Design of a Distributed Kite Power Control System

Uwe Fechner and Roland Schmehl
Institute for Applied Sustainable Science, Engineering and Technology
Delft University of Technology, The Netherlands
Email: u.fechner@tudelft.nl, r.schmehl@tudelft.nl

Abstract—Kite power is a promising innovative technology for converting wind energy into electricity at a higher capacity factor and, for many applications, at a lower cost than conventional wind turbines. However, accessing this potential depends substantially on the availability of sophisticated control systems. Delft University of Technology is developing a kite power generator which operates a tethered inflatable membrane wing in a pumping cycle. The flight trajectory is controlled by an actuator unit suspended below the wing and communicates with the ground station control centre via a fast and reliable wireless link. The link is also used to transmit the data of the on-board sensors to the ground. In a future wind park of many kite power systems, the individual kites and ground stations have to communicate among each other, to avoid collisions and to optimize the total energy output of the park. A preparatory analysis has shown that the current prototype would significantly benefit from a distributed control system approach, achieving higher efficiency and increased operational flexibility. For larger installations a distributed control system would be mandatory anyway. For these reasons, a distributed control system with a flexible architecture has been developed. The unique design and first test results are presented.

I. INTRODUCTION

Wind at higher altitudes is stronger and steadier compared to wind accessible by tower-based wind turbines. One of the possible solutions to access this energy potential is to convert the traction power of a kite by an electrical generator on the ground. Continuous operation of such a system requires a periodic alternation between reel-out and reel-in of the tether (see Fig. 1). In the reel-out phase of each cycle, the kite is operated in "crosswind" figure-of-eight flight manoeuvres, delivering a high traction force which is converted into electrical energy by the drum/generator module of the ground station. In the reel-in phase, the generator is operated as a motor. The wing is "de-powered" by reducing its angle of attack such that it can be pulled back towards the ground station using only a small fraction of the generated energy.

The prototype system of Delft University of Technology is based on an inflatable membrane wing which is controlled by an actuator unit (control pod) suspended below the wing (see Fig. 2). This control pod incorporates two micro winches, which operate on the two steering lines attached to the rear ends of the wing tips. For steering, one line is pulled while the other line is released. For modifying the angle of attack of the wing both lines are pulled or released simultaneously. However, the large gear ratio of the micro winches allows only for limited actuation speeds and, consequently, they can not be used to adapt the wing to rapid changes of the wind environment, e.g. wind gusts. This functionality is provided by the drum/generator module of the ground station. With short response time and high reeling speeds, its constant force mode can effectively be used to avoid overload situations.

A main advantage of the airborne control pod is the minimal mechanical delay between activation of the micro winches and dynamic response of the wing. Another advantage is that the kite can still be controlled if the main tether ruptures. However, the airborne unit requires a reliable long-distance communication link to the ground station, an on-board power supply and its extra weight has a negative impact on the power output. Other prototype systems implement the steering and de-powering actuators as part of the ground station [1], [2]. Although this solution eliminates the added mass of the airborne unit, the multiple
tethers increase the aerodynamic drag and thus reduce the achievable traction power during reel-out.

II. DESIGN OBJECTIVE

The prototype system has originally been designed for manual control of ground station and kite. Each having their own operator, these two subsystems have in fact only been coupled mechanically, by the tether. The current development target is automatic operation and to achieve this, a fundamental architectural change of the control system has been required, with software and hardware components having to comply to a number of specific requirements.

The design of the control system accounts for the modular architecture of the prototype system and particularly aims at an easy integration of new control computers, sensors and actors. Physical links between the control nodes can be established by different types of wired links (e.g. serial, Ethernet, CAN bus) or wireless links. This framework, in which the clocks of nodes are synchronized with a precision of at least 1 ms, also allows for redundant links to achieve the necessary reliability. The software design incorporates distributed data processing including transfer of software components like state estimators and controllers from one node to another without changes to the code. Also included is a central logging with straightforward addition of new fields to the log messages.

The initial idea was based on an implementation in the Open Robotics Control Framework OROCOS. However, OROCOS offers by default only CORBA (Common Object Request Broker Architecture) as a communication middleware, which is complex to use (requires a separate request broker) and does not offer any real-time behaviour by default. For these reasons, a new high-level framework for distributed control was developed. OROCOS can still be used on single nodes within the new framework, if special features of OROCOS are needed that are not yet offered by the new framework.

