ArduSmartPilot mit WLAN-Kommunikation

Motivation

Die größte Herausforderung beim Fliegen eines ArduSmartPilots ist dessen geringe Funkreichweite von ca. 70 m. Der Grund hierfür ist die Kommunikation via Bluetooth (BT), insbesondere die geringe Sendeleistung im Smartphone mit ca. 4 dBm (2,5 mW), da hier ein BT „Class 2“ Funkmodul verbaut ist. Aus diesem Grund wurde am ArduSmartPilot mit dem HC-05 ebenfalls ein Class 2 Funkmodul eingesetzt.

Die in mobilen Endgeräten verbauten WLAN-Module haben generell eine wesentlich höhere Sendeleistung von typisch etwa 20 dBm (100 mW). Daher ist es naheliegend, die BT Funkkommunikation durch eine WLAN Verbindung zu ersetzen. Dabei ist der ArduSmartPilot ein WLAN Access Point, mit dem sich das Smartphone verbindet.

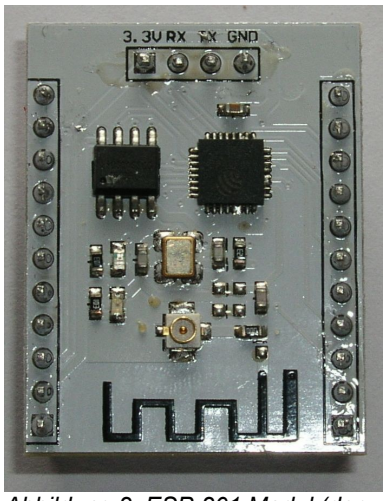


Abbildung 2: ESP-201 Modul (der ESP8266 µC ist der quadratische Chip).

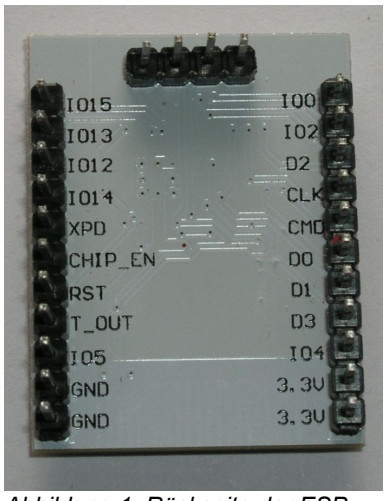


Abbildung 1: Rückseite der ESP-201 Moduls mit der Bezeichnung der Anschlüsse.

Im Sommer 2014 tauchte ein für dieses Vorhaben ideal geeignetes Modul auf dem Markt auf: **Der ESP8266 Mikrocontroller (µC) mit integrierter WLAN-Schnittstelle (ca. 20 dBm Funkleistung).**

Dabei handelt es sich um ein sogenanntes „**S**ystem **o**n a **C**hip“ (SoC), womit man die Integration aller Systemkomponenten wie Prozessor, Speicher und Funkschnittstelle auf einem gemeinsamen Chip bezeichnet. Inzwischen kann auch die Arduino IDE zum Programmieren des ESP8266 verwendet werden. Dies erleichtert die Programmierung erheblich, und es können große Teile des ArduSmartPilot-Programms unverändert übernommen werden.

Den ESP8266 gibt es auf verschiedenen Break Out Platinen, die zwischen 2 und 10 € kosten. Hier wird die in Abb. 1 und 2 dargestellte Variante „ESP-201[1]“ verwendet, da diese mit einer externen Antenne ausgestattet ist und drei Pins des µC zugänglich macht, die man als PWM-Ausgänge ansteuern kann. Dieses Modul wird ähnlich wie der Arduino Pro Mini über seine serielle Schnittstelle mit einem Foca oder FTDI Adapter programmiert.

Die Idee, den ESP8266 für den ArduSmartPilot zu verwenden, hatte Hr. Dr. Brandhorst, der über die Publikationen im Internet und im Make Magazin auf den ArduSmartPilot aufmerksam geworden war. Diese Publikation basiert auf seinen Arbeiten und Quellcode, die er freundlicherweise zur Verfügung gestellt hat. Herzlichen Dank an Hrn. Dr. Brandhorst!

1 Dieses Modul ist ähnlich dem „ESP-12“ aus dem Buch von N.Kolban [1]. Leider ist im Internet kein Schaltplan dieser Platine zu finden, was vielleicht das Funktionieren dieses Moduls ohne Pull Up bzw. Pull Down Widerstände erklärt.

