UNIVERSITATEA TITU MAIORESCU

FACULTATEA: INFORMATICĂ
DEPARTAMENT: INFORMATICĂ
Programa de studii: INFORMATICĂ
DISCIPLINA: INTELIGENȚĂ ARTIFICIALĂ

IA - *Testul de evaluare nr.* 4

Robot autonom aerian – HEXACOPTER

EXAMEN

Nr. crt.	Grupa	Numele și prenumele	Semnătură student	Notă evaluare
1				
2				
3				
4				
5				
6				

Data: __/__/_____

CS I dr.ing.

Lucian Ștefăniță GRIGORE

Conf.dr.ing.

Iustin PRIESCU

Ș.l. dr.ing

Dan-Laurențiu GRECU

Cuprins

1. COMPONENTE COMANDĂ și CONTROL

1.1 Arduino

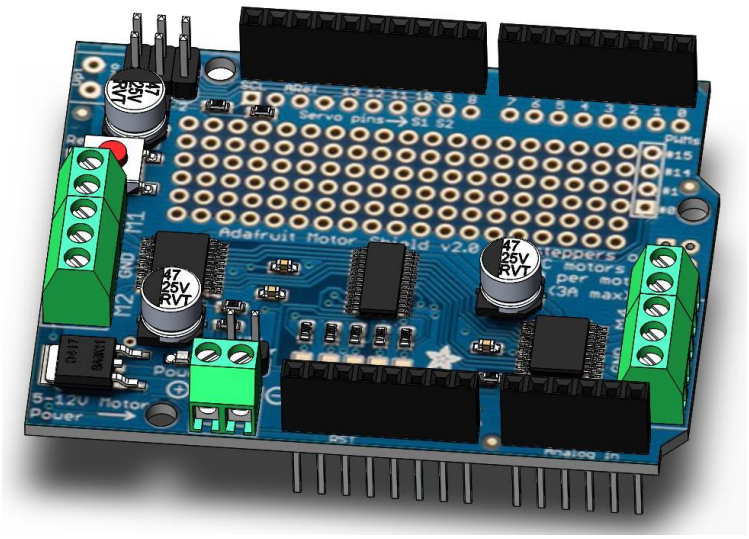

Fig. 1-1 Arduino Atmega.

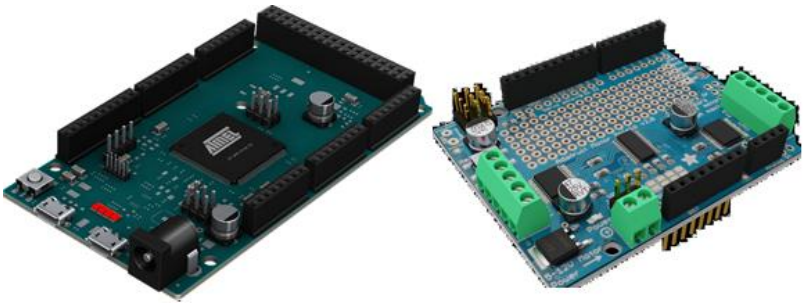

Fig. 1-2 Arduino Atmega.

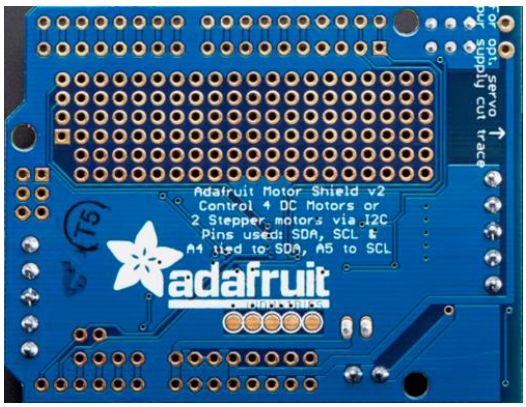

Fig. 1-3 Arduino Atmega.

1.2 Raspberry PI

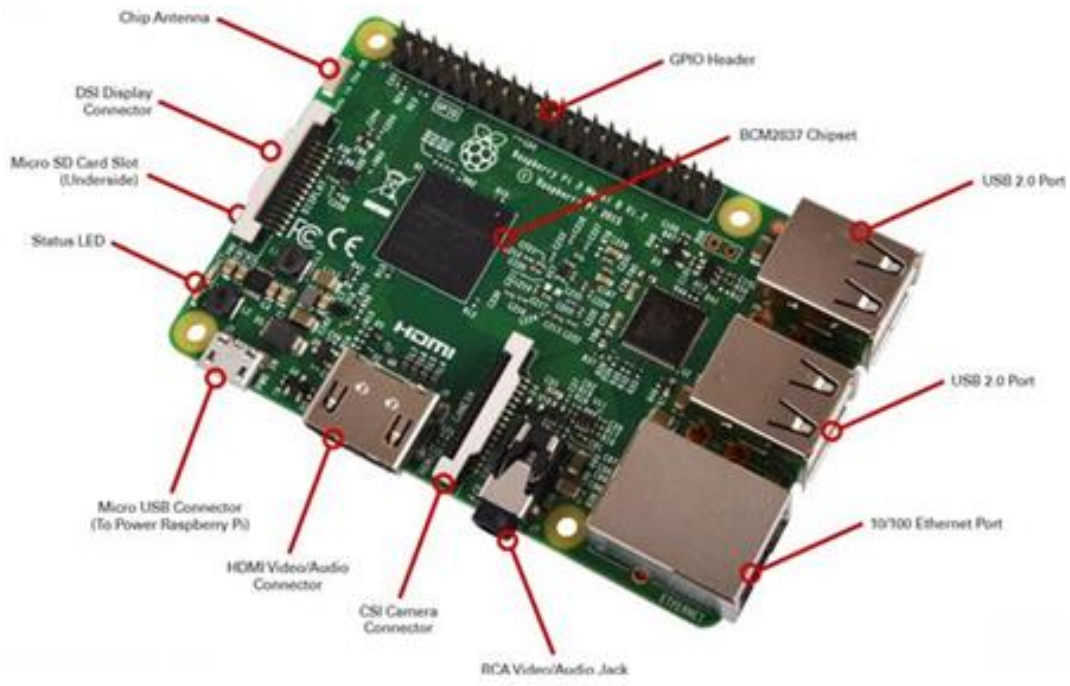

Fig. 1-4 Raspberry Pi B+ V2.1

http://www.cnx-software.com/2016/02/29/raspberry-pi-3-board-is-powered-by-broadcom-bcm2827-cortex-a53-processor-sells-for-35/

Spre deosebire de soluțiile distribuite, Platforma pune în aplicare toate acestea componente într-o arhitectură omogenă care utilizează o singură platformă hardware un Raspberry PI Compute Module Dev Kit, un singur limbaj de programare (C ++) cu următoarele specificații tehnice:

- Broadcom BCM2835 CPU 700MHz;
- SODIMM sized (6.5cm by 3cm) Raspberry Pi board;
- 512MB RAM;
- 4GB eMMC Flash memory;
- micro USB connector type B;
- 2 x CSI ports for camera boards;
- 2 x DSI ports for display boards;
- HDMI port;
- Micro USB power connector.

Acest design conduce la o arhitectură deschisă, care este mai puțin complexă, mai ușor de utilizat, și mai ușor expandabilă. În special, zona de cooperare cu mai mulți roboți și care asigură un mijloc simplu și flexibil de comunicare.

S-a urmărit crearea unei platforme comune generic, care poate fi reutilizată de către cercetători pentru diferite aplicații. Având în vedere varietatea de aplicații robotice și domenii de cercetare, aceasta este o sarcină dificilă. Din cauza lipsei de flexibilitate și performanță a controler-ului robotului, s-a preferat realizarea unei biblioteci de module de control pentru diverse aplicații.

Acest tip de arhitectură are următoarele avantaje:

- *Simplitate*. O arhitectură omogen nedistribuită este mult mai mic și mai simplă decât o arhitectură distribuită neomogen. Este mai ușor de configurat, ușor de înțeles, și mai ușor să se extindă. Simplitatea este critică prin prisma motivării de a putea reutiliza platforma pentru diferite aplicații.
- *Flexibilitate* la toate nivelurile. Toate componentele platformei sunt deschise pentru extensii și modificări.
- *Integrarea ridicată*. Din moment ce toate componentele pot rula pe aceeasi platformă, s-a urmărit o integrare mare, ceea ce permite o cooperare mai simplă și mai eficientă între componente. Respectiv, de comunicare între componentele sale și cu cele ale altor roboți tereștri sau aerieni, sistem GPS, etc.

Interfațare Hardware	Programarea în timp real	Simultaneitatea
Generarea traiectoriei	Arhivarea și prelucrarea datelor în timp real	Comunicații între procesoare
Sistem de proiectare orientat pe obiect	Sistem complex de proiectare	Funcții matematice pentru descrierea mișcării robotului
Programe utilitare: calibrare etc.	Suport pentru diferite instrumente, senzori	Network
Posibilitatea de implementare a sistemelor de procesare 3D	Interfața grafică	Interfațare software

1.3 Modulul de navigație Navio

Fig. 1-5 Modul navigație NAVIO2.

http://www.robotshop.com/blog/en/navio2-or-the-raspberry-pi-flight-controller-18362

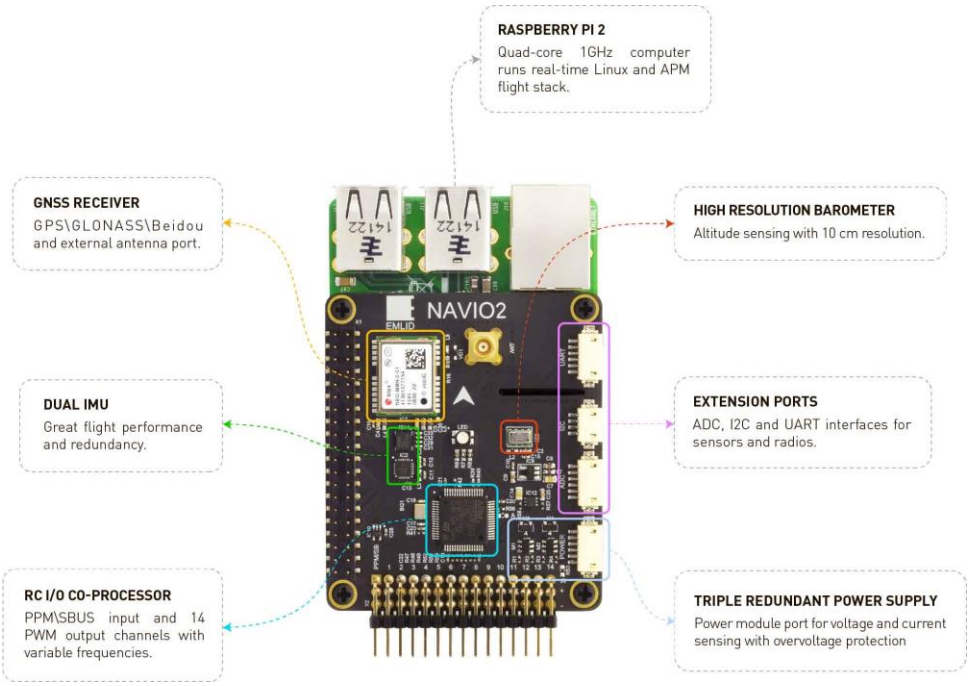

Fig. 1-6 Descriere modul navigație NAVIO2.

http://www.robotshop.com/en/navio2-autopilot-kit-raspberry-pi-2-3.html

Sistemele de control al zborurilor pentru UAV-uri și Drone se dezvoltă foarte repede. În acelasi timp, se fac multe dezvoltări pentru reducerea costurilor, iar Raspberry Pi este unul dintre liderii în acest domeniu. Acest lucru a ridicat întrebări în mintea oamenilor de la Emlid și, prin urmare, au amestecat ambele sisteme într-un sistem bazat pe ArduPilot pentru Raspberry Pi, rezultând un HAT care adaugă tot hardware-ul necesar unui controlor de zbor la bordul inițial Raspberry Pi. De asemenea, au portat complet proiectul ArduPilot la Linux, care oferă o soluție reală de controler de zbor.

