WindESCo

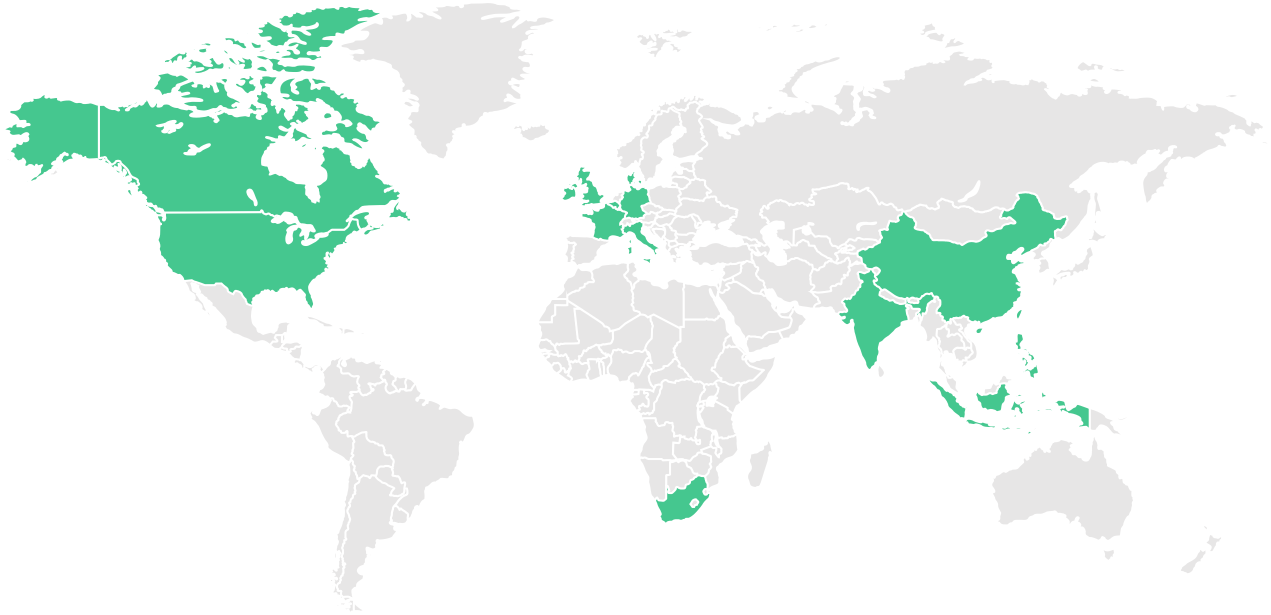
Systems engineering after the fact

Paying back technical debt in wind energy technology

Pete Bachant, PhD, Director of Software Engineering

WindESCo

Trusted by the largest wind energy operators globally



Unlocking value for wind energy

14
Countries

42
Customers

10 GWs
Under Contract

58 turbine models
Analyzed

19
OEMs

WindESCo

WindESCo optimization solutions

#1 Find, Fix, Measure Turbine-level optimization	**#2 Swarm** Wind plant-level optimization
✓ Domain expertise combined with AI turns data into increased revenue	✓ Autonomous, cooperative control system for wind plant-level optimization
✓ Scalable and cost-effective solution to deliver a 1–2% increase in revenue	✓ Delivering 3–5% increase in plant output
✓ Leverages high-speed data (600x more data than industry standard)	✓ OEM agnostic retrofit solution for both onshore and offshore wind plants
✓ Software-only solution; no warranty impact	✓ Combined software and IoT hardware solution without warranty impact
The most advanced wind energy performance analytics platform	First commercially available retrofit solution for plant level optimization
10 GW under contract	4 wind plants (400 MW) under contract within 12 months of product launch

WindESCo

Technical debt

Usually talked about in the context of software engineering

"…the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer" – Wikipedia

We can draw some parallels with wind energy technology:

An engineered-in characteristic that makes a system harder to maintain or improve

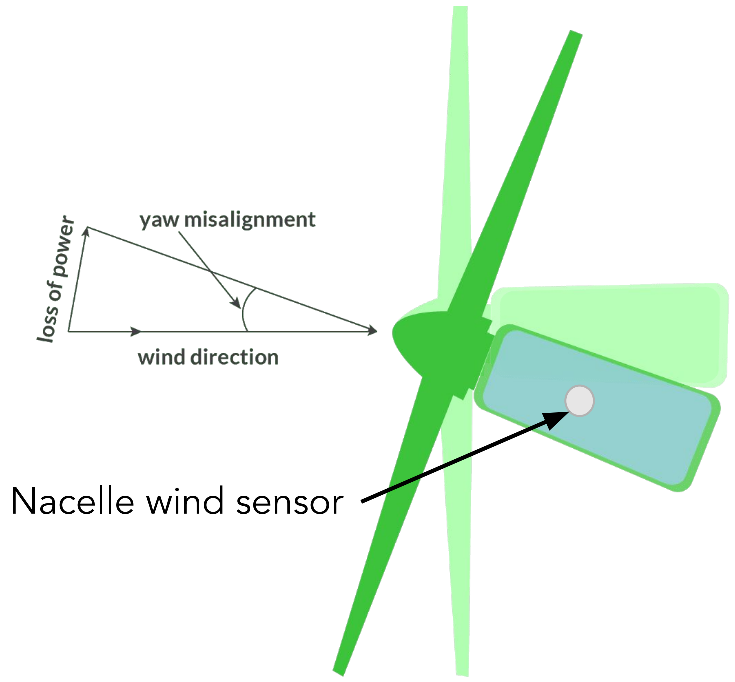
Can incur when requirements aren't perfectly defined, but with imperfect knowledge, requirements will always be imperfect → debt is harder to pay back when systems are not evolvable

