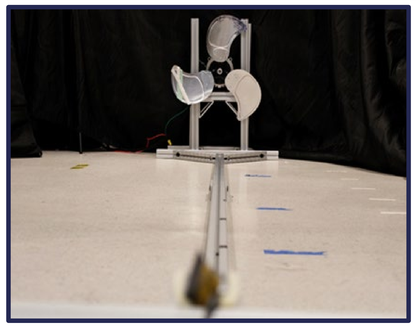
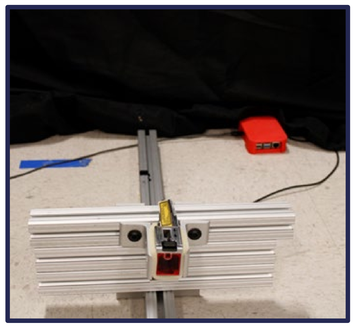
FRC Blade Height

School of Engineering + Technology

Problem Statement

- The Whirlstand has outdated electronics.
- Some tests freeze the computer and cannot be used.
- The camera cannot sense the blades during intense sunlight.
- The capstone team will design a new and easily maintainable system for measuring the height of the blades. (All documentation will be given to the sponsor upon completion)





Requirements

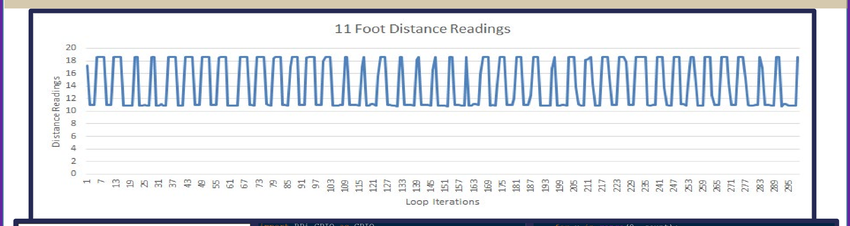
- Lower cost, keeping within a budget of $6500.
- Device and housing must take up a volume of 125 cubic inches or less.
- Measurements must be taken between 6-12 feet with a tolerance of 1/8th inch.
- Must be able to distinguish between a control blade and two test blades.
- Data gathered must be stored and displayed for ease of use.
- The device must not cause damage to any user's eyes.

Concepts

- Initial concepts: Photo Analysis In plane with Blades, Infrared Rangefinder, Focused Light, and Laser Sensor.
- Narrowed down to the Laser Sensor using Analysis of Alternatives and council from the sponsor.

	Alternatives			
Quantitative Requirements (Weight)	Focused Light	IR Rangefinder	Laser	Photo Analysis
Volume < 125 cu in (1.0)	0	1	1	1
Range 6 ft ≤ x ≤ 12 ft (2.0)	1	2	2	2
Tolerance < 1/8 in (2.0)	1	1.5	1.5	1.6
Total	2	4.5	4.5	4.6
Comments		Can be inaccurate at long ranges		Inaccuracies dependent on angle
Qualitative Requirements (Weight)				
Distinguishes Blades (Yes, No)	No	No	No	Yes
Compiles Data (Yes, No)	Yes	Yes	Yes	Yes
Comments				
Risks (Likelihood/impact)				
Eye-Safety (L-M-H)	M/L	L/L	L/L	L/L
Instable Mounting (L-M-H)	M/H	M/H	M/H	M/H
Comments				
Cost (low to high)	Low	Low	Low	Low
Assumptions				
Feasibility		May not refresh fast enough		
Issues	May not focus very well			Sponsor would like to stray from this idea

