**System description**

of “Pendulum”

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This document describes the process of integrating all subsystems, both software and hardware, into one functional structure. *Pendulum* is a mini project aimed at evaluating basic software for later use in flying machines.

Jakub Mnich – MikroCpp (Universal Flying Platforms) 24 October 2016

1. **General description**

*Pendulum* is a proof-of-concept device that allows to test drive regulation algorithms for copters. Mechanical part is composed of 3D printed stand with a rotational joint which gives one degree of freedom to the arm holding two BLDC motors with propellers. This setup allows for quick and safe evaluation and comparison of regulation algorithms which is a very valuable capability for Universal Flying Platforms as a custom fuzzy logic algorithm is considered as a replacement for standard PID in constructed copters.

Fig. 1 - the final version of Pendulum

1. **Mechanics**

The mechanical part of *Pendulum* was designed in student version of Autodesk Inventor 2016. Using CAD software and 3D printing allowed to build operational device with only one iteration of the design as no improvements were required.

Rotational joint is built of two bearings clenched in cavities of two plastic parts held together by a set of screws. The top part serves as a base for mounting aluminum arm profiles and specially perforated platform for electronics and wiring. The platform’s base is designed to provide stability to the entire setup and has four holes to attach weights. It also has two curves to tie limiters which prevent propellers from touching the ground and breaking.

Fig. 2 - 3D model prepared with Autodesk Inventor

1. **Electronics**

*Pendulum* is equipped with a very minimalistic set of electronics needed to mimic the real drones’ subsystems. It includes the following main components:

* Raspberry Pi 3B (RPI) **[1]**
* Pololu AltIMU-10 v4 (IMU) **[2]**
* Logic voltage converter (LVC)
* ESC regulators (ESC) **[3]**

RPI acts as a brain and communication relay with data acquiring application on PC. It is compatible with 3,3V logic signals which allows it to directly interface IMU via I2C bus but all ESCs which will be used with this device require 5V PWM or I2C so an additional LVC module is needed to allow RPI to control the rotors.

This setup contains additionally a high power li-po battery to run the motors which might drain up to 60A at about 12V.

1. **Software**

Software used in *Pendulum* is written entirely in Java which is fully supported on RPI. Using this high level language instead of C/C++ allowed to finish most parts of the software in less than three months which is a very good result as all code with the exception of data filter was developed by only one person.

*Pendulum* requires for proper operation both *Bixie* *Station* on PC and *Perun* kernel with *ADCS* (Attitude Determination and Control System) module running on the device itself. This paper does not fully cover *Perun* module and only explains Pendulum-related components of *Bixie* *Station*.

Data exchange between modules is implemented with FFDE network. **[4]**

1. *Perun* **[5] [11]**

Perun is a Java application which acts as a kernel in Pendulum and will be used also in copters. It performs several tasks:

* Handles communication via Internet or local network with the *Bixie station*.
* Identifies commands and directs them to the right listeners.
* Provides event logging functionality to all software modules.
* Runs FFDEKernel instance.

1. *Bixie station* **[6]**

*Pendulum* is mostly autonomous device therefore *Bixie’s* role is in that case very limited. It performs the following tasks:

* Allows the PC to bind with *Perun’s* communication gate
* Allows the user to set parameters like regulation algorithm type, starting angle etc.
* Gathers and visualizes data produced by *Pendulum*. (Bixie provides only online visualization and logging data to files)

