Flightsim-Avionics-Panel

Mit dem Arduino wurde eine Bedieneinheit als Testaufbau erstellt. Für den FSX wurde ein SimConnect-Client erstellt, der über Ethernet mit der Bedien-Einheit kommuniziert. Die Bedieneinheit sendet Kommandos an den SimConnect-Client, der diese verarbeitet und in FSX-Befehle umsetzt. Die Anzeige-Daten werden an die Bedieneinheit zurückgesendet und dort auf dem LC-Display angezeigt.

Im folgenden möchte ich die Hard- und Software-Komponenten vorstellen, mit denen das realisiert wurde.

Experimentierboard

Arduino Uno

Als Hauptkomponente wurde ein Arduino-Uno verwendet. Die Anzahl der verwendeten Anschlüsse wird auch die Verwendung eines ArduinoPro-Mini ermöglichen.

Eine Beschreibung ist hier zu finden: Funduino

LCD-Anzeige

Es wird eine zweizeilige Anzeige mit 16 Zeichen verwendet. Die Anzeige hat eine Erweiterungsplatine, die den Anschluss über I²C ermöglicht. Die I²C-Adresse des verwendeten Moduls ist 0x27. Je nach Lieferant sind auch andere Adresse möglich (PCF8574 0x20/0x27, PCF8574A 0x38/0x3F). Bei Funduino ist das Modul beschrieben, hat dort 5,90€ gekostet.

Ethernet-Modul 1

Für meine ersten Versuche habe ich ein einfaches Modul verwendet. Dieses ist mit einem ENC28J60 aufgebaut und bietet eine einfache Ethernet-Unterstützung. Die Kommunikation mit dem Prozessor erfolgt über SPI. Das Board hat keinen integrierten IP-Stack. Aber für erste Versuche hatte ich es als geeignet befunden. Preiswerte Module kosten ca. 2,-€.

Quellen:
http://www.arduino-projekte.de/index.php?n=24
http://www.tweaking4all.com/hardware/arduino/arduino-enc28j60-ethernet/
http://jeelabs.org/pub/docs/ethercard/index.html
https://github.com/jcw/ethercard

Ethernet-Modul 2

Bei weiteren Versuchen wurde auch ein Ethernet-Modul mit einem WIZnet W5500 angeschlossen und die Software entsprechend angepasst. Die Kommunikation mit dem Prozessor erfolgt über SPI.

Das Board hat einen integrierten IP-Stack. Preiswerte Module kosten ca. 5,-€.

Begonnen hatte ich mit folgender Bibliothek:
https://github.com/Wiznet/WIZ_Ethernet_Library
In der Version 1.8.8 der Arduino-IDE funktioniert die integrierte Bibliothek.

Encoder und Taster

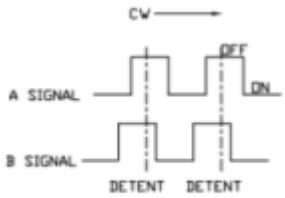
Für den Drehimpulsgeber wird ein Typ verwendet, der eine komplette Impulsfolge pro Rastung erzeugt. Für meinen Testaufbau habe ich einen REB162PVBS verwendet.

Für nachfolgende Entwicklungen habe ich dann Drehimpulsgeber in kleiner und flacherer Bauweise verwendet, wie z.B.

- RE111F-41B3N-20F-20P von alpha bei Mouser
- EN11-HSB1AF20 von BI Technologies bei Mouser
- CI-11CT-V1Y22-IFAAF von Piher (veraltet)
- EC11B2024 von Alps (veraltert)

Wichtig bei der Auswahl ist, dass die gleiche Anzahl von Rastungen und Impulsen angegeben ist, z.B 20 / 20.

