Android + Arduino

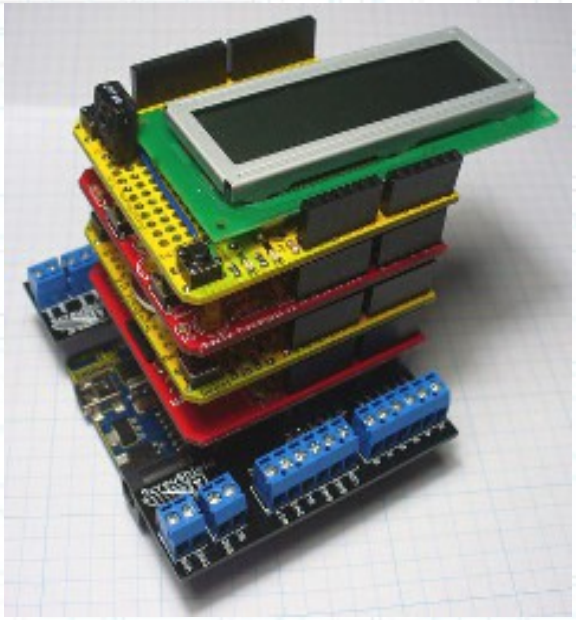
Hardware steuern mit Android

Entwicklertag Karlsruhe, 09.05.2012

Sebastian Wastl

Arduino

- Opensourceprojekt
- Auf Basis eines 8-Bit Mikrocontroller (ATMEL ATmega)
- Modular aufgebaut
  - Erweiterbar durch Shields

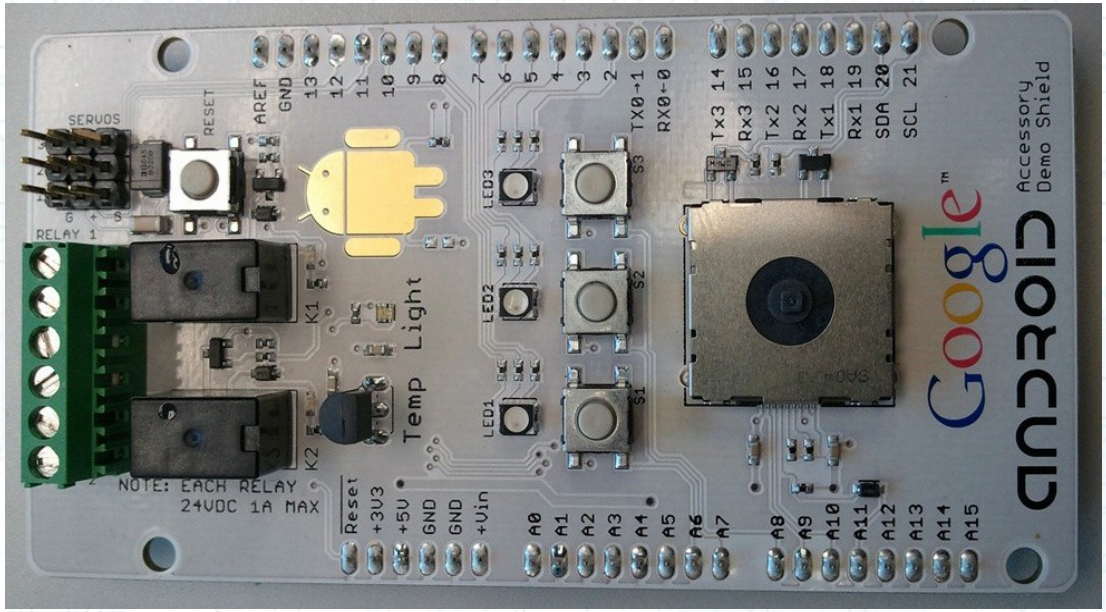


Hardwareeigenschaften ADK

- 8-Bit Mikrocontroller (ATmega 2560)
- 256kb Speicher
- 54 I/O Pins
- 15 analoge Ein- und Ausgänge
- 10 Bit AD-Wandler
- 4 serielle Ein- und Ausgänge
- 6 Hardware-Interrupts

Google Demo Board

- Demoapp „Demokit“ für das Demoboard
- 3 Multicolor LEDs
- 2 Relais
- Möglichkeiten der Steuerung von Servos
- Kapazitiver Sensor
- Joystick

