SNOW DEPTH INFORMATION AND MITIGATION BEFORE AVALANCHES

SIMBA

Manufacturing Status Review

Presenters
Max Fidler, Travis Griffin, Brett Papenfuss, Aidan Sesnic, Saad Syed, Kevin Yevak, Sean Yoo

Customer
Copper Mountain Resort: Hagen Lyle

Advisor
Professor Zachary Sunberg

Additional Team Members
Lucas Dickinson, Adam Gourmos, Stephen Peng, Devon Ricken, Jordan Walters

SPONSORS:

COPPER MOUNTAIN

Smead Aerospace
UNIVERSITY OF COLORADO BOULDER

Overview

Project Purpose

The Problem: Avalanche Mitigation

- Currently
  - Deep pits in avalanche prone areas
  - Dangerous, laborious, and time consuming
- Our system provides data that minimizes
  - Number of snow pits required
  - Time and effort in avalanche prone areas
- This makes avalanche mitigation safer and more efficient

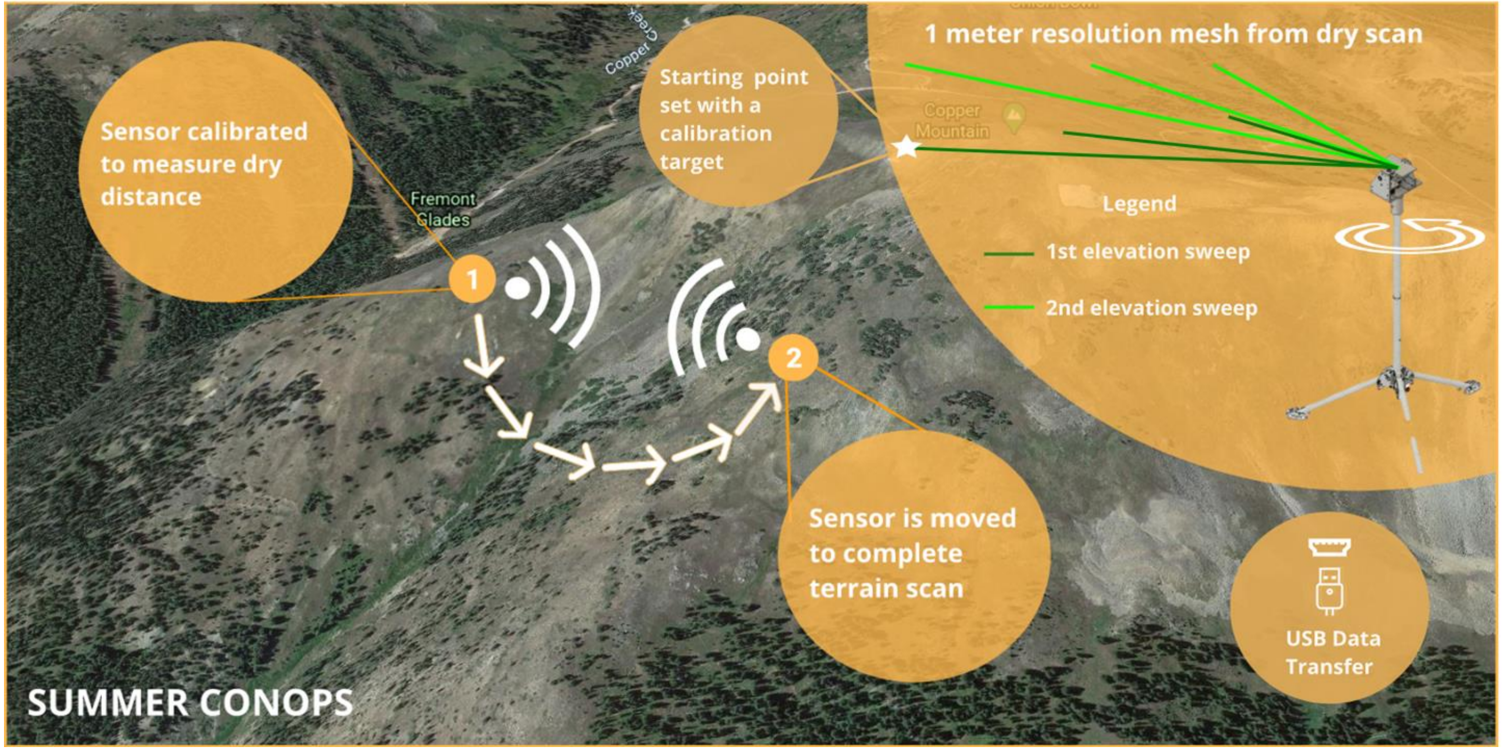
Mission Statement

- Team SIMBA shall provide a mobile snow depth measuring system to help Copper Mountain Ski Patrol mitigate the risk of avalanches.

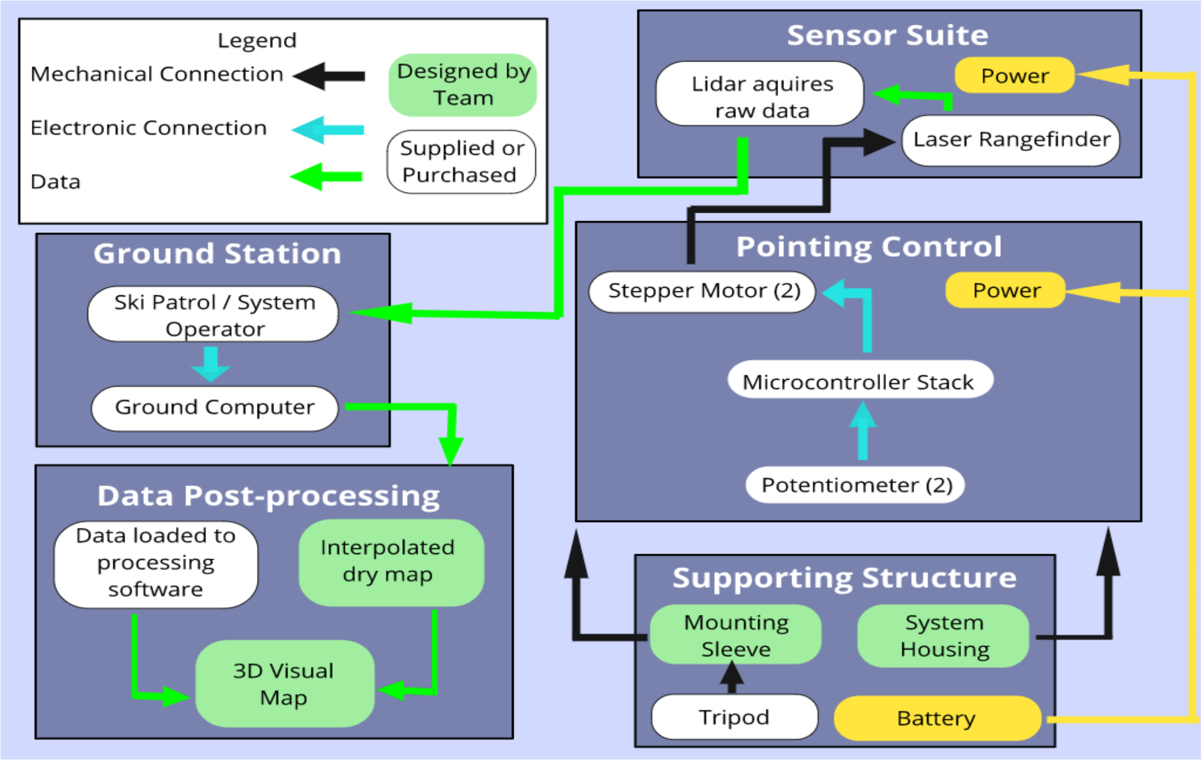
Main Objectives

- Measure snow depth to within ±10 cm at 400 m with 1 m spatial resolution
- Produce heat map displaying snow depth
- Operate at minimum temperature of -20°C

