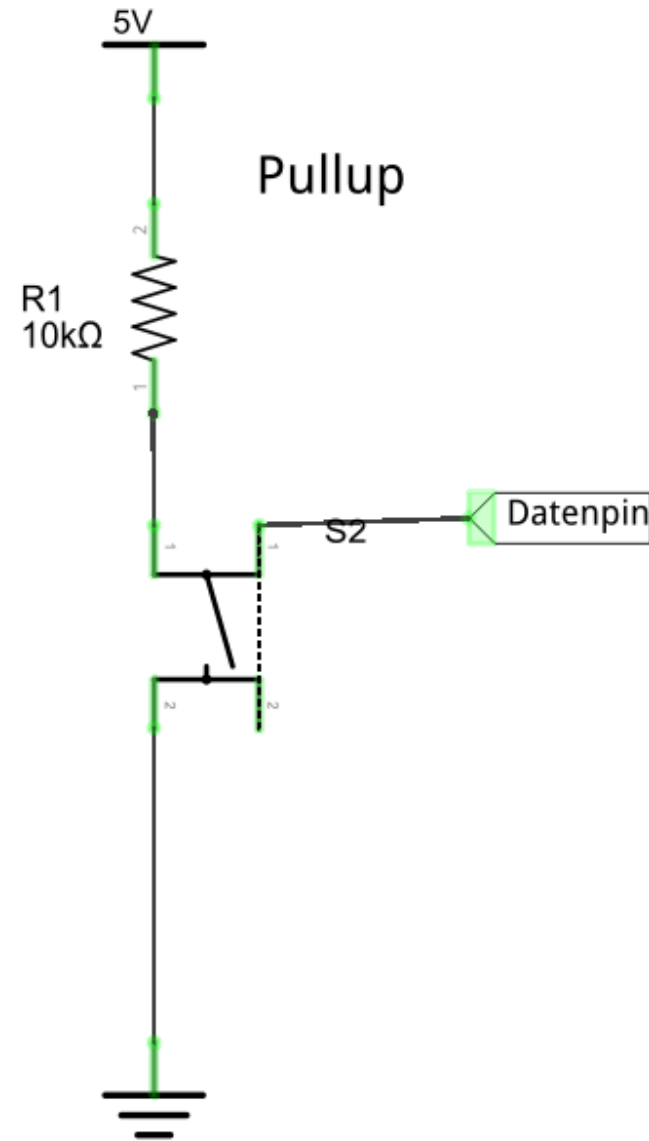
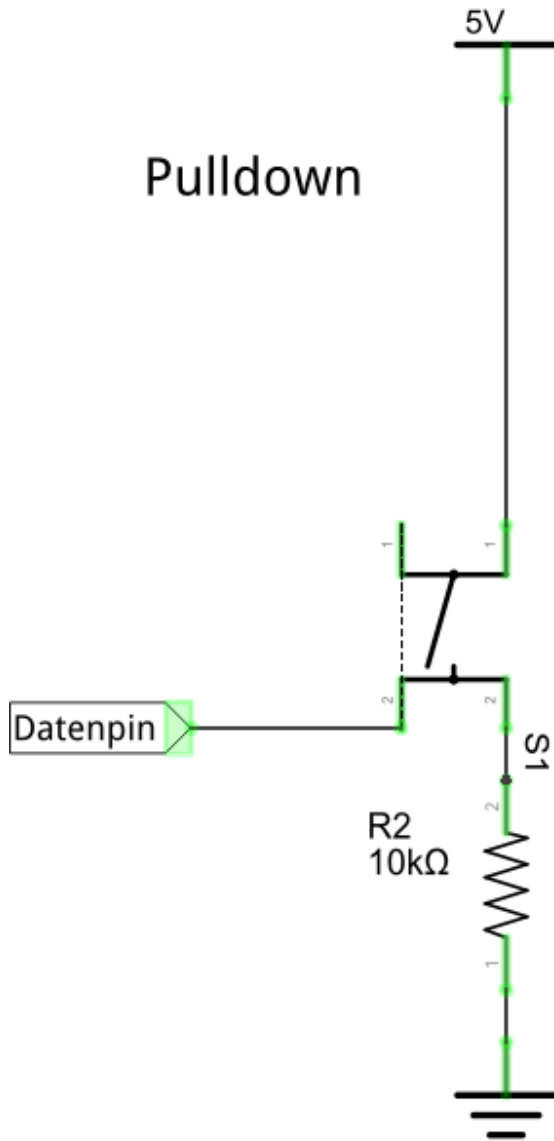
- **Warum?**
- Damit wir ein Signal eindeutig definieren
- Im schlimmsten Fall verfälschte Werte,
kann keiner vorhersagen

