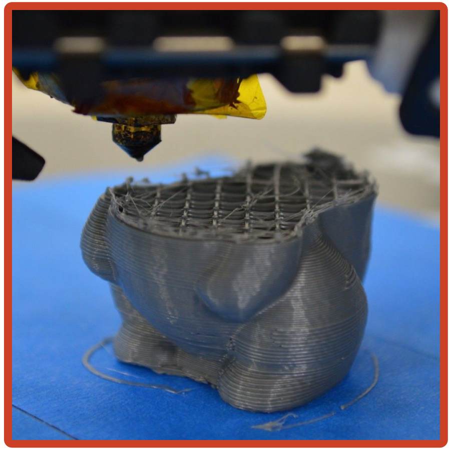
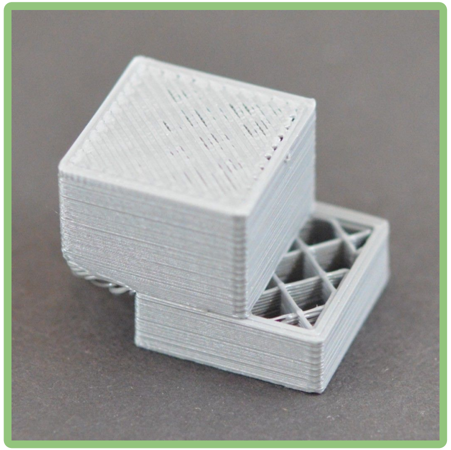
3D Printing Error Detection System

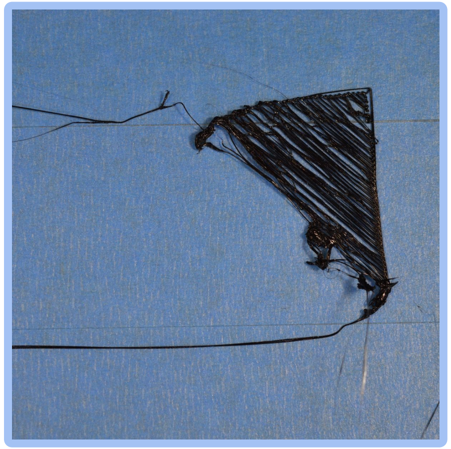
Team E1
Joshua Bas, Hannah Preston, Lucas Moiseyev

Project Summary

- Monitor active 3D prints, detecting errors as they occur, and alert users of potential errors
- Errors to Detect:
  - Extrusion stops mid-print
  - Layer shifting
  - Failing to adhere to the print bed
  - “Hairball”
- Target Printers: Dremel 3D40, Ultimaker U3+, Ramps 1.4 (PrinterBot configuration)





