Cahit Yusuf Taş
cyusuftas@gmail.com

DRONEKIT

Part 1: Connection, Read and Set Values

- Import necessary libraries like below:

```
from dronekit import connect, VehicleMode
```

  - ➔ connect: It is used to connect to the vehicle and returns a Vehicle object. Also it takes parameters including IP, wait_ready, baud.
  - ➔ VehicleMode: This object is used to get and set the current flight mode.
- Connect to the vehicle by using the following:

```
vehicle = connect(connection_string, wait_ready=True, baud=921600)
```

- Now we have a Vehicle object named vehicle. Vehicle state is exposed through 'attributes' (e.g heading). All attributes can be read however only some of them are settable (mode, armed and airspeed). Commands to change a value is not guaranteed to succeed and code should be written with this in mind. Check whether the set value is really set. Example is shown below:

```
print 'Roll: %s, Pitch: %s' % (vehicle.attitude.roll, vehicle.attitude.pitch)  #READ

vehicle.mode = VehicleMode('GUIDED')                                          #SET

vehicle.armed = True                                                          #SET
```

- Vehicle 'settings' (parameters) are read/set using the parameters attribute. Parameters can be iterated and are also individually observable. Full parameter list can be accessed from the Ardupilot Website. They can be read and set as follows:

```
print 'Param: %s' % vehicle.parameters['Q_ENABLE']      #READ

vehicle.parameters['Q_ENABLE'] = 1                      #SET
```

Part 2: Observing Attribute Changes

- Attribute changes can be observed and necessary actions can be done in the callback functions. To do that we need a callback function and add a listener to the observed attribute.
- Define a callback function which takes three parameters: self, attr_name, value. attr_name represents the name of the changed attribute, value represents the value of that attribute. Value of the attribute can be manipulated inside the callback function.

```
def callback(self, attr_name, value):
    print 'New value of %s: %s' % (attr_name, value)
```

- Assume we are going to observe the changes in the heading. Listener can be added by using add_attribute_listener method. It takes two parameters: Attribute name and callback function. Then whenever the listened attribute changes the callback function is called.

```
vehicle.add_attribute_listener('heading', callback)
```

- Any parameter change also can be observed by using wildcard symbol '*'.

```
vehicle.parameters.add_attribute_listener('*', callback)
```

Cahit Yusuf Taş
cyusuftas@gmail.com

Part 3: Mavlink Messages

- dronekit-python introduces easy ways to interact and communicate with the vehicle. The communication is done by using Mavlink messages. Alsocspecified Mavlink messages may be used instead of ready to use functions.
- To send a Mavlink message we can use send_mavlink function of dronekit Vehicle class. It takes the message to be sent as only argument.
- The message needs to be encoded properly. To do that message_factory() uses a factory method for the encoding. The name of this method will always be the lower case version of the message/command name with _encode appended. Also proper arguments need to be given. All Mavlink messages and argument lists can be found in this link.
- If arguments of the message includes target_system and target_component, they can be set to 0 since Dronekit automatically fills them.
- Let's investigate a message: PARAM_SET. It is used to set a parameter. We can see that it takes 5 arguments: target_system, target_component, param_id, param_value and param_type. target_system and target_component may be set to 0 as we discussed. Parameter's name is given as param_id and the value to be set is given as param_value. Also the type of the param_value we set is given as param_type. See the descriptions of the arguments in the link given above. Assume you connected succesfully so you have a vehicle object of Vehicle class and you want to set 'FLTMODE1' parameter. It could be any parameter. The code for this specific Mavlink message(PARAM_SET) would be like this:

```
msg = vehicle.message_factory.param_set_encode(
0,0,                    #target_system, target_component
'FLTMODE1',             #param_id
1,                      #param_value
1)                      #1: 8-bit unsigned integer. See MAV_PARAM_TYPE enum.

vehicle.send_mavlink(msg)
```

- Now if we look at the value of the parameter from any ground station software or by using ready to use parameters attribute of dronekit it must have changed to 1.
- Alright, we have set the value but how can we read any parameter value or all of the parameters or any other Mavlink messages emitted from the vehicle. After we send the command to set the parameter value, vehicle emits the PARAM_VALUE Mavlink message. Same message is also emitted by the vehicle after we requested to read any parameter by using PARAM_REQUEST_READ command or all of the parameters by using PARAM_REQUEST_LIST command. We need to listen for this message and act on it. Dronekit has an elegant way to handle received Mavlink messages. It uses decorators and we just need to specify the name of the message to be listened. The next function from the decorator has exactly 3 arguments: self, name, message. Manipulation of the message takes place in this function.
- An example code to send PARAM_REQUEST_READ message to the vehicle:

```
msg = vehicle.message_factory.param_request_read_encode(
0,0,                    #target_system, target_component
param_id,               # string id of the parameter
-1)                     #see the argument descriptions
vehicle.send_mavlink(msg)
```

- An example code for the PARAM_VALUE message to be listened:

```
@vehicle.on_message('PARAM_VALUE')
def my_method(self, name, msg):
    print '%s: %s' % (name, msg)              #prints the name and the message of the function
```