Robert Jaworski 81591

Projekt "Kokpit samolotu FA/18C Hornet" stworzony do współpracy z symulatorem lotu firmy Eagle Dynamics Digital Combat Simulator (DCS)

Pomysł projektu powstał w 2021 roku. W pierwotnym zamyśle miały to być pojedyncze panele do Horneta, mające na celu zwiększenie realistycznych odczuć w czasie używania symulatora. Podczas realizacji pomysłu i w trakcie nabierania nowych umiejętności i wiedzy w zagadnieniach projektowania 3D (aplikacja FREECad) drukowania 3D na drukarce Prusa MK3U+ MMU2 oraz projektu Arduino , początkowy pomysł przerodził się w chęć zbudowania całego kokpitu w 100% wiernie oddającego działanie i wymiary rzeczywistego wnętrza FA/18C Hornet. Głównym założeniem było korzystanie z symulatora za pomocą gogli w technologii VR 3D PIMAX 8KX DMAS , co wymuszało bardzo wierne oddanie kokpitu wymiarowo, ponieważ podczas rozgrywki gracz nie widzi fizycznie urządzenia, widzi tylko to co ma na ekranach gogli VR. Rozwiązanie sprawdziło się idealnie dzięki dobremu wymiarowaniu i rewelacyjnej zdolności człowieka tzw. pamięci mięśniowej, (co widać na załączonym krótkim filmie pokazowym)

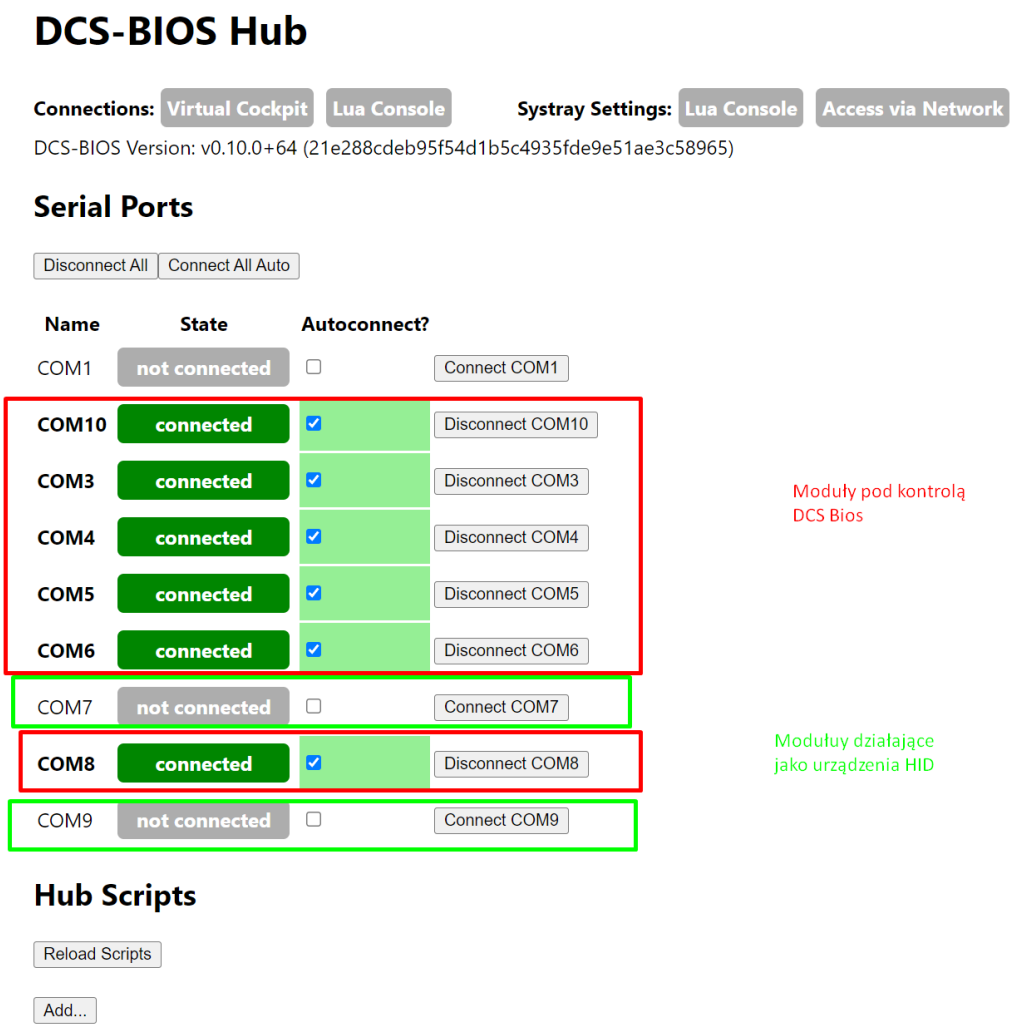
Prace na projektem trwały 7 miesięcy, jego poszczególne etapy można zobaczyć na foto zamieszczonych w katalogu ze zdjęciami.