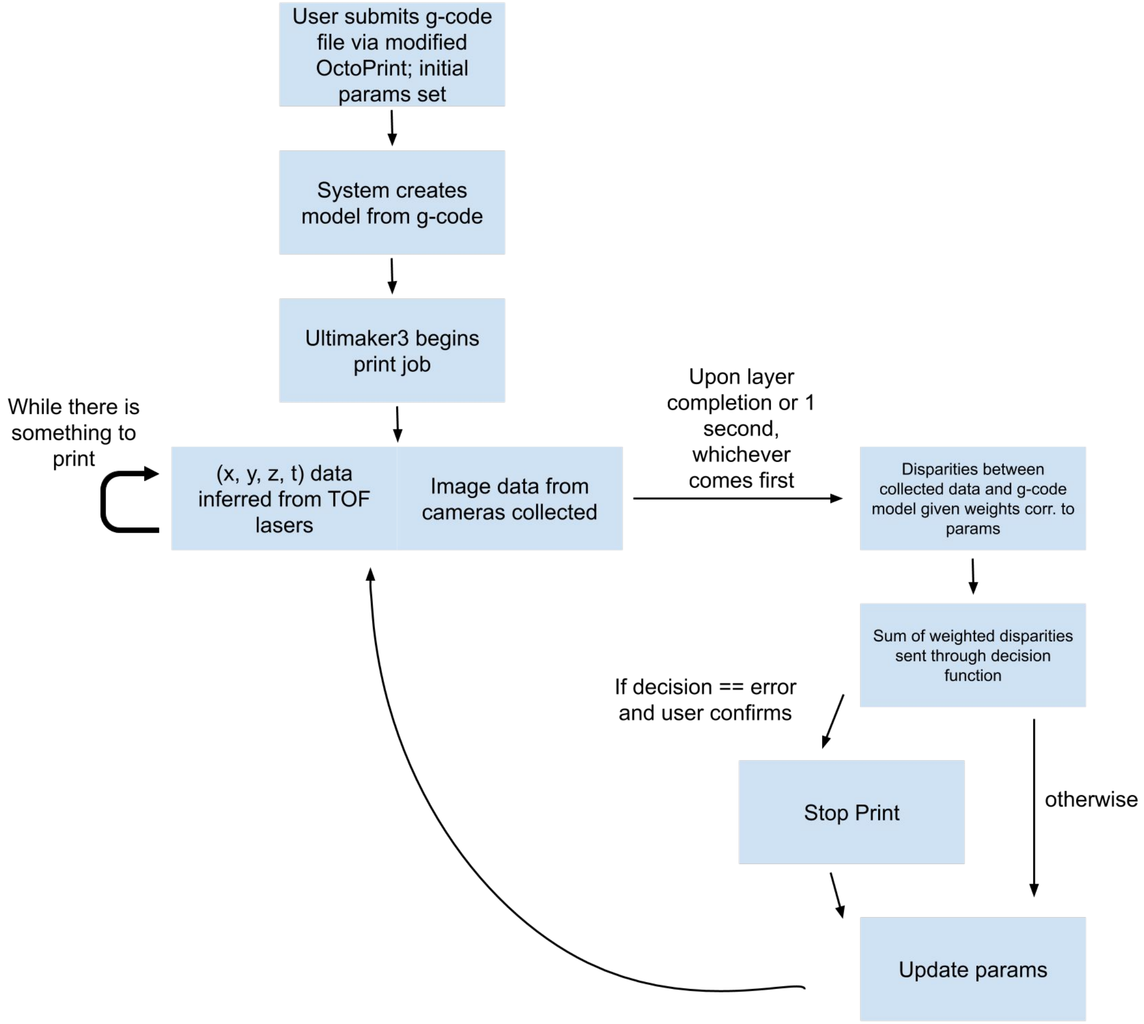


System Requirements

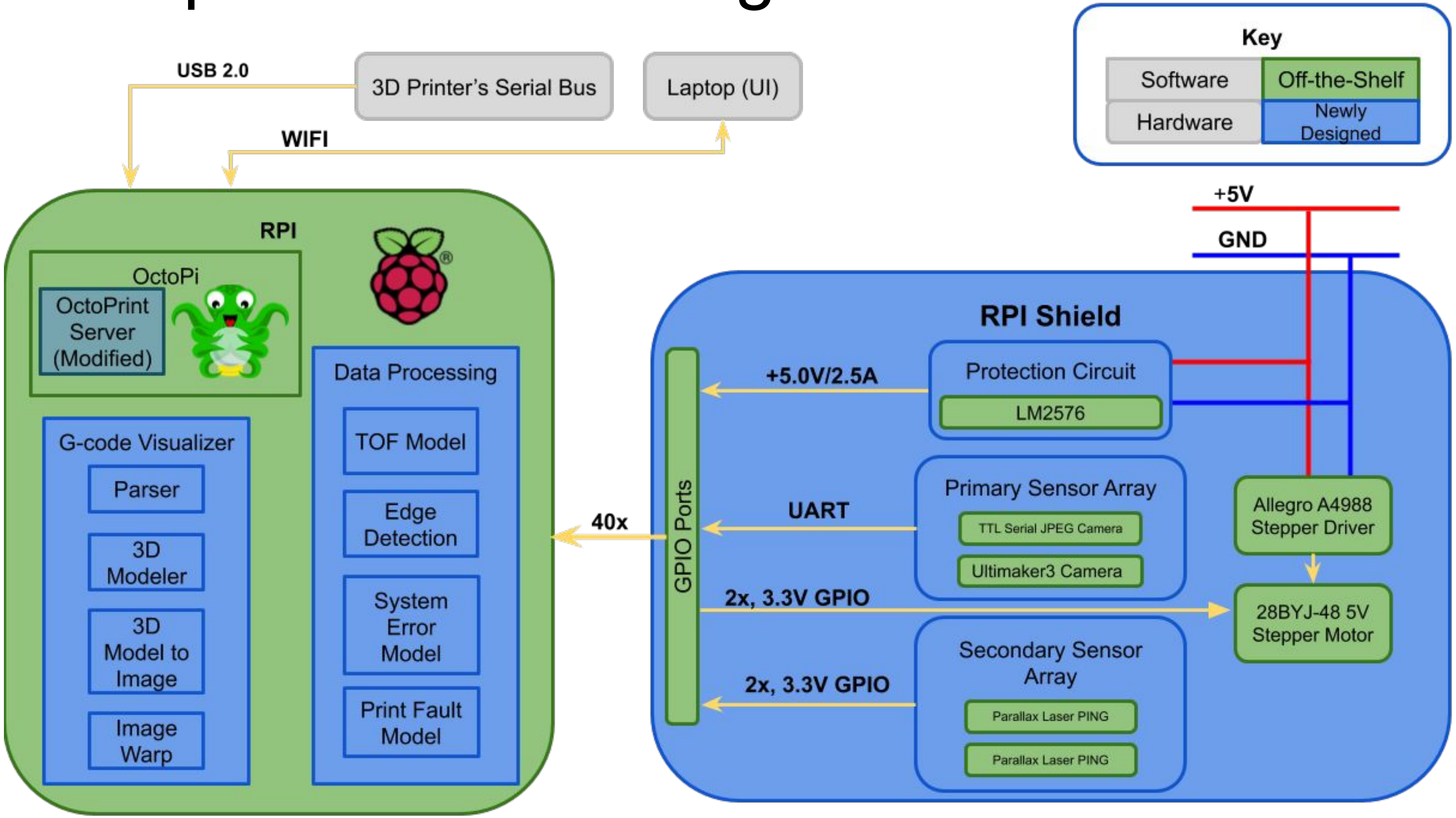
01	**Error Check Rate**	• Calculate when a layer should be completed • Check on layer completion • Otherwise, check every second
02	**Error Detection Rate**	• Detected within 10 checks (~1mm)
03	**Error Detection Accuracy (Average)**	• 85% accurate
04	**False Positive Rate**	• 20% of each detected error is actually not an error
05	**False Negative Rate**	• 10% of each real error is not detected
06	**Runtime**	• Must run at least 6 hours uninterrupted
07	**Size & Weight**	• Within 6 x 3 inches • Weights less than 4lbs
08	**Sensor Coverage Region**	• Covers a 8.9L x 6.7W x 6.7H inch space

Process Flow

- Given an armature length and module angle, we can find the (x, y, z) position of the object.
- Based off of the known speed of the print (read from g-code), the system is able to project how far along the print should be at any given time.
- TOF lasers are narrow, but accurate sensing
- Cameras for “big picture”

