Elektronische Steuerung für F1E

Teil 2 - Fortsetzung aus Heft 1/22 / Paul Seren

Motivation

Im Teil 1 über meine elektronische Steuerung für F1E hatte ich im Fazit erwähnt, dass auf dem Weg dorthin eine Menge an Neuem zu entdecken gab und so nebenbei ein weiteres modelflugnahes Hobby entstanden ist.
Zwar hatte ich mich in den letzten Jahren schon immer wieder mit Elektronik beschäftigt: Von den Anfängen in der Jugend mit dem ersten zusammengebratenen Detektor-Empfänger, bis zu selbst gelöteten Elektronik-Bausätzen im Umfeld der RC-Modellfliegerei.
Vor einigen Jahren habe ich für mich die programmierbaren Mikroprozessoren entdeckt, mit denen sich auf erstaunlich einfache Weise und dazu noch günstige schnelle Erfolgserlebnisse entstehen: Waren es in meiner Jugend noch die unerschwinglichen Elektronikbaukästen von KOSMOS oder PHILIPS, sind es heute universelle Mikroprozessoren für ein paar wenige Euro.
In Kombination mit dem eigenen Rechner und der obligatorischen USB-Schnittstelle sind schnell einfache bis komplexe Lösungen entwickelbar. Unsere heutige vernetzte Welt gibt uns hierbei eine mannigfaltige Unterstützung: Unmengen an Ideen, Software-Bibliotheken und Programmbeschreibungen finden sich im Internet. Foren rund um die Robotik bilden eine sehr virtuelle Gesprächs- und Informationsbasis. Es gibt kaum etwas, was nicht schon in irgendeiner Form schon mal angedacht wurde.

Einstieg

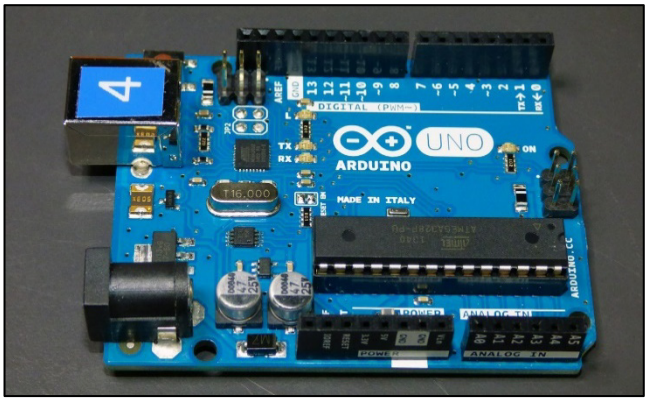
Für den Einstieg in die Programmierung von Mikroprozessoren für unsere Modellbauzwecke benötigt man erstmal nur wenige Dinge. Man muss noch nicht mal löten können oder gar viel Geld ausgeben:
Für unsere Zwecke reicht:

- ein normaler PC/ein Notebook (hat man ja sowieso)
- eine Experimentierplatine (Steckboard) und Steckkabel, ca. 10,- €
- ein Mikroprozessor, z.B. aus der Arduino Entwicklungs-Serie, ca. 10,- €
- ein passendes USB-Kabel für den Mikroprozessor (Micro, Mini oder Sub C). Liegt meistens ja schon im Haushalt irgendwo rum oder wird mit dem Mikroprozessor mitgeliefert.
- Die Software-Entwicklungsumgebung für den Mikroprozessor, ca. ziemlich genau 0,- €, da „Open Source" und damit frei verfügbare Software.
- Eine handelsübliche Rudermaschine/ein Servo. Liegt irgendwo in einer der Schubladen im Keller oder kann aus einem Modell ausgebaut werden. Ggf. 5,-€ Investition bei Ebay & Co.
- Eine Idee, welche man mit dem Mikroprozessor und dem Servo ausprobieren möchte. (Meine konkrete Idee war eben die Umsetzung einer elektronischen Magnet-Steuerung für F1E)
- Ein bisschen Lust, Laune und Zeit, sich mit einfacher Software-Programmierung zu beschäftigen