III. STRUCTURE OF THE CONTROL SYSTEM

A. Overview

In Fig. 3 the overall structure of the control system is illustrated. The User Interface (UI) provides top and front view displays for the current position of the kite, numerical displays for the current altitude, tether reel-out speed and force, generated power etc. Connected to the UI is the finite state machine (FSM) controlling the operational state of the system: starting, landing, reel-out and reel-in. It gives control commands to the flight path planner and the winch controller. The flight path planner calculates the optimal flight trajectory depending on the current wind velocity, the system state and the current tether force and generator power.

The winch controller manages the drum/generator module. Mounted on a sled which is driven by a spindle motor, this linear-displacement module ensures that the tether is evenly fed onto the drum. At low wind speeds it optimizes the combination of reel-out speed and force to achieve maximum traction power. At high wind speeds it keeps the average force as high as possible without exceeding the maximal allowed value. In the reel-in phase it ensures that the tether tension never drops below the required minimal value for robust flight control of the kite, while minimizing the reel-in time and reel-in energy. The objective of the flight path controller is to minimize the error between the actual and the planned flight path. It uses an inner loop, which controls the heading of the kite, and an outer loop, which controls its position.

The kite state estimation is based on the information from the following sensors:

- Global Satellite Navigation System (GNSS), attached to the main strut of the kite;
- Inertial Measurement Unit (IMU), at the same location;
- Wind speed sensor, suspended in the bridle system between the kite and the control-pod (see Fig. 2).

B. Small earth reference frame

To understand how the control system is working it is necessary to introduce the small earth reference frame. The position of the kite and the ground station are measured in the "Earth Centered Earth Fixed" reference frame. They first have to be converted into the "Wind Reference Frame" as shown in Fig. 4. The origin of the wind reference frame is at the lower end of the tether and it’s $x_w$ coordinate is always pointing in downwind direction. To obtain the coordinates of the kite in the small earth reference frame it’s position is projected on the unit sphere around the origin of the wind reference frame. Now the position of the kite can be described with two angles, the azimuth angle $\xi$ and the elevation angle $\eta$. The movement of the kite in the direction of the tether is controlled by the winch controller and can be ignored by the kite controller. Currently the kite shall always fly on a prescribed trajectory. The heading angle of the kite can be used to steer the kite from its current position to the desired position. If we assume a straight tether and further assume that the heading of the kite is perpendicular to the tether than the angle $\Theta_{SE}$ is the same as the heading of the kite. The control objective of the kite control subsystem is to control this angle. This can be achieved by controlling the
steering signal that is sent to the control pod of the kite. By introducing the small earth reference frame the kite control problem has been reduced to a SISO problem [3].

C. Detailed control structure

In Fig. 5 the resulting, simplified control structure is shown. On the right hand side the software components that provide the sensor data from the kite and the ground are shown. In principle only the position and orientation of the kite, the position of the lower end of the tether (the ground-station) and windspeed and direction are needed. From this data the SystemStateEstimator calculates the coordinates of the kite in the small earth reference frame (elevation and azimuth) and the heading of the kite.

The SystemStateControl block at the top left generates the desired trajectories. In the most simple case for the reel-in phase a one point trajectory is generated (zenith) and for the reel-out phase a two point trajectory (one point on the right and one on the left side of the wind window).

The FlightPathPlanner chooses an appropriate point on the desired trajectory as current flight destination. The CourseController calculates the heading, that is needed to reach this destination (the bearing) using the great circle navigation [3]. In the most simple case a parameter varying PI controller is used with the bearing (desired heading) as set point and the measured heading as actual value. The output of this LPV controller is then used as steering input for the kite. Currently, the depower setting of the kite is the only parameter that is used to vary the settings of the PI controller. This is needed because the response of the kite to a steering input heavily depends on the depower settings. Better results (more constant power output) can be achieved with a four point trajectory. Even better results can be achieved with a controller that compensates the non-linear behaviour of the plant with feedback linearisation. Currently a one point LPV controller is used for the reel-in phase and a two-loop bearing/attitude controller is being used for the reel-out phase.

D. Distributed control

Distributed control systems use sensors and actuators at different places, that are part of at least one critical control loop and usually connected via a wireless link. When designing the control algorithm the properties of this link (latency, packet losses) have to be taken into account. Using distributed control is necessary for the implementation of a reliable kite power control system mainly because using a GNSS is not sufficient to determine the position of the kite reliably. GNSS sensors can fail for a lot of different reasons. Therefore for commercial kite power systems at least an additional position sensor is needed, for example angular sensors that measure the elevation and azimuth angles directly at the place where the tether is leaving the winch. For research and development there are additional reasons why a distributed control system with the controller at the ground has big advantages: Test and development of new controllers on a desktop computer is much easier and faster than doing the same thing on an embedded computer attached to the kite. And for implementing and testing of computational intensive control algorithms like "non-linear model predictive control" it is a big advantage if a powerful but heavy control computer at the ground can be used.