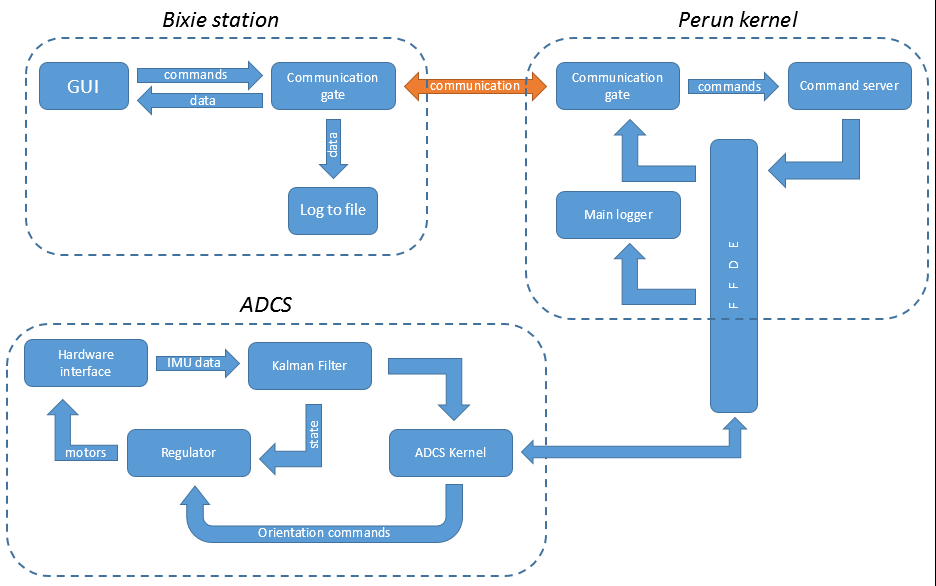


Fig. 3 - data flow diagram for software used in Pendulum

1. *ADCS* **[7]**

*ADCS* stands for *Attitude Determination and Control System.* It performs the following tasks:

* Acquires raw sensor readings from IMU
* Performs filtering on IMU data with a dedicated Kalman Filter algorithm
* Uses fuzzy logic or PID regulator to calculate an optimal drive response
* Directly interfaces ESCs of BLDC motors
* Provides flight data to the other software modules.

*ADCS* used in *Pendulum* is significantly different from the ones which will be used in real drones as it needs to handle only one degree of freedom while UAVs have three which require rapid response to ensure the vehicle’s stability (planning transitional movement is not that critical for maintaining stable flight).

This *ADCS*, despite being simplified, consist of identical functional blocks which are visualized on figure 3. Each block’s functionality is described in the following section.

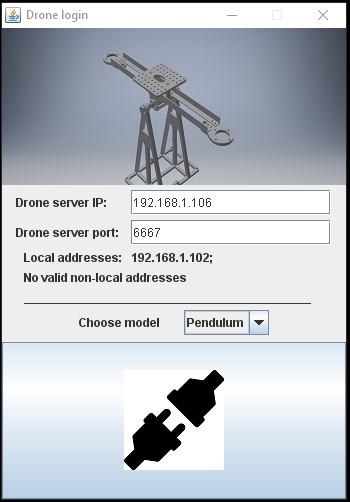
* ADCS Kernel – responsible for handling external communication through FFDE Network and therefore needs to run an instance of FFDE Server. It additionally does some preprocessing and redirects commands coming from higher-level behavioral modules.
* Hardware interface – provides the other submodules of *ADCS* with the ability to communicate with *IMU* (through I2C) and with *ESCs* (both through I2C and with PWM signal).
* Regulator **[8]** – calculates an optimal drive response. Pendulum supports PID and Fuzzy logic regulators. This feature is required for *Pendulum* to provide the team with the ability to compare effectiveness of different algorithms.
* Kalman Filter **[9]** – implements advanced data filtering and sensory fusion algorithm which accepts raw data from accelerometer, gyroscope and magnetometer and optimally predicts the most plausible attitude of the UAV in next time step.

1. **Integration process**

Overall time taken to finish *Pendulum* is difficult to estimate as obtaining all necessary parts, acquiring knowledge related to UAV technology and preparing basic software components took a few months and, at least during the initial phase, these activities were irregular and rather chaotic. First important software components were created on June while the final assembly of Pendulum took place in October 2016.

Integration process as such began one month earlier and was required due to modular design of the software made possible by FFDE Network. Integration process can be split into several phases:

1. **Preparing framework for Bixie Station [10]**

No device built for Universal Flying Platforms project can operate without human who initiates mission / experiment. The whole user interface is implemented on PC in common for all machines application called *Bixie Station*. *Bixie’s* core is responsible for establishing and maintaining communication link with the drone and allows the user to select the right interface depending on a model and type of an experiment. The way Pendulum is built allows it to use *Bixie Station* like the real drones do.