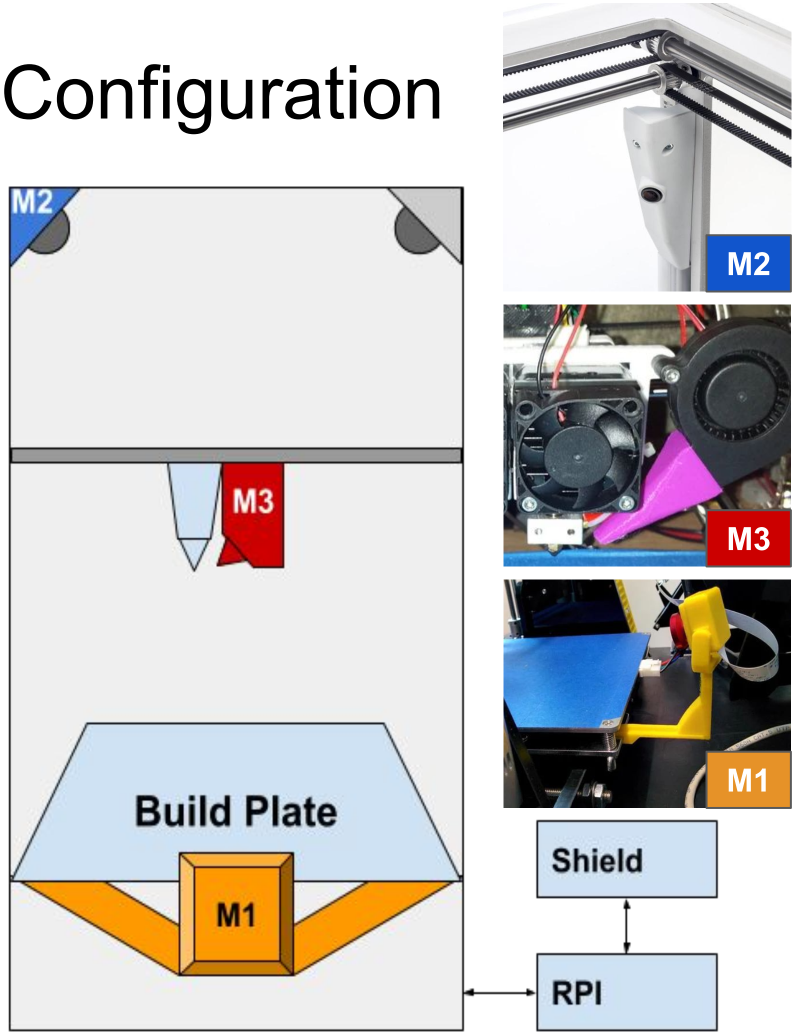


System Specs & Block Diagram



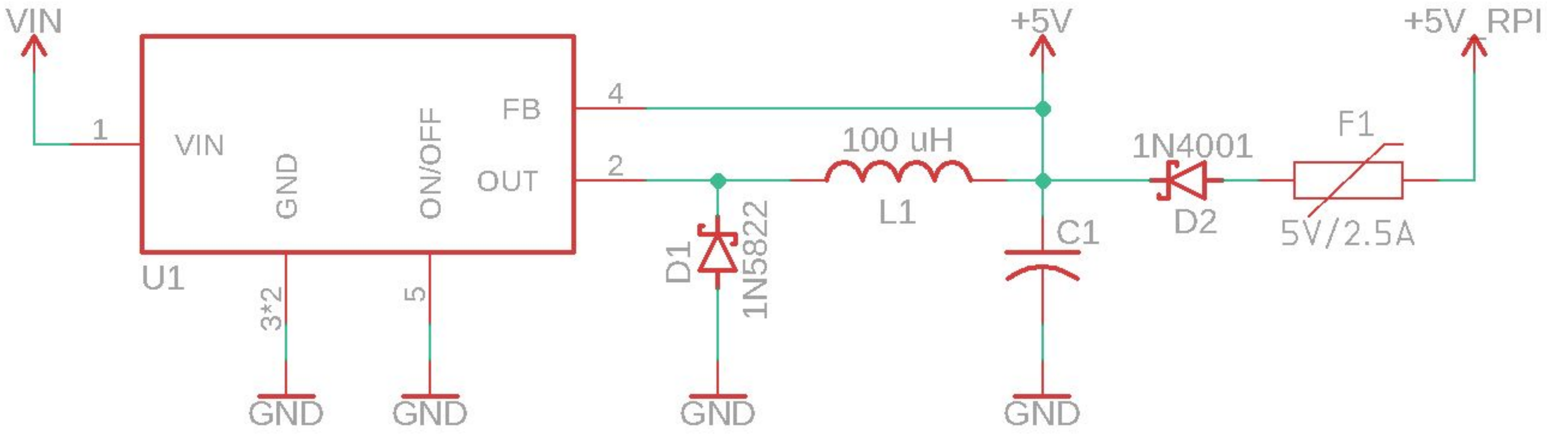
Implementation Plan: Variable Configuration

- Three different custom mounts:
  - M1: Attached on build plate
    - Camera
    - Optionally additional motorized TOF laser (L2) to check layer 1
  - M2: Attached at top corner
    - Camera
  - M3: Attached on extruder
    - TOF Laser (L1)
    - Optionally additional TOF laser (L2)
- Different mounts will be configured based on specific printer configuration:
  - For example, the Ultimaker has no space for an M3 mount, while a PrinterBot style Ramps 1.4 printer has no upper frame to attach an M2