Results



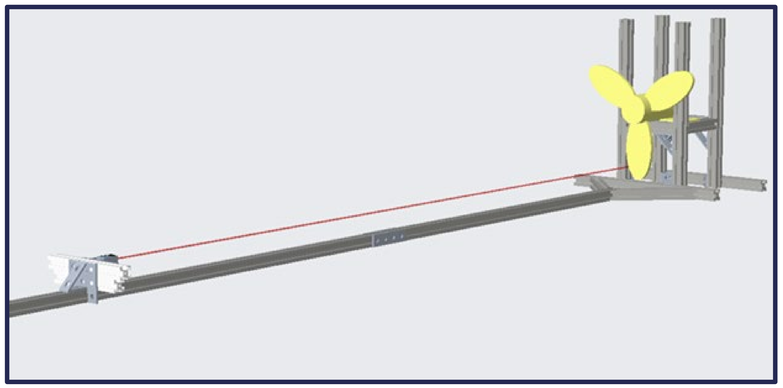
Top: Chart showing test results.
Left: Example of GUI.
Right: Snippets of sampling code.

```
import RPi.GPIO as GPIO
import board
import busio
import math
import adafruit_ads1x15.ads1015 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from openpyxl import Workbook
import datetime

wb = Workbook()
ws = wb.active
ws['A1'] = "Start Time"
ws['B1'] = datetime.datetime.now()
ws['A2'] = "End Time"

i2c = busio.I2C(board.SCL, board.SDA)
ads = ADS.ADS1015(i2c)

GPIO.setmode(GPIO.BCM)
digitIn = GPIO.setup(26,GPIO.IN)

count = 0
counting = 0
denominator = 0
add = 0
position = 1

displayB1 = 0
displayB2 = 0
displayB3 = 0

record = []
blade1 = []
blade2 = []
blade3 = []

while True:
    print("start recording")
    for x in range(0, 1400):
        distance = AnalogIn(ads, ADS.P0)
        record.append(distance.voltage)

    count = len(record)
    for x in range(0, count):
        if record[x] >= 1.27:
            record[x] = 0

    print("finished recording, start data analysis")


    for x in range(0, count):
        if record[x] == 0:
            if add == 0:
```

```
                continue
            average = add/denominator
            add = 0
            denominator = 0
            counting +=1
            remainder = math.fmod(counting,3)
            if remainder == 1:
                blade1.append(average)
            elif remainder == 2:
                blade2.append(average)
            else:
                blade3.append(average)
        elif record[x] > 0:
            denominator += 1
            add += record[x]
add = 0
count = len(blade1)
for x in range(0, count):
    add += blade1[x]
average = add/count
ws.cell(row = position, column = 1, value = average)
displayB1 = (displayB1+average)/2
print("Blade 1 =", displayB1)

add = 0
count = len(blade2)
for x in range(0, count):
    add += blade2[x]
average = add/count
ws.cell(row = position, column = 3, value = average)
displayB2 = (displayB2+average)/2
print("Blade 2 =", displayB1)

add = 0
count = len(blade3)
for x in range(0, count):
    add += blade3[x]
average = add/count
ws.cell(row = position, column = 5, value = average)
displayB3 = (displayB3+average)/2
print("Blade 3 =", displayB1)

position+=1
wb.save("runData04.xlsx")

record.clear()
blade1.clear()
blade2.clear()
blade3.clear()
```

- Initial testing showed issues with timing of the blades.
- The code has been mostly debugged but cannot be verified.
- Multiprocessing has been implemented in the GUI to prevent freezing.

Final Design



- Constructed a motor stand and rail using 8020 Aluminum as a testing rig.
- Made use of a PLC cart for the motor and controller.
- Machined a store-bought fan to fit the motor.
- Used a Keyence LR-TB5000 laser sensor to make distance measurements while the fan is running.
- Collected data using a Raspberry Pi, where it would then be processed by a Python code and displayed on a GUI.

Conclusion

- The team was able to make a GUI to control the laser sensor and the execution of code.
- A remote desktop environment was created for easy access of the Raspberry Pi.
- The team was able to get distance measurements and save them to a spreadsheet, but unfortunately was unable to verify if the blade assignments were correct.
- The distance measurements are out of tolerance.

Team & Acknowledgements

- Wyatt Allen - ECET
- Owen Edwards - EE
- Nicholas Queen - ECET
- Devin Smith - ME
- Paul Yanik - Mentor
- Fleet Readiness Center East - Sponsor Company —Chance Houston and John Hinson - Sponsors