Was ist was und wofür? Die nächsten Zeilen dienen einer kurzen Übersicht der in diesem Kontext auftauchenden Fachbegriffe und deren Bedeutung:

Arduino

Das wird uns nun öfter begegnen: Im Grunde handelt es sich dabei um eine Mikrocontroller-Produktreihe auf einer schon fertig konfektionierten Eingabe/Ausgabe-Platine, bzw. -Board. Je nach Bauart ist dieses Board mit einer USB-Schnittstelle/-Verbindung zur dazugehörenden Software-Entwicklungsumgebung (Arduino IDE) aufgebaut. Die unterschiedlichen Bauarten unterscheiden sich in Größe, Anzahl von Programmspeicherplätzen und Anzahl der Eingabe-/Ausgabe-Pins.
An die Eingabe-Pins können Sensoren (Zum Beispiel ein Schalter, ein Drucksensor, ein Einstellregler, eine Funkbremse, etc.) als Input angeschlossen werden. An die Ausgabe-Pins können wiederum Aktuatoren/Ausgabeeinheiten (LEDs, ein Servos, Summer, ein weiterführende Schaltung, Relais, etc.) angeschlossen werden.
Mein wesentlicher Eingabe-Sensor für meine Idee war der Magnetfeldsensor, der Ausgabe-Aktuator natürliche das Servo zur Ansteuerung des Ruders.



Arduino Uno

Arduino IDE

Das ist nun unser eigentlicher „Arbeitsplatz": Es handelt sich um die kostenfreie (!) Programmier-Software. Mit dieser entwickeln wir die Programme für unseren Arduino-Mikroprozessor und übertragen die Programme über die USB-Schnittstelle des Rechners/Notebooks/PC auf den jeweiligen Arduino.
Über die serielle USB-Schnittstelle können wir während des Programmablaufs auch die Print-Ausgaben beobachten, sofern wir diese im Programm eingebaut haben. Diese IDE verfügt über sehr viele Schnittstellen zu den unterschiedlichen Arduino-Boards, diese können beliebig um neue Arduino-Ableger erweitert werden. Für die unterschiedlichen Sensoren können die dazugehörigen

Programm-Bibliotheken eingebunden werden. Eine Vielzahl von einfachen Programmierbeispielen ist in der Arduino IDE schon verfügbar, mit jeder Einbindung weiterer Bibliotheken erweitert sich das Beispiel-Angebot. Die grundsätzliche Programmiersprache ist „C" – ähnlich.

Bibliotheken

Für die unterschiedlichsten Arduino Anwendungen gibt es eine schier unüberschaubare Anzahl von fertigen Software-Paketen mit einer Vielzahl von Beispielen. Robotik-Foren und aus dem RC-Bereich bieten vieles von dem, was wir für Anwendungen im Freiflugbereich benötigen können. Die Suche lohnt sich. Gerade durch die Menge an Beispielen, wie man einen bestimmten Sensor auswertet, den internen Timer nutzt, eine oder mehrere Servos ansteuert oder irgendetwas piepsen oder leuchten lässt passt sehr gut für Projekte im Modellflugbereich.

Programmiersprache C (C#, C++)