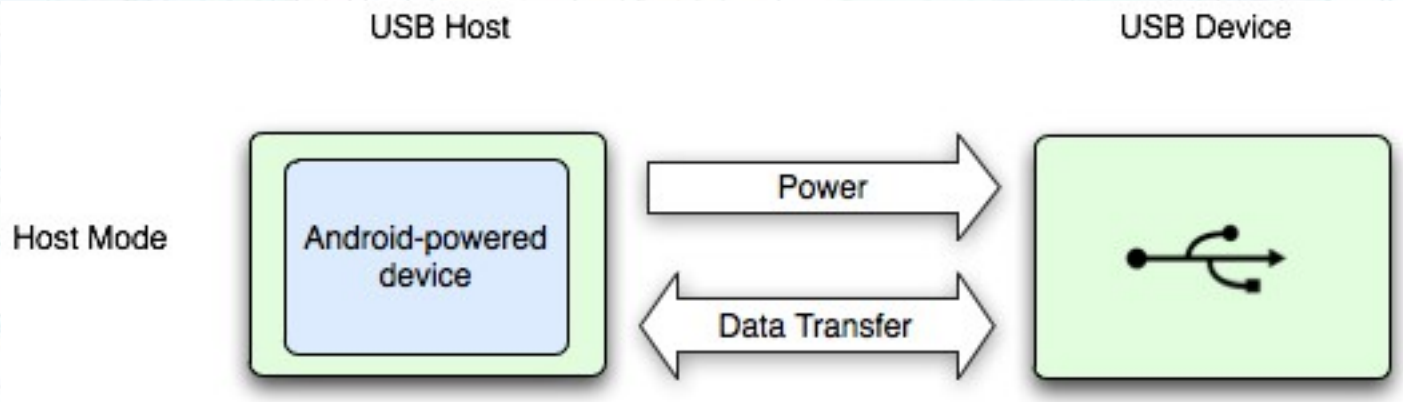


Anwendungsbeispiele

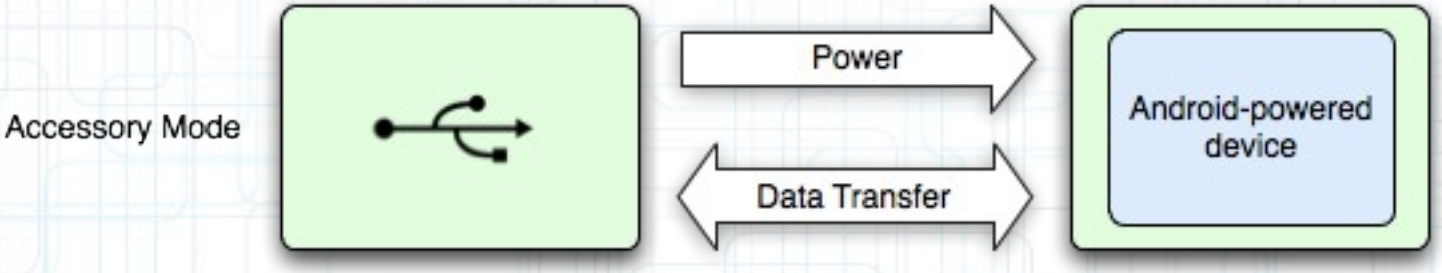
- Telemetrieerfassung
- Remotesteuerung von Geräten
  - Webserver auf Androidbasis
- Remoteerfassung von Daten
  - REST-Service

USB

- USB-Host-Mode
  - Unterstützung ab API-Level 12 (3.1)



- USB-Accessory-Mode
  - Unterstützung ab API-Level 10 (2.3.4)



Workflow der Entwicklung

1. Entwicklung der Hardware
2. Entwicklung der Arduino Firmware
3. Test der Ausgabe
4. Entwicklung der Android App (Kommunikation)
5. Entwicklung des Android-UI

Programmierung Arduino

- IDE basierend auf Processing
- Entwickelt für Künstler, Designer und Bastler
- Entwicklung in C/C++
- Libraries um Bitmanipulation zu minimieren
- Libraries für verschiedene Hardwarebausteine
    - Kapazitiver Sensor
    - 1-wire Sensor
    - SD-Karte
    - LCD-Display
    - ...