Folgende Signalfolge wird verwendet:

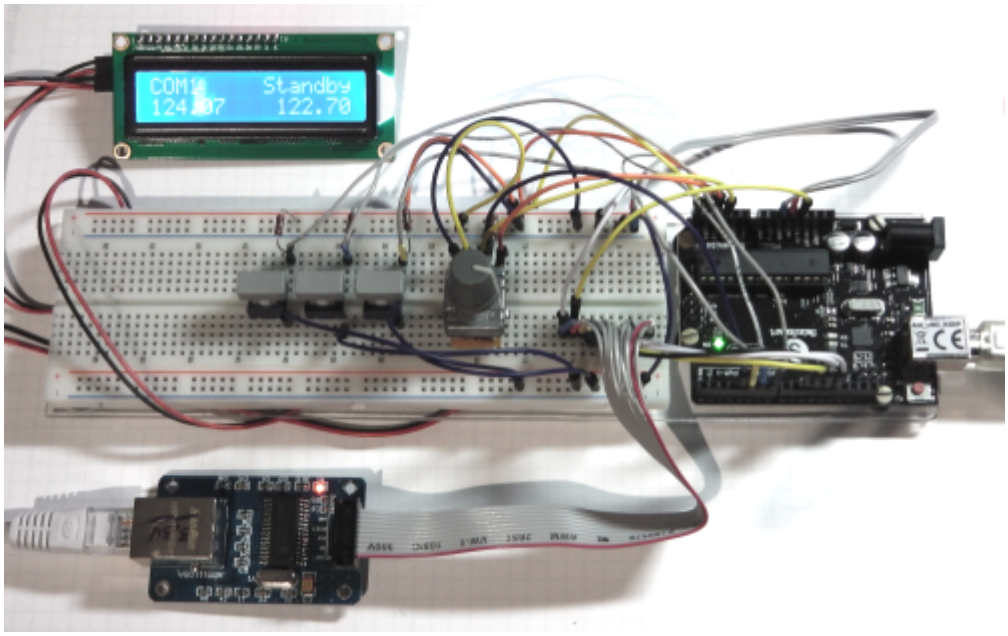


For 20 PPR (B leads A)

Drucktaster sind unkritisch, ich habe welche vonConrad Electronic verwendet. Dort gibt es auch Kappen dafür.

Zusammenbau

Ich habe für die Experimente eine Grundplatte, auf der ich den Arduino-Uno und ein Steckbrett montiert habe. Das Bild zeigt die Zusammenstellung der Einzeilkomponenten.



Die LCD-Anzeige ist an den Ports A4/A5 angeschlossen, das sind die Standard-Ports für SDA und SCL.

Das Ethernet-Modul nutzt für die Kommunikation über SPI die Ports D11...13. Port10 ist der Select-Port.

Der Encoder ist an A2/A3 angeschlossen, was die Verwendung von Interrupts ermöglichen würde, ich habe aber fastgestellt, dass mit der jetzigen Programmstruktur ein Polling-Betrieb ausreicht.

Der Push-Button ist an Port D5, die Taster sind an Port D6...D8 angeschlossen, je mit einem Pull-Up von 22kOhm.