IV. SOFTWARE ARCHITECTURE

A. Operating system and framework

1) Operating system: Since several different computers are used for control and measurement purposes (currently three at the ground and two airborne) a modular software structure is required with components that can be developed independently and can be deployed to any of the kite power control computers. The requirements regarding the timing of the software components and communication infrastructure are tight: Some kite control algorithms fail, if the latency between measuring the state and the reaction of the actor exceeds 100 ms. A test of a kite control system, using a predecessor of the control algorithm described in [3] did not show a stable control behaviour. One possible reason was,
that the latencies in the control loop might have been too high.

Because the round-trip delay is composed of many component delays (see table I) each of these delays must be as short as possible. To fulfill these requirements, Linux configured for low latency was chosen as operating system as suggested by Bruzzone [4].

<table>
<thead>
<tr>
<th>Software components</th>
<th>High level framework DKCS</th>
<th>Low level framework OROCOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework for software components</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Components as threads</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Components as programs</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Components on many different computers</td>
<td>++</td>
<td>+/-</td>
</tr>
<tr>
<td>Many programming languages supported</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Hard real-time capable</td>
<td>+/-</td>
<td>++</td>
</tr>
<tr>
<td>Good Ethercat drivers</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Toolbox for estimation and control</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Tools for central configuration</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Simple to use and install</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Latency monitoring</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Clock, synchronized between computers</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

2) High level framework: For the communication between computers and between the not-so-very time critical software components the “distributed kite control software” (DKCS) framework was developed. It is based on the ZeroMQ (see section: IV-C.1) messaging library and the Google Protocol buffer data format (see section: IV-C.2). For time synchronization a network time protocol (NTP-) time server with a GNSS time reference and “chrony” “time clients are used. The software “chrony” has a very good performance for computers that don’t have a permanent internet connection (see: http://chrony.tuxfamily.org/). A startup time (until a control computer is synchronized with an error of less than one ms to the main time server) of less than one minute was measured. The latency of all data links and all critical software components is permanently monitored.

3) Low level framework: For the winch control computer even tighter timing requirements should be met. The generator/ motor of the winch can be very fast, e.g. accelerating from zero to full speed in one second. To control this kind of acceleration, medium-hard real-time is required. For this purpose the software framework “OROCOS” [5] together with the “Xenomai” Linux real-time extensions [6] shall be used. For this combination many software drivers for Ethercat I/O modules are available. This makes it easy to implement fast motor control loops.

A comparison of the properties of the high level framework and the low level framework is shown in table II.

B. Software components

The following software components are currently implemented:

- **KiteCtrlDaemon**: A daemon that is reading the data from the GNSS and IMU, doing analog measurements of the cable force and wind speed and sending control commands to the motor controller. At the same time it receives commands from the ground and sends the current state of the kite and the control-pod to the ground.

- **FlightPathPlanner**: Calculates the optimal flight tra-
jectory based on the system state, the wind speed and direction and the force and power data from the winch.

- **CourseController:** Keeps the kite on the calculated flight path, using the estimated kite state and the data from the flight path planner as input.

C. Communication between software components

The communication between software components shall happen using a protocol and transport layer compatible to different CPU architectures and having the same interface for in-process, interprocess and inter-computer communication. Also multiple programming languages shall be supported (some students and team members prefer Java or C# to C++). Hard real-time is normally not required, but shall be possible if needed.

1) Transport layer: To fulfill these requirements the following solution was chosen:

- For the transport layer the ZeroMQ [7] library was chosen. From our own experience we can say that it is easy to use, does not require a "message broker" and offers bindings to very many programming languages. According to [8] it is one of the fastest middleware solutions for the transport of structured data, if combined with "Google protocol buffers". It has very good support for the "publish subscribe" pattern. This is very useful in the given application. All sensors can publish their measurements to make them available for all software components.

- Alternatively, UDP can be used for message transport. Currently it is used on the wireless link, because it has a lower overhead than ZeroMQ.

- As a third alternative a serial link can be used. This is helpful, if embedded computers must be connected, that have no Ethernet interface, or for slow, long-range wireless links.