Prin aceasta, echipa Emlid a sporit foarte mult posibilitățile, deoarece procesorul Raspberry Pi este mult mai avansat decât microcontrolerele standard găsite în controlorii de zbor deja disponibili pe piață. Bazat pe Raspberry Pi, acest sistem poate fi hacked ușor și noi caracteristici adăugate fără compromisuri. Acest lucru poate deschide un număr mare de posibilități, cum ar fi senzori noi și platforme complet autonome. Deoarece proiectul ArduPilot nu se limitează doar la Multirotori, Navio2 poate fi folosit și pentru multe aplicații, inclusiv planuri și roți de teren.

Unele dintre Caracteristicile promovate de echipa Emlid:

- APM FLIGHT STACK: Navio2 rulează stivă de zboruri APM bine dovedită și poate funcționa în diferite moduri de zbor, inclusiv manual, stabilizator, follow-me și auto. Codul este executat direct pe Raspberry Pi cu kernel Linux în timp real și puteți rula aplicațiile alături. Copters, Planes și Rovers sunt suportate;
- MULTI-PLATFORMA GCS: APM susține comunicarea MAVLink cu o gamă largă de opțiuni GCS pentru Win, Mac, Linux, precum și pentru Android și IOS. Navio va trimite telemetrie prin Wi-Fi, LTE, BT sau orice alt modem. Cu conectare la distanță se poate monitoriza sistemul, executa comenzi și executa scripturi în zbor;
- CONECTAREA INTERNETULUI: se utilizează modemul LTE sau conexiunea Wi-Fi de mare putere pentru a face accesibilitatea prin internet sau în rețeaua locală. Se efectuează streaming video de la aparatul de fotografiat Raspberry Pi, joystick pentru a vă controla dronul de oriunde din lume. Navio este un pilot automat multimedia.

  Ce este inclus:

- MPU9250 9DOF IMU;
- LSM9DS1 9DOF IMU;
- MS5611 Barometer;
- U-blox M8N Glonass/GPS/Beidou;
- RC I/O co-processor;
- HAT EEPROM;
- RGB LED.

Specificații:

- 14 PWM servo outputs;
- PPM/S.Bus input;
- Triple redundant power supply;
- Power module connector;
- UART, I2C, ADC for extensions;
- Size: 55x65mm;
- Weight: 23gr.

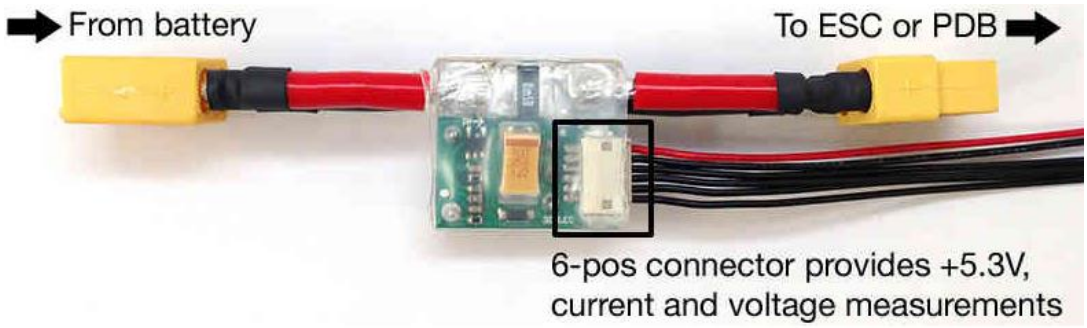

Fig. 1-7 Driver motor MODULE v1.0.

http://ardupilot.org/copter/docs/common-3dr-power-module.html

Modul de configurare al modulului de alimentare comun pentru a măsura tensiunea bateriei și consumul de curent. Majoritatea controlerelor de zbor, inclusiv Pixhawk, au un conector dedicat pentru atașarea modulului de alimentare. Acest lucru este util deoarece:

- oferă o sursă de alimentare stabilă de 5.37 V și 2.25 Amp, ceea ce reduce șansele de apariție a unui semnal de alarmă;
- permite monitorizarea tensiunii și curentului bateriei și declanșarea unei reveniri la lansare atunci când tensiunea devine scăzută sau puterea totală consumată în timpul zborului se apropie de capacitatea bateriei;
- permite firmware-ului autopilotului să compenseze mai exact interferențele de pe compas din alte componente;
- master-ul acceptă o tensiune maximă de intrare de 18V (până la bateria 4S Lipo) și un curent maxim de 90Amp. Atunci când se utilizează un APM, se poate utiliza întregul interval de detectare a curentului de 90 Amp, cu ajutorul măsurătorilor PX4 / Pixhawks până la 60Amp.

Tensiunea maximă de intrare a modulului de alimentare este de 18V. Acesta este nivelul maxim permis de regulatorul de la bord (4 celule LiPo max). Modulul de alimentare furnizează suficientă energie pentru controlerul de zbor, receptorul, perifericele de putere inferioară, cum ar fi

un lidar de putere redusă și un aparat de telemetrie, dar nu dispune de suficientă energie pentru servomotoare sau dispozitive de curent înalt precum transmițătoarele FPV și radiatoarele de telemetrie RFD900.

Configurația și calibrarea modulului de alimentare (PM) implică măsurarea tensiunii bateriei și a consumului de curent în mod continuu. Un modul de alimentare poate fi utilizat pentru a furniza o sursă stabilă de alimentare a sistemului și pentru a măsura cu precizie tensiunea / curentul bateriei pentru a declanșa o revenire, adică de a reîncărca acumulatorul descărcat. ArduPilot este compatibil cu un număr de module de alimentare. Măsurarea bateriilor se poate activa din următorul ecran:

INITIAL SETUP | Hardware opțional | Ecran monitor baterie.

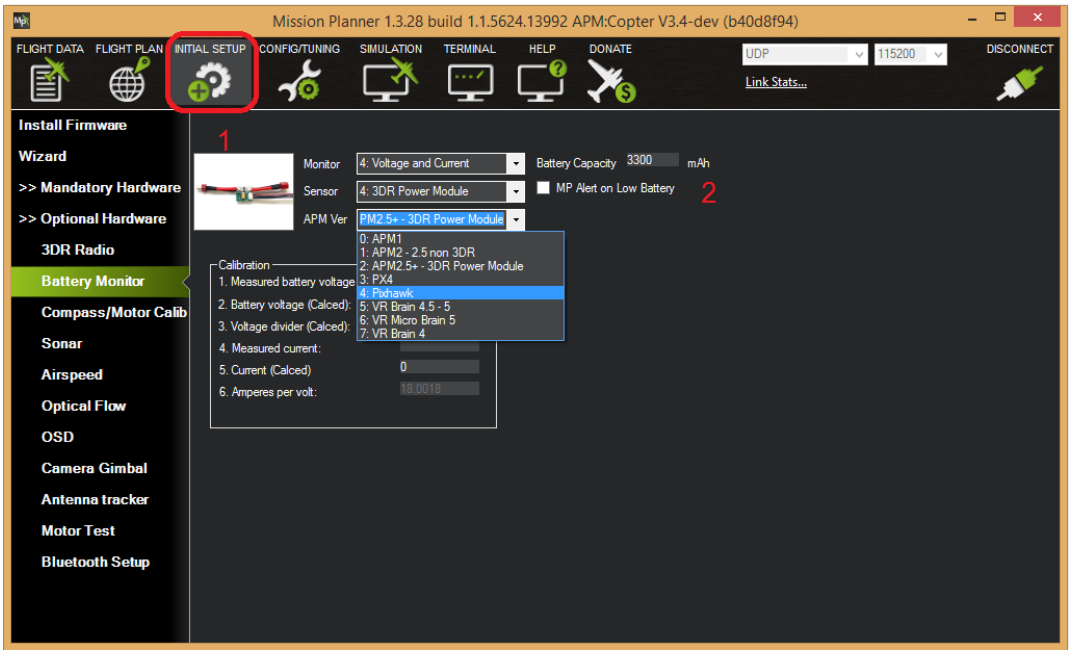

Fig. 1-8 Mission Planner: Battery Monitor Configuration.

http://ardupilot.org/copter/docs/common-power-module-configuration-in-mission-planner.html#common-power-module-configuration-in-mission-planner

Se activează detectarea tensiunii și a curentului, se introduc proprietățile modulului utilizat pentru a le putea măsura, tipul de modul, tipul de controler de zbor și capacitatea bateriei:

Monitor:	Voltaj și curent sau volți de baterie
Senzor:	Modul de alimentare acceptat sau "Altele"
APM ver:	controler de zbor (de exemplu, Pixhawk)
Capacitatea bateriei:	Capacitatea bateriei în mAh

Lista de selecție a senzorilor oferă un număr de module de alimentare (inclusiv modele populare din 3DR și Autopilot) care se pot selecta pentru a configura modulul în forma automat control. Dacă PM-ul nu este pe listă, se poate selecta altul și apoi se efectuează o calibrare manuală:

- în partea de jos a ecranului pentru monitorul bateriilor se poate efectua calibrarea și măsurarea tensiunii/curentului pentru a verifica dacă tensiunea măsurată a bateriei este corectă;
- se activează butonul Other și apoi se efectuează calibrarea pentru a configura un modul de alimentare „necunoscut”;
- calibrarea pentru citirea tensiunii bateriei LiPo cu un voltmetru de mână sau un analizor de putere:
    - se conectează APM / PX4 la computer și apoi se conectează acumulatorul LiPo;
    - se verifică tensiunea prin intermediul:

      INITIAL SETUP Hardware opțional | Ecranul

      pentru monitorul bateriei sau pe ecranul HUD sau în fereastra de stare a datelor de zbor.