Beispiel_Thermiksense

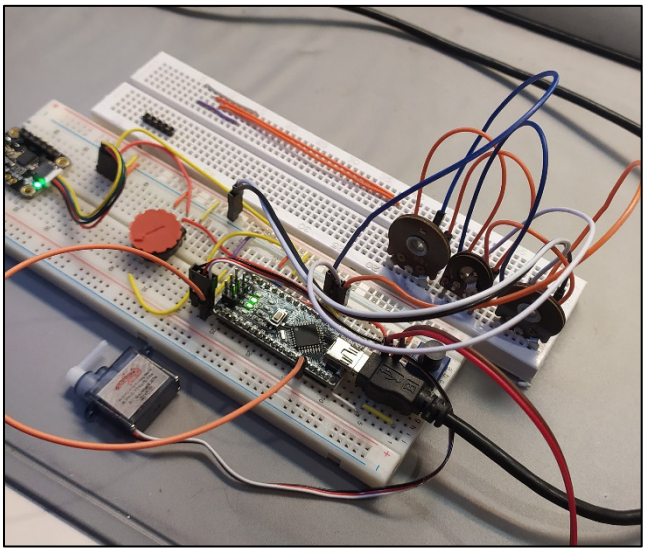
```
// Die setup-Funktion läuft nach Programmstart einmal.
// Hier werden die Grundeinstellungen getroffen.

void setup() {
  // PIN 13 des angeschlossenen Arduino wird als Output-Pin festgelegt
  pinMode(13, OUTPUT);
}

// Die loop-Funktion läuft in einer Endlosschleife - immer wieder....

void loop() {
  digitalWrite(13, HIGH); // Pin 13 bekommt den Zustand "An" - eine angeschlossene LED würde leuchten.
  delay(1000);            // Warte 1 Sekunde (Die Angaben sind in Milli-Sekunden)
  digitalWrite(13, LOW);  // Pin 13 bekommt den Zustand "Aus" - - eine angeschlossene LED würde dunkel.
  delay(1000);            // Warte wieder 1 Sekunde
                          // das ganze wiederholt sich, bis der Strom abgeschaltet wird.
}
```

C ist eine der weitesten verbreitete Programmiersprache, die in den 1970er Jahren entwickelt wurde. Viele Betriebssysteme (Unix, Linux, …) sind in dieser Sprache programmiert. Zahlreiche weitere Programmier-Sprachen, wie C++, Objective-C, C#, D, Java, JavaScript, LSL, PHP oder Perl, orientieren sich an der Syntax und anderen Eigenschaften von C.
Also: Wer sich C als Programmiersprache aneignet, kann diese Kenntnisse schnell auf andere Sprachen anpassen. Es lohnt sich also mit dem Arduino mit jedem geglückten Beispiel diese Sprache ein Stück weiter zu lernen.



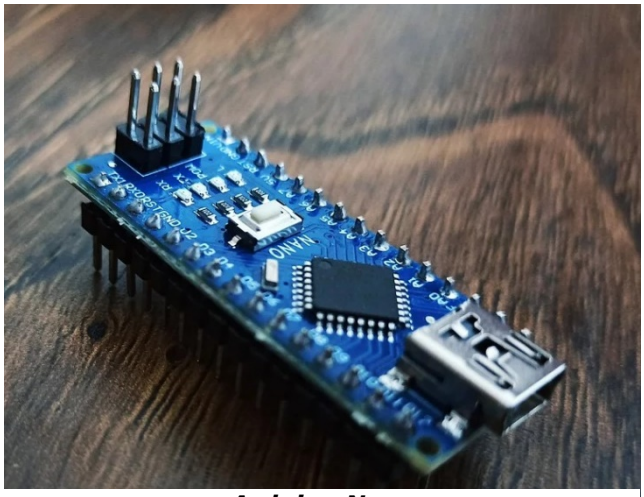
Steckplatine

Experimentierplatine/Steckplatine

Um unseren Arduino-Mikroprozessor mit Sensoren, zusätzlichen elektronischen Bauelementen oder einer Rudermaschine zu verbinden hilft eine Steckplatine. Hier kommen wir dann ohne Lötarbeiten nur mit Kabeln und den steckbaren elektronischen Bauteilen schnell zu einem Prototypen, bei dem man schnell was umstecken und anpassen kann.