Programmierung Arduino

- setup() und loop() notwendig
- pinMode zum setzen von Aus- oder Eingang
- digitalWrite, digitalRead

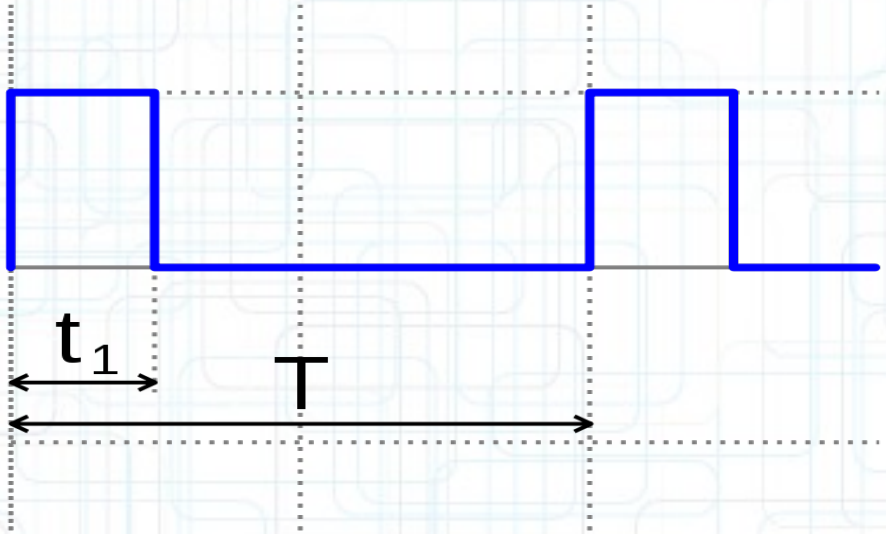
```
int ledPin = 10; // PIN 10
int lastStatus = LOW;

void setup() {
    pinMode(ledPin, OUTPUT); // Ausgang
}

void loop() {
    if (lastStatus == HIGH)
      lastStatus = LOW;
    else
      lastStatus = HIGH;
    digitalWrite(ledPin, lastStatus); // LED an
    delay(1000); // 1 Sekunde warten
}
```

Analoge Steuerung

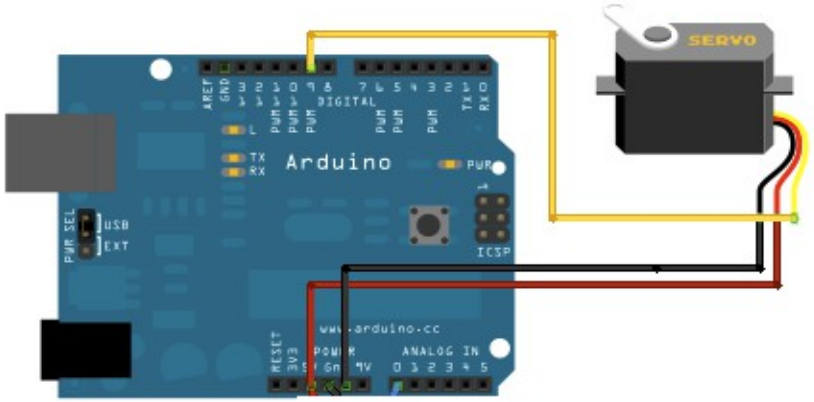
- Eingang: Spannungsmessung
- 8-Bit Pulsweitenmodulation (PWM)



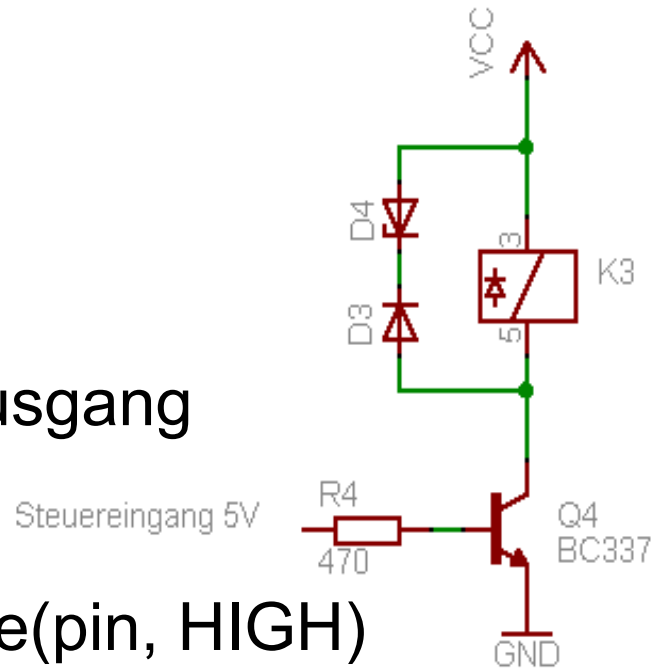
- Verwenbar um die durchschnittliche Spannung anzupassen → z.B. Dimmen von LEDs
- writeAnalog → 0-255