Od strony elektronicznej, projekt wykorzystuje 8 modułów Arduino.
Dwa z tych modułów Arduino Leonardo i Arduino Micro które ze względu na inny niż Arduino Mega czy Uno kontroler USB mogą pracować w systemie Windows jako urządzenia HID. (na załączonym screenie to COM7 i COM9). Ich oprogramowanie wykorzystuje biblioteki Keypad.h i Joystick.h. Przyciski są mapowane za pomocą matrycy pinów, w przeciwieństwie do rozwiązania pracującego na DCS Bios, gdzie jednemu pinowi odpowiada jedna funkcja.
Plik wsadowy dla tych dwóch modułów emulujący pracę zwykłego kontrolera gier, konfigurowanego z poziomu gry. (kod 1)

Pozostałe moduły (różne, zastosowano MEGA, Mini, i Uno) pracują pod kontrolą specjalnie napisanej do współpracy paneli z DCS World aplikacji o nazwie DCS Bios. Róznica zasadnicza polega na tym, ze w symulatorze już niczego nie trzeba konfigurować, podłącza się port USB i urządzenie od razu jest gotowe do pracy, bez żmudnego mapowania klawiszy w grze.

Ich plik wsadowe ze względu na swoją obszerność można zobaczyć w załączonym pliku xls opisującym wszystkie połączenia wykonane w kokpicie, będącym w swoim rodzaju dokumentacją techniczną urządzenia sporządzoną na użytek własny w przypadku konieczności serwisowania. (kolejne zakładki skoroszytu reprezentują kolejne moduły Arduino i ich połączenie do fizycznych przełączników)

Rysunek 1



KOD1.

```
#include <Keypad.h>
#include <Joystick.h>



#define NUMBUTTONS 56
#define NUMROWS 4
#define NUMCOLS 14
```

```
byte buttons[NUMROWS][NUMCOLS] = {
  {0,1,2,3,4,5,6,7,8,9,10,11,12,13},
  {14,15,16,17,18,19,20,21,22,23,24,25,26,27},
  {28,29,20,31,32,33,34,35,36,37,38,39,40,41},
  {42,43,44,45,46,47,48,49,50,51,52,53,54,55},
};



byte rowPins[NUMROWS] = {21,20,19,18};
byte colPins[NUMCOLS] = {16,15,14,10,9,8,7,6,5,4,3,2,1,0};



Keypad buttbx = Keypad( makeKeymap(buttons), rowPins, colPins, NUMROWS, NUMCOLS);


Joystick_ Joystick(JOYSTICK_DEFAULT_REPORT_ID,
  JOYSTICK_TYPE_JOYSTICK, 56, 0,
  false, false, false, false, false, false,
  false, false, false, false, false);



void setup() {
  Joystick.begin();
}



void loop() {
  CheckAllButtons();
delay(0);
}



void CheckAllButtons(void) {
     if (buttbx.getKeys())
    {
      for (int i=0; i<LIST_MAX; i++)
       {
         if ( buttbx.key[i].stateChanged )
          {
          switch (buttbx.key[i].kstate) {
                case PRESSED:
                case HOLD:
                        Joystick.setButton(buttbx.key[i].kchar, 1);
```

```
                    break;
                case RELEASED:
                case IDLE:
                    Joystick.setButton(buttbx.key[i].kchar, 0);
                    break;
            }
           }
         }

    }
}
```