"Interest" on debt → lost energy

Example: Yaw misalignment

Turbines use a relative wind direction measurement at the nacelle, yaw error, seek to keep this at zero, assuming that will optimize performance

Flow is distorted behind the rotor, so controllers attempt to correct for this in various ways: mechanical offset, a hard-coded offset in control software, parameterized offset, lookup table as a function of wind speed



WindESCo

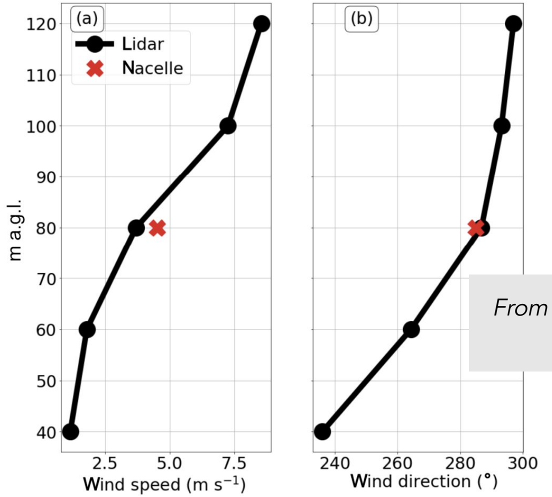
Example: Yaw misalignment

The vertical location of the rotor that should be aligned with the flow changes due to shear/veer—not always the center!

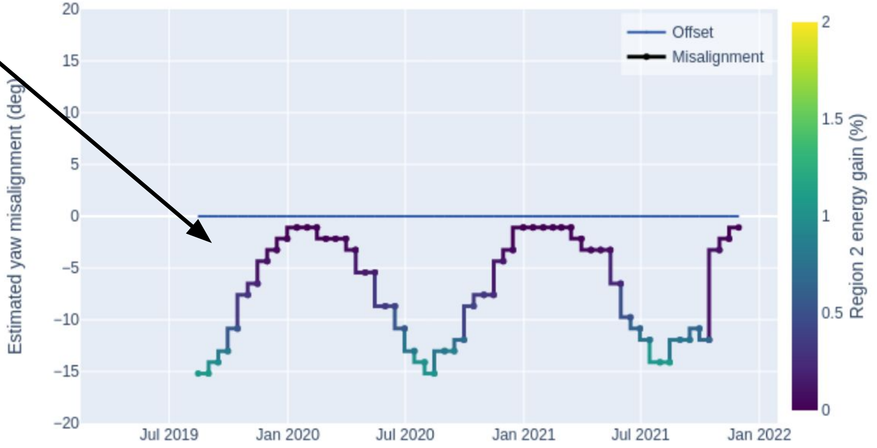
Through our yaw misalignment detection algorithm we have also found that this can be highly seasonal

Can also change due to sensor replacement, mounting issues, etc.

Tech debt: There is no one-size-fits-all yaw error offset, and it will need to be updated in time to maximize performance



From Murphy et al. (2020)



WindESCo

Example: Controller parameter peer outliers

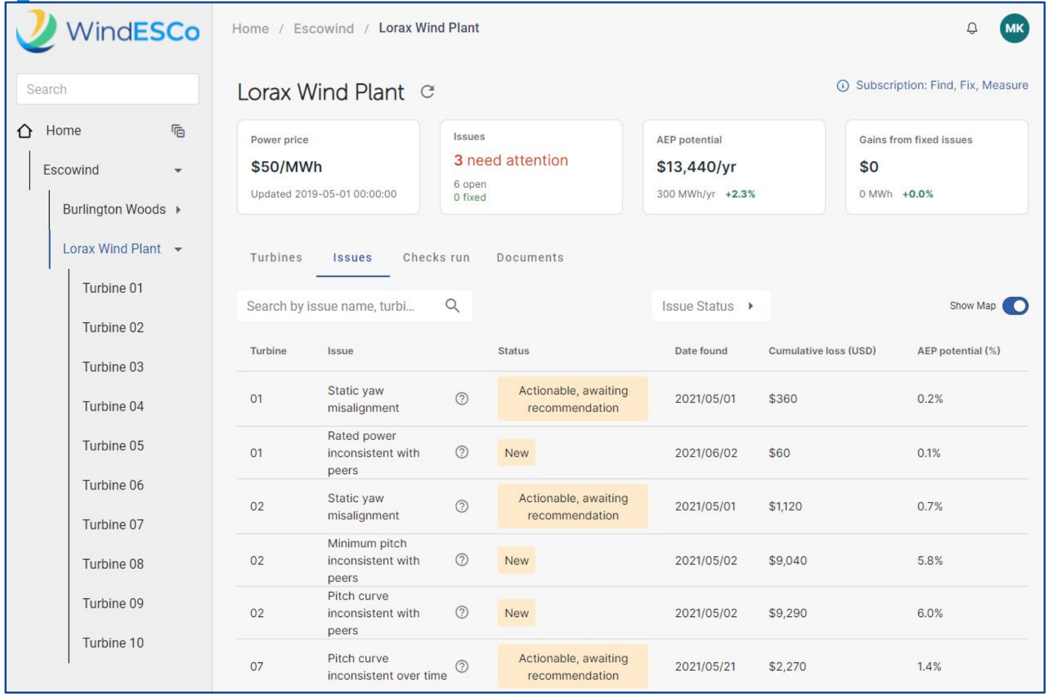
Sometimes a turbine is behaving “normally,” but there was some kind of human error