Prof. Dr. rer. nat. Stefan Mack, Fakultät Technik / Mechatronik, Hochschule Reutlingen, Alteburgstraße 150, 72762 Reutlingen,
www.reutlingen-university.de, T. +49 (0)7121 271-7070, F. +49 (0)7121 271-7004, stefan.mack@reutlingen-university.de

Systemdesign

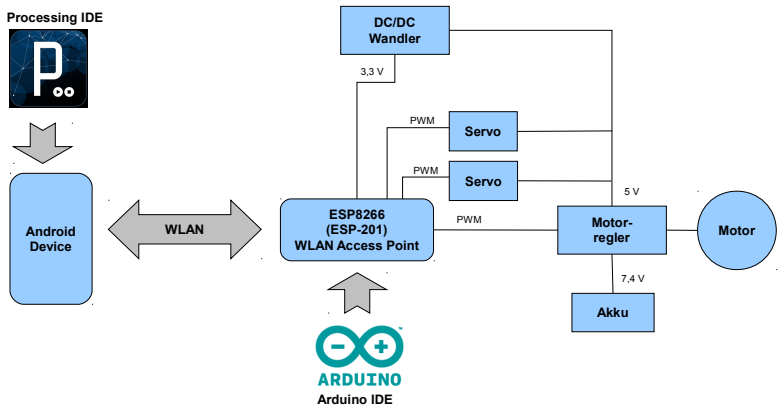


Abbildung 3: Systemdesign des ArduSmartPilot WLAN

Anders als bei der BT-Variante ist der µC und der WLAN-Funkbaustein des ESP8266 auf dem selben Chip untergebracht. Die Kommunikation mit dem Smartphone muss also nicht mehr wie beim Arduino Pro Mini den Umweg über die serielle Schnittstelle nehmen.

Genau so wie bei der BT-Variante steuern drei PWM-Ausgänge des µC den Motorregler (ESC) und die beiden Servos an. Unverändert ist auch die Versorgung der Servos über den 5 V Spannungsausgang des Motorreglers.

Der ESP-201 benötigt jedoch eine Versorgungsspannung von 3,3 V, die mit einem zusätzlichen DC/DC-Wandler[2] (Kosten ca. 2-3 €) aus der 5 V Spannung des Motorreglers generiert werden muss. **Eine Spannung von 5 V egal an welchen Pins führt mit hoher Wahrscheinlichkeit zur Zerstörung des ESP8266!**

Das komplette Systemdesign ist in Abb. 3 dargestellt.
Bei der Entwicklung der Hard- und Software bietet sich eine Steckbrettspannungsversorgung mit 3,3 V und 5 V an, wie sie in Abb 4 links am Steckbrett zu sehen ist.
Der ESP-201 agiert als WLAN Access Point, mit dem sich das mobile Endgerät (ohne Passwort) verbindet.

In den kostenlosen E-Book von Neil Kolban [1] werden der ESP8266 und dessen verschiedene Break Out Boards sehr gut erklärt.
Grundlegende Informationen zur zur sogenannten „TCP/IP“-Kommunikation mittels WLAN finden Sie z.B. in den Büchern von E. Bartmann [2] oder T. Igoe [3].

Hardwareaufbau

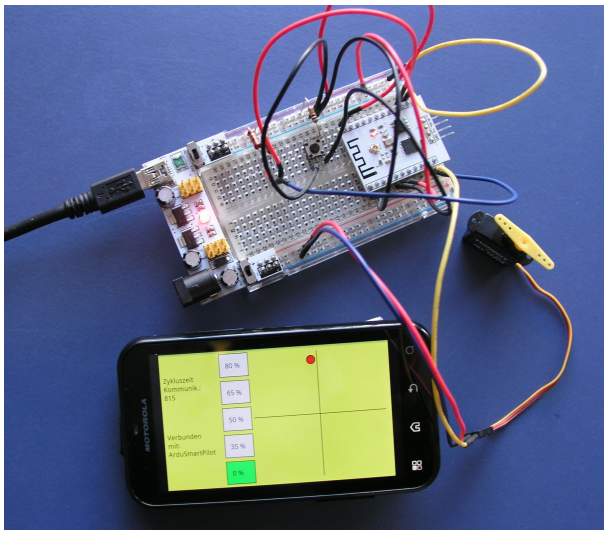


Abbildung 4: Betrieb mit einer Steckbrett-Spannungsversorgung (3,3 und 5 V).