Videoclip vom COM1-Panel
Videoclip vom NAV1-Panel

Hardware - kompakt

Arduino Pro Mini

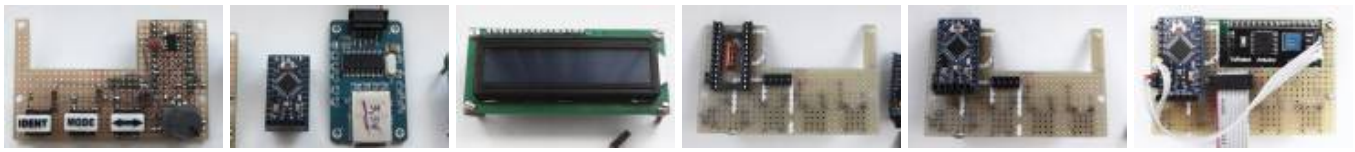
Aus dem Experimentier-Panel wurde ein erster kompakter Aufbau erstellt. Hierfür sollte dann der kleine Arduino Pro Mini eingesetzt werden. Diese kleinen Controller gibt es preiswert im Internet. Es gibt unterschiedliche Ausführungen. Die von mir verwendete Version ist mit ATmega328P, 5V, 16MHz aufgebaut, hat incl. des USB-Seriell-Adapters 5,40€ gekostet. Es gibt preiswerte Einzel-Module ab ca. 2,-€. Andere Versionen Versionen wurden nicht getestet, diese passte von den Systemeigenschaften zum oben genannten Uno. Falls im späteren Einsatz eine andere Versorgungsspannung als 5V verwendet werden soll, so hat der Pro Mini einen 5V-Regler auf der Platine, die Spannungszufuhr muss dann über das RAW-Pin erfolgen. Dies wurde nicht getestet, es ist zu beachten, dass auch LC-Display und Ethernet-Modul damit versorgt werden müssen.

USB-Seriell-Adapter

Der Arduino Pro Mini hat keinen eigenen USB-Anschluss, über den die Programme hochgeladen werden können. An der Stirnseite des Pro Mini sind die Anschlüsse für den USB-Seriell-Wandler vorhanden. Rx, TX sind für die Datenübertragung, das DTR-Signal steuert das Reset des Pro Mini. Es gibt Ausführungen u.a. mit einem FTDI-Chip oder auch mit einem CH340G-Chip.

Muster mit Lochraster-Platine

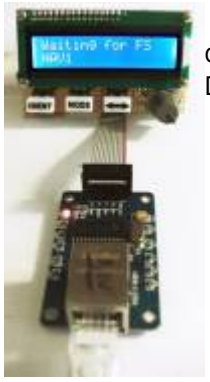
Alle Taster, der Drehgeber, eine Fassung für den Pro Mini und ein 3,3V-Regler für das Ethernet-Modul wurden auf eine Lochrasterplatte gebracht und verdrahtet. In der Leiterplatte befindet sich eine Aussparung für die Huckepack-Platine des LC-Displays.



Das Ethernet-Modul wird über ein 10poliges Flachband-Kabel angeschlossen. In der Programmier-Phase wird das System über den Seriell-USB-Adapter aus dem USB mit 5V versorgt. Anschließend kann das Panel mit einem 5V-Steckernetzteil versorgt werden.

Muster in Betrieb

Ich zeige hier die Version des NAV1-Moduls.



Das Panel wartet nach der Inbetriebnahme auf die Zuteilung einer DHCP-IP-Adresse. Hat das Panel diese erhalten, dann wird auf eine Verbindung zu einem Flugsimulator gewartet. Dies wird im Display angezeigt.

- Anzeige der zugewiesenen IP-Adresse für ca. 3s
- **ACTIVE/STANDBY**-Mode - Anzeige der Active/Standby Frequenzen
- **ACTIVE/CDI**-Mode - Anzeige der aktiven Frequenz, des Course Deviation Indicators (CDI) und des OBS-Kurses
- **ACTIVE/BEARING**-Mode - Anzeige der aktiven Frequenz und des Kurses zur Station (Bearing TO)
- **ACTIVE/RADIAL**-Mode Anzeige der aktiven Frequenz und des Kurses weg von der Station (Radial FROM)

Weitere Informationen zu den Funktionen sind im Bereich der FS-Hardware unter Arduino Avionics Panel zu finden.

Software

Arduino-IDE

Die Arduino-IDE kann auf der Webseite von Arduino geladen werden. Die Installation sollte ohne Komplikationen funktionieren.

Bibliotheken

Für das oben beschriebene Projekt werden zusätzliche Bibliotheken notwendig. Diese können in den Arduino/libraries-Ordner unterhalb der Programm-Installation eingefügt werden oder über den Menüpunkt in der IDE zugefügt werden.