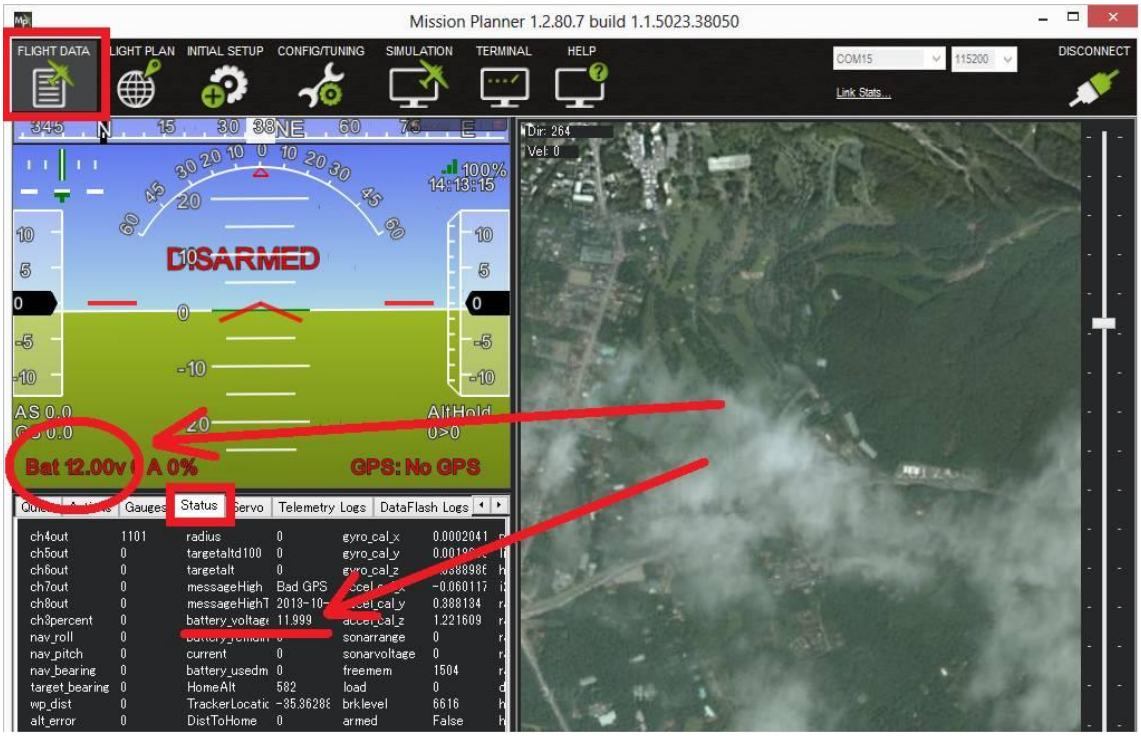

Fig. 1-9 Mission Planner: Battery Monitor Configuration.

http://ardupilot.org/copter/docs/common-power-module-configuration-in-mission-planner.html#common-power-module-configuration-in-mission-planner

2. SIMULAREA SISTEMULUI DE COMANDĂ AL HEXACOPTERULUI ÎN SOLID WORKS

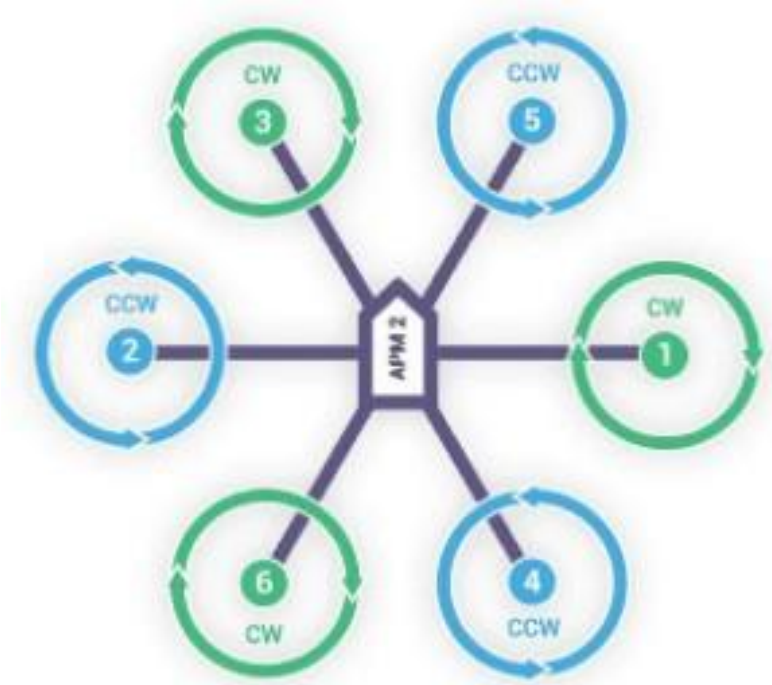

Fig. 2-1 Schematizarea funcțională a unui hexacopter
https://oscarliang.com/types-of-multicopter/

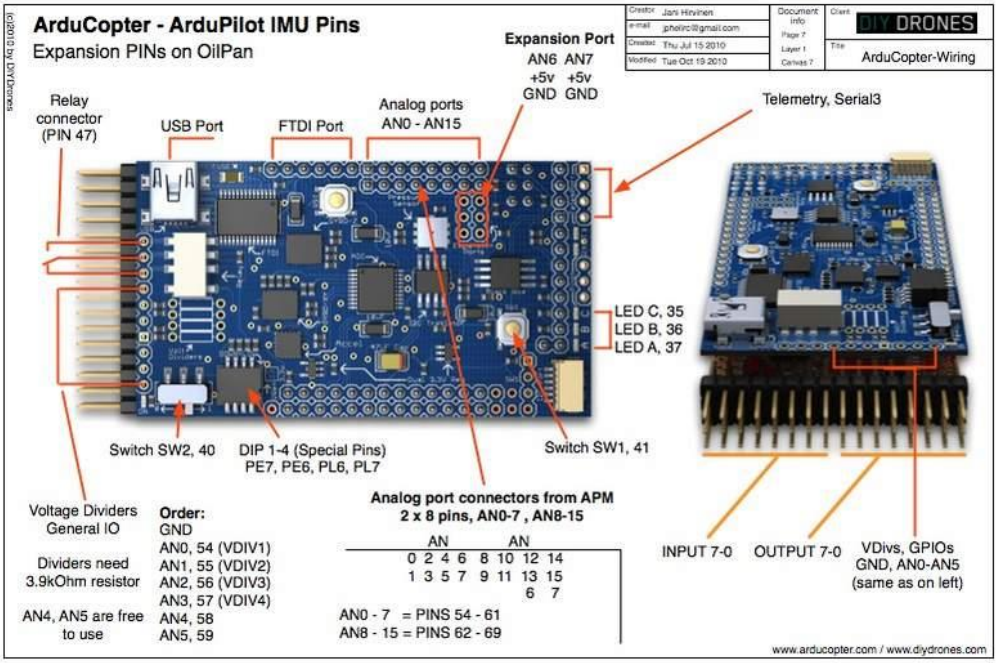

Fig. 2-2 ArduPilot platformă dezvoltare multirotor.
https://www.rcgroups.com/forums/attachment.php?attachmentid=3597095

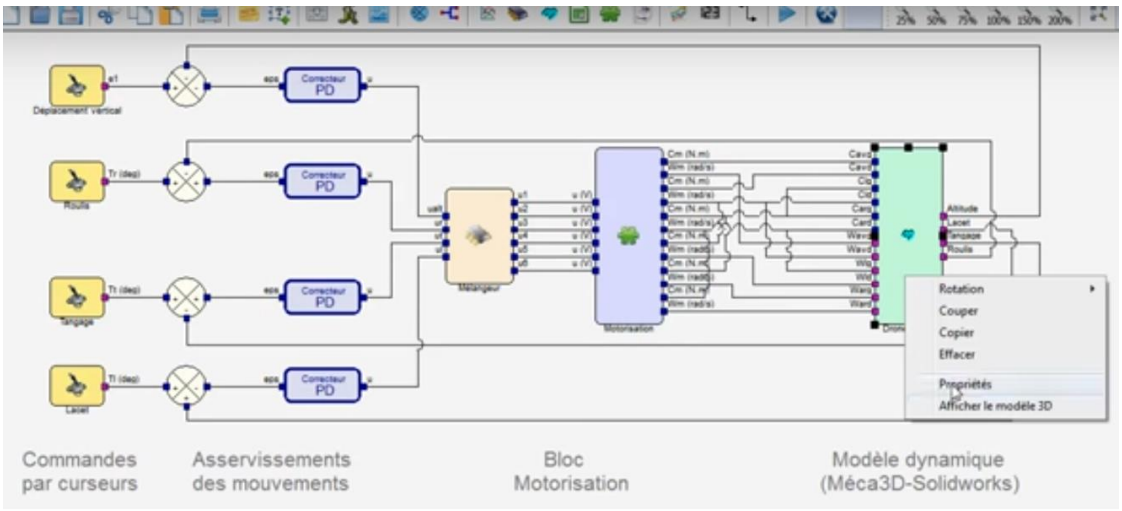

Fig. 2-3 Schema bloc de programare a unui hexacopter
https://www.youtube.com/watch?v=WMcoXfWTr08

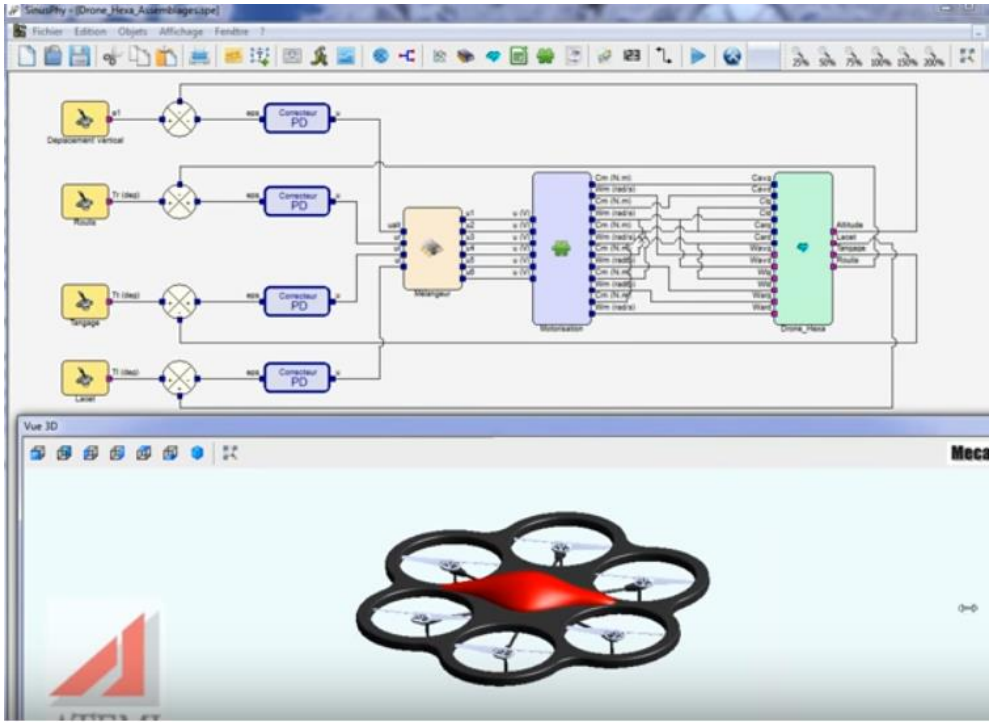

Fig. 2-4 Stabilirea punctului de plecare, în modul simulat.
https://www.youtube.com/watch?v=WMcoXfWTr08