Examples: Power limit was reduced to keep turbine online until gearbox could be replaced, minimum pitch was adjusted as part of an experiment

Typically, there is no farm-level checking for turbine control parameters, so they may be set with unintended values (losing potential energy production) and not caught for a long time

SCADA alarms don’t detect every possible issue at a wind plant, so we set out to turn our analytics into a SaaS product to help address some of these deficiencies

Making this into a product: *Find, Fix, Measure*



WindESCo

Search

Home

Escowind

Burlington Woods

Lorax Wind Plant

Turbine 01
Turbine 02
Turbine 03
Turbine 04
Turbine 05
Turbine 06
Turbine 07
Turbine 08
Turbine 09
Turbine 10

Home / Escowind / Lorax Wind Plant

MK

Lorax Wind Plant

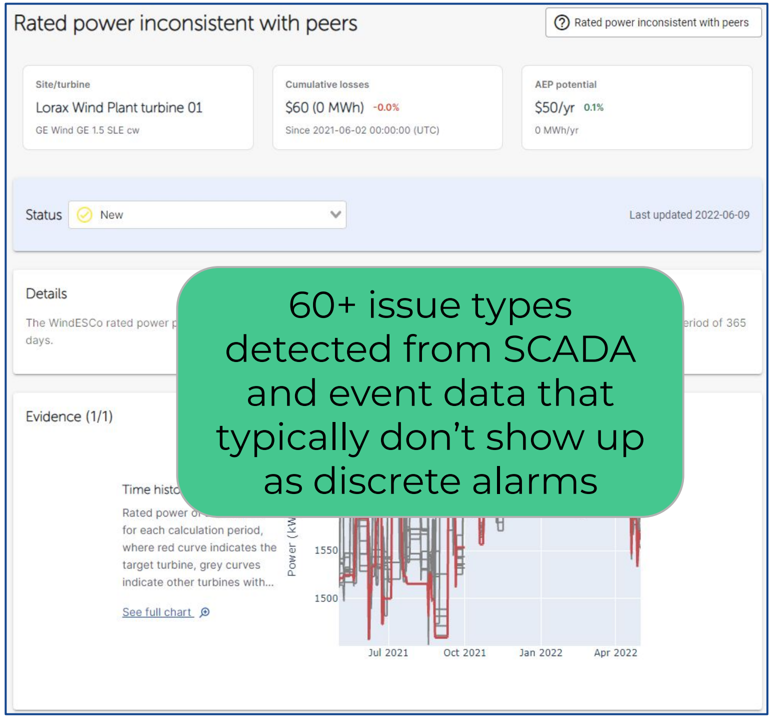
Subscription: Find, Fix, Measure

Power price	Issues	AEP potential	Gains from fixed issues
$50/MWh	3 need attention	$13,440/yr	$0
Updated 2019-05-01 00:00:00	6 open, 0 fixed	300 MWh/yr +2.3%	0 MWh +0.0%

Turbines | Issues | Checks run | Documents

Search by issue name, turbi... | Issue Status | Show Map

Turbine	Issue	Status	Date found	Cumulative loss (USD)	AEP potential (%)
01	Static yaw misalignment	Actionable, awaiting recommendation	2021/05/01	$360	0.2%
01	Rated power inconsistent with peers	New	2021/06/02	$60	0.1%
02	Static yaw misalignment	Actionable, awaiting recommendation	2021/05/01	$1,120	0.7%
02	Minimum pitch inconsistent with peers	New	2021/05/02	$9,040	5.8%
02	Pitch curve inconsistent with peers	New	2021/05/02	$9,290	6.0%
07	Pitch curve inconsistent over time	Actionable, awaiting recommendation	2021/05/21	$2,270	1.4%



Rated power inconsistent with peers

Rated power inconsistent with peers

Site/turbine	Cumulative losses	AEP potential
Lorax Wind Plant turbine 01	$60 (0 MWh) -0.0%	$50/yr 0.1%
GE Wind GE 1.5 SLE cw	Since 2021-06-02 00:00:00 (UTC)	0 MWh/yr

Status: New

Last updated 2022-06-09

Details

The WindESCo rated power p… eriod of 365 days.

Evidence (1/1)

Time histo…

Rated power o… for each calculation period, where red curve indicates the target turbine, grey curves indicate other turbines with...

