



# DistoY

By: Isaac Kirsch

[kirschic@mail.uc.edu](mailto:kirschic@mail.uc.edu)



# Abstract

DistoY is a cave surveying device aimed at providing cavers with a larger variety of functionality for surveying at a much lower cost than current products on the market. The DistoY will allow for taking distance measurements that can also record angles, track cardinal direction, and store data for later use. Furthermore, it will also be built to survive in the harsh conditions of caves, such as being dropped from heights, exposed to lots of dust, cold temperatures, and wetness or submersion in water.

# Goals

- Create a device that can measure distances, angles, and cardinal direction
- Record data points for further use
- Provide a more robust alternative to the devices on the market
- Remain affordable



# Intellectual Merits

Standalone design

Open-source and  
simplistic

Cave-specific  
solution

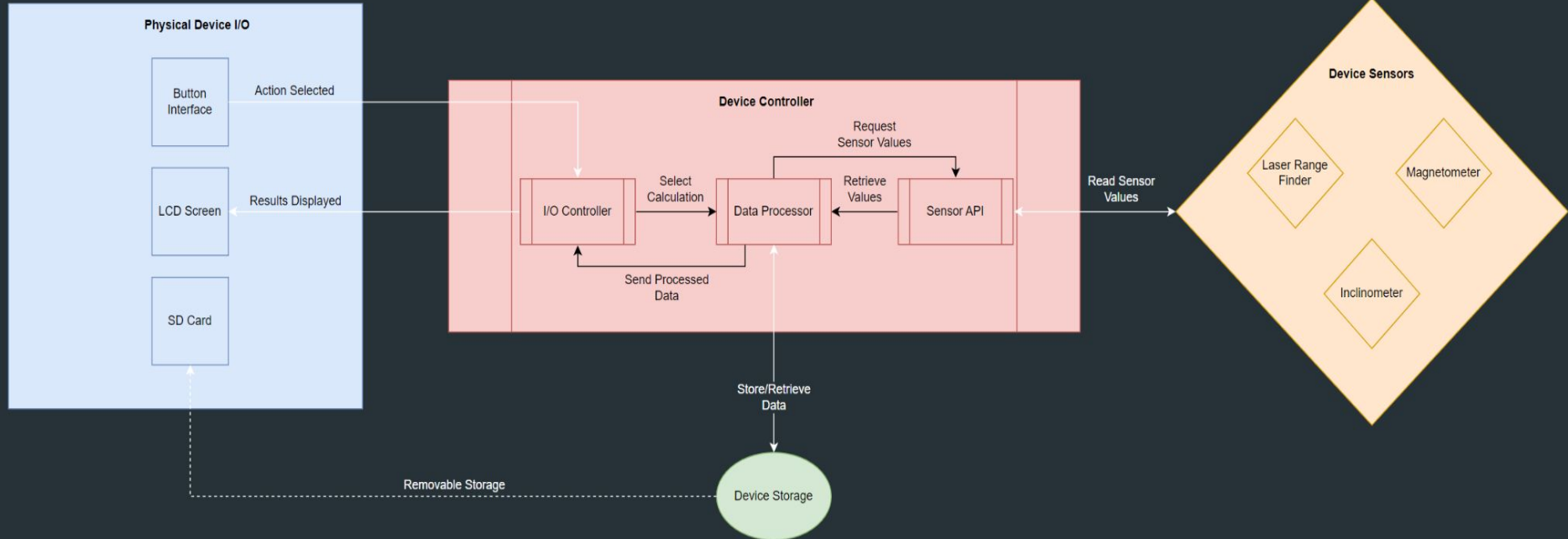
Modularity and  
upgradeability

# Broader Impacts

- Provide more detailed cave topographies and maps
  - Safer recreational use of caves
  - Increased chance of survival in case of accidents
- Increased speed at which cavers can collect information
  - Less time spent in cold and uncomfortable conditions
  - Increased number of caves that can be explored
- Basis for cave device innovation
  - Provides an open-source example of what cavers need and want

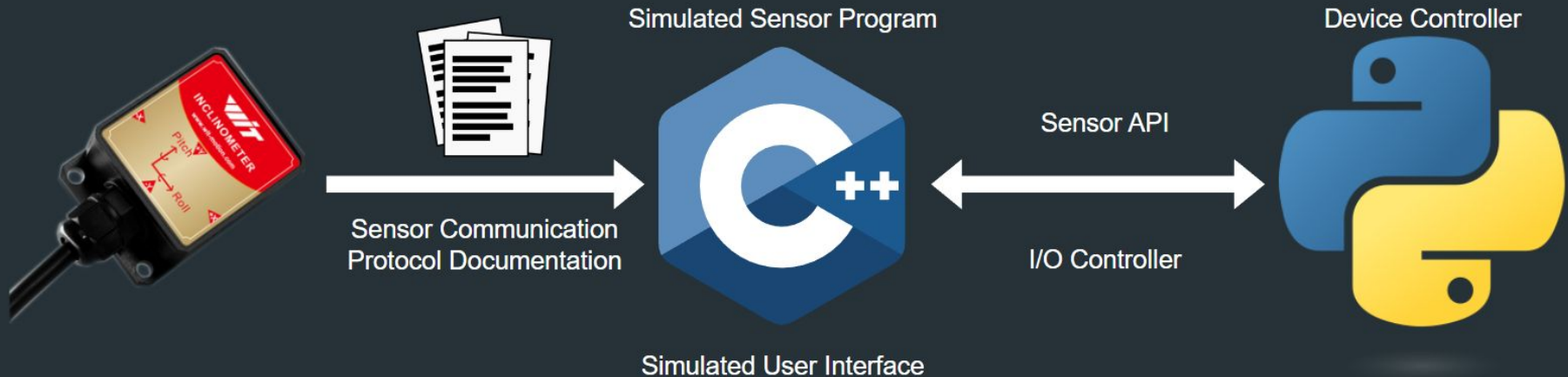


# Design Diagram



# Simulation Specifications

- Run as multiple program on a single machine
- Program interactions occurs over a file system
- Device I/O is performed through the use of a terminal



# Technology Used



C++

Used to simulate the sensors and microcontroller functionality



Python

Implemented the device program functionality, such as taking a measurement and calibration



Raspberry Pi

Used as the microcontroller for the device prototype



Sensors

Used to provide functionality of the device in the real world. Documentation was used to create the simulation programs.

# Milestones

---

## 1. Sensor Selection

Selected microcontroller, sensors, battery, physical connectors, and I/O needed for the device.

Identified code libraries necessary for speeding up development time.

Completion: 10/2022

## 2/5. Device Simulation

Simulated sensors and microcontroller I/O functionality using documentation.

Implemented as separate programs interacting over a filesystem, used to develop without access to all the sensors.

Completion: 12/2022

## 3. Sensor API

Implemented the ability to request and read values from all the sensors.

Designed all code logic to avoid dealing directly with sensor languages, allowing it to be easily switched to a physical device.

Completion: 01/2023

## 4. Measurement & Calibration

Developed a measurement routine combines readings from all the sensors to give a full data point.

Designed a basic calibration function that could identify and correct sensor drift.

Completion: 02/2023

## 6. Data Storage

Implemented a method for converting sensor data into a standardized JSON format.

Included additional metadata to the data point, and allowed for storage on an SD card for later use.


Completion: 03/2023



# Results

- Simulation programs for each sensor
- Sensor API for interacting with each sensor
- Measurement and calibration routines
- Data and metadata storage

# Challenges & Future Work



---

---

- Supply chain issues delaying sensors and components arriving
- Designing mockups using only available documentation
- More rigorous testing for calibration and sensor accuracy
- Increased functionality, such as comparison shots, burst measurements, and multi-stepping
- Constructing a final physical design