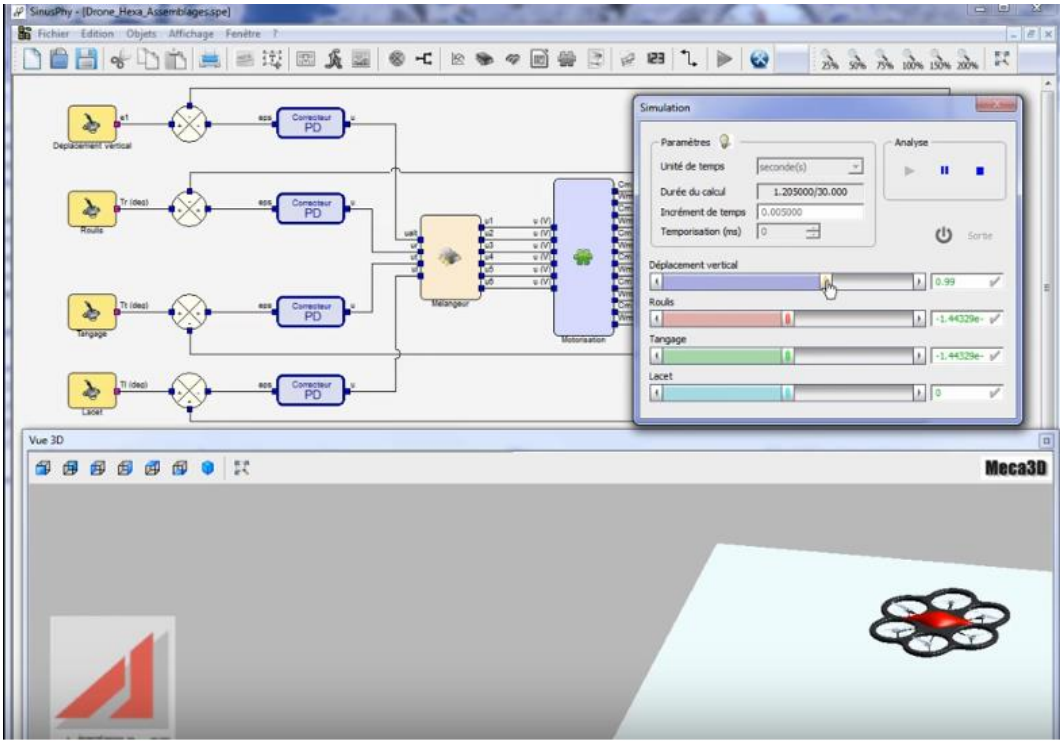

Fig. 2-5 Vizualizarea deplasării simulate a hexacopterului.
https://www.youtube.com/watch?v=WMcoXfWTr08

3. SISTEM IDENTIFICARE MOTOARE ELECTRICE

Scopul este de a obține modelele[1] liniare si neliniare ale motoarelor DC Brushed, motoarelor DC brushless (BLDC), motoarelor sincrone cu magnet permanent (PMSM) și motoarelor cu inducție AC. În prezent, s-a implementat pe un stand simplu de testare pentru a obține modelul linear al motoarelor DC Brushed. Acest stand de testare este alcătuit dintr-un transportor Driver Pololu MC33926, un prototip mic, o poartă NAND quad cu 2 intrări IC 74LS00, o placă PCI-6229 de achiziție de date multifuncțională cu tehnologie low-cost. Motorul de curent continuu prezentat în figură are un codificator optic de cvadratură cu 300 PPR (impulsuri per revoluție) și un raport de transmisie de 30: 1, 200 rpm, 12 V.

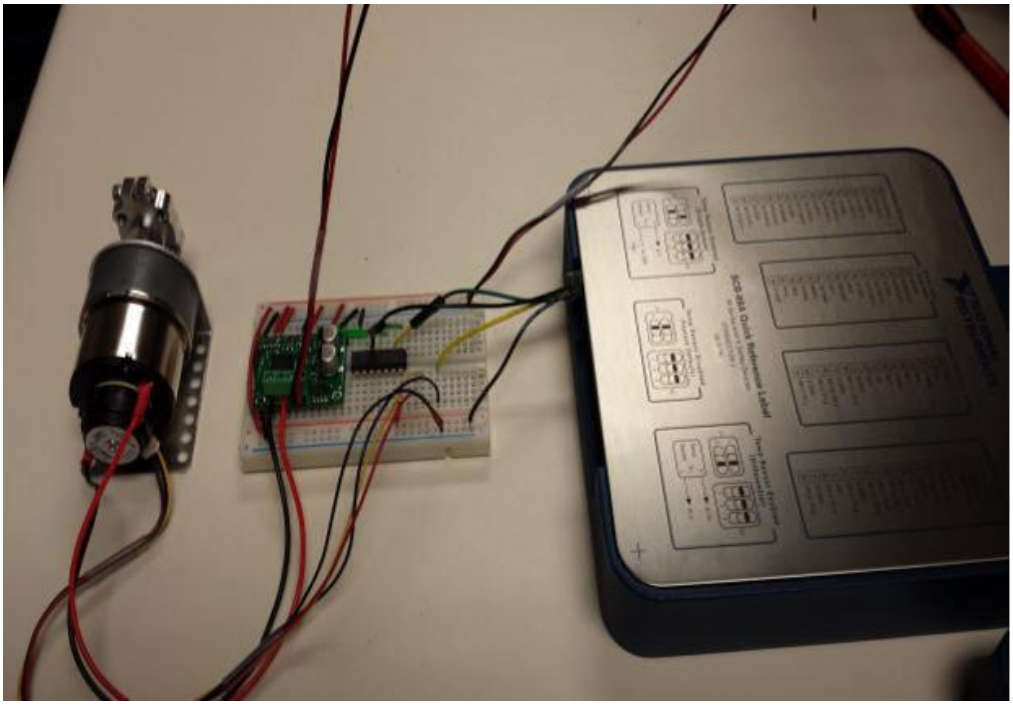

Fig. 3-1 Stand testare motor Pololu pe o platformă LabVIEW.
http://faculty.atu.edu/ccastillo/projects/projects.html

Sub LabVIEW - VI s-a dezvoltat o platformă de simulare, pentru a înregistra răspunsul la viteză atunci când ciclul de funcționare al punții H, conectată la motor, este variat aleator. Semnalul PWM are o frecvență de 20 KHz. Ciclul de sarcină a fost permis să varieze de la aproximativ 10% la 100%. Imaginea următoare prezintă Panoul frontal al LabVIEW - VI.

Pentru a simula un motor electric, trebuie pregătiți parametrii și datele referitoare la modelul motor care urmează să fie utilizat. Diferitele modele necesită resurse diferite în etapa de pregătire. Pentru citirea parametrilor constanți sau modelul liniar, trebuie cunoscuți parametrii motorului electric ce urmează a fi simulat. Simulatorul VI citește parametrii motorului și imită comportamentul

[1] http://faculty.atu.edu/ccastillo/projects/projects.html

unui motor real. În figura următoare este un exemplu care arată modul în care simulatorul VI citește parametrii motorului. În acest exemplu, simulatorul VI este PMSM Constant Parameter Model.vi. Trebuie specificați parametri motorului, cum ar fi numărul de poli, rezistența, inductanța și legarea fluxului, la simulatorul VI.

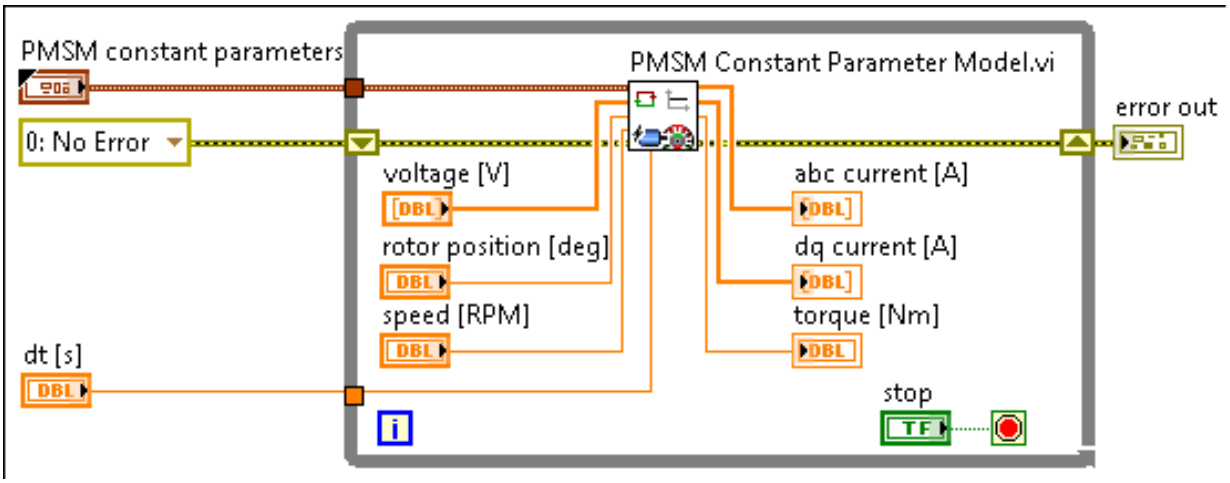

Fig. 3-2 Citirea PMSM Constant Parameter Model.vi.
http://zone.ni.com/reference/en-XX/help/373998C-01/lvemsim/emsim_rtt/

Pentru a utiliza modelul de analiză a elementelor finite (FEA) sau modelul de parametru variabil, se pot citi parametrii motorului și datele brute dintr-un fișier RTT. Cu ajutorul Instrumentului de simulare a motorului electric, se poate citi un fișier RTT pentru a obține o referință de clasă LabVIEW a modelului. Referința claselor LabVIEW conține parametrii motorului în formatul LabVIEW (lvclass), care este util pentru simulatorul VIs pentru a obține caracteristicile motorului electric. Următoarea figură este un exemplu care arată cum se citește fișierul RTT și se utilizează referința de clasă LabVIEW cu simulatorul VI.