See full chart

60+ issue types detected from SCADA and event data that typically don't show up as discrete alarms

Continuously scan through HF SCADA data from a wind plant, finding issues not caught by existing systems, estimating losses, tracking them over time to minimize the negative impact of the installed assets' technical debt

WindESCo

Example: Wake losses and wake steering

Farms were historically not engineered to have collective control systems, e.g., to mitigate wake losses

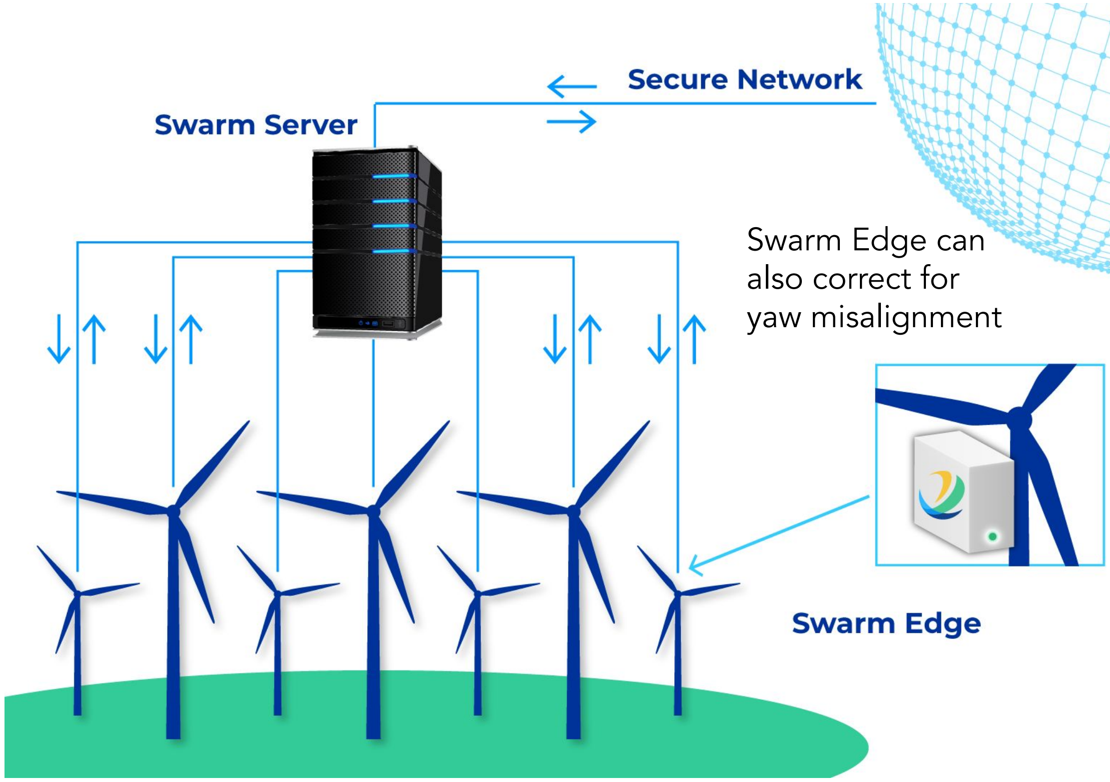
A common idea to design a wake steering controller that adds an offset to the wind direction sensor signal to achieve a yaw misalignment

Potential technical debt with these kinds of wake steering controllers

- Measured yaw error is not necessarily linear, i.e., when yawed, measurement of the relative wind direction is distorted differently → hard to achieve desired yaw
- Using a lookup table for yaw misalignment as a function of wind speed and direction is typically done assuming a more global estimate, which may not be the same as that measured at the individual turbine
- If the steering turbine does not have knowledge of the waked turbine's operation, it may steer a wake away from a turbine that is offline

WindESCo's take on wind farm control: Swarm

Install a device (Swarm Edge) on each turbine, and connect all of these to a central Swarm Server



Fully managed, cloud integrated wind farm control as a service, with automated software updates, cloud monitoring

We just signed our 4th Swarm contract!

WindESCo

How we address common wake steering system flaws

Use absolute nacelle position as control setpoint → Patented solution using GNSS compass

Use data from all turbines to estimate wind conditions

Take into account derating and offline turbines

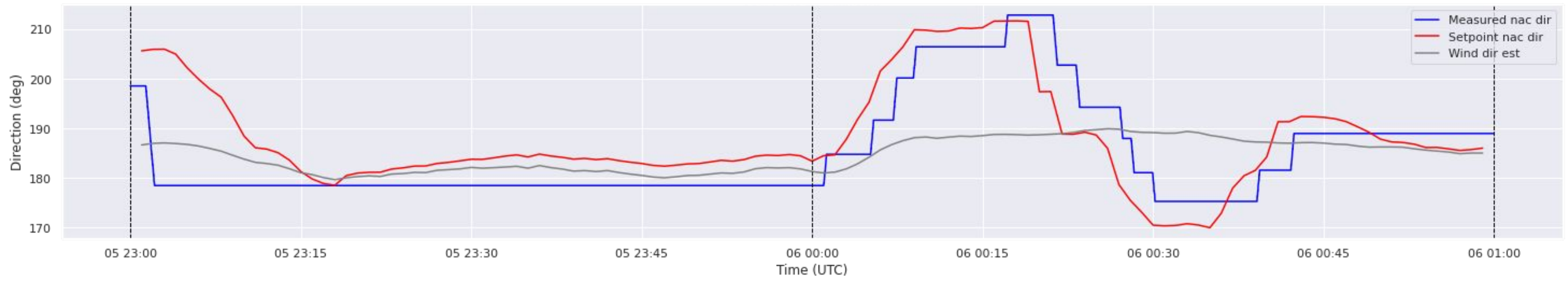
Lookup tables would require too many dimensions → model in the loop