Adaptation of *Bixie Station* for *Pendulum* included preparing specialized telemetry window for GUI and creating a logger. The window contains six dynamic charts for visualizing readings from IMU and Kalman filter output. It also contains interactive components required for conducting experiments on Pendulum.

1. **ADCS**

*ADCS* for *Pendulum* is a simplified version of what real drones need to maintain a flight – it servers here mainly as a technology demonstrator and evaluation platform. Its submodules are described in great detail in other papers from MikroCpp documentation repository.

1. **Implementing FFDE in all modules**

Software driving *Pendulum* is designed in the same modular way that the drones’ software will. FFDE enables all components to work independently and still allows them to perform simple and reliable communication through local host TCP ports. The following modules are fitted with separate FFDE Servers:

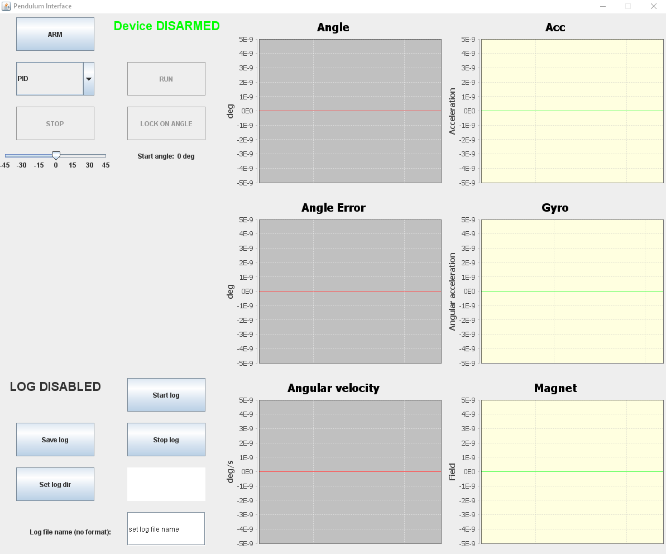
* Global Communication Gate
* Global logger
* Global command server
* ADCS kernel

Fig. - Bixie Station Pendulum interface

*ADCS* components cannot be internally connected via FFDE Network as they need to operate in real time and FFDE would introduce a delay of about 3-4ms in data exchange.

Fig. - Bixie Station login window

1. **Mechanical construction of Pendulum**

This phase was one of the quickest as *Pendulum* relays on 3D printed plastic and standard aluminum parts – both types can be obtained in a short time.

1. **Connection and a basic test**

As simple as *Pendulum* is it still required a few days of debugging in order to fully launch all of its basic functionalities but it was much easier to identify and eliminate problems thanks to clear distribution of tasks performed by each subsystem and high reliability of FFDE.

**References:**

1. Raspberry Pi official webpage [https://www.raspberrypi.org/]
2. Pololu AltIMU-10 v4 manufacturer’s webpage [https://www.pololu.com/product/2470]
3. **Szymon Rzążewski, Jakub Mnich** (2016) Custom ESC for BLDC motor - design and implementation. *MikroCpp documentation repository*.
4. **Jakub Mnich** (2016) FFDE Network – Foundation For Data Exchange. *MikroCpp documentation repository*.
5. Github repository with Perun [https://github.com/jmnich/UFP\_Pendulum\_Perun]
6. Github repository with Bixie Station [https://github.com/jmnich/UFP\_Bixie\_Station]
7. Github repository with ADCS for Pendulum [https://github.com/jmnich/UFP\_Pendulum\_ADCS]
8. **Jakub Mnich** (2016) Fuzzy regulator for UAV – Simulation approach. *MikroCpp documentation repository*.
9. **Paweł Polański** (2016) Kalman Filter for UAV. *MikroCpp documentation repository*.
10. **Jakub Mnich** (2016) Bixie Station – Description. *MikroCpp documentation repository*.
11. **Jakub Mnich** (2016) Perun Kernel – Description. *MikroCpp documentation repository*.