Arduino – Familie

Die zahlreichen Varianten von Arduinos sind mittlerweile zu einer Großfamilie gewachsen. Daher hier nur eine



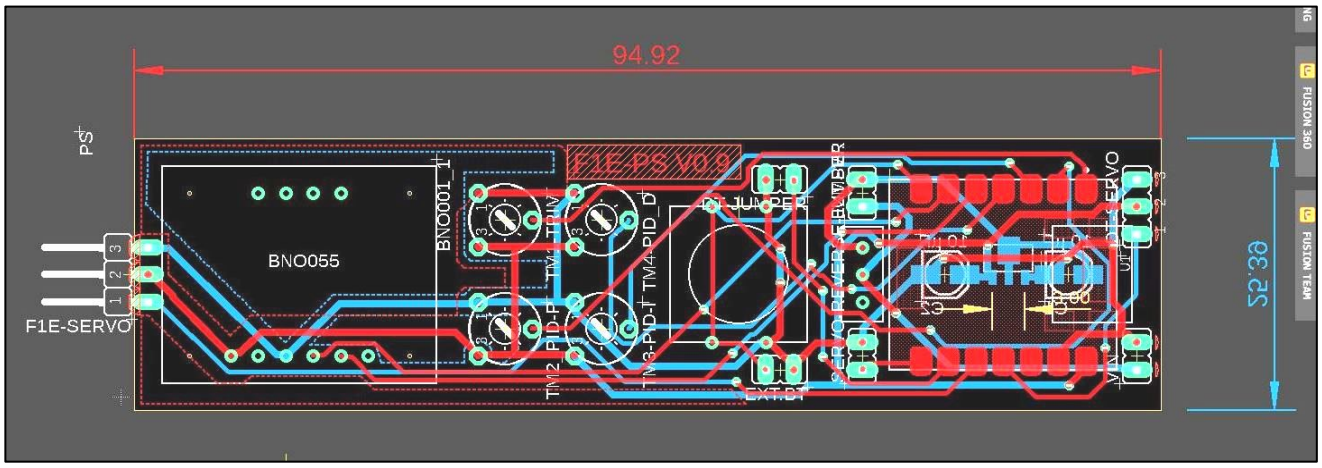
Arduino Nano

kleine Übersicht der Möglichkeiten:

- Arduino Uno: Der „Ur-Vater" aller Dinge. Etwas groß und sperrig für unsere Anwendungen im Freiflug, aber ein kostengünstiger Einstieg in die Welt der Arduino-Programmierung. Sollte in keiner Elektronik-Sammlung fehlen.
- Arduino Nano: der kleine Bruder vom Uno: Klein und leicht (passt in jeden F1A-Rumpf locker hinein), viele Speicherplätze und ausreichend schneller Mikroprozessor. Aktuell sehr schwankende Preise, aber alles unter 10 €. Gehört ebenfalls in die Grundausstattung der neuen Elektronik-Werkstatt.
- Seeduino: Ein neuerer Vertreter in der Szene – noch kleiner als der Nano, zwar ein paar Eingangs/Ausgangs-Pins weniger als der Nano, dafür mehr Speicherkapazität für größere Programme. Meine Wahl bei der Auslegung der F1E- Steuerung.

Eagle

Wenn wir aus unserer Experimentier-Schaltung einen Schritt in Richtung einer Elektronik-Platine weitergehen wollen, können wir mit diesem ebenfalls kostenfreien Programm aus unserer funktionierenden elektronischen Schaltung das Platinen-Layout grafisch und programmgestützt gestalten.
Für die üblichen Elektronik-Bauteile sind lauter Grafik-Bibliotheken integriert, weiter können wiederum aus dem Internet dazu geladen und mit der Elektronik verknüpft werden.



Platinenlayout in Eagle erstellen

Die Software hilft die Pins der elektronischen Bauteile sinnvoll auf Ober- und Unterseite einer Platine zu Planen und in die Fertigung zu geben.