Swarm pilot

Instrumented 10 turbines on a ~50 turbine farm in Fall 2021

Have been running model-in-the-loop wake steering since then—15,000+ turbine-hours of operation

Swarm Edges successfully follow nacelle position setpoint:

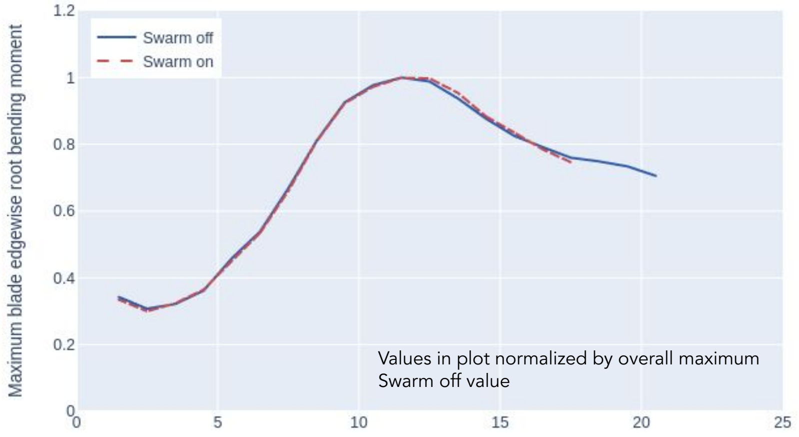


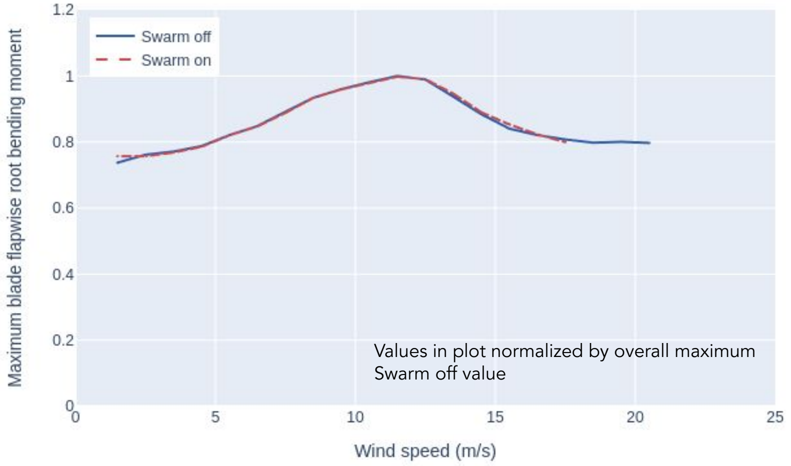
Swarm off: Wake steering could be beneficial, but disabled for comparison

Swarm on, tracking optimum setpoint

Swarm pilot: Loads

One turbine was instrumented with a blade root moment measurement system—no appreciative change in fatigue or peak loads