Pull-Down Widerstände



Pull-Up Widerstände





fritzing

Das Framework

- **Delay einfügen**
- Wollen wir zeitverzögert arbeiten, benötigen wir `delay(ms)`
- Beim Aufrufen blockiert der μC
- z.B zum LED blinken
- `ms` = Millisekunden `1s` = `1000ms`

Das Framework

- **Serial**
- Wenn wir mit dem PC kommunizieren wollen
- Z.b Text schreiben, Debug Ausgaben für die Systemkontrolle
- Baud unser Einheit (Symbole in der Sekunde)
Unser ESP8266 schafft locker 115200 Baud

Das Framework

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  Serial.write(45); // Sende ein Byte mit dem Wert 45  
  
  int bytesSent = Serial.write("hello"); // Sende die Zeichenfolge "Hallo" und gib die Länge der Zeichenfolge zurück.  
}
```

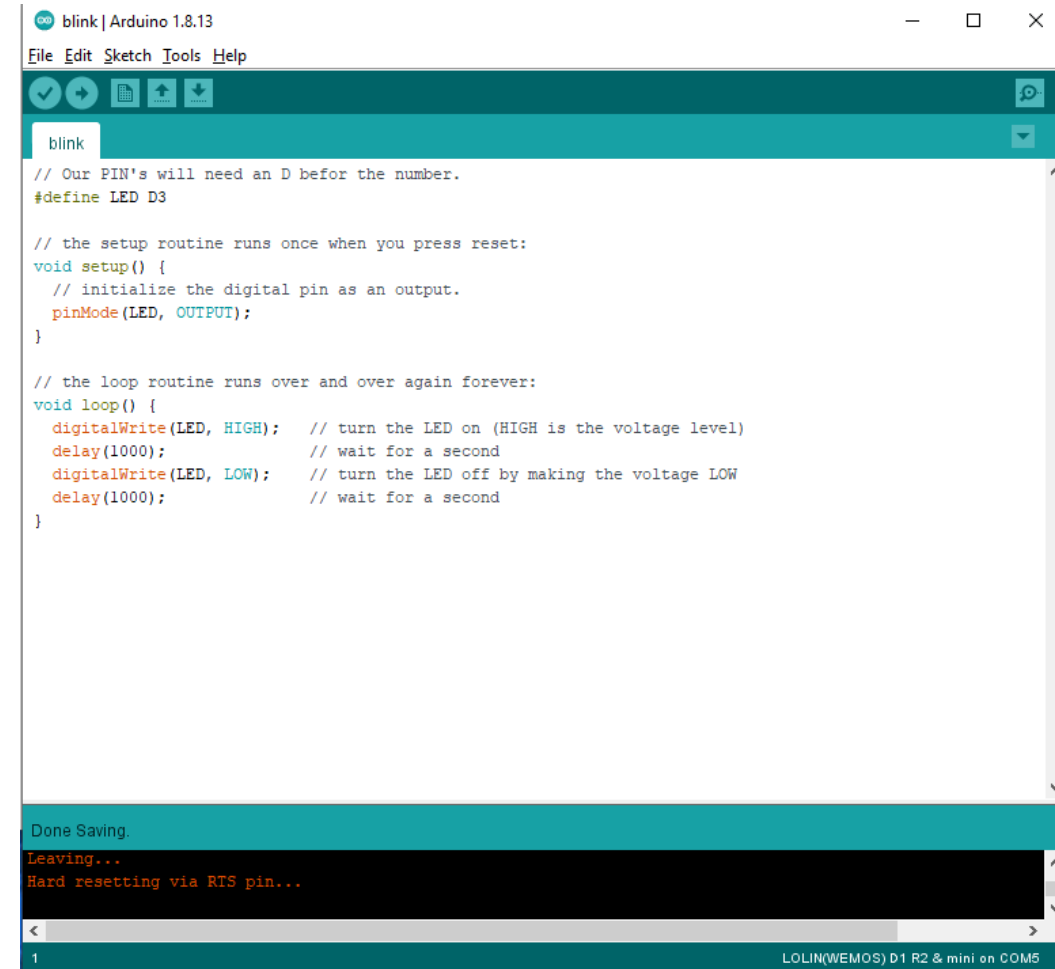
- **Baudrate**
- Anzahl Symbole pro Sekunde
- In diesem Fall 9600 Baud, wir benutzen 115200

Arduino Referenz

- Ausführliche Zusammenfassung vom Framework
- https://github.com/przygodyzkodem/Arduino-Cheat-Sheet/blob/master/Arduino_Cheat_Sheet_1-en.pdf
- <https://www.arduino.cc/reference/en/>

Blink LED

- Programmcode für die Blink LED
- 1000ms für ein Blinken
- An Pin 3 muss eure Anode von der LED
Die Kathode zu GND bzw. G



The screenshot shows the Arduino IDE interface with the 'Blink' sketch loaded. The code is written in C++ and is designed to blink an LED connected to digital pin 3. The code includes a preprocessor directive to define the LED pin, a setup function to initialize the pin as an output, and a loop function that turns the LED on and off with a 1000ms delay between each state change. The IDE window title is 'blink | Arduino 1.8.13'. The status bar at the bottom indicates the board is 'LOLIN(WEMOS) D1 R2 & mini on COM5'.