Eurocircuit

Dies ist ein Anbieter, welcher aus dem Platinen-Entwurf die Platinen ätzt. Die Layout-Dateien im Austauschformat vom Programm Eagle werden auf diese Online-Plattform hochgeladen.
Nach einer automatisierten Prüfung mit Fehlerprotokollen/Änderungsvorschläger erfolgt noch eine manuelle Überprüfung des Entwurfes, erst danach kann man die Bestellung auslösen. Je nach Menge (von 1 – beliebig) ergibt sich der Preis. Je mehr, umso deutlicher günstiger. Der Preis richtet sich auch nach der gewünschten Fertigungs-Vorlaufzeit. Nach wenigen Tagen hat man die professionell gefertigte Platine in der Post. Natürlich sollt man nun auch schon Löten können – aber das ist ja für uns Modellbauer wohl kein wirkliches Problem.

Sensor

Hier gibt es eine Unmenge an Sensoren, welche als Input für den Arduino genutzt werden können. Es fängt beim Schalter und Potentiometer als einfachste Sensoren an. Es geht bis hin zu barometrischen Sensoren oder eben Magnetfeldsensoren. Diese Sensoren liefern entweder einen analogen Widerstandswert oder konkrete digitale Werte als Input für den Arduino. Diese Werte-Inputs könnten dann im Programmlauf ausgewertet und aufgrund im Programm festgelegter Größen entsprechende Aktionen – zum Beispiel für ein Servo auslösen. Beim Arduino können die Input-Pins beim Start des Programms als analoger oder digitaler Input flexibel festgelegt werden. Eine weitere Variante ist die Werte-Übertragung über die I2C – Schnittstelle mehrerer adressierten Sensoren über zwei spezielle Pins des Arduino.

Aktor/Aktuator

Dies ist die Bezeichnung für antriebstechnische Baueinheiten, welche ein elektronisches Signal in eine mechanische Bewegung umsetzen. Die Rudermaschine/das Servo ist solch ein besonders erwähnenswertes Bauteil. Im weitesten Sinne ist es auch eine LED, welche man leuchten lässt oder ein Signalgeber, welcher auf ein Signal hin Geräusche von sich gibt.

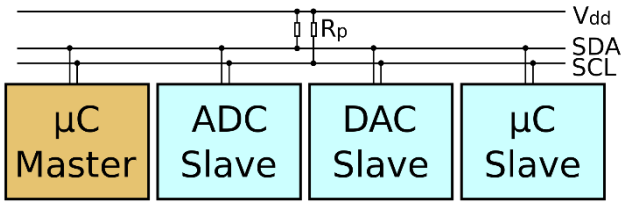
Analoger Input

Dies ist die einfachste Möglichkeit über einen sogenannten Spannungsteiler einem der anlogen PINs des Arduino einen Spannungswert aufgrund des Sensor-Zustandes (Temperatur, Druck, Beschleunigung, Magnetfeld, ….) zu geben – und dann das Programm darauf reagieren zu lassen.

Digitaler Input

Hier gibt der Sensor einen konkreten digitalen Wert an den Input-Pin des Arduino. Dieser Wert variiert z.B. zwischen „0" und einem Sensor-spezifischen Maximal-Wert (z. B.: 1024), welcher wiederum als Steuergröße im Programm genutzt werden kann.

I2C – Schnittstelle



I2C-Schnittstelle

Dies ist eine 4polige Datenleitung, über die einerseits die Spannungsversorgung des Sensors oder eines Aktors (z. B. ein Display) erfolgt, und gleichzeitig über die jeweilige Adresse des Sensors/Aktors dessen aktueller Messwert abgefragt wird, bzw. ein Steuersignal gesendet wird. Dadurch kann man beliebig viele Sensoren und Aktoren an die gleiche Datenleitung anschließen und die Sensoren nacheinander abfragen, bzw. die Aktoren nacheinander ansteuern. Im Fall meiner F1E-Steuerung wird über diese Schnittstelle der Richtungswert abgefragt und gleichzeitig das optionale Display angesteuert.

Links zu diesem Beitrag

Paul Seren hat eine Liste mit hilfreichen Links zu dieser Arbeit erstellt. Sie ist auf www.thermiksense.de unter Infothek -> Elektronik abgelegt. So können die Infos durch ein Klick direkt erreicht werden.