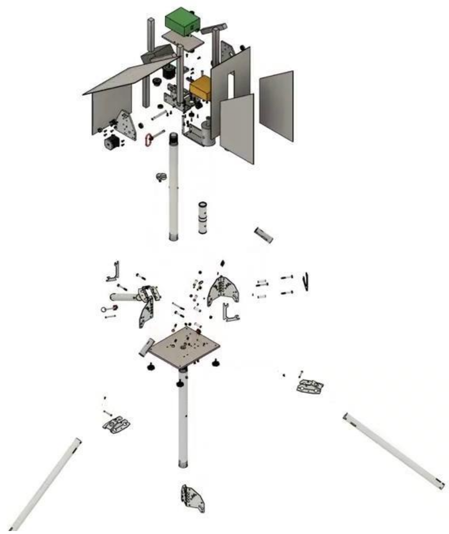
CONOPS


1 meter resolution mesh from dry scan
Starting point set with a calibration target
Copper Mountain
Copper Creek
Sensor calibrated to measure dry distance
Fremont Glades
Legend
1st elevation sweep
2nd elevation sweep
1
2
Sensor is moved to complete terrain scan
USB Data Transfer
SUMMER CONOPS

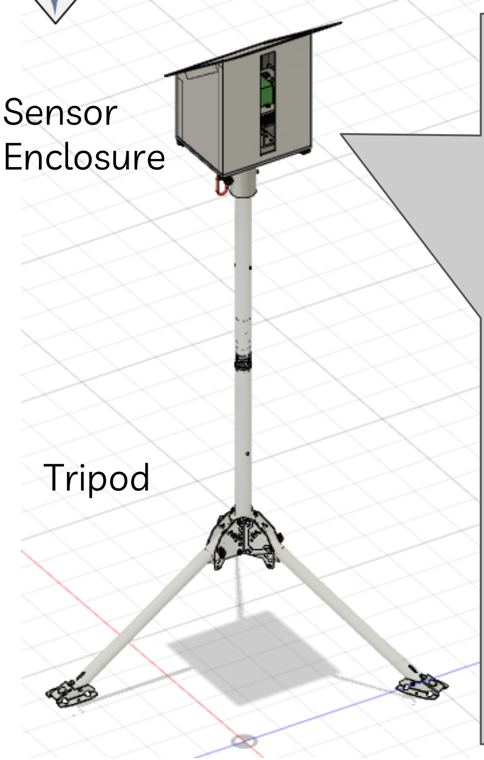
FBD

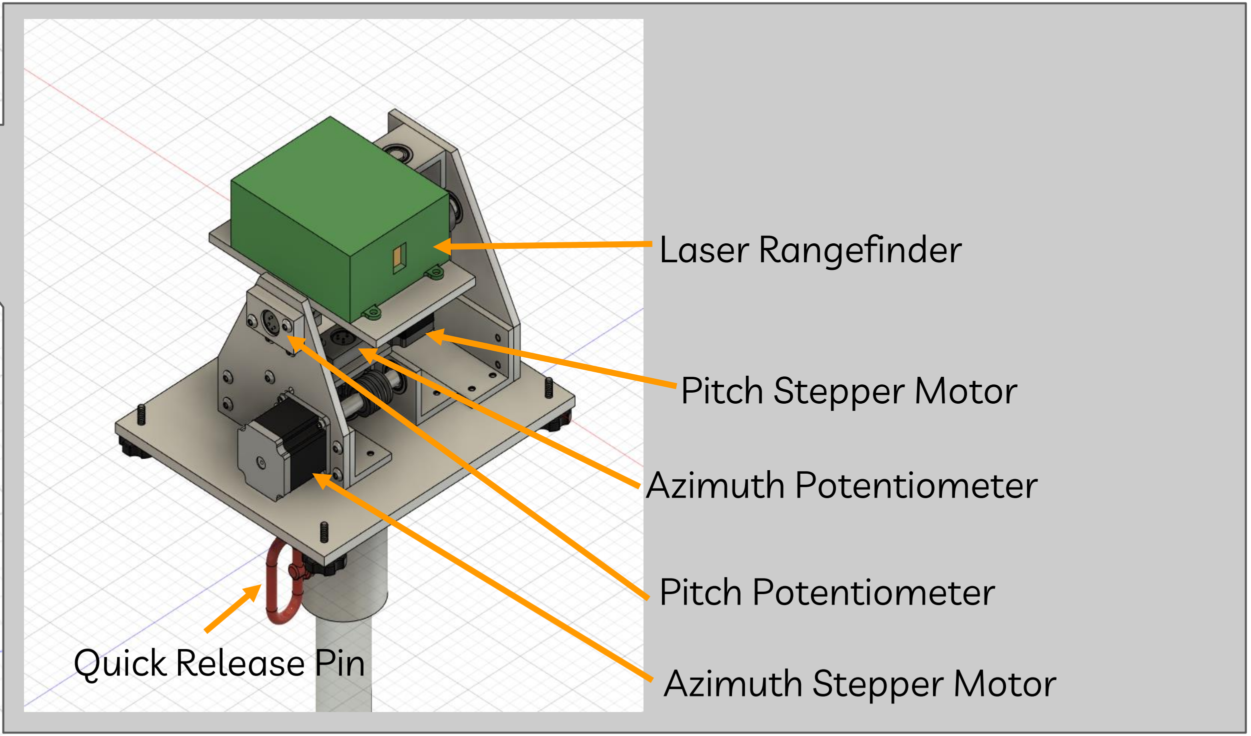

Legend
Mechanical Connection
Electronic Connection
Data
Designed by Team
Supplied or Purchased
Sensor Suite
Lidar aquires raw data
Power
Laser Rangefinder
Ground Station
Ski Patrol / System Operator
Ground Computer
Pointing Control
Stepper Motor (2)
Power
Microcontroller Stack
Potentiometer (2)
Data Post-processing
Data loaded to processing software
Interpolated dry map
3D Visual Map
Supporting Structure
Mounting Sleeve
System Housing
Tripod
Battery

Baseline Design



Baseline Design





Project Levels of Success

	Sensor Package	Software	Pointing Accuracy and Control	Output
1	Snow depth accurately measured within ±50 cm at 1 location at 400 m	Data of one distance measurement by sensor is saved	Laser pointing is able to be determined to 0.01 degrees. No feedback present. Motors ±1° of desired position	Compile data to form a plane to serve as origin for height measurements
2		Distance and attitude of each measurement is recorded for attitude control	Feedback is present allowing the motors to readjust as needed	Display snow depth calculated for one location
3	Snow depth accurately measured **within ±15 cm at 400m**	**Distance, attitude, time & temperature of each measurement** is recorded	Motor initial move **±0.1° of desired position. Feedback allows for ±0.01°**	Produce **map displaying snow depth**
4	Snow depth accurately measured within ±10 cm at 400 m with 1 m spatial resolution		Motor initial move ±0.01° of desired position. Feedback allows for ±0.001°	Produce topographical snow depth map to within ±10 cm

Major Changes Since FFR

1. Raspberry **Pi is now mounted on the roof** instead of with the laser range finder
2. Sensor **enclosure is enlarged** to accommodate Raspberry Pi in order **to maintain full degrees of motion** in pitch
3. Sensor package will use **one battery for both wet and dry scans** instead of two separate batteries
4. Ceramic **resistors selected for conductive heating** around potentiometers
5. **Physical buttons** provides user friendly interface for functional system