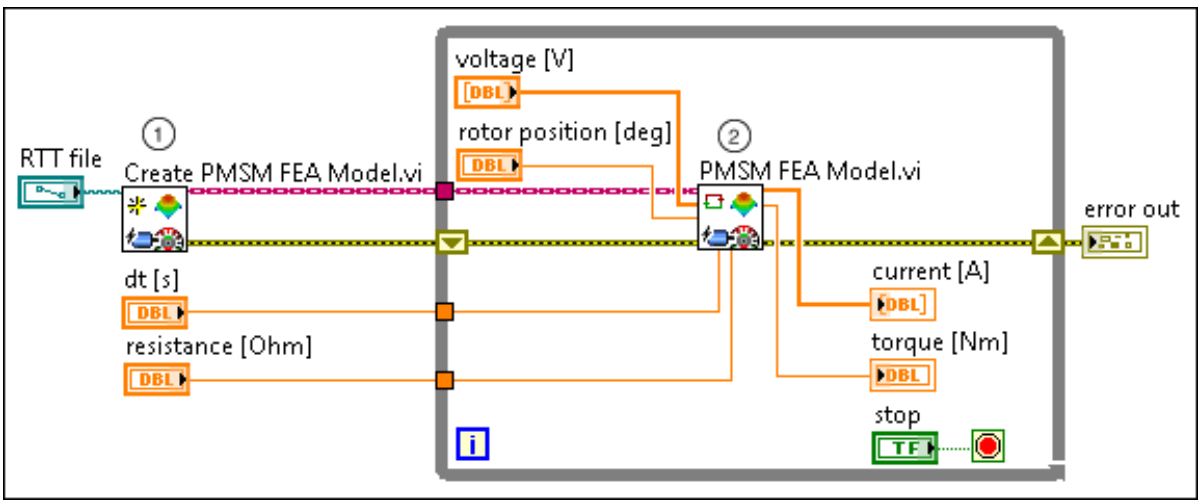

Fig. 3-3 Citirea unui fișier RTT.
http://zone.ni.com/reference/en-XX/help/373998C-01/lvemsim/emsim_rtt/

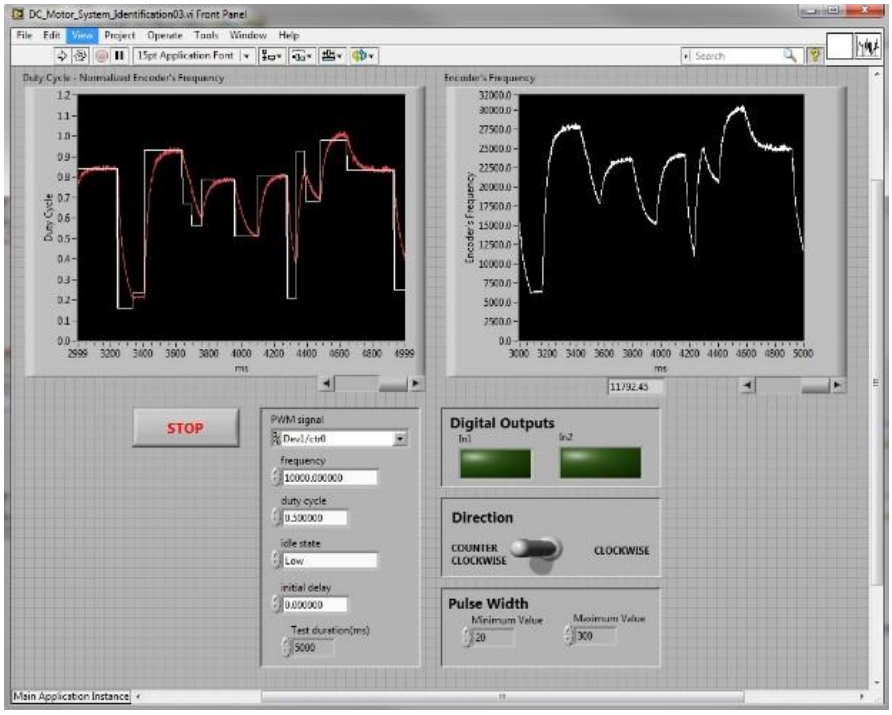

Fig. 3-4 Platformă simulare motoare GearMotor DC.
http://faculty.atu.edu/ccastillo/projects/projects.html

Acestei diagrame bloc, îi corespunde următoarea schemă de conexiuni și reglaje.

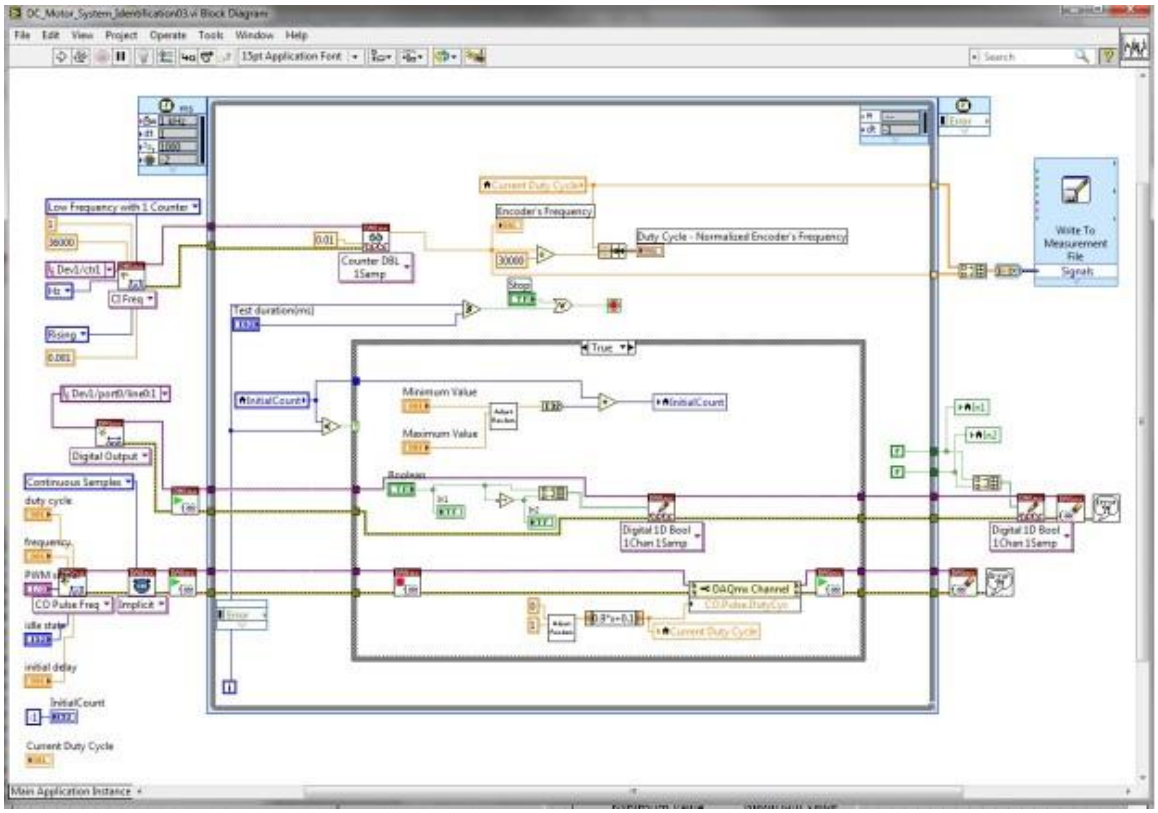

Fig. 3-5 Diagramă bloc pentru conexiuni și reglaje.
http://faculty.atu.edu/ccastillo/projects/projects.html

Semnalul ciclului de sarcină este generat aleatoriu și răspunsul la ieșire (frecvența encoderului) au fost salvate într-un fișier. După aceasta, datele au fost importate în MATLAB și au fost prelucrate utilizând tool-ul disponibil UNIC din Newcastle. Modelul de viteză a motorului DC cu timp discret obținut împreună cu unitatea este un ARX (Autoregresiv Exogen) cu următoarea structură:

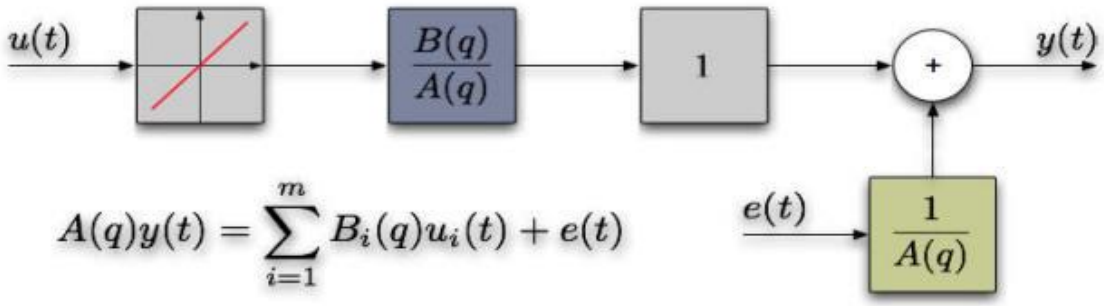

Fig. 3-6 Diagramă bloc pentru conexiuni și reglaje.
http://faculty.atu.edu/ccastillo/projects/projects.html

În continuare sunt prezentate câteva secvențe ale programului de mai sus.

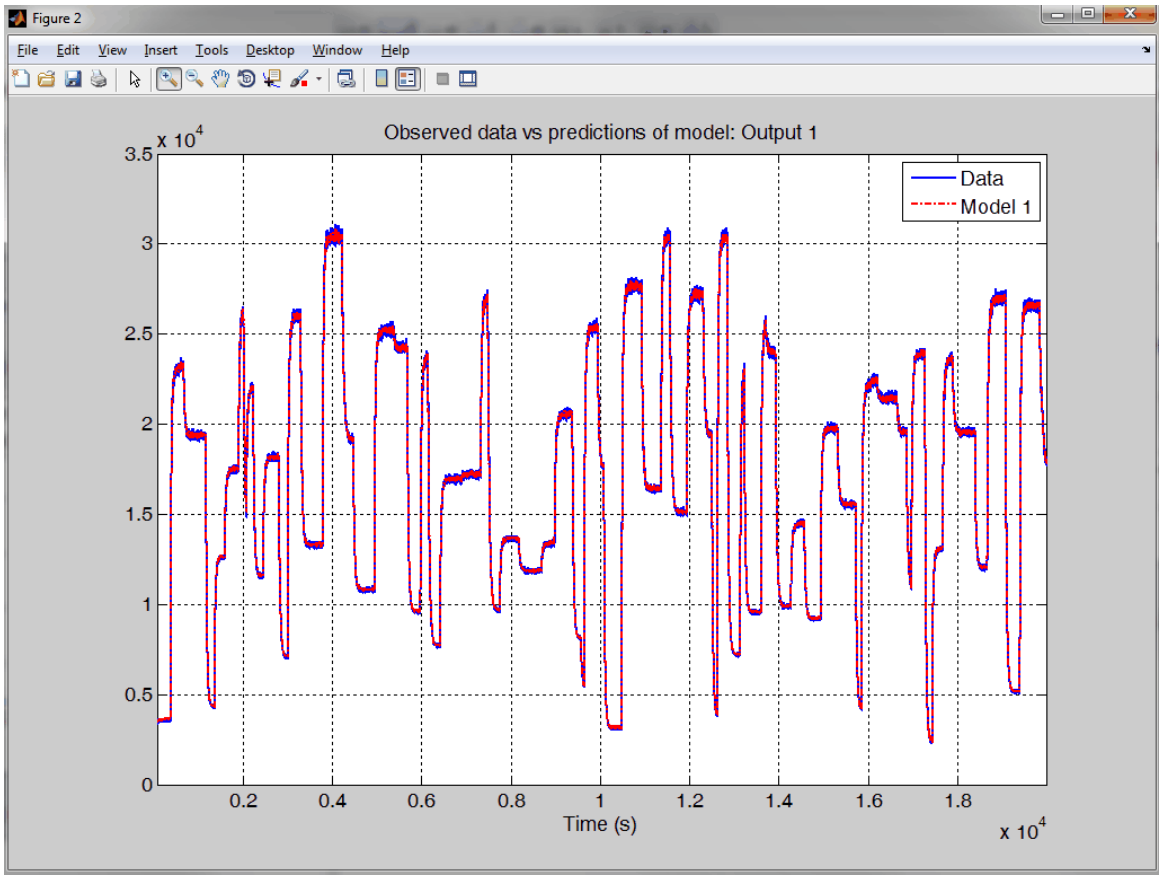

Fig. 3-7 Diagramă bloc pentru conexiuni și reglaje.
http://faculty.atu.edu/ccastillo/projects/projects.html