```
blink | Arduino 1.8.13
File Edit Sketch Tools Help

blink
// Our PIN's will need an D befor the number.
#define LED D3

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(LED, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(LED, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}

Done Saving.
Leaving...
Hard resetting via RTS pin...

LOLIN(WEMOS) D1 R2 & mini on COM5
```


Arbeitsaufgaben

– Für die Arbeitsaufgaben, bitte dem Link folgen


<https://docs.google.com/document/d/1crygGyjAOg8ejlXzY7S6yqlrE2T3OuCbwkW2H2NfKgo/edit?usp=sharing>

Zeit für ein paar Fragen

- Schreib mir eine Mail: malte.grave@uol.de
- Schreib ins Forum der Veranstaltung

Danke für eure Zeit!

- Freut euch schon auf komplexere Programme!



```
File Edit Sketch Tools Help
LaserCat BTHC05.cpp BTHC05.h VarSpeedServo.cpp VarSpeedServo.h
// Include Libraries
#include "Arduino.h"
#include "BTHC05.h"
#include "VarSpeedServo.h"

// Pin Definitions
#define BTHC05_PIN_RXD 10
#define BTHC05_PIN_TXD 11
#define LASER_PIN_S 2
#define SERVO9G1_PIN_SIG 3
#define SERVO9G2_PIN_SIG 4

// Global variables and defines
// object initialization
VarSpeedServo servo9g1;
VarSpeedServo servo9g2;
BTHC05 bthc05(BTHC05_PIN_RXD, BTHC05_PIN_TXD);

bool laserState = 0;
bool autoplayState = 0;

const int servoSpeed = 10;
const int seqIntervalDelta = 50;
const int manuallyServoMinStep = 5;
const int manuallyServoMaxStep = 20;
const int minimalRangeSize = 10;

// Change these parameters to define the rectangular play area
int servolMin = 80;
int servolMax = 110;
int servo2Min = 20;
int servo2Max = 50;

int servolpos = (servolMin + servolMax) / 2;
int servo2pos = (servo2Min + servo2Max) / 2;
```