IoT und Smart-Home im schulischen Kontext

ESP32 und Ardiuno IDE

ESP32

- WiFi 802.11 b/g/n
- BT und BLE
- Dual Core Microprozessor
- 34 GPIOS
- 12Bit ADC auf 18 Pins
- 2x 8Bit DAC
- 10x Touch Sensor

ESP32

- 4x SPI
- 2x I2C und I2S
- 3x UART
- CAN 2.0
- Motor PWM
- LED PWM
- Hall Sensor

Einsatzgebiete

- Low Power IoT Sensor/Data Logger
- Video Streaming
- Bild und Spracherkennung
- Heimautomatisierung
- Spielzeuge
- Wearables

Lolin D32

- ESP32-WROOM Modul
- 240 MHz Clock
- 4MB Flash Speicher
- 3.3V Pins
- LiPo Schnittstelle mit Lade-IC
- Kompatibel zum Arduino Framework und MicroPython

Lolin D32

- Analog In (VP, VN, 32, 33, 34, 35)
- Analog Out (25, 26)
- Touch (4, 0, 2, 15, 13, 12, 14, 27, 33, 32)
- Interner Temperatursensor
- Interner Hall-Sensor

Arduino Framework

- Hardware
  - Diverse Mikrocontroller Boards wie z.B. Uno, Nano, Lilypad etc..
- Software
  - Arduino IDE
  - „Programmiersprache“
    - C/C++ Dialekt
  - Kompatible Bibliotheken

Arduino Framework

- Void Setup()
  - Wird nur 1x aufgerufen
  - Initialisierung
    - Schnittstellen bereitstellen
    - DataDirectionRegister
    - Objekte erzeugen
    - Zustände festlegen

```
255  void setup() {
256    delay(3000); // 3 second delay for recovery
257    pinMode(D5, INPUT);
258    digitalWrite(D5, HIGH);
259    FastLED.addLeds<WS2811, DATA_PIN, GRB>(leds, NUM_LEDS).setCorrection(TypicalLEDStrip)
260    FastLED.setMaxPowerInVoltsAndMilliamps(5, 1800);
261    FastLED.setBrightness(90);
262    Serial.begin(115200);
263    Serial.println("Booting");
264    WiFi.mode(WIFI_STA);
265    WiFi.begin(ssid, password);
266    while (WiFi.waitForConnectResult() != WL_CONNECTED) {
267      Serial.println("Connection Failed! Rebooting...");
268      delay(5000);
269      ESP.restart();
270    }
271
272    // Port defaults to 8266
273    // ArduinoOTA.setPort(8266);
274
275    // Hostname defaults to esp8266-[ChipID]
276    ArduinoOTA.setHostname("baum");
```

Arduino Framework

- Void Loop()
  - Wird als Endlosschleife ausgeführt
  - Messungen durchführen
  - Zustand der Pin ändern
  - Auf Datenpakete warten

```
327  void loop() {
328    ArduinoOTA.handle();
329
330    if (!client.connected()) {
331      reconnect();
332    }
333    client.loop();
334
335    if (animation_enabled) {
336      if (cur_animation == RAINBOW) {
337        rainbow();
338
339      }
340      if (cur_animation == JUGGLE) {
341        juggle();
342      }
343      if (cur_animation == FIRE) {
344        fire();
345      }
346      if (cur_animation == CYCLON) {
347        if (cyclon_dir == 0) {
348          cyclon_i++;
349        }
350        if (cyclon_dir == 1) {
351          cyclon_i--;
352        }
353
354        if (cyclon_i == 300) {
355          cyclon_dir = 1;
356        }
357
358        if (cyclon_i == 0) {
359          cyclon_dir = 0;
360        }
361
362        leds[cyclon_i] = CHSV(hue++, 255, 255);
363
364      }
365      FastLED.show();
366      if (cur_animation == CYCLON) {
367        fadeall();
368      }
```

Arduino Framework

- Konstanten
  - HIGH: High Pegel = 3.3V, max 40 mA pro GPIO
  - LOW: Low Pegel = 0V
  - INPUT: Pin wird als Eingang genutzt
  - OUTPUT: Pin wird als Ausgang genutzt