Steuerung von Relais und Servos

- Library zur Ansteuerung von Servos
- Ansteuerung mittels PWM
- servo.write(90) → Servomittelstellung



- Relaissteuerung über digitalen Ausgang
- Freilaufdiode notwendig
- digitalWrite(pin, LOW), digitalWrite(pin, HIGH)



Programmierung Android

- Platform API (API ≥ 12) oder Add-On API (API ≥ 10)
- Unterstützung von Hersteller abhängig
- <meta-data> mit Filter um die passende App via Intent zu starten
- Arduino sendet „Hersteller“, „Model“ und „Version“ an Android nach erfolgreichem Verbindungsaufbau

Android Security

- Kein Security Eintrag in der Manifest.xml notwendig
- Zugriff auf die USB-Schnittstelle muss vom User bestätigt werden
- Registrierung des USB-Event am BroadcastReceiver

```
private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (ACTION_USB_PERMISSION.equals(action)) {
        synchronized (this) {
          UsbAccessory accessory = UsbManager.getAccessory(intent);
          if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
            openAccessory(accessory);
          } else {
            Log.d(TAG, "permission denied for accessory " + accessory);
          }
        }
      }
    }
  }
};
```

Übertragungsprotokoll

- Übertragung von Bytes von Arduino an die Serielle Schnitstelle

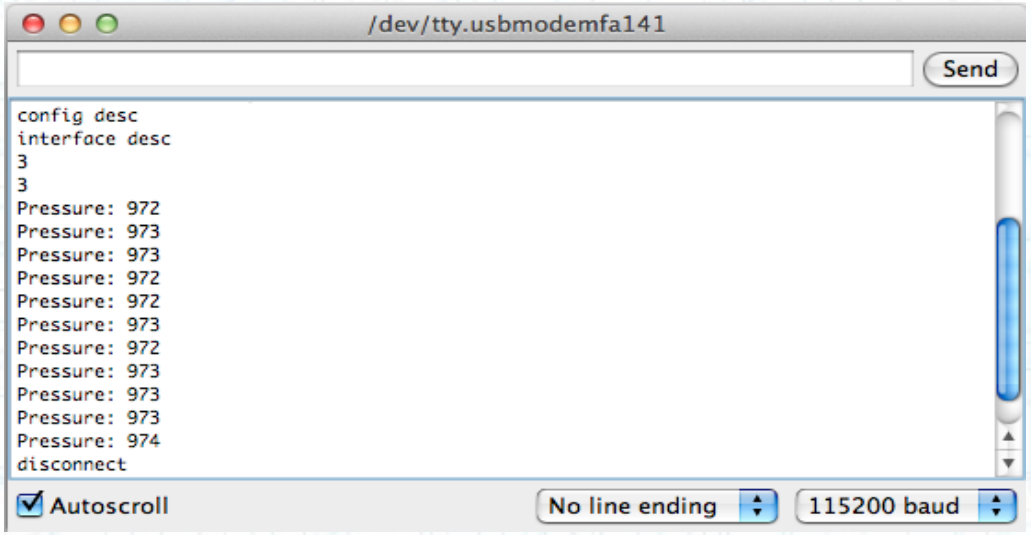
```
msg[0] = 0x1;
msg[1] = 0x1;
msg[2] = rpm >> 8;
msg[3] = rpm & 0xff ;
acc.write(msg, 4);
```

- Behandlung in Android (Java) als InputStream bzw. OutputStream

```
...
switch (buffer[i]) {
  case 0x1:
    if (len >= 4) {
      Message m = Message.obtain(mHandler, MESSAGE_RPM);
      RPMMsg rpm = new RPMMsg(composeInt(buffer[i + 2], buffer[i + 3]));
      m.obj = rpm;
      mHandler.sendMessage(m);
    }
    i += 4;
    break;
...
```