Critical Project Elements

Battery
Laser Rangefinder
Pointing Accuracy
Sensor Environment

Sensor Package

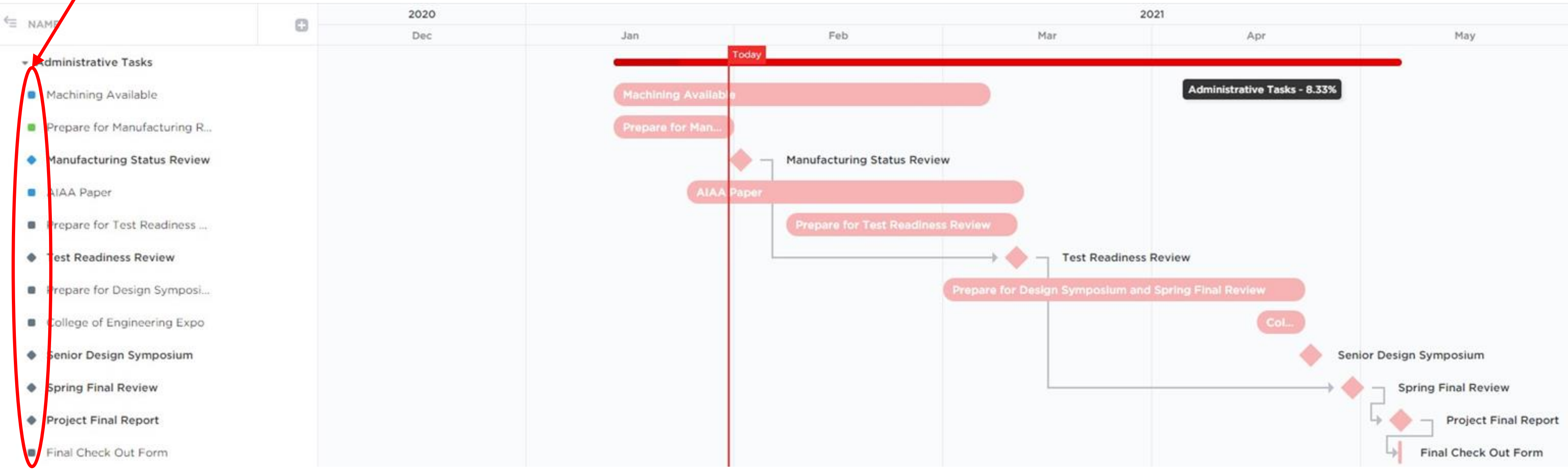
Control System
Data Visualization
Georeferencing
Microcontroller

Software Package

Schedule

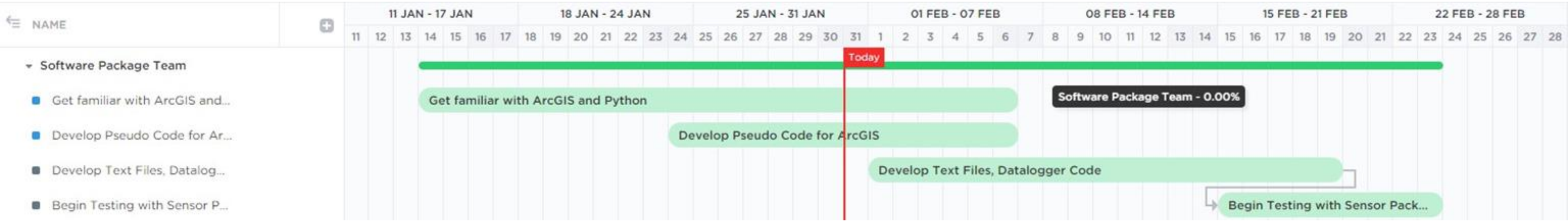
Administrative Work Plan

Status Key: = To-do = In progress = Completed



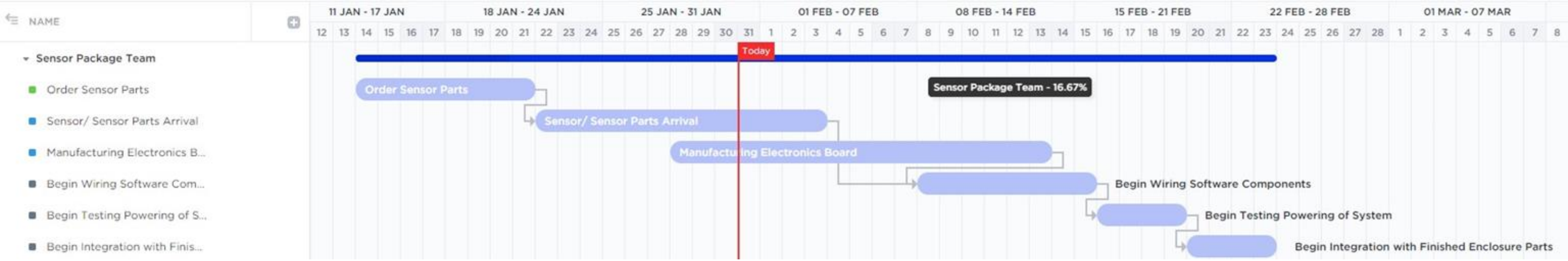
Enclosure and Software Team Work Plan

Status Key: = To-do = In progress = Completed



Sensor Team Work Plan

Status Key: = To-do = In progress = Completed

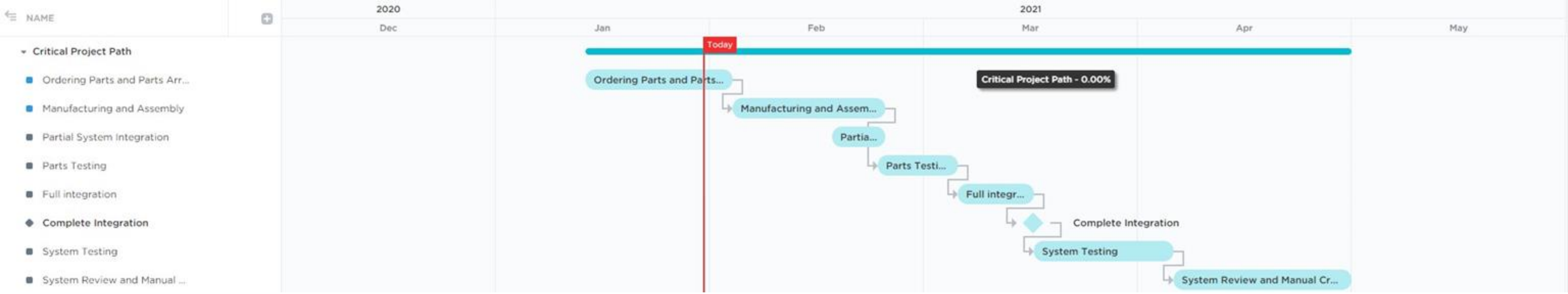


Testing Plan

Status Key: = To-do = In progress = Completed

Critical Path

Status Key: = To-do = In progress = Completed



Manufacturing

Pointing Accuracy

Summary: The pointing accuracy consists of *COTS laser range finder*, *2x potentiometers*, and *2x stepper motors*. It is dependent on interaction between COTS sensors, manufactured enclosure components, and code based controls

Current Status:

- COTS hardware components have been purchased

Remaining Work:

- COTS sensor assembly between sensor enclosure and microcontroller "stack"
- Sensors need to be individually tested to validate manufacturer spec sheets before TRR

Main Concerns:

- Meeting Pointing Accuracy Requirements

Mitigation:

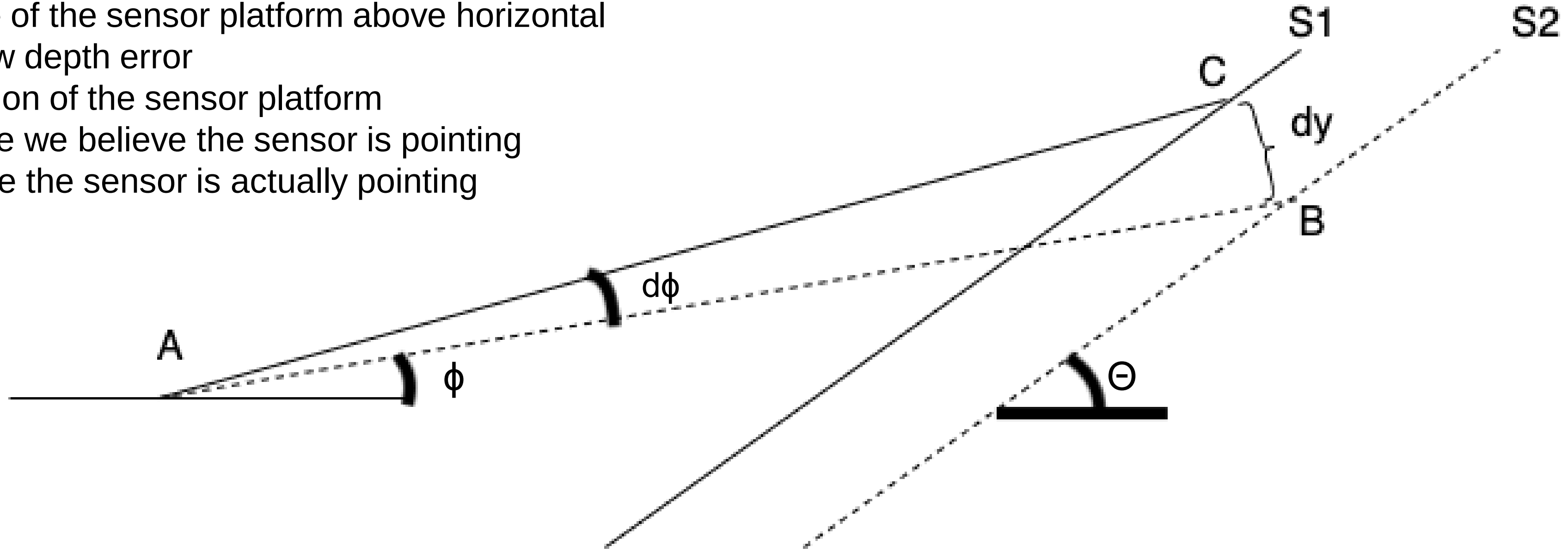
- Detailed Error budget to predict accuracy based on component performance

Pointing Error

$$dy = (AC + dAC) * (sin(\phi + d\phi) - [sin(\phi) * sin(\theta - \phi - d\phi)/sin(180 - \theta + \phi)])$$

Θ - angle of the slope
dϕ - potentiometer error + tripod tip deflection
ϕ - angle of the sensor platform above horizontal
dy - snow depth error
A - location of the sensor platform
B - where we believe the sensor is pointing
C - where the sensor is actually pointing

S1 - Surface of the slope after snowfall
S2 - Surface of the slope without snow



Pointing Accuracy

Error Value Calculating using the previous equation with $\Theta = 30°$, $\phi = 0°$, $d\phi = 7.58°$, AC = 400m, and dAC = 4cm

Snow Depth Error, dy = 52.8 m, which will not allow for useful data collection

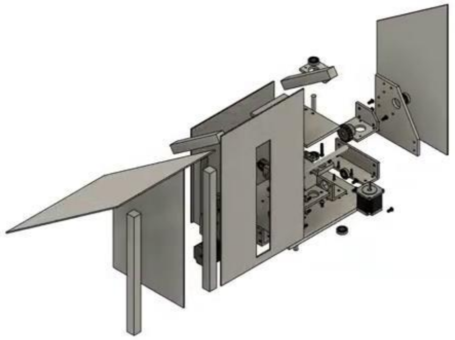
Change in Snow Depth Error vs Range Error

dAC	Snow Depth Error
0.10 m	0.01808 cm
0.20 m	0.018090 cm
0.30 m	0.018095 cm

Change in Snow Depth Error vs Potentiometer Error

$d\phi$	Snow Depth Error
.001°	0.698 cm
.01°	6.98 cm
.1°	69.8 cm

Sensor Environment



Sensor Environment

Summary: The housing of the sensor package shields sensor components from a winter environment

Current Status:

- Aluminum and HDPE sheets: Delivered
- Structure height increased 3” for layout change
- Ceramic Resistors for conductive heating

Remaining Work:

- Width of opening for laser
- Machining & Assembly
- Power Regulation

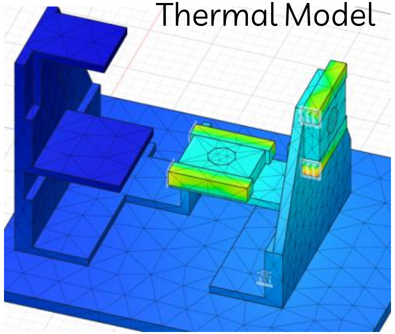
Main Concerns:

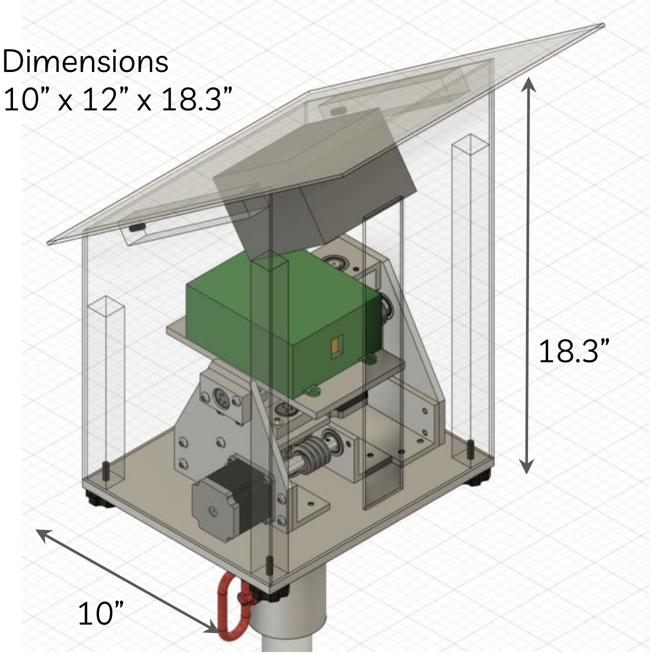
- Snow accumulation

Mitigation:

- Component IP rating
- Drainage for melting snow

Thermal Model





Microcontroller

Summary: The microcontroller "stack" is made up of COTS products.

Current Status:

- All parts have been received
- Modelling and layout complete

Remaining Work:

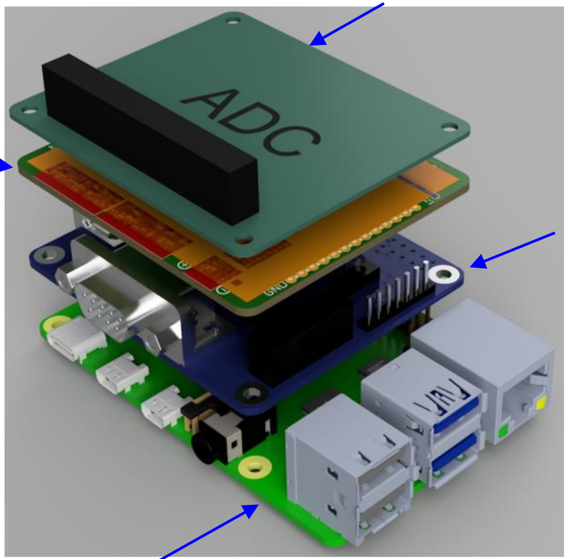
- Testing and final assembly

Main Concerns:

- Connecting 3 "hats" to Pi

Mitigation:

- Early testing in case problems arise



Control System

Summary: Software to control the sensor's pointing on the microcontroller.