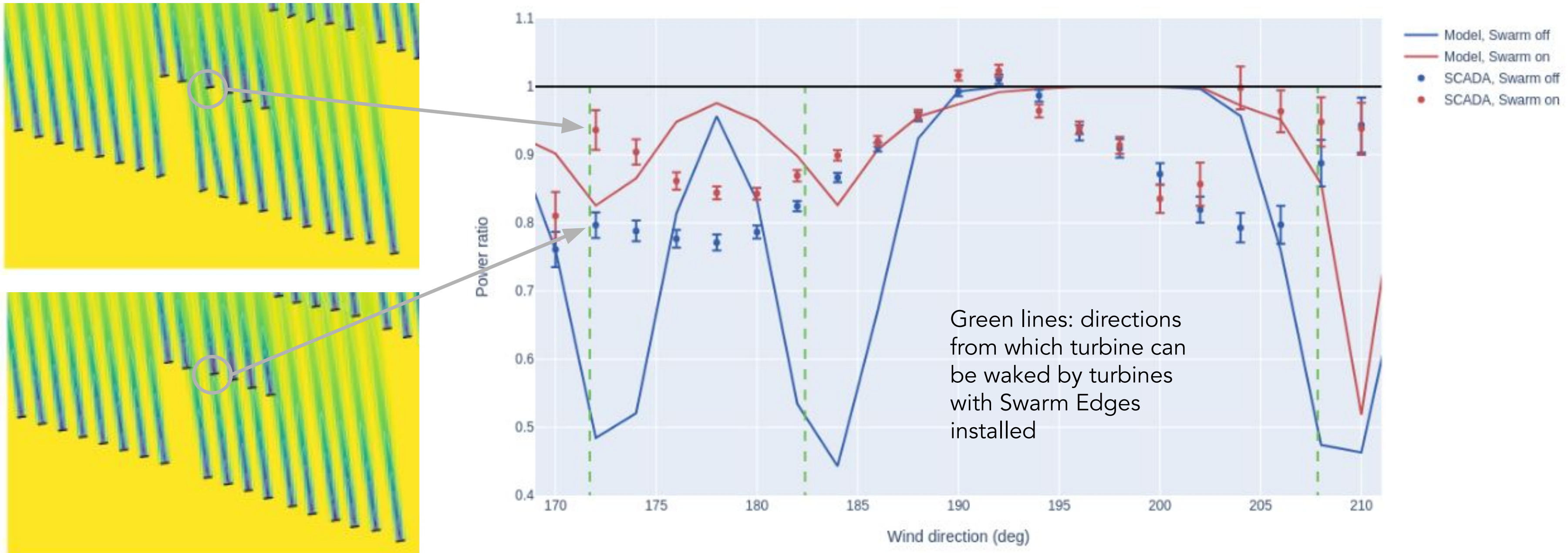




Swarm pilot: Wake steering performance

Successful wake steering, but still have algorithm improvements to make

With additional control applications, we predict we will achieve 3–5% AEP gain across 165 turbines instrumented for the full deployment

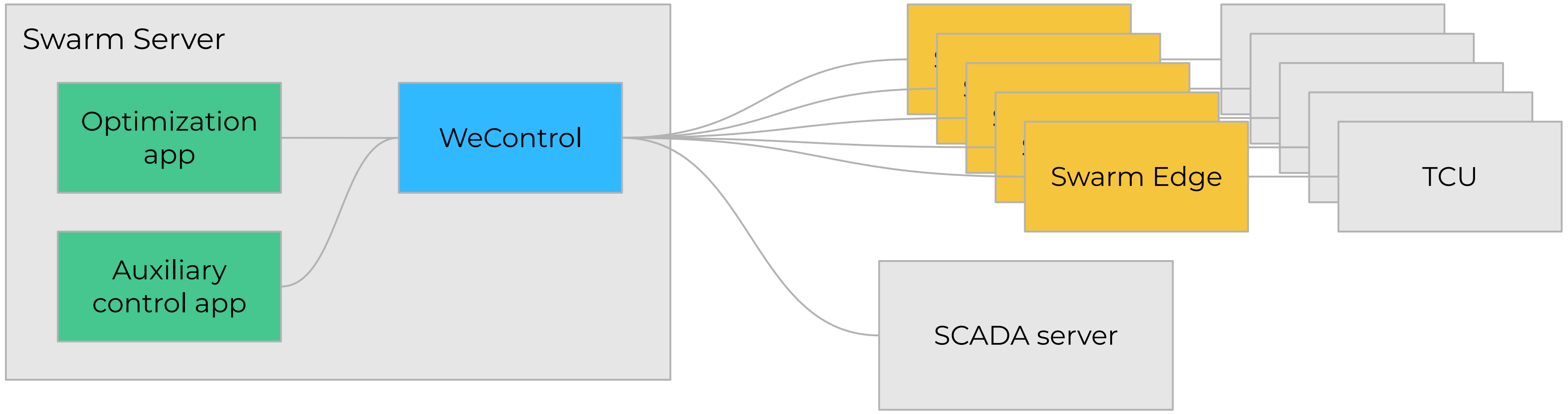


Green lines: directions from which turbine can be waked by turbines with Swarm Edges installed

Swarm system architecture

Our WeControl app abstracts away communication so optimization app only needs to worry about computing and sending optimum setpoints