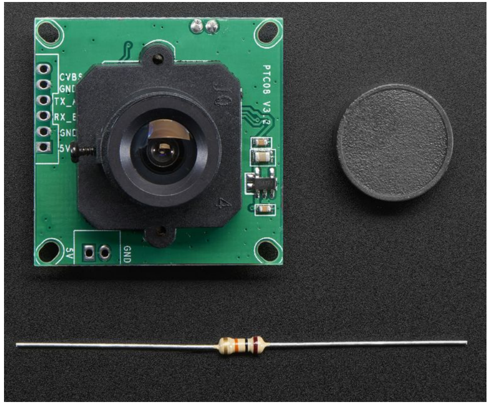
Implementation Plan: Protection Circuit

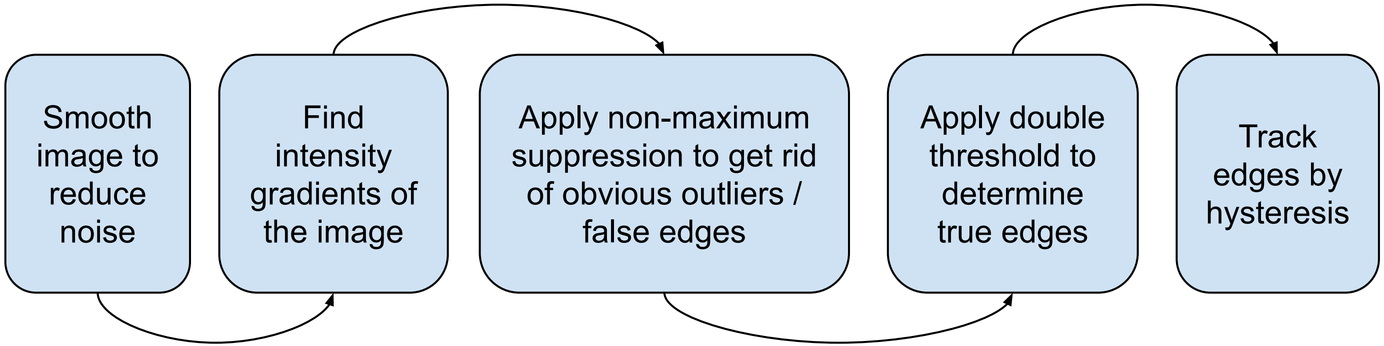
- Powering from the unregulated 5V rail on RPI
- Uses buck converter to efficiently regulate the wall-wart voltage

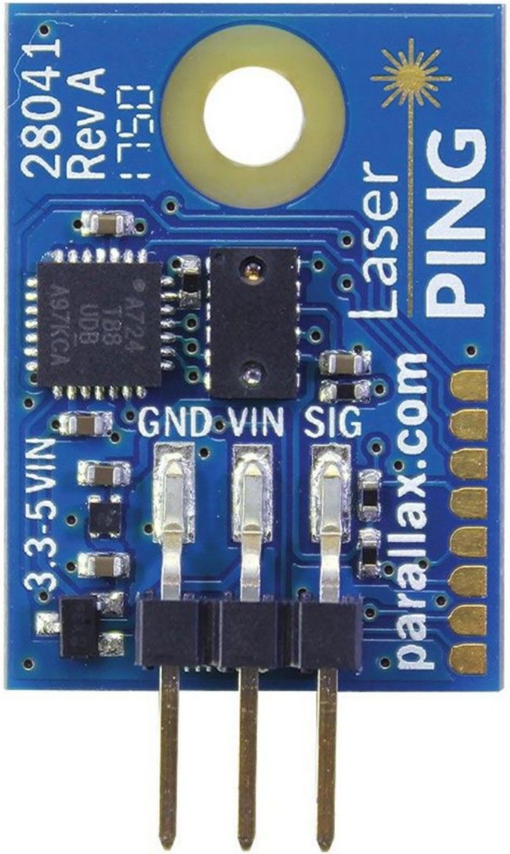


Implementation Plan: Camera

- TTL Serial JPEG Camera with NTSC Video
  - Both snapshot and video features
  - Communication via 3.3V TTL
  - Cost: $39.95
- Existing Ultimaker3 Camera
- UCAM-III 116LENS
  - 116° viewing angle
  - Cost: $10.99
- Camera Use: Edge Detection







Implementation Plan: Time of Flight Laser Rangefinders

- Parallax ToF Laser PING 2m Rangefinder:
  - Range: 2 –200cm
  - Resolution: 1 mm
  - Laser: Class 1 850 nm VCSEL (Vertical Cavity Surface Emitting Laser)
  - Typical refresh rate: 15 Hz PWM mode, 22 Hz serial mode
  - Power requirements: +3.3V DC to +5 VDC; 25 mA
  - Communication: PWM (idle low) or serial 9600 baud (idle high); logic level = VIN
- Cost: $30.00
  - Specifications just barely fulfill our requirements
  - Focusing on reducing cost as much as possible
  - We will re-spec if this module proves ineffective
- Connect via software pwm on one RPI GPIO pin