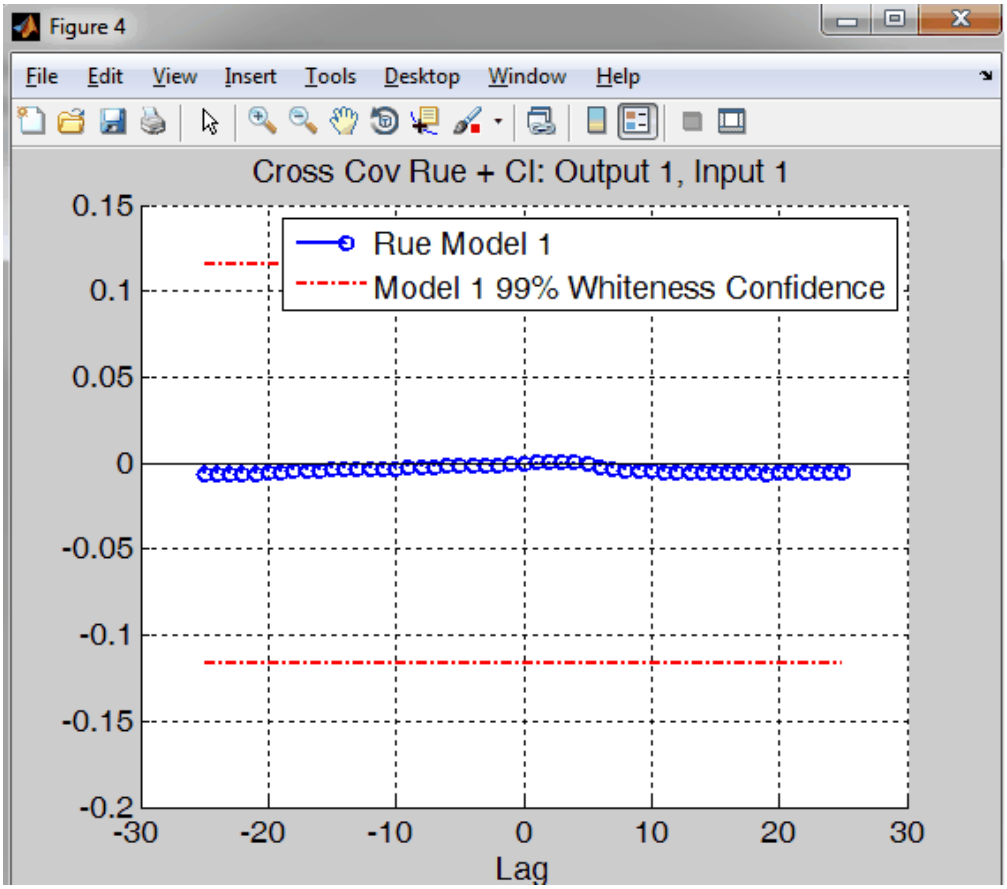

Fig. 3-8 Diagramă bloc pentru conexiuni și reglaje.
http://faculty.atu.edu/ccastillo/projects/projects.html

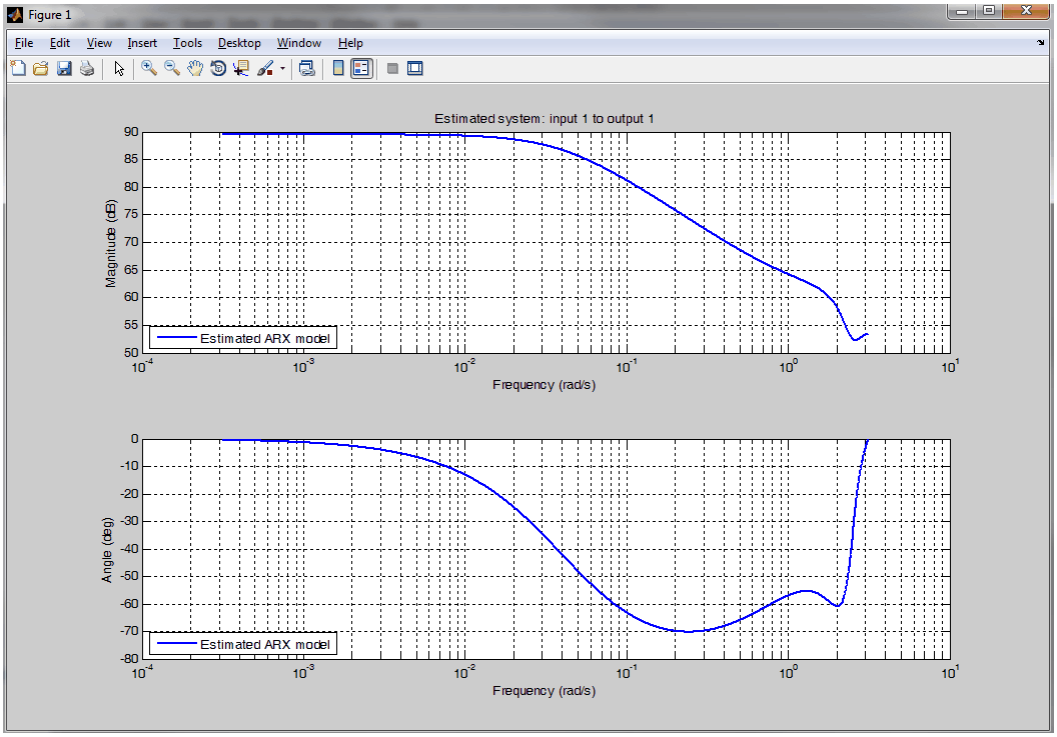

Fig. 3-9 Diagramă bloc pentru conexiuni și reglaje.
http://faculty.atu.edu/ccastillo/projects/projects.html

El ilustrează următoarea secvență de pași:

- crearea modelului PMSM FEA Model.vi citește fișierul RTT pentru a obține parametrii motorului și datele;
- simulatorul VI utilizează parametrii motorului și datele pe care le citează partea anterioară a acestui program pentru a simula comportamentul unui motor.

În acest caz, simulatorul VI este modelul PMSM FEA Model.vi.

Citirea unui fișier de model motor ANSYS utilizând modulele LabVIEW VI:

- pentru a utiliza modelul de parametru variabil pentru simularea motoarelor sincrone cu magnet permanent (PMSM), se pot citi parametrii motoarelor și datele brute dintr-un fișier model de motor ANSYS;
- cu ajutorul Instrumentului de simulare electrică pentru motoare, se poate citi un fișier model de motor ANSYS pentru a obține o referință de clasă LabVIEW a modelului parametrului variabil PMSM. Referința de clasă LabVIEW conține parametrii motorului în formatul LabVIEW (lvclass). Formatul este util pentru simulatorul VIs pentru a obține caracteristicile motorului electric.

Următoarea figură este un exemplu care arată cum se citește un fișier model de motor ANSYS și se utilizează referința de clasă LabVIEW cu simulatorul VI.

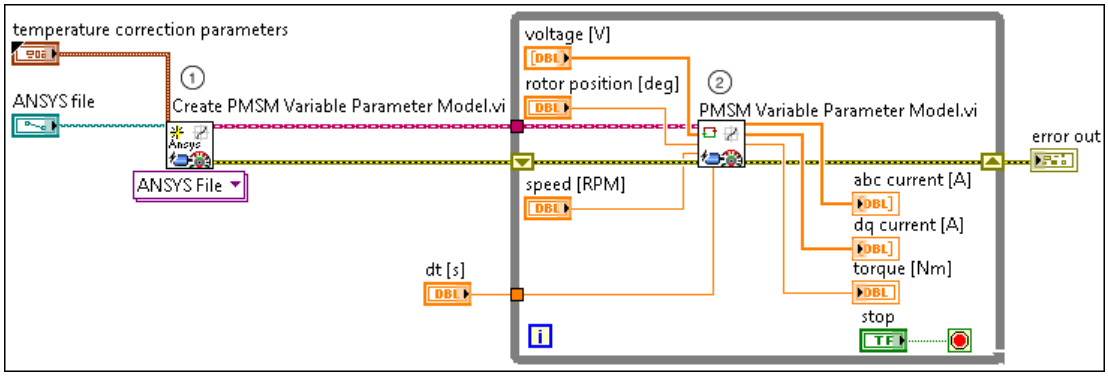

Fig. 3-10 Simularea parametrilor care țin de temperatura mediului ambiant și a componentelor.
http://zone.ni.com/reference/en-XX/help/373998C-01/lvemsim/emsim_rtt/

Ilustrația prezintă următoarea secvență de pași:

- Modelul de creare a parametrilor variabili PMSM.vi citește fișierul modelului de motor ANSYS pentru a obține parametrii motorului și datele.
- Simulatorul VI utilizează parametrii motorului și datele pe care le citează partea anterioară a acestui program pentru a simula comportamentul unui motor. În acest caz, simulatorul VI este modelul parametru variabil PMSM.vi.

Importarea unui model extern:

- pentru a utiliza modelul de parametru variabil, puteți obține parametrii motorului și datele brute prin importarea unui model extern;
- cu ajutorul Instrumentului de simulare a motorului electric, puteți importa un model extern pentru a genera o referință de clasă LabVIEW a modelului de parametru variabil. Referința de clasă LabVIEW conține parametrii motorului în formatul LabVIEW (lvclass). Formatul este util pentru simulatorul VIs pentru a obține caracteristicile motorului electric.

Următoarea figură este un exemplu care arată cum să importați un model extern și să folosiți referința de clasă LabVIEW cu simulatorul VI.