Current Status:

- Currently in pseudocode stage
- Github for version control

Remaining Work:

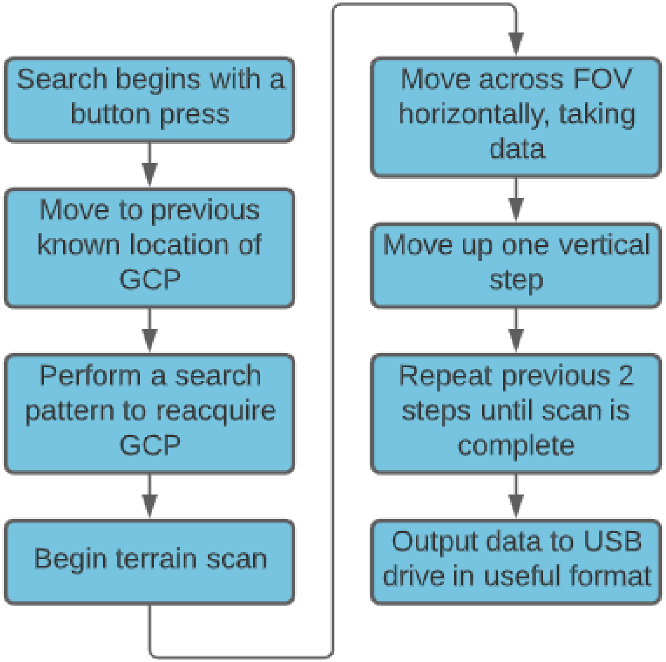
- Final Programming
- Testing

Main Concerns:

- Deciding what scan pattern to use

Mitigation:

- Not difficult to change scan pattern in software to run multiple tests



Detailed pseudocode in backup slides

Data Visualization

Summary: Data visualization is how the team will output the collected data onto an intuitive heat map.

Current Status:

- Pseudocode developed for basic flow and function required by our code
- ArcGIS downloaded on personal computers

Remaining Work:

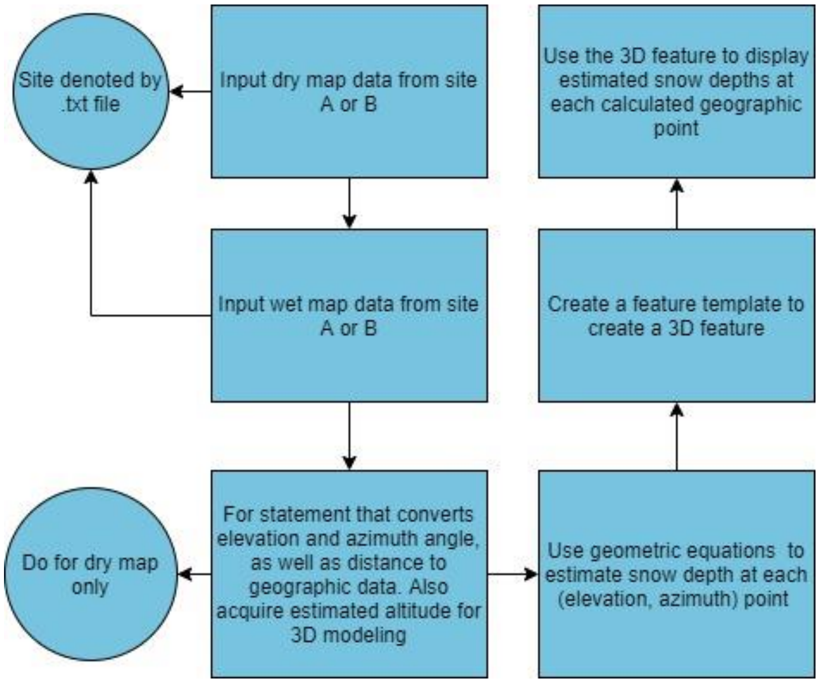
- Start code development in ArcGIS

Main Concerns:

- Making data processing program intuitive to use by inexperienced personnel

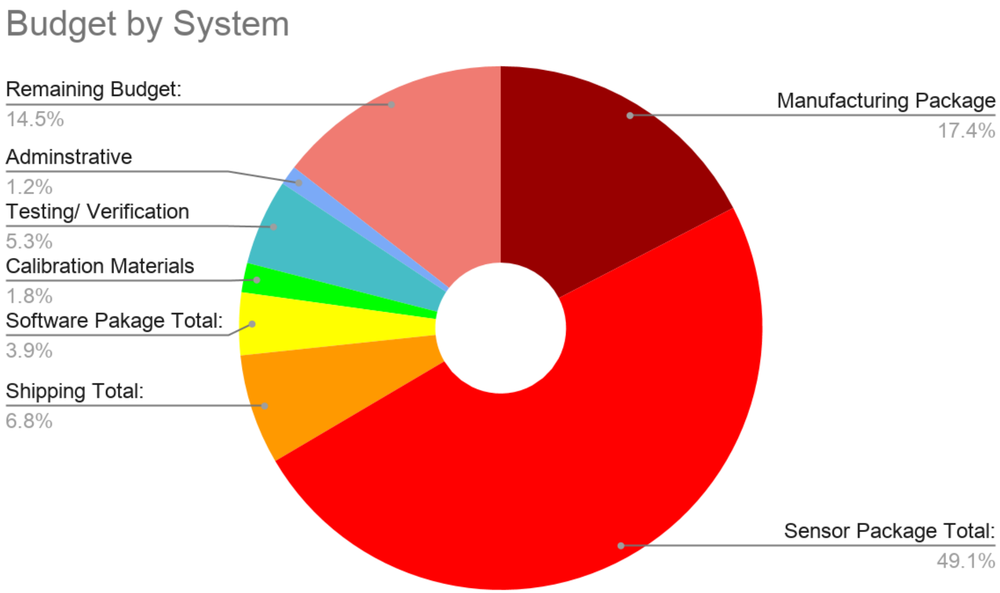
Mitigation:

- Create easy to follow user manual



Budget

Financial Budget Overview



Manufacturing Package Total	$870.51
Sensor Package Total	$2,455.61
Shipping Total	$340.23
Software Package Total	$193.01
Calibration Materials	$90.83
Testing/ Verification Equipment	$265.46
Administrative	$60
Total w/ Margin	$4,275.66
Remaining Budget	$724.34

* 5% Margin applied to total to account for any costs not considered (i.e. extras of sensitive components, incorrect parts, missing components)

Financial Budget Status

Order/ Subscription Name	Order Description	Date Ordered/ Processed	Date Received	Total Cost
Click-Up October - December Subscription		12/27/2020	12/27/2020 (Done)	$45.87
McMaster Order	All parts for manufacturing the enclosure for the sensor package.	12/22/2020	1/4/2021 (Done)	$799.79
Forestry Supplier	TruPulse x200	1/14/2021	1/21/2021 (Done)	$1,822.20
Adafruit	Raspberry Pi 4 Model B	1/12/2021	1/19/2021 (Done)	$45.50
Adafruit	Stepper motor hat	1/13/2021	1/14/2021 (Done)	$28.32
PiShop US	Pi power cable, HDMI, and SD card	1/19/2021	02/01/2021 (Done)	$31.55
Amazon	RS-232 cable	1/20/2021	01/26/2021 (Done)	$6.15
Amazon	Resistors, serial hat, 100-ft tape measure	1/21/2021	01/26/2021 (Done)	$50.13
Amazon	AWG22 Wires	1/22/2021	01/27/2021 (Done)	$15.99
Digikey	Stepper motors, potentiometers, ADCs	1/27/2021	02/01/2021 (Est.)	$184.50
			Total:	$3,030.00
			Remaining Budget:	$1,970.00