In Tabelle 1 und 2 ist die Anschlussbelegung des ESP-201 für die Programmierung bzw. für den Betrieb angegeben. Die darin genannten 10 kΩ Pull Up und Pull Down Widerstände sind nicht zwingend erforderlich, werden aber laut Datenblatt des ESP8266 empfohlen, um undefinierte Zustände an den Eingängen und eine noch höhere Stromaufnahme zu vermeiden.

Den Hardwareaufbau baut man am besten zuerst auf einem Steckbrett auf (siehe Abb. 4). Dabei ist zu beachten, dass die Steckbrett-Spannungsversorgung einen genügend hohen Strom liefern kann: Der ESP8266 kann Ströme bis ca. 200 mA ziehen. Der Betrieb des ESP-201 mit dem Foca- oder FTDI-Adapter mit aktiviertem WLAN ist aus diesem Grund instabil. Zum Programmieren liefern diese beiden Adapter jedoch ausreichend Strom.

2 Kosten ca. 2 €. Die Ausgangsspannung muss im Bereich 3,0 – 3,6 V einstellt werden, siehe Datenblatt ESP8266. Preiswerte DC/DC Wandler wie der hier verwendete, haben einen Mindestspannungsabfall von fast 2 V, weshalb hier der ESP mit einer Versorgungsspannung von 3,1 V betrieben wird.

Die PWM-Ausgänge werden am besten einzeln mit einem Servo getestet, ebenfalls damit die Strombelastung nicht zu hoch wird. Meistens ist aber gar nicht die Steckbrettspannungsversorgung für einen Spannungseinbruch verantwortlich: Manche USB Anschlusskabel haben derart kleine Adernquerschnitte, so dass der Spannungsabfall nicht vom Spannungsregler auf dem Adapter sondern vom USB-Kabel verursacht wird!

Auf dem ESP-201 ist eine Platinenantenne vorhanden. Über den U.FL Stecker kann mit einer externen Antenne die Reichweite etwas vergrößert werden. Dieser Effekt ist aber nicht sonderlich hoch.

Pin am ESP-201...	**… wird laut Datenblatt zum Programmieren belegt mit:**
IO0	Direkt auf GND
IO2	Über 10 kΩ Pull Up Widerstand auf 3,3 V [3] [4]
IO15	Über 10 kΩ Pull Down Widerstand auf GND [3]
CHIP_EN bzw. CHIP_PD	Direkt auf 3,3 V
RST (Reset)	Über 10 kΩ Pull Up Widerstand auf 3,3 V, bei Reset kurz direkt auf GND oder floatend (=unbeschaltet) Dann Reset beim Einschalten.
3,3 V	Direkt auf 3,3 V (Stromversorgung)
RX	TX (Foca/FTDI auf 3,3 V einstellen!)
TX	RX (Foca/FTDI auf 3,3 V einstellen!)
GND	Direkt auf GND (Stromversorgung)

Tabelle 1: Belegung der Anschlusspins des ESP-201 für die Programmierung.

Pin am ESP-201...	**… wird laut Datenblatt im Betrieb belegt mit:**
IO0	Direkt auf 3,3 V [4]
IO2	Über 10 kΩ Pull Up Widerstand auf 3,3 V [3] [4]
IO15	Über 10 kΩ Pull Down Widerstand auf GND [3]
CHIP_EN bzw. CHIP_PD	Direkt auf 3,3 V
RST (Reset)	Floatend (=unbeschaltet). Dann automatisch Reset beim Einschalten.
3,3 V	Direkt auf 3,3 V (Stromversorgung)
GND	Direkt auf GND (Stromversorgung)
IO13	Servo Höhenruder (PWM)
IO12	Servo Seitenruder (PWM)
IO14	Eingang ESC Motorregler (PWM)

Tabelle 2: Belegung der Anschlusspins des ESP-201 für den Betrieb.

Programmieren („Flashen“)

Bei der Arduino IDE (Version 1.6.5 oder neuer) muss zuerst der ESP8266 als neue Platine hinzugefügt werden: Dazu sind folgende Schritte notwendig:

1) In der Arduino IDE unter „*Datei -> Voreinstellungen -> Additional Boards Manager URLs:*“ den Link „*http://arduino.esp8266.com/staging/package_esp8266com_index.json*“ eingeben.