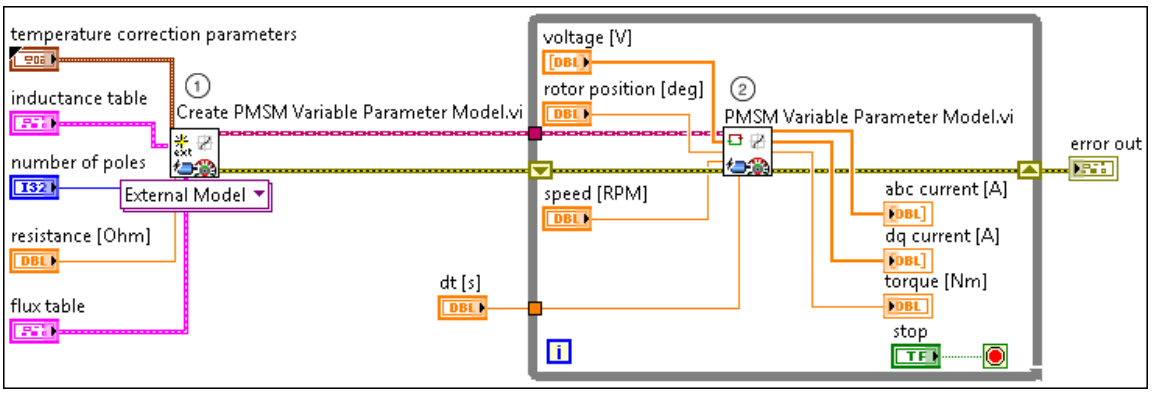

Fig. 3-11 Citirea parametrilor termodinamici.
http://zone.ni.com/reference/en-XX/help/373998C-01/lvemsim/emsim_rtt/

4. SOFTWARE CARTOGRAFIERE 3D

```
/**
* 3D modelling with spiral pattern
* Detect an object and set initial radius.
* Set an initial radius.
*/
 /*
Final functions structure:
   ->>>> After pressing the env mapping.
 Actions
 (1) Lift of the GND (3 meters)
 (2) Rotate to locate object (colour set by user)
 (3) When object found, stop rotating and move towards the object (at a slow speed).
 (4) Stop when the object is fiting the whole screen
 (5) Rotate around the POI whilst taking pic and increasing alt. (Height increment after completing every loop.
 (6) If the object lost return to action (2)
  End after completing the loops set by user
 */
#include "common.h"
#include "env_mapping.h"
#include "utility.h"
#define _USE_MATH_DEFINES
#include <cmath>
using namespace picopter;
using namespace picopter::navigation;
EnvironmentalMapping::EnvironmentalMapping(Options *opts, int radius)
: m_finished{false}
, m_radius(radius) {
}
EnvironmentalMapping::EnvironmentalMapping(int radius)
: EnvironmentalMapping(NULL, radius) {}
EnvironmentalMapping::~EnvironmentalMapping() {
}
void EnvironmentalMapping::GotoLocation(FlightController *fc, Coord3D l, Coord3D roi, bool relative_alt)
   {    GPSData d;
   double wp_distance, wp_alt_delta;
       fc->fb->SetGuidedWaypoint(0, 3, 0, l, relative_alt);
       fc->fb->SetRegionOfInterest(roi);
       do {
             if (fc->lidar) {
                  float v = fc->lidar->GetLatest() / 100;
                  if (v > 0.1 && v < 2) {
                           //STAHP
                          fc->fb->Stop();
                          break;
                   }
             }
             fc->gps->GetLatest(&d);
            wp_distance = CoordDistance(d.fix, l);
            wp_alt_delta = l.alt != 0 ?
                   std::fabs((d.fix.alt-d.fix.groundalt)-l.alt) : 0;
             fc->Sleep(100);
   } while ((wp_distance > 2 || wp_alt_delta > 0.2) && !fc->CheckForStop());
}
```

```
/*
void EnvironmentalMapping::SearchingMapping(FlightController *fc) {
    //Rotate to search
    while (!fc->CheckForStop()) {
            fc->fb->SetYaw(5, true);
             fc->Sleep(100);
             fc->cam->GetDetectedObjects(&objects);
              if (objects.size() > 0) {
                    //We have a detected object
                    break;
               }
     }
     //Move forward slowly
    Vec3D msp{0, 0.2, 0};
    while (!fc->CheckForStop()) {
       fc->fb->SetBodyVel(msp);
       fc->cam->GetDetectedObjects(&objects);
       if (objects.size() > 0) {
          ObjectInfo o = objects.front();
          if (o.bounds.area() > 0.3*(o.image_width * o.image_height)) {
             fc->fb->Stop();
             break;
             break;
          }
       }
    }
}
*/
void EnvironmentalMapping::Run(FlightController *fc, void *opts) {
    Log(LOG_INFO, "Environmental mapping initiated; awaiting authorisation...");
    SetCurrentState(fc, STATE_AWAITING_AUTH);
    if (!fc->WaitForAuth()) {
       Log(LOG_INFO, "All stop acknowledged; quitting!");
       return;
    }
    Log(LOG_INFO, "Authorisation acknowledged.");
    if (!fc->gps->WaitForFix(200)) {
       Log(LOG_WARNING, "No GPS fix; quitting.");
       return;
    }
    //Takeoff if we're not in the air     if (!fc->fb->IsInAir()) {
       UtilityModule module(UtilityModule::UTILITY_TAKEOFF);
       module.Run(fc, reinterpret_cast<void*>(3));
    }
    SetCurrentState(fc, STATE_ENV_MAPPING);
    std::vector<ObjectInfo> objects;
    //while (!fc->CheckForStop()){
    //    o.bounds.area > 0.1*(o.image_width * o.image_height))
     //double start_radius = fc->lidar->GetLatest();
    //double alt  = fc->gps->GetLatestRelAlt();
    GPSData d;
    fc->gps->GetLatest(&d);
   Coord3D centre = {d.fix.lat, d.fix.lon, d.fix.alt - d.fix.groundalt};
    centre = CoordAddOffset(centre, 7, 90-fc->imu->GetLatestYaw()); //10m in front of copter
    //centre = CoordAddOffset(centre, Vec3D{-7, 0, 0}); //5m To west of copter.
    //Ah, magic offsets
    float initial_yaw = 270-fc->imu->GetLatestYaw();
```

```
fc->fb->SetRegionOfInterest(centre);
for (int j=1; j<4; j+=1) {
   for (int i=0; i<360 && !fc->CheckForStop(); i=i+5) {
      Coord3D location = CoordAddOffset(centre, m_radius /*7 start_radius*/, i+initial_yaw);
      GotoLocation(fc, location, centre, false);
      Log(LOG_DEBUG, "%d", i);
               fc->cam->GetDetectedObjects(&objects);
      if (objects.size() > 0) {
         //Log(LOG_DEBUG, "%.7f, %.7f", centre.lat, centre.lon);
         std::string path = std::string(PICOPTER_HOME_LOCATION "/pics/mappingimages") +"_" +
         std::to_string(location.lat) + "_"+
         std::to_string(location.lon) + "_" +
         std::to_string(location.alt) +
         std::string(".jpg");
         fc->cam->TakePhoto(path);
      }
      fc->Sleep(100);
   }
   centre.alt += j;
   Log(LOG_DEBUG, "NEW LOOP");
}
//}
fc->fb->UnsetRegionOfInterest();
fc->fb->Stop();     m_finished = true; }
 bool EnvironmentalMapping::Finished() {
return m_finished; }
```

5. SOFTWARE HEXA NAVIO

```
site_name: Navio2 docs
        theme_dir: docstheme/material
        site_favicon: favicon.ico
        repo_url: https://github.com/emlid/navio2-docs
        extra:
          palette:
            primary: 'light blue'
            accent: 'light blue'
          logo: 'img/logo.png'
          social:
            - type: 'github'
              link: 'https://github.com/emlid'
            - type: 'twitter'
              link: 'https://twitter.com/emlidtech'
            - type: 'facebook'
              link: 'https://facebook.com/emlidtech'
          google_analytics:
            - 'UA-49508097-6'
            - 'auto'
        markdown_extensions:
         - markdown.extensions.admonition
         - markdown.extensions.codehilite(guess_lang=false)
         - markdown.extensions.def_list
         - markdown.extensions.footnotes
         - markdown.extensions.meta
         - markdown.extensions.toc(permalink=true)
        pages:
        - Introduction : 'index.md'
        - Hardware setup: 'ardupilot/hardware-setup.md'
        - Raspberry Pi configuration: 'common/ardupilot/configuring-raspberry-pi.md'
        - ArduPilot:
          - 'Typical setup schemes': 'ardupilot/typical-setup-schemes.md'
          - 'Installation and running' : 'common/ardupilot/installation-and-running.md'
          - 'Configuration' : 'ardupilot/tips.md'
          - 'Upgrading' : 'common/ardupilot/upgrade.md'
          - 'Building from sources': 'common/ardupilot/building-from-sources.md'
```

```
- ROS : 'common/dev/ros.md'
- Navio2 for developers:
  - 'Pinout': 'dev/pinout.md'
  - 'Video streaming': 'common/dev/video-streaming.md'
  - 'Wi-Fi Broadcast' : 'common/dev/wbc.md'
  - 'Examples setup': 'common/dev/navio-repository-cloning.md'
  - 'Barometer' : 'common/dev/barometer.md'
  - 'ADC' : 'dev/adc.md'
  - 'PWM output' : 'dev/pwm-output.md'
  - 'RC input' : 'dev/rc-input.md'
  - 'RCIO' : 'dev/rcio.md'
  - '9DOF IMU' : 'common/dev/imu-mpu9250_lsm9ds1.md'
  - 'AHRS' : 'common/dev/ahrs.md'
  - 'GPS' : 'common/dev/gps-ublox.md'
  - 'GPS via u-center' : 'common/dev/gps-ublox-ucenter.md'
  - 'RGB LED' : 'common/dev/rgb-led.md'
- Changelogs: 'common/dev/changelogs.md'
- Community projects:
  - 'Windows IoT': 'common/community-projects/Windows-IoT.md'
```

