Vývoj ADB Protokolu pre ESC

V tomto dokumente je uvedený doterajší postup a zmeny, ktoré boli vykonané vo FW zariadenia Pixhawk. Všetky (aj upravené) zdrojové kódy sú dostupné na: https://github.com/eMrazSVK/ardupilot/tree/ardupilot_adb/libraries

Implementácia knižnice “ADB_Proto”

Túto knižnicu definjú dva súbory, a to: ADB_Proto.h a ADB_Proto.cpp. Funkcie, ktoré zabezpečia základnú funkcionalitu sú (uvedené zatiaľ bez vstupov):

- ADB_Proto::init()

- ADB_Proto::tick()

- ADB_Proto::send_frame()

Ďalej v ADB_Proto.h je definovaný dátový frame, ktorý bude posielaný jednotlivým ESC. Ďalšie premenné potrebné k fungovaniu sú `AP_HAL::UARTDriver *ADB_Port` `a AP_SerialManager::SerialProtocol ADB_protocol.` Smerník na ADB_Port bude odkazovať na sériový port, cez ktorý budeme komunikovať. ADB_protocol slúži ako kontrola, či zvolený sériový port komunikuje pomocou nášho protokolu.

Funkcia **init(const AP_SerialManager &serial_manager)** vykoná inicializáciu, a konrétne inicializuje premenné frame-u (pri opakovanom posielaní sa tieto premenné budú len meniť). Ďalej pomocou knižnice AP_SerialManager nájdeme voľný sériový port na komunikáciu. Ak nájde funkcia **AP_SerialManager::find_serial** vhodný port, premennej **ADB_Protocol** priradíme náš protokol (zadefinovaný v AP_SerialManager.h – k tomuto neskôr). Ďalší dôležitý krok, ktorý sa vykoná pri inicializácií zaregistrovanie procesu (ADB_Proto.cpp, riadok 66), zaregistruje sa tam funkcia **ADB_Proto::tick()**. Ako vstup inicializačnej funkcie je serial_manager, čo je objekt, ktorý volá metódy pre správu sériových portov (UART). (pozn. funkcia register_timer_process() by mala zaregistrovať proces do systému a mal by sa volať s 1KHz frekvenciou)

Funkcia **tick()** obsahuje také príkazy, ktoré sa budú cyklicky vloať (preto bola v init() zaregistrovaná ako proces do systému). **ADB_Port→begin()**, spustenie UART sa vykoná tu, pretože tento príkaz musí byť zavolaný z vlákna, z ktorého sa bude komunikovať. Z iného by to nefungovalo. Ďalej je tu zatiaľ obsiahnutá aj funkcia **send_frame()**, ktorá by mala posielať cez sériový port jeden dátový frame – toto sa ešte môže zmeniť, neviem, či to nebude lepšie posielať po bytoch atď., zatiaľ to nie je podstatné. Príkaz na posielanie cez sériový port je ***ADB_Port→write(uint8_t byte);***

Závislosti v systéme

V samotnej implementácií knižnice som ešte až tak neriešil celkové zasadenie protokolu do systému. Opíšem preto jednotlivé kroky, ktoré som uskutočnil pre to, aby som protokol ako tkaý implementoval do systému samotného.

- **config.h**

```
#ifndef ADB_PROTO_ENABLED
        # define ADB_PROTO_ENABLED ENABLED
 #endif
```

- **Copter.h**

```
#if ADB_PROTO_ENABLED == ENABLED
#include <ADB_Proto/ADB_Proto.h>
#endif
```

Vytvorenie objektu adb_light_proto, aby mohli byť volané metódy knižnice

```
#if ADB_PROTO_ENABLED == ENABLED
ADB_Proto adb_light_proto;
#endif
```

- **system.cpp**

Tu sa volá inicializačná funkcia

```
#if ADB_PROTO_ENABLED == ENABLED
//initialize adb_protocol
adb_light_proto.init(serial_manager);
#endif
```

- **make.inc**

Pridanie ADB_Proto, aby pre “build” bol tento priečinok viditeľný ako knižnica

-**AP_Arming.cpp.d**

Pridaná cesta k ADB_Proto.h

-AP_SerialManager.h

Riadok 98 – pridanie ADB protokolu ako **SerialProtocol_ADB_Proto** s identifikátorom **14**(tento identifikátor je dôležitý, domnievam sa, ak sa nastavuje protokol napr. Cez Mission planner, je tam parameter serial5 protocol, tak tam sa napíše príslušné číslo).

Riadky 73, 74, 75 - definovanie BAUDu a veľkosti zásobníkov pre RX a TX (takisto sa zmenia, zatiaľ len približné hodnoty)

-AP_SerialManager.cpp

Riadok 216 – nastavenie Baudu pre príslušný port