2) In der Arduino IDE wählen: „*Werkzeuge -> Platine -> Boards Manager -> esp8266 installieren*“

3 Die Pull Up bzw. Down Widerstände sind nicht zwingend nötig, siehe Abschnitt Hardwareaufbau.

4 Kann auch floaten (= unbeschaltet). Jedoch dann störanfälliger z.B. bei Berührung der Platine oder elektromagnetischer Einstrahlung.

Für das Programmieren des ESP-201 sind die in Abb 5 dargestellten Einstellungen bei der Arduino IDE vorzunehmen.

Für das Übertragen des Programms ist ein USB-Seriell-Wandler wie z.B. der Foca- oder der FTDI-Adapter nötig. Der Aufbau (siehe Abb. 6) hierfür erfolgt auch am besten auf einem Steckbrett, wobei bei einem guten USB-Kabel die 3,3 V Spannungsversorgung durch den Adapter ausreicht. Anders als beim Arduino Pro Mini wird der ESP beim Herunterladen des Programms nicht automatisch in den Programmiermodus versetzt und erhält auch keinen Resetbefehl.

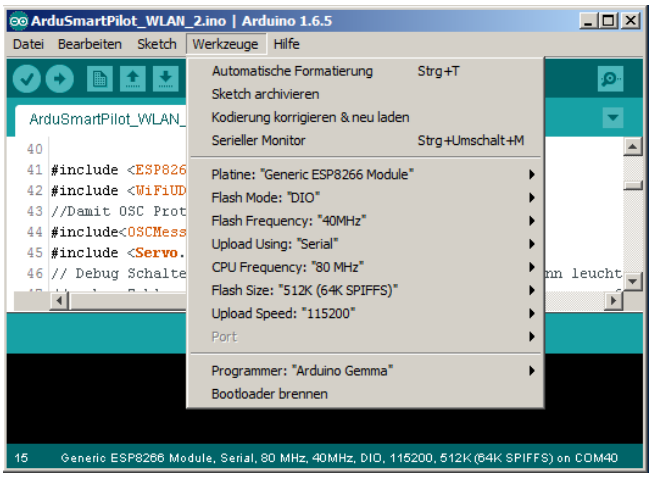


Abbildung 5: Einstellungen der Arduino IDE zur Programmierung des ESP8266.

Ist der RST Pin nicht angeschlossen (floatet), so führt der ESP8266 nach dem Einschalten der Spannungsversorgung automatisch einen Reset aus, was durch ein kurzes Aufblinken der blauen LED angezeigt wird. Ansonsten mus der RST-Pin kurz auf GND gelegt werden.

Die Kommunikation mittels WLAN verwendet das UDP Protokoll, welches über die Bibliothek `WiFiUDF` implementiert wird. Zusätzlich wird (nur) für den Empfang der Steuerdaten das OSC-Protokoll verwendet, das über die Bibliothek `OSCMessage` aus `esp8266-OSC-master` eingebunden wird (www.github.com/sandeepmistry/esp8266-OSC). Man kann das Empfangen aber auch ohne OSV nur via UDP programmieren.

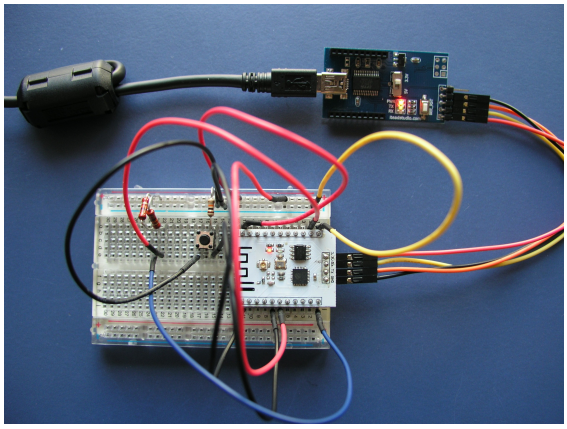


Abbildung 6: Aufbau zum Programmieren ("Flashen") des ESP-201 mit einem Foca-Adapter.