Swarm Edge handles safety

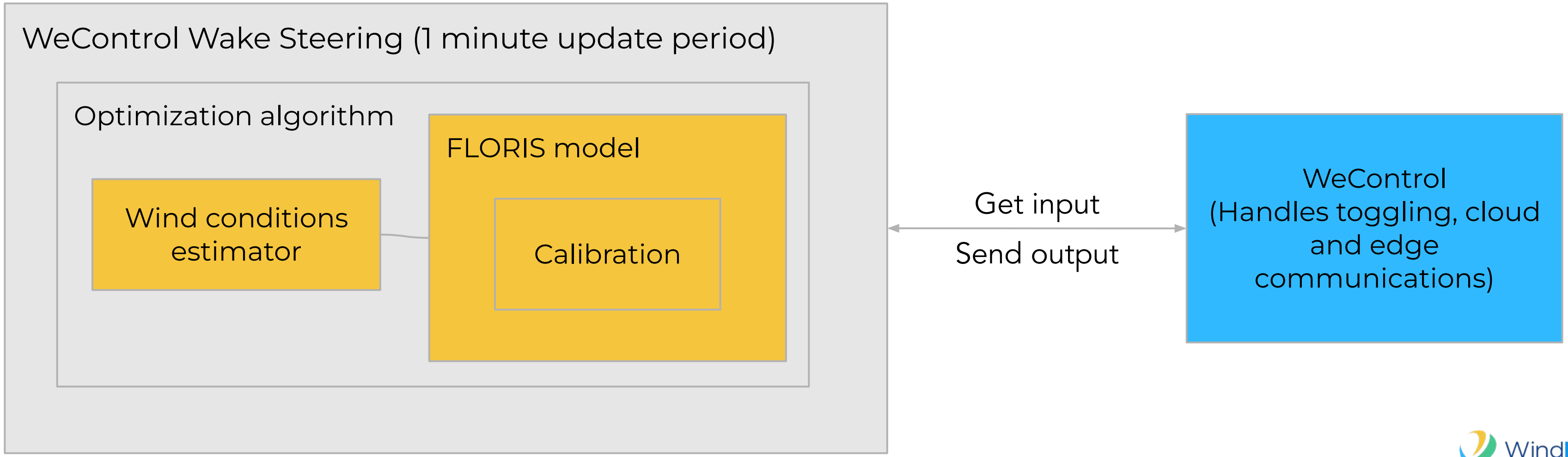


Support for additional control apps, e.g., for controlling lights, other IoT devices connected to Swarm Edge, yaw misalignment correction

Swarm algorithms

App specifies an optimization algorithm object, currently uses FLORIS with our own wrapper for clustering and calibration, but is not locked in to any particular model type (might be tech debt if we were!)

Wind conditions estimators: Farm average, gaussian weighted average (excluding waked turbines), pre-trained forecasting


WeControl Wake Steering (1 minute update period)
Optimization algorithm
Wind conditions estimator
FLORIS model
Calibration
Get input
Send output
WeControl (Handles toggling, cloud and edge communications)

Ideas for collaboration

Swarm is a platform that helps abstract away communications and safety concerns so developers can focus on writing the best optimization algorithms

So far we've done this for internal developers, but would it make sense to offer externally?

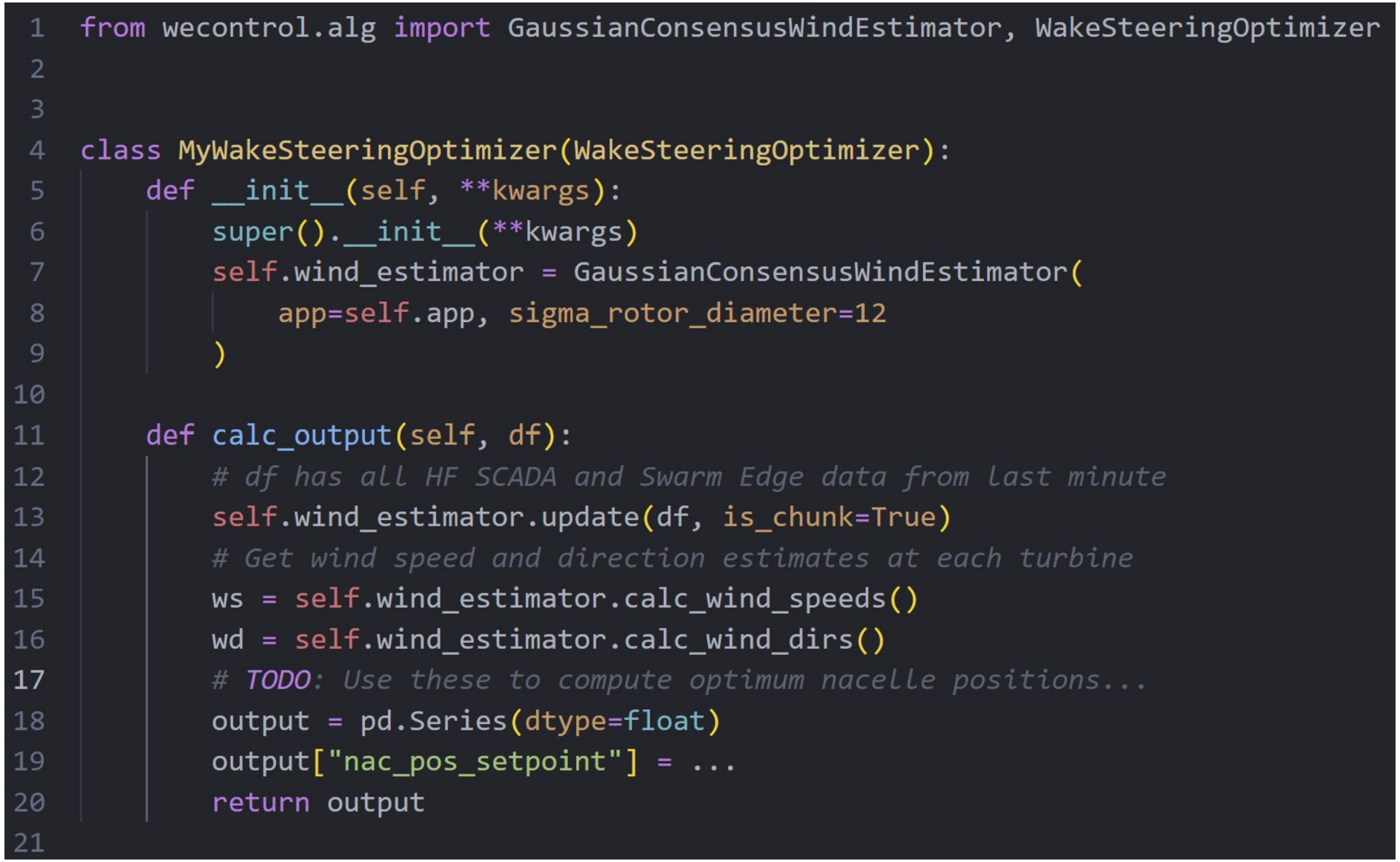
Would benefit developers who want to write algorithms and applications in Python

Turbine metadata, pretrained models can be fetched from the cloud via our API, and control data is sent there to be analyzed alongside SCADA data

Could also give access to our high frequency unsteady simulation framework to evaluate algorithms before deploy

If you want your research tech to be available to use in our controller, publish a Python package!