Components Procurement Status

Key

* = Critical Project Component | = Arrived at AERO | = Shipping | = Order created | = Order not yet made

Sensor Environment*

- Angle irons
- Ball bearing
- Worm
- Gear
- Shafts
- Plates
- Bars
- Quick releases pin
- HDPE Sheets
- Washers
- Screws

Computer Hardware

- Microcontroller*
- Serial Hat
- Wire
- Resistor Set
- ADC
- Stepper Motor Hat
- Pi power cable
- Pi HDMI adapter
- Pi microSD Card
- Plastic enclosure for microcontroller

Sensor/Pointing Accuracy

- Battery*
- Lidar*
- RS-232 cable
- Potentiometer*
- Stepper motor

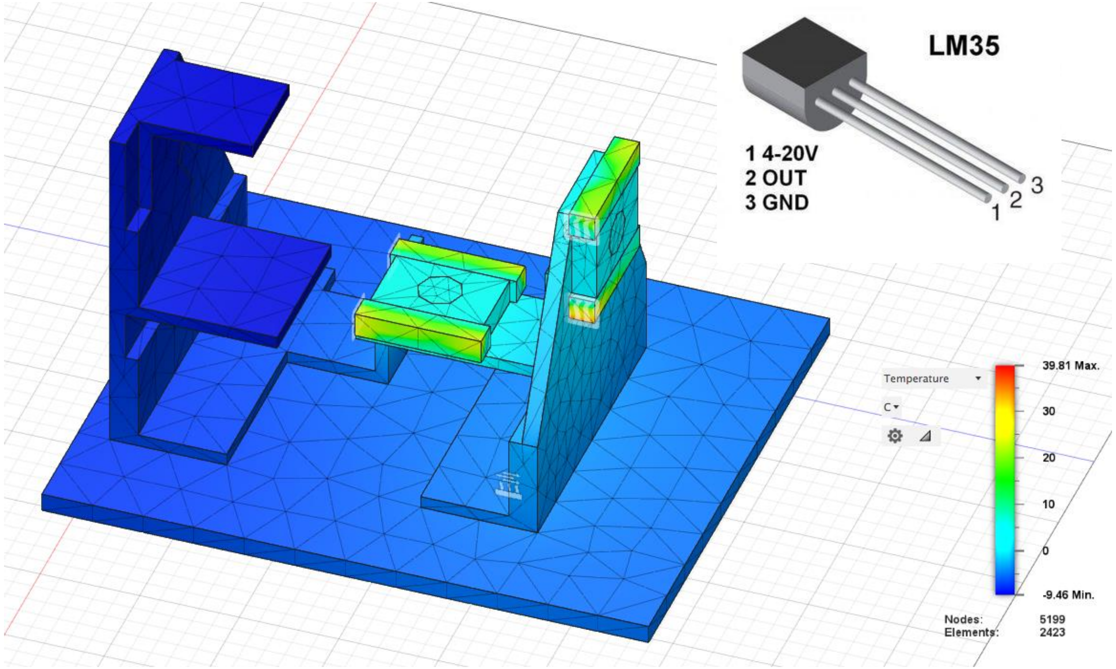
Calibration/ Testing

- BCA Stealth 300 Avalanche Probe
- 100 ft Tape Measure
- Ultra Bright Green Laser Pointer
- Reflectors
- Calibration sign

Backup Slides

Detailed Thermal Model

- -20C ambient
- 60 W/m^2 convection coef.
- 60 W input per potentiometer
  - 2 power resistors
  - 30 W each
- Thermal controller
  - Closed loop
  - LM35 temp sensor
  - On/off for two seperate channels (one channel for each potentiometer)
  - Threshold temp TBD
- Model results
  - Potentiometers <5C
  - Continuous aluminum piece

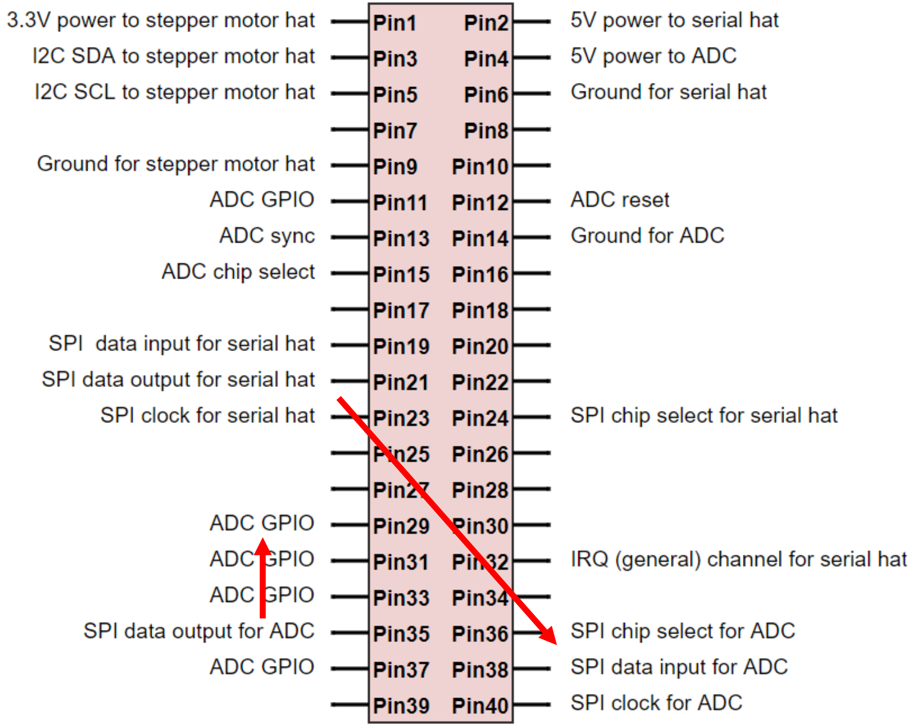


Pinout for GPIO pins on Pi

3.3V power to stepper motor hat	Pin1	Pin2	5V power to serial hat
I2C SDA to stepper motor hat	Pin3	Pin4	5V power to ADC
I2C SCL to stepper motor hat	Pin5	Pin6	Ground for serial hat
	Pin7	Pin8	
Ground for stepper motor hat	Pin9	Pin10	
ADC GPIO	Pin11	Pin12	ADC reset
ADC sync	Pin13	Pin14	Ground for ADC
ADC chip select	Pin15	Pin16	
	Pin17	Pin18	
SPI data input for serial hat	Pin19	Pin20	
SPI data output for serial hat	Pin21	Pin22	
SPI clock for serial hat	Pin23	Pin24	SPI chip select for serial hat
	Pin25	Pin26	
	Pin27	Pin28	
ADC GPIO	Pin29	Pin30	
ADC GPIO	Pin31	Pin32	IRQ (general) channel for serial hat
ADC GPIO	Pin33	Pin34	
SPI data output for ADC	Pin35	Pin36	SPI chip select for ADC
ADC GPIO	Pin37	Pin38	SPI data input for ADC
	Pin39	Pin40	SPI clock for ADC

Changed Pin Locations

- **ADC SPI pins 35, 36, 38, 40 moved to alternate SPI pins**
- **General purpose pin moved from pin 35 to pin 29 to allow for use of alternate SPI channels**