2) Serialization format: As serialization library "Google protocol buffers" [9] was chosen. This library supports the strongly typed transport of messages over any binary link. The format of the messages is very flexible, for example you can use optional or required fields, variable sized arrays etc. It can be compared to XML, but is much faster and much easier to use. All major programming languages are supported. The binary encoding is quite compact. The size of the messages is about one third of the size of ASCII encoded messages, but there remains an overhead of about 30% compared to hard coded messages with a fixed size.

Currently, 19 message types and two enumerations for the communication with the control-pod are defined, 10 message types and three enumerations for the communication with the winch and eight message types and three enumerations for internal system communication.

3) Wireless link: The XBee link that was used in the past has a round trip delay (for messages with 20 bytes uplink and 80 bytes downlink) of about 110 ms in average, but on a distance of 400 m the package loss rate is about two percent. If maximal two subsequent packages are lost, this is an equivalent of a worst case round trip delay of 330 ms. This is not sufficient for a reliable kite control system.

Therefore a new link was developed. Basic design ideas:

- Use at least two antennas in the control-pod, to be less dependent on the relative orientation of sender and receiver and of fading.

- Use a link with a high data rate and short ping time, so that even if a message has to be repeated three times the maximal allowed time limit is still met.

- Avoid the 2.4 GHz frequency band, because it is very crowded and the risk of interference with other users is high.

Wewetzer writes in [10] about his tests using a WLAN 802.11a link for vehicle to vehicle communication:

"We measured a maximum communication range around 1450 meters, while the first packets got lost at about 750 meters distance."

This sounded promising. WLAN 802.11n should have an even better performance, because it can be used with two antennas on each side and an improved modulation scheme. To improve the range even further, it was decided to use a directed antenna at least at the ground. With this setup (a WLAN-n router attached to the kite and an access point with a directed antenna mounted at the ground station in such a way, that it is always directed at the kite) the performance that is shown in Fig. 6 was measured. This is the statistics of one flight including start and landing and flying figures of eight as in normal operation. The tether length (distance) during operation was between 300 and 620 m and the frequency 5.5 Ghz. The packet frequency was 20 Hz, the packet size about 150 Bytes. The mean round-trip-time was 4.7 ms, the max. round-trip-time 12.5 ms. Not a single packet loss was recorded.

The longest distance that was tested was 1000 m. Even on this distance no packet losses were recorded. For the link on both sides antennas with an opening angle of about 60 degree were used.

V. PERFORMANCE OF THE CONTROL SYSTEM

The control system was successfully tested on more than 15 flights of about two hours each. It is very easy to maintain: even during the flight it is possible to change the source code of any of the controllers, recompile it and run it again.
Fig. 7 shows the flight path of the kite at a wind speed of 2.8 m/s at the ground in 6 m height. It shows two figures of eight during the reel-out phase, then the kite is moving upwards to increase the elevation angle and reduce the force and then it is reeled in again. The duration of the cycle was 109 s. Fig. 8 shows a screenshot from the front view display.

It shows the desired trajectory (the closed figure of eight), the last 20 seconds of the actual trajectory, the heading, bearing and course. It can be seen that the tracking performance is already very good.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

For kite power systems that use an airborne control pod a fast and reliable distributed control system is proposed. The design of a distributed control system was presented that meets all the requirements of a modular kite control system. It is well suited for research, because many programming languages can be used and components can be developed independently, based on a common communication protocol. But it is also feasible for commercial control applications, because it is very easy to add redundant communication links and control components.

By using standard Ethernet connections, the wired connections became faster, more reliable and more simple compared to the USB/serial adapters and the many serial cables, that were used in the past.

It has been shown that the suggested wireless link is fast enough for a tracking kite controller with a good accuracy. This can significantly reduce the costs of kite power systems. A SISO LPV controller is sufficient to obtain stable control of the given kite-power system. With a two-loop controller that uses feedback linearisation a good tracking performance can be achieved.

B. Future Work

Different control strategies have to be tested and compared. The new winch control system has to be implemented. A control system for automated starting and landing has to be implemented. The power output has to be optimized. The reliability has to be proven. And finally: The control system has to be extended for the control of many kites in a kite-power park.

VII. ACKNOWLEDGMENTS

The authors would like to thank fellow team members Aart de Wachter and Rolf van der Vlugt for the development and testing of the kite power system, Antonio Ramos for the research on fast and reliable wireless links, Filip Saad and Sergiy Ulyashyn for their contribution to the software development and Claudiaus Jehle for proof-reading the paper. The financial support of the Rotterdam Climate Initiative and the Province of Friesland is gratefully acknowledged.

REFERENCES