Debugging

- Arduino: Serial Monitor



- Android: USB mit Arduino belegt

  → Debugging über TCP/IP

  - $ adb tcpip 5555
  - $ adb connect 192.168.42.42:5555

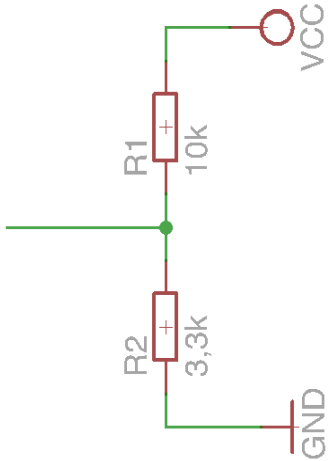
  - $ adb usb

RC-Telemetrie

- Erfassung von Messwerten
- Speicherung von Messwerten
- Nachträgliche Auswertung

RC-Heli-Telemetrie Hardware

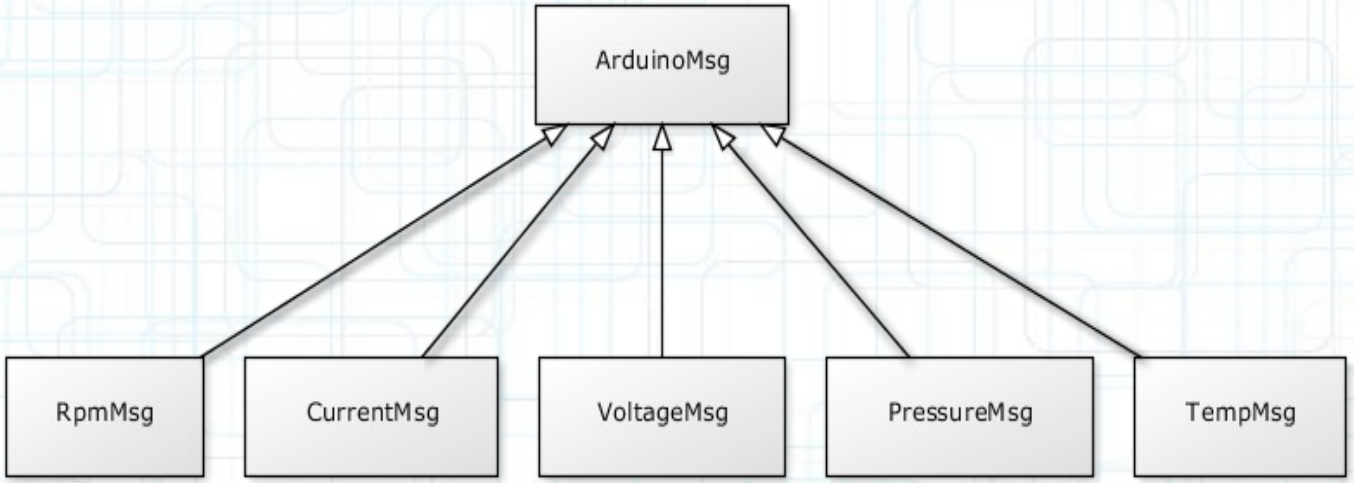
- Akkutemperatur
  - 1-wire-Sensor (DS1820)
- Akkuspannung
  - Spannungsteiler



- Drehzahlsensor
  - Messung über einen bipolaren Hallsensor
- Luftdruck (MPX4115a)
  - Spannungsmessung an analogem Eingang

Android-App

- Arduino: Größe 16kb
- Anzeigen der Daten
- Visuelle Darstellung der Daten



- Eigene Klasse je Sensortyp (Drehzahl, Temperatur, Strom, Spannung, Luftdruck)

Mögliche Erweiterungen

- Steuerung von Servos.
- Erfassung der momentanen Position über GPS.
- Erkennung der Lage im Raum über Beschleunigungs- und Drehratensensoren.

→ Eigenständige Steuerung im Raum möglich, mittels geeigneter Algorithmen (z.B. Kalman-Filter)

Demo

ANDROID

Fragen?

- www.arduino.cc
- www.arduino-shop.de

Sebastian Wastl
sebastian.wastl@arconsis.com