Writing a new wake steering optimization algorithm



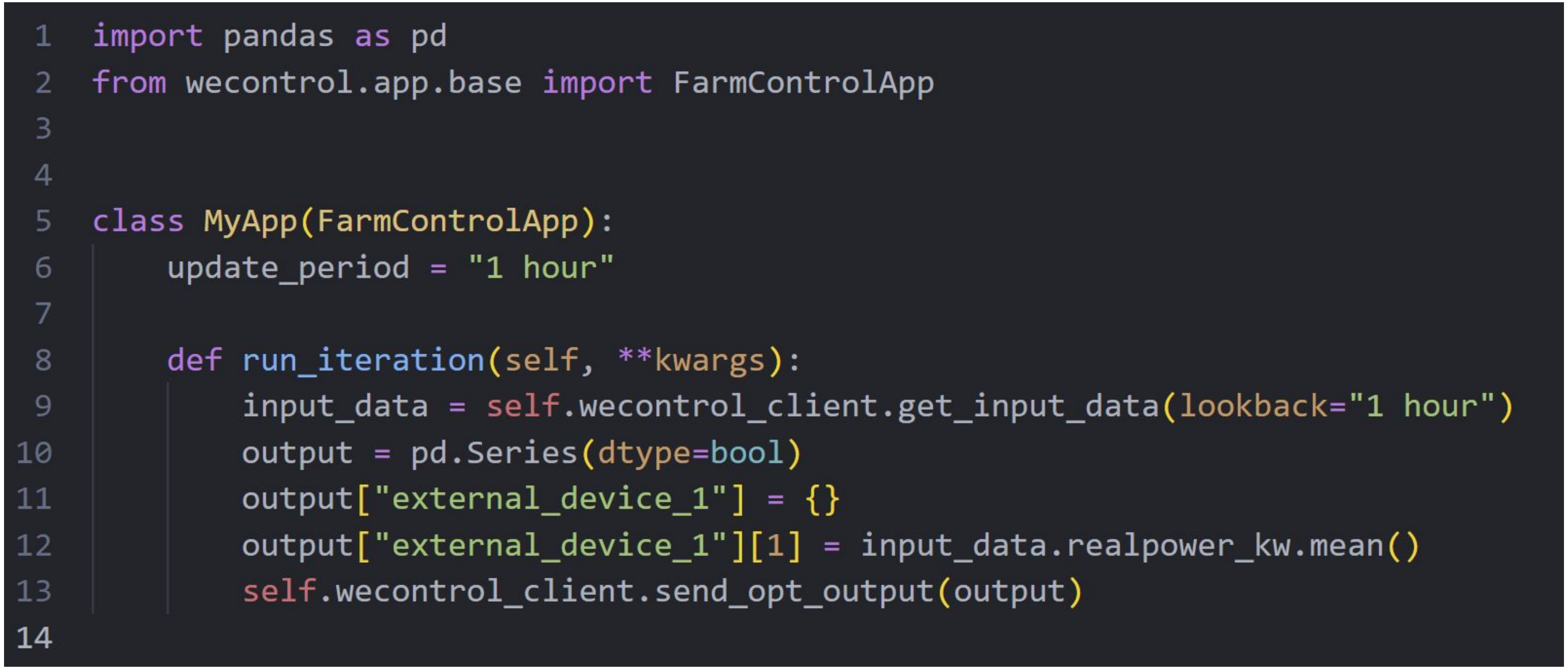
```python
from wecontrol.alg import GaussianConsensusWindEstimator, WakeSteeringOptimizer


class MyWakeSteeringOptimizer(WakeSteeringOptimizer):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.wind_estimator = GaussianConsensusWindEstimator(
            app=self.app, sigma_rotor_diameter=12
        )

    def calc_output(self, df):
        # df has all HF SCADA and Swarm Edge data from last minute
        self.wind_estimator.update(df, is_chunk=True)
        # Get wind speed and direction estimates at each turbine
        ws = self.wind_estimator.calc_wind_speeds()
        wd = self.wind_estimator.calc_wind_dirs()
        # TODO: Use these to compute optimum nacelle positions...
        output = pd.Series(dtype=float)
        output["nac_pos_setpoint"] = ...
        return output

```

Writing a new control app



```python
import pandas as pd
from wecontrol.app.base import FarmControlApp


class MyApp(FarmControlApp):
    update_period = "1 hour"

    def run_iteration(self, **kwargs):
        input_data = self.wecontrol_client.get_input_data(lookback="1 hour")
        output = pd.Series(dtype=bool)
        output["external_device_1"] = {}
        output["external_device_1"][1] = input_data.realpower_kw.mean()
        self.wecontrol_client.send_opt_output(output)

```

Questions?

Contact: pete@windesco.com