Die Android App verwendet das OSC-Protokoll sowohl zum Senden der Steuerdaten als auch zum Empfangen der Arduinodaten. Dafür werden die Bibliotheken `oscP5` und `netP5` verwendet.

Damit die Android App mit dem ESP-201 kommunizieren kann, muss sich das Endgerät vorher mit dem WLAN des ESP-201 verbinden. Der Namen dieses WLANs ist im Arduinoprogramm festgelegt, ein Passwort ist für den Verbindungsaufbau nicht nötig.

Informationen zum Umgang mit der OSC-Bibliothek im Processing-Androidprogramm finden sich im Buch von D. Sauter [4].

Praktische Erfahrungen

In der Praxis zeigte sich, dass man von der korrekten Beschaltung nach Datenblatt abweichen darf und ohne größere technischen Probleme die Minimalvarianten (siehe Abb. 7 und 8) verwenden kann.

In Abb. 10 und 9 ist eine Platine mit dem ESP-201, dem DC/DC-Wander sowie den Anschlusspins für die zwei Servos und den Motorregler zu sehen. Diese Platine wird am ArduSmartPilot mit Klettpunkten befestigt.

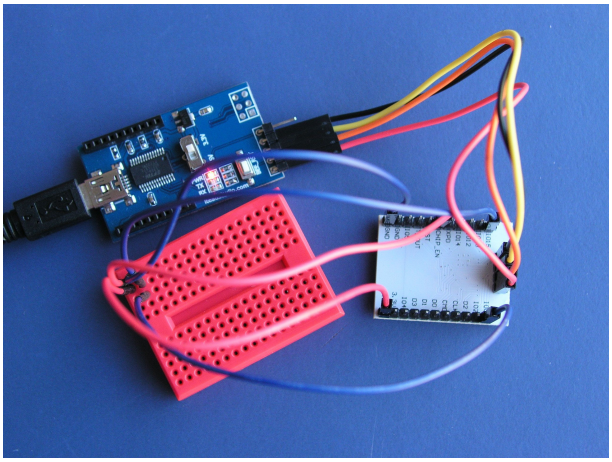


Abbildung 7: Minimale Beschaltung des ESP-201 für das Programmieren.

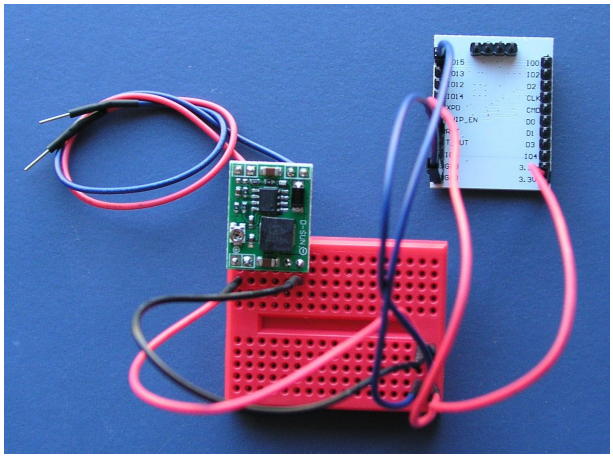


Abbildung 8: Minimale Beschaltung des ESP-201 für den WLAN-Betrieb.

Bei Testflügen zeigte sich eine Reichweite von über. 200 m.

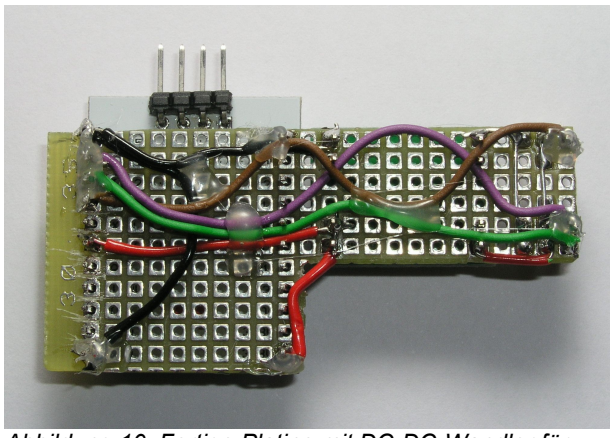


Abbildung 10: Fertige Platine mit DC-DC-Wandler für den ArduSmartPilot (Rückseite).