ArduPilot

```
#!/usr/bin/env python
    # encoding: utf-8
    from __future__ import print_function
    import os.path
    import sys
    sys.path.insert(0, 'Tools/ardupilotwaf/')
    import ardupilotwaf
    import boards
    from waflib import Build, ConfigSet, Configure, Context, Utils
    # TODO: implement a command 'waf help' that shows the basic tasks a
    # developer might want to do: e.g. how to configure a board, compile a
    # vehicle, compile all the examples, add a new example. Should fit in
    # less than a terminal screen, ideally commands should be copy
    # pastable. Add the 'export waf="$PWD/waf"' trick to be copy-pastable
    # as well.
    # TODO: replace defines with the use of the generated ap_config.h file
    # this makes recompilation at least when defines change. which might
```

```
    # be sufficient.
    def _set_build_context_variant(variant):
        for c in Context.classes:
            if not issubclass(c, Build.BuildContext):
                continue
            c.variant = variant
    def init(ctx):
        env = ConfigSet.ConfigSet()
        try:
            p = os.path.join(Context.out_dir, Build.CACHE_DIR, Build.CACHE_SUFFIX)
            env.load(p)
        except:
            return
        Configure.autoconfig = 'clobber' if env.AUTOCONFIG else False
        if 'VARIANT' not in env:
            return
        # define the variant build commands according to the board
        _set_build_context_variant(env.VARIANT)
    def options(opt):
        opt.load('compiler_cxx compiler_c waf_unit_test python')
        opt.load('ardupilotwaf')
        opt.load('build_summary')
        g = opt.ap_groups['configure']
        boards_names = boards.get_boards_names()
        g.add_option('--board',
            action='store',
            choices=boards_names,
            default='sitl',
            help='Target board to build, choices are %s.' % boards_names)
        g.add_option('--debug',
            action='store_true',
            default=False,
            help='Configure as debug variant.')
        g.add_option('--no-autoconfig',
            dest='autoconfig',
            action='store_false',
            default=True,
            help="'
```

```
Disable autoconfiguration feature. By default, the build system triggers a
reconfiguration whenever it thinks it's necessary - this option disables that.
''')
    g.add_option('--no-submodule-update',
        dest='submodule_update',
        action='store_false',
        default=True,
        help='''
Don't update git submodules. Useful for building with submodules at specific
revisions.
''')
    g.add_option('--enable-benchmarks',
        action='store_true',
        default=False,
        help='Enable benchmarks.')
    g.add_option('--disable-lttng', action='store_true',
        default=False,
        help="Don't use lttng even if supported by board and dependencies available")
    g.add_option('--disable-libiio', action='store_true',
        default=False,
        help="Don't use libiio even if supported by board and dependencies available")
    g.add_option('--disable-tests', action='store_true',
        default=False,
        help="Disable compilation and test execution")
    g.add_option('--static',
        action='store_true',
        default=False,
        help='Force a static build')
def _collect_autoconfig_files(cfg):
    for m in sys.modules.values():
        paths = []
        if hasattr(m, '__file__'):
            paths.append(m.__file__)
        elif hasattr(m, '__path__'):
            for p in m.__path__:
                paths.append(p)
        for p in paths:
            if p in cfg.files or not os.path.isfile(p):
```

```
                continue
            with open(p, 'rb') as f:
                cfg.hash = Utils.h_list((cfg.hash, f.read()))
                cfg.files.append(p)
def configure(cfg):
    cfg.env.BOARD = cfg.options.board
    cfg.env.DEBUG = cfg.options.debug
    cfg.env.AUTOCONFIG = cfg.options.autoconfig
    cfg.env.VARIANT = cfg.env.BOARD
    if cfg.env.DEBUG:
        cfg.env.VARIANT += '-debug'
    _set_build_context_variant(cfg.env.VARIANT)
    cfg.setenv(cfg.env.VARIANT)
    cfg.env.BOARD = cfg.options.board
    cfg.env.DEBUG = cfg.options.debug
    # Allow to differentiate our build from the make build
    cfg.define('WAF_BUILD', 1)
    cfg.msg('Autoconfiguration', 'enabled' if cfg.options.autoconfig else 'disabled')
    if cfg.options.static:
        cfg.msg('Using static linking', 'yes', color='YELLOW')
        cfg.env.STATIC_LINKING = True
    cfg.load('ap_library')
    cfg.msg('Setting board to', cfg.options.board)
    cfg.get_board().configure(cfg)
    cfg.load('clang_compilation_database')
    cfg.load('waf_unit_test')
    cfg.load('mavgen')
    cfg.load('uavcangen')
    cfg.env.SUBMODULE_UPDATE = cfg.options.submodule_update
    cfg.start_msg('Source is git repository')
    if cfg.srcnode.find_node('.git'):
        cfg.end_msg('yes')
    else:
        cfg.end_msg('no')
        cfg.env.SUBMODULE_UPDATE = False
    cfg.msg('Update submodules', 'yes' if cfg.env.SUBMODULE_UPDATE else 'no')
    cfg.load('git_submodule')
    if cfg.options.enable_benchmarks:
```

```
            cfg.load('gbenchmark')
        cfg.load('gtest')
        cfg.load('static_linking')
        cfg.load('build_summary')
        cfg.start_msg('Benchmarks')
        if cfg.env.HAS_GBENCHMARK:
            cfg.end_msg('enabled')
        else:
            cfg.end_msg('disabled', color='YELLOW')
        cfg.start_msg('Unit tests')
        if cfg.env.HAS_GTEST:
            cfg.end_msg('enabled')
        else:
            cfg.end_msg('disabled', color='YELLOW')
        cfg.env.append_value('GIT_SUBMODULES', 'mavlink')
        cfg.env.prepend_value('INCLUDES', [
            cfg.srcnode.abspath() + '/libraries/',
        ])
        # TODO: Investigate if code could be changed to not depend on the
        # source absolute path.
        cfg.env.prepend_value('DEFINES', [
            'SKETCHBOOK="' + cfg.srcnode.abspath() + '"',
        ])
        # Always use system extensions
        cfg.define('_GNU_SOURCE', 1)
        cfg.write_config_header(os.path.join(cfg.variant, 'ap_config.h'))
        _collect_autoconfig_files(cfg)
    def collect_dirs_to_recurse(bld, globs, **kw):
        dirs = []
        globs = Utils.to_list(globs)
        if bld.bldnode.is_child_of(bld.srcnode):
            kw['excl'] = Utils.to_list(kw.get('excl', []))
            kw['excl'].append(bld.bldnode.path_from(bld.srcnode))
        for g in globs:
            for d in bld.srcnode.ant_glob(g + '/wscript', **kw):
                dirs.append(d.parent.relpath())
        return dirs
    def list_boards(ctx):
```

```
    print(*boards.get_boards_names())
def _build_cmd_tweaks(bld):
    if bld.cmd == 'check-all':
        bld.options.all_tests = True
        bld.cmd = 'check'
    if bld.cmd == 'check':
        if not bld.env.HAS_GTEST:
            bld.fatal('check: gtest library is required')
        bld.options.clear_failed_tests = True
def _build_dynamic_sources(bld):
    bld(
        features='mavgen',
        source='modules/mavlink/message_definitions/v1.0/ardupilotmega.xml',
        output_dir='libraries/GCS_MAVLink/include/mavlink/v2.0/',
        name='mavlink',
        # this below is not ideal, mavgen tool should set this, but that's not
        # currently possible
        export_includes=[
            bld.bldnode.make_node('libraries').abspath(),
            bld.bldnode.make_node('libraries/GCS_MAVLink').abspath(),
        ],
    )
    if bld.get_board().with_uavcan:
        bld(
            features='uavcangen',
            source=bld.srcnode.ant_glob('modules/uavcan/dsdl/uavcan/**/*.uavcan'),
            output_dir='modules/uavcan/libuavcan/include/dsdlc_generated',
            name='uavcan',
            export_includes=[
                bld.bldnode.make_node('modules/uavcan/libuavcan/include/dsdlc_generated').abspath(),
            ]
        )
    def write_version_header(tsk):
        bld = tsk.generator.bld
        return bld.write_version_header(tsk.outputs[0].abspath())
    bld(
        name='ap_version',
        target='ap_version.h',
```

```
            vars=['AP_VERSION_ITEMS'],
            rule=write_version_header,
        )
        bld.env.prepend_value('INCLUDES', [
            bld.bldnode.abspath(),
        ])
    def _build_common_taskgens(bld):
        # NOTE: Static library with vehicle set to UNKNOWN, shared by all
        # the tools and examples. This is the first step until the
        # dependency on the vehicles is reduced. Later we may consider
        # split into smaller pieces with well defined boundaries.
        bld.ap_stlib(
            name='ap',
            ap_vehicle='UNKNOWN',
            ap_libraries=bld.ap_get_all_libraries(),
        )
        if bld.env.HAS_GTEST:
            bld.libgtest(cxxflags=['-include', 'ap_config.h'])
        if bld.env.HAS_GBENCHMARK:
            bld.libbenchmark()
    def _build_recursion(bld):
        common_dirs_patterns = [
            # TODO: Currently each vehicle also generate its own copy of the
            # libraries. Fix this, or at least reduce the amount of
            # vehicle-dependent libraries.
            '*',
            'Tools/*',
            'libraries/*/examples/*',
            'libraries/*/tests',
            'libraries/*/utility/tests',
            'libraries/*/benchmarks',
        ]
        common_dirs_excl = [
            'modules',
            'libraries/AP_HAL_*',
            'libraries/SITL',
        ]
        hal_dirs_patterns = [
```

```
            'libraries/%s/tests',
            'libraries/%s/*/tests',
            'libraries/%s/*/benchmarks',
            'libraries/%s/examples/*',
        ]
        dirs_to_recurse = collect_dirs_to_recurse(
            bld,
            common_dirs_patterns,
            excl=common_dirs_excl,
        )
        for p in hal_dirs_patterns:
            dirs_to_recurse += collect_dirs_to_recurse(
                bld,
                [p % l for l in bld.env.AP_LIBRARIES],
            )
        # NOTE: we need to sort to ensure the repeated sources get the
        # same index, and random ordering of the filesystem doesn't cause
        # recompilation.
        dirs_to_recurse.sort()
        for d in dirs_to_recurse:
            bld.recurse(d)
    def _build_post_funs(bld):
        if bld.cmd == 'check':
            bld.add_post_fun(ardupilotwaf.test_summary)
        else:
            bld.build_summary_post_fun()
        if bld.env.SUBMODULE_UPDATE:
            bld.git_submodule_post_fun()
    def build(bld):
        config_hash = Utils.h_file(bld.bldnode.make_node('ap_config.h').abspath())
        bld.env.CCDEPS = config_hash
        bld.env.CXXDEPS = config_hash
        bld.post_mode = Build.POST_LAZY
        bld.load('ardupilotwaf')
        bld.env.AP_LIBRARIES_OBJECTS_KW.update(
            use=['mavlink'],
            cxxflags=['-include', 'ap_config.h'],
        )
```

```
    if bld.get_board().with_uavcan:
        bld.env.AP_LIBRARIES_OBJECTS_KW['use'] += ['uavcan']
    _build_cmd_tweaks(bld)
    if bld.env.SUBMODULE_UPDATE:
        bld.add_group('git_submodules')
        for name in bld.env.GIT_SUBMODULES:
            bld.git_submodule(name)
    bld.add_group('dynamic_sources')
    _build_dynamic_sources(bld)
    bld.add_group('build')
    bld.get_board().build(bld)
    _build_common_taskgens(bld)
    _build_recursion(bld)
    _build_post_funs(bld)
ardupilotwaf.build_command('check',
    program_group_list='all',
    doc='builds all programs and run tests',
)
ardupilotwaf.build_command('check-all',
    program_group_list='all',
    doc='shortcut for `waf check --alltests`',
)
for name in ('antennatracker', 'copter', 'plane', 'rover', 'sub'):
    ardupilotwaf.build_command(name,
        program_group_list=name,
        doc='builds %s programs' % name,
    )
for program_group in ('all', 'bin', 'tools', 'examples', 'tests', 'benchmarks'):
    ardupilotwaf.build_command(program_group,
        program_group_list=program_group,
        doc='builds all programs of %s group' % program_group,
    )
```