LCD-Anzeige

Für die LCD-Anzeige habe ich die Bibliothek von Francisco Malpartida installiert. Diese kann die bisherige Bibliothek ersetzen und bietet zusätzlich die notwendige Unterstützung für I²C.
https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads

EtherCard

Für das Ethernetmodul mit dem ENC28J60 gibt es mehrere Bibliotheken. Diese sind hier beschrieben. Ich verwende die EtherCard-Bibliothek von Jean-Claude Wippler (jcw). Der Hinweis auf die Nutzung von Pin8 anstelle Pin10 stimmt nicht mehr, in der aktuellen Version der Bibliothek wird auch Pin10 für CS verwendet, wie in der readme beschrieben.
https://github.com/jcw/ethercard/

Ethernet

Für das Ethernetmodul mit dem WIZnet W5500 wurde die WIZ_Ethernet_Library eingebunden. Wie dies gemacht wird, ist in der zugehörigen readme beschrieben. In der verwendeten IDE 1.8.8 funktioniert auch die integrierte Bibliothek.

Drehimpulsgeber

Für den Drehimpulsgeber wurde eine Biblothek von Matthias Hertel eingebunden. Diese ermöglicht auch Interrupt-Betrieb, den ich aber derzeit nicht nutze.
http://www.mathertel.de/Arduino/RotaryEncoderLibrary.aspx

Die Pushbutton der Drehimpulsgeber sind an einem Port-Expander angeschlossen, der das Ereignis über Interrupt an einen Port des Arduino meldet. Für die Kommumikation mit dem Port-Expander ist die I²C-Bibliothek notwendig. Diese ist standardmäßig in der Arduino-IDE enthalten.
Für einfachere Zugriffe auf den Port-Expander PCF8574 wurde eine Bibliothek eingebunden:
https://arduino-projekte.webnode.at/meine-libraries/portexpander-pcf8574/

Arduino-Sketch

Das Arduino-Panel soll sich mit einer PC-Software verbinden, Ereignisse an diese Übertragen und die von der PC-Software übertragenen Werte anzeigen.

Das Ethernet-Modul erhält über DHCP eine Netzwerk-Adresse und nutzt das UDP-Protokoll für die Datenverbindung. Hier ist zu beachten, dass immer eine eindeutige MAC-Adresse und ein eindeutiger Port zugewiesen wird. Diese ist im Sketch definiert.

Die gestartete Software zeigt kurz die zugewiesene Netzwerk-Adresse an und wartet dann auf den Verbindungsaufbau durch die PC-Software. Durch den Verbindungsaufbau weiß die Arduino-Software, wohin sie die Daten senden soll.

Der Arduino-Sketch fragt den Encoder und die Tasten ab und sendet Informationen darüber über Ethernet zum PC-Programm. Dies kann ein FSX/P3D-SimConnect-Client, ein X-Plane-Plugin oder weitere Software sein. Von der PC-Software werden die Ereignisse verarbeitet und die aktualisierten Anzeigedaten zurück zum Arduino gesendet und dort angezeigt.

PC-Software

Die Client-Software stellt die Verbindung zwischen dem Flugsimulator und dem Arduino-Panel her. Über den Netzwerkadapter wird an die Broadcast-Adresse auf dem definierten Port (diese muss mit dem Sketch übereinstimmen) das Arduino-Panel gesucht und dann die Verbindung aufgebaut.

Jetzt können Ereignisse des Arduino-Panels über UDP zum Client gesendet werden, der diese dann in Flugsimulator-Ereignisse umsetzt bzw. Anzeige-Varianten einstellt. Die aktualisierten Variablen werden zurück an das Arduino-Panel gesendet und dort auf dem LCD-Panel angezeigt.

Weitere Informationen zur Software sind im Bereich der FS-Hardware unter Arduino Avionics-Panel zu finden.