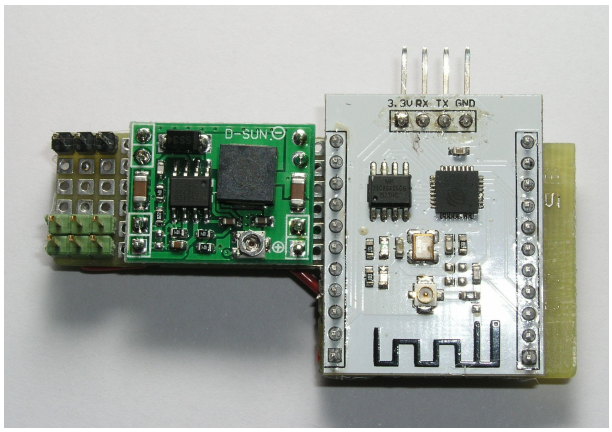


Abbildung 9: Fertige Platine mit DC-DC-Wandler für den ArduSmartPilot.

Ausblick

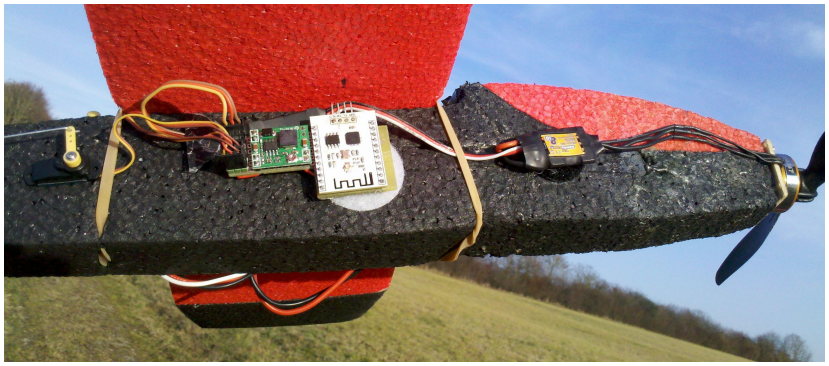
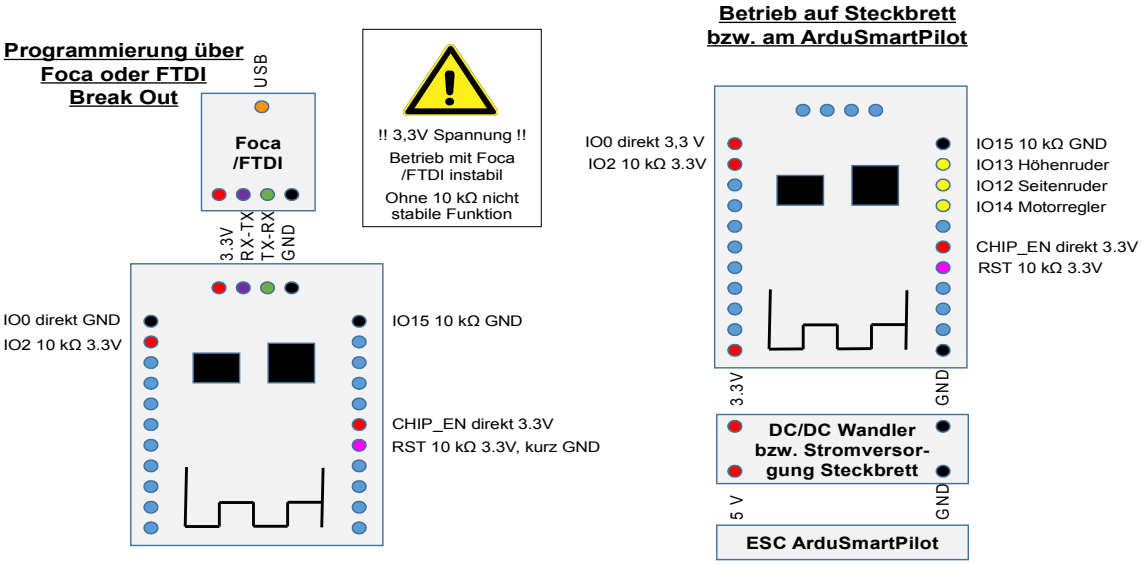