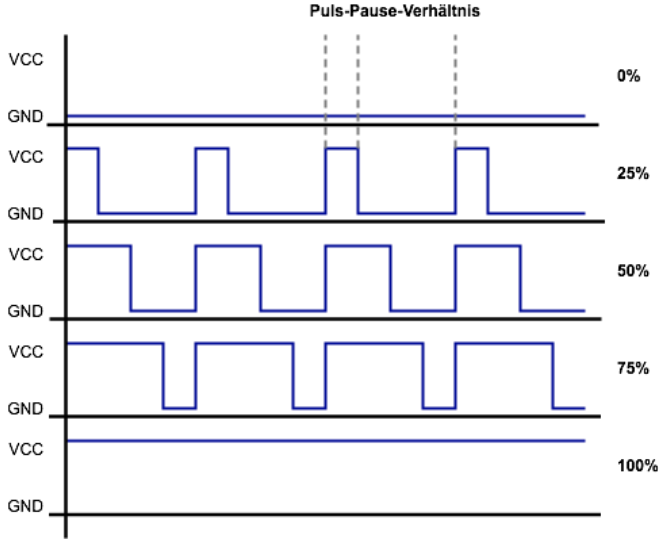
Arduino Framework

- Digital I/O
  - pinMode(**pin** / [INPUT, OUTPUT])
    - Setze Pin **pin** als Eingang (INPUT) oder Ausgang (OUTPUT)
  - digitalWrite(**pin**, [HIGH, LOW])
    - Setze den Spannungspegel an Pin **pin** auf HIGH oder LOW
  - int digitalRead(**pin**)
    - Lese den Spannungspegel an Pin **pin**.
      - 0 bis 0.825V LOW
      - 2.475V bis 3.3V HIGH

Arduino Framework

- Analog I/O
  - int analogRead(pin)
    - Da 12Bit ADC, kann der gesamte Spannungsbereich 0-3.3V auf insgesamt 4096 (2^12) diskrete Werte aufgeteilt werden
    - Angenommen analogRead gibt 500 zurück, dann beträgt die Spannung am Pin etwa 0.4V
    - 500/4095 = 0.4V/3.3V

Arduino Framework



- Analog I/O
  - analogWrite(pin,[0 - 255])
    - Pulsweitenmodulation
    - Funktioniert nicht beim ESP32!
  - dacWrite([25, 26], [0 - 255])
    - Geht leider nur an 2 Pins
  - LED PWM nutzen
    - ledcSetup(ledChannel, freq, resolution);
    - ledcAttachPin(LED_BUILTIN, ledChannel);
    - ledcWrite(ledChannel, dutyCycle);

Arduino Framework

- ledcSetup(ledChannel, freq, resolution);
  - Einmal initial ausführen
  - ledChannel
    - 16 unabhängige Channel
    - [0 - 15]
  - freq
    - Frequenz des PWM Signals
    - Max 78 kHz
    - 5000 Hz ist ein guter Wert

Arduino Framework

- ledcSetup(ledChannel, freq, resolution);
  - resolution
    - Wie viele diskrete Schritte soll es geben?
    - Auswahl zwischen 1 bis 16Bit Auflösung
    - 8Bit resolution = 256 Werte
- ledcAttachPin(pin, ledChannel);
  - 1x ausführen
  - pin wählen und den konfigurierten Channel festlegen

Arduino Framework

- ledcWrite(pin, [0 – 2^resolution-1])

```ino
#include <Arduino.h>

int freq = 5000;
int ledChannel = 0;
int resolution = 8;

void setup() {

  ledcSetup(ledChannel, freq, resolution);
  ledcAttachPin(LED_BUILTIN, ledChannel);

}
```

```ino
void loop() {

  for (int dutyCycle = 0; dutyCycle <= 255; dutyCycle++) {
    ledcWrite(ledChannel, dutyCycle);
    delay(7);
  }

  for (int dutyCycle = 255; dutyCycle >= 0; dutyCycle--) {
    ledcWrite(ledChannel, dutyCycle);
    delay(7);
  }

}
```

Arduino Framework

- Time
  - unsigned long millis()
    - Systemzeit in Millisekunden
  - unsinged long micros()
    - Systemzeit in Mikrosekunden
  - delay(zeit_in_ms)
    - Warte zeit_in_ms Millisekunden
    - Blockierend!

Arduino Framework

- Serial
  - Nützliche Statusmeldungen vom Mikrocontroller
  - Serial.begin(BAUDRATE);
    - BAUDRATE = Übertragende Symbole pro Sekunde
    - 115200 guter Wert für ESP32
  - Serial.println(„Hallo“);
    - Mikrocontroller sagt Hallo über USB-Schnittstelle
    - Serial Monitor kann Kommunikation anzeigen

Arduino Framework

- Hall Sensor
  - Messung von Magnetfeld
  - Einheit Tesla
  - int hallRead()
- Touch Sensor
  - int TouchRead(pin)

Weitere Frameworks

- Mongoose OS (JS)
- MicroPython (Python)
- ESP-IDF (C++)
- Lua RTOS (Lua)
- Pumbaa (Python)
- Mruby (Ruby)

Weitere IDE

- Atom + PlatformIO
- Eclipse
- Sloeber

Arbeitsaufträge

1.) LED blinken lassen

2.) Touch-Sensor testen. Wenn das Kabel am Pin berührt wird, soll der Mikrocontroller „Aua“ im Serial Monitor anzeigen.

3.) LED mit Hilfe des Touch-Sensors toggeln.

4.) Das Licht geht automatisch nach 30 Sekunden aus.

5.) Bonus: Statt ausschalten, LED ausfaden lassen.