Risk Factors

- System not being able to make up for accumulated calibration errors
    - Solution: user confirmation has greater say in the decision
- Higher false positive rate than desired:
    - Solution: fine-tune initial parameters
    - Drastic solution: spec higher resolution sensors
- Mounts are heavier than desired:
    - Solution: use lighter materials e.g. less infill
    - Drastic solution: spec lighter sensor modules
- RPI not able to process OctoPrint with our software customizations
    - Solution: Parallelize work between two RPI's (one running lightly modified OctoPrint, the othe running our CV)
    - Drastic solution: spec heavier-duty microcontroller/sbc

Metrics and Validation Plan

1. Method: Physical Error
    a. Begin printing
    b. Pause printer and our system
    c. Physically cause an error
    d. Resume printer and our system
    e. Record if the error was detected
2. Method: Programmatic Error
    a. Load faulty g-code into printer; load correct g-code into error detector
    b. Begin printing
    c. Record if the error was detected
3. Method: Real World Case Error
    a. Use a print prone to certain errors
    b. Record if error was detected

Project Management

- Project broken down into key areas
- Shifted previous tasks to better align with newer technical approach
- Built in slack weeks over Spring Break and week of 4/12

Lucas, Joshua, Hannah, L+J, L+H, J+H, Team	Week of 1/19	Week of 1/26	Week of 2/2	Week of 2/9	Week of 2/16	Week of 2/23	Week of 3/1	Week of 3/8	Week of 3/15	Week of 3/22	Week of 3/29	Week of 4/5	Week of 4/12	Week of 4/19	Week of 4/26	Week of 5
Course Logistics																
Proposal Presentation			Hannah ✓													
Design Presentation						Joshua										
Make Video Documentation										Lucas			Team			
Make Poster Board/Presentation													Team			
Final Presentation															Lucas	
Final Demo																Team
Research																
Research translating g-code into images			Hannah	✓												
Research camera views				✓												
Trade Study for Camera			L+H		✓											
Trade Study for Camera Lenses			L+H		✓											
Explore Device Positioning on 3D Printers				Team	✓											
Explore Remote 3D Printer Access				Team	✓											
Software Design																
Design edge detection (block diagram, documentation, etc.)				Hannah												
Design g-code visualizer (block diagram, documentation)				J+H												
Write edge detection function					Hannah											
Write g-code parser function						Hannah										
Write g-code visualizer (3D model)							Hannah									
Write g-code visualizer (3D model to 2D image)									Hannah							
Write g-code visualizer (warp 2D image w/ fisheye)									Hannah							
Set Up Octo-Print on Ultimaker with baseline webcam					Lucas											
Hardware Design																
Design MCU/Processing Subsystem				Lucas												
Design Power Management Subsystem				Joshua	✓											
Design Preliminary Mounting System					Lucas											
Design Preliminary Shell					Lucas											
Build out Prototype with COTS Dev Boards						L+J										
Preliminary RPI Shield Power Schematics/Layout							Joshua									
Final RPI Shield Schematic/Layout Iteration											L+J					
Design Final Mounting System												Lucas				
Design Final Shell												Lucas				
Integration																
Load edge detection onto prototype							Team									
Load g-code visualizer onto prototype							Team									
Load edge detection onto p-SBC											Team					
Load g-code visualizer onto p-SBC											Team					
Load edge detection onto f-SBC														Team		
Load g-code visualizer onto f-SBC														Team		
Testing																
Test g-code parsing function						J+H										
Test edge detection function						Hannah										
Test g-code visualizer (3D model)									Hannah							
Test g-code visualizer (3D model to 2D image)									Hannah							
Test g-code visualizer (warp 2D image w/ fisheye)											Hannah					
Test WIFI Output Subsystem (connects to website)							Team									
Test Power Management Subsystem (good voltage/currents)							Team									
Run Preliminary System on Multiple Printers												Team				
Run Final System on Multiple Printers (nice)														Team		
Implement Core Website Backend (MVP)							L+H									
Implement Website UI/UX/Frontend (MVP)									Hannah							
Iterate on Website Design											Team					