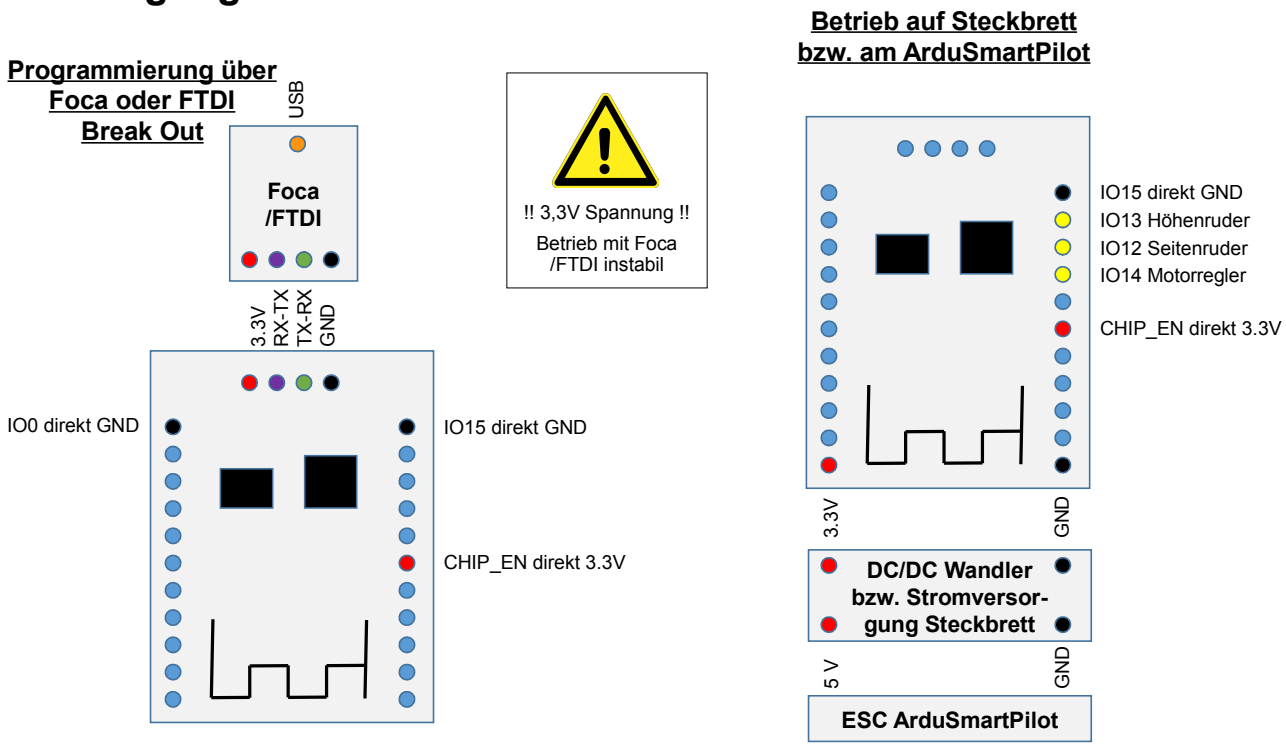
Abbildung 11: Flugfertiger ArduSmartPilot (der Akku befindet sich auf der gegenüber liegenden Seitenfläche).

Bei den Schülerprojekten stellte sich heraus, dass viele Schüler über ein Apple Mobilgerät statt über ein Androidmobilgerät verfügen. Leider kann die Processing IDE nur für das Programmieren von Android Apps verwendet werden.
In einer zukünftigen Version könnte der ESP8266 mit einem Webserver ausgestattet werden. Dann kann auf diesem Webserver über einen Browser unabhängig vom Betriebssystem des Smartphones zugegriffen werden. Die damit im Browser dargestellte Java Script Webseite würde dann über entsprechende Buttons den ArduSmartPilot steuern. Eine ArduSmartPilot-App auf dem Smartphone wäre dann nicht mehr nötig.
Hierzu muss jedoch eine ESP8266-Variante mit einem größeren Speicher verwendet werden.

Pinbelegung nach Datenblatt ESP8266



Pinbelegung: Minimalversion



Literaturverzeichnis

1: Kolban, Neil, Kolban's Book on ESP8266, 2015, neilkolban.com/tech/esp8266/

2: Bartmann, Erik, Die elektronische Welt mit Arduino entdecken, 2014

3: Igoe, Tom, Making Things Talk, 2011

4: Sauter, Daniel, Rapid Android Development, 2013