Paintball Environment Tactical Engagement Recon System (P.E.T.E.R.S.)

Group Number: BCC-4
Richard Taylor – C.E.
Anthony (AJ) Schmidt – E.E.
Kenneth Hale – E.E.
Brett Reich – C.E., E.E.
Antonio Foster – E.E.

Advisor
Dr. Christopher Peters – E.C.E.

May, 2016 – Spring Quarter

Presentation Outline

- Problem Statement
- Proposed Solution
- Electrical Schematics
- Standards
- Team Member Responsibility
- Societal Impact
- Budget – Out of Pocket
- Testing and Validation Results
- Demo Video

Problem Statement