Microcontroller Pseudocode

```
##########################################################################################
# Description: pseudocode for microcontroller for ASEN 4028 Senior Projects
# Notes: code is a pseudo-mixture of Python and Linux shell scripts
# Author: Brett Papenfuss
# Team 2: SIMBA
# Last updated: 1/21/2021
##########################################################################################


# Imports: various libraries are provided for the Raspberry Ri hats, as well as some general libraries

# Stuff for serial hat from Waveshare.com -----------------------------------------------
#Open the terminal and modify config.txt file by commands:
sudo nano /boot/config.txt
#Add the line below to the file, the int_pin should be set according to the actual welding:
dtoverlay=sc16is752-spi1,int_pin=24
#Then restart Raspberry Pi
sudo reboot
#After rebooting, the driver of SC16IS752 will be loaded into the system kernel. You can run command...
ls /dev
# to check the devices

# Install wiringpi
sudo apt-get install wiringpi
cd /tmp
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# Running gpio-v to check if the version is 2.52, If it is not, you need to check the installation again.
# Install the python3 library
sudo apt-get update
sudo apt-get install python3-pip
sudo pip3 install RPi.GPIO
sudo apt-get install python3-serial

# waveshare provides a test program as well, see https://www.waveshare.com/wiki/2-CH_RS232_HAT

# Stuff for stepper motor hat via Adafruit -----------------------------------------------
# Begin by enabling I2C on Pi (look up a tutorial)
# Install adafruit stepper motor library
sudo pip3 install adafruit-circuitpython-motorkit

from adafruit_motor import stepper as STEPPER
from adafruit_motorkit import MotorKit

# other recommended libraries
import time
import atexit
import threading
import random
```

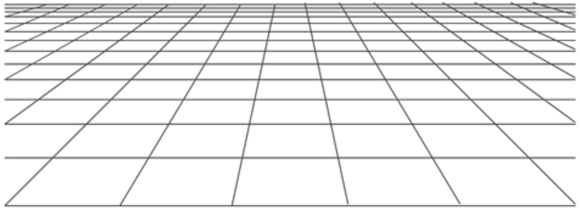
```
# Stuff to receive data from serial hat ---------------------------------------------------

import serial
import os
import sys
import logging
import time

logging.basicConfig(level=logging.INFO)
libdir = os.path.join(os.path.dirname(os.path.dirname(os.path.realpath(__file__))), 'lib')
if os.path.exists(libdir):
    sys.path.append(libdir)

from waveshare_2_CH_RS232_HAT import config

ser1 = config.config(Baudrate=xxxxxxx, dev = "/dev/ttySC0")  # retrieve location and baudrate of serial input

try:
    while(1): # infinite loop, we want to listen all the time for serial input
        currentDistance = ser2.Uart_ReceiveString(stringLength) # update curretn distance
        print currentDistance
        # not sure the format that we will receive distances in, so that may take some work

except # some exception to stop listening like a keyboard interrupt or stop scan


# Begin search pattern to find GCP ---------------------------------------------------------
motorsObject = MotorKit(i2c=board.I2C()) # create object for motor input
# for now, motor 1 is the horizontal plane and motor 2 is vertical plane

def stepper_function(motornum, direction, step_type): # function to step motor once
    print('step') # debug purposes
    motornum.onestep(direction=direction, style = step_type)


currentDistance =
searchRangeSize = # yet to be determined, probably about 1 degree
minimumStepSize = 0.0056 # degrees, needs to be confirmed

numsteps = searchRangeSize/minimumStepSize
while 399 < currentDistance < 401 # again, may need to process data from rangefinder to put in useful format
for i in range(numsteps): # vertical steps
    for i in range(numsteps):
        stepper_function(motorsObject.stepper1,STEPPER.FORWARD,STEPPER.MICROSTEP) # horizontal
    stepper_function(motorsObject.stepper2, STEPPER.FORWARD, STEPPER.MICROSTEP) # vertical
```

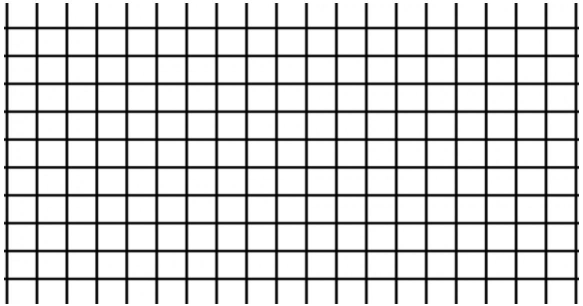
Scan Patterns

- Testing required to determine most effective scan pattern

- Need to achieve 6m x 6m horizontal resolution

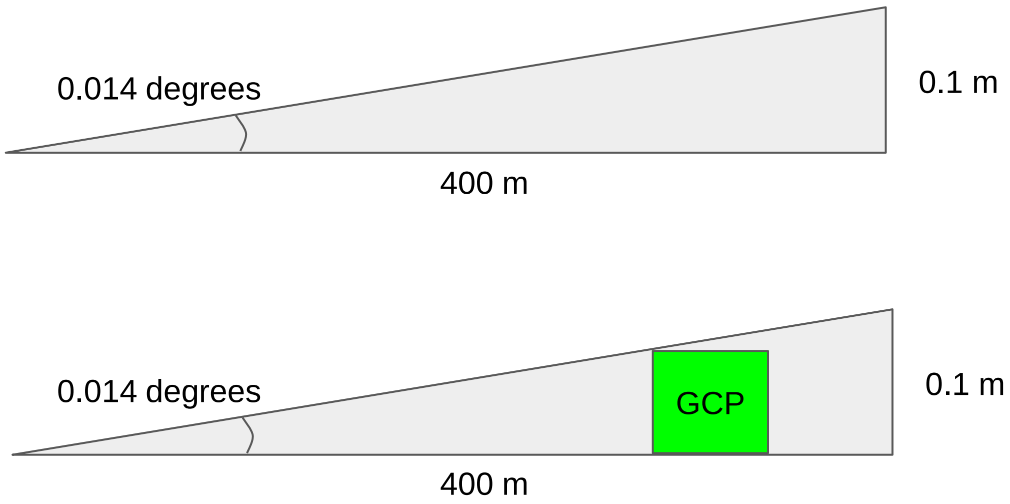
- Options include…
    - Perspective based scan like the top image
    - A orthogonal grid like the bottom one
    - Some combination of the two





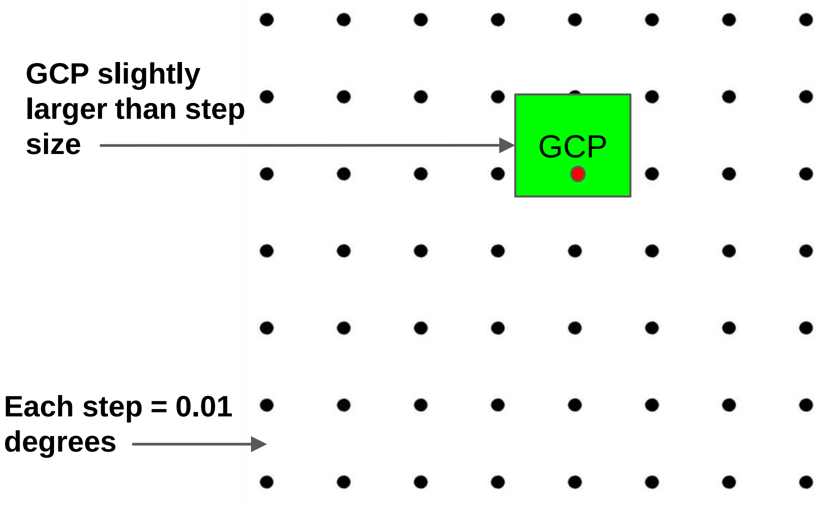
Calibration Search

- Will use a Ground Control Point (GCP) to help achieve desired accuracy
- Ensures the scan starts in the same location each time
- Size GCP s.t. it is within desired angular size
- Then need a software search pattern to find the GCP



Calibration Search cont.

- The software will move the rangefinder to the last known location of the GCP
- Since the GCP is fixed, this should get close to hitting it
- Then the search pattern begins
- Continues searching until known distance of GCP is acquired

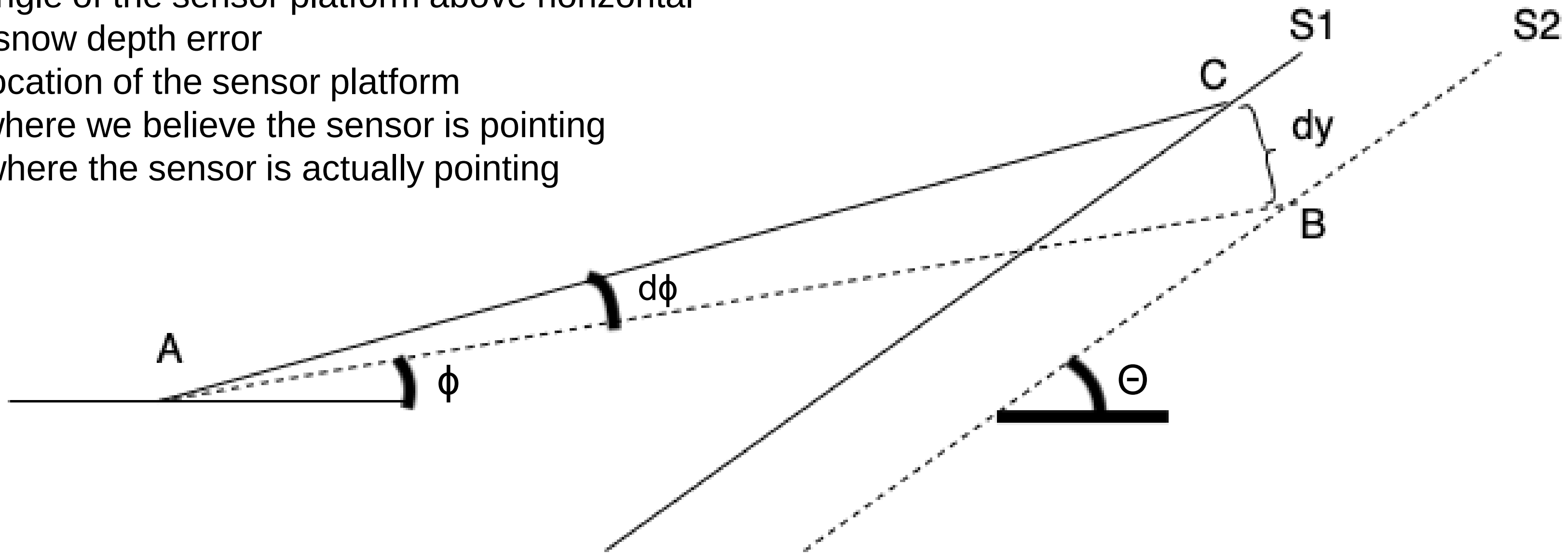


Total size of search grid needs to be determined with testing

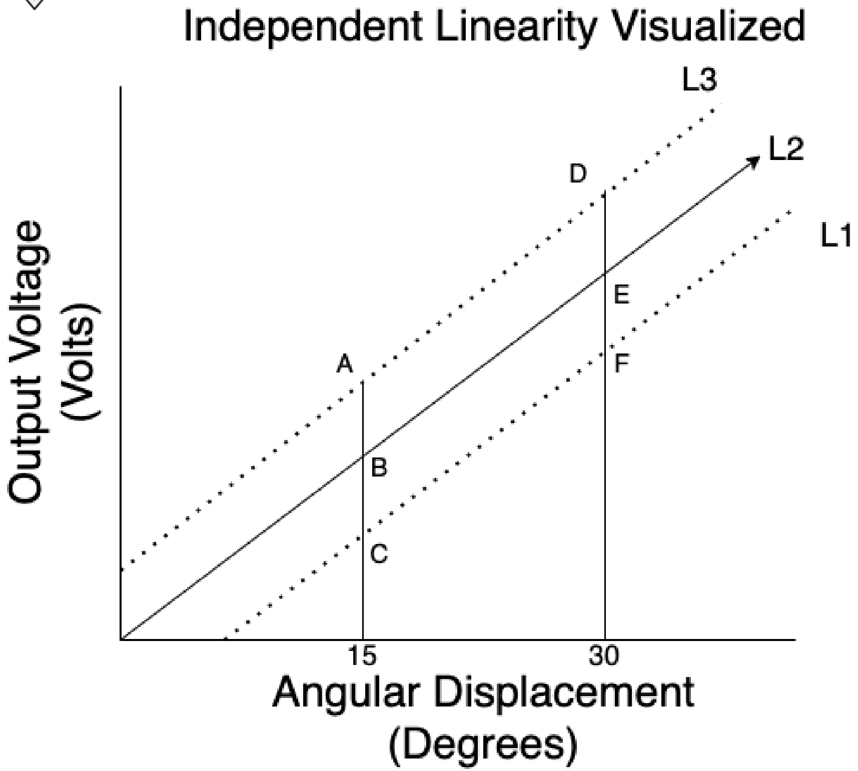
Pointing Error Backup

$$dy = (AC + dAC) * (sin(\phi + d\phi) - [sin(\phi) * sin(\theta - \phi - d\phi)/sin(180 - \theta + \phi)])$$

Θ - angle of the slope
dϕ - potentiometer error
ϕ - angle of the sensor platform above horizontal
dy - snow depth error
A - location of the sensor platform
B - where we believe the sensor is pointing
C - where the sensor is actually pointing



Potentiometer Independent Linearity



L2 - Expected potentiometer behavior
L1/L3 - Possible potentiometer output due to independent linearity
A - Upper bound on potentiometer output at 15° displacement
B - Expected potentiometer output at 15° displacement
C - Lower bound on potentiometer output at 15° displacement
D - Upper bound on potentiometer output at 30° displacement
E - Expected potentiometer output at 30° displacement
F - Lower bound on potentiometer output at 30° displacement

Potentiometer Independent Linearity

At **30°** displacement, the resistance of the potentiometer will be 30° * (0.003636°per Ohm) + 1000 Ohms because that is the resistance of the potentiometer with no displacement, which equals 9250 Ohms

Using Ohm's Law V = IR, V = (0.00012A) * 9250 Ohms = **1.11V** across the potentiometer

But the voltage read may actually be 1.11 V ∓0.12V

Upper resistance bound : R = V/I = 1.23V / 0.00012A = 10250 Ohms

Lower resistance bound : R = V/I = 0.98V / 0.00012A = 8166.67 Ohms

Upper - Lower = 10250 Ohms - 8166.67 Ohms = 2083.33 Ohms of variation

2083.33 Ohms * (0.003636° per Ohm) = **7.58° of error**

- The potentiometer acts as a 1K Ohm resistor at 0° displacement and changes 1 Ohm of resistance per 0.003636° of displacement
- The independent linearity is 1%, which is applied to the input voltage. So if we supply the potentiometer 12V, then our output voltage may deviate ∓0.12V from the expected value
- If we feed the potentiometer 12V, it will have a maximum resistance of 100K Ohms. Using Ohm's Law, V = IR, the current of the circuit is 0.00012 Amps
- Current will be constant in the circuit, so as the resistance changes with displacement, the voltage output will vary

Detailed Structural Model

- Dimensions according to manufacturer specifications and CAD model
- Using stainless steel variant of tripod
- Loading: estimated based on drag on vertical square plate and flow over cylinder
    - 11.4N point load at top of tripod
    - 4.43N total load on tripod mast; load is linearly distributed
- Constraints: guy wire bottom, tripod bottom completely fixed in 3 dimensions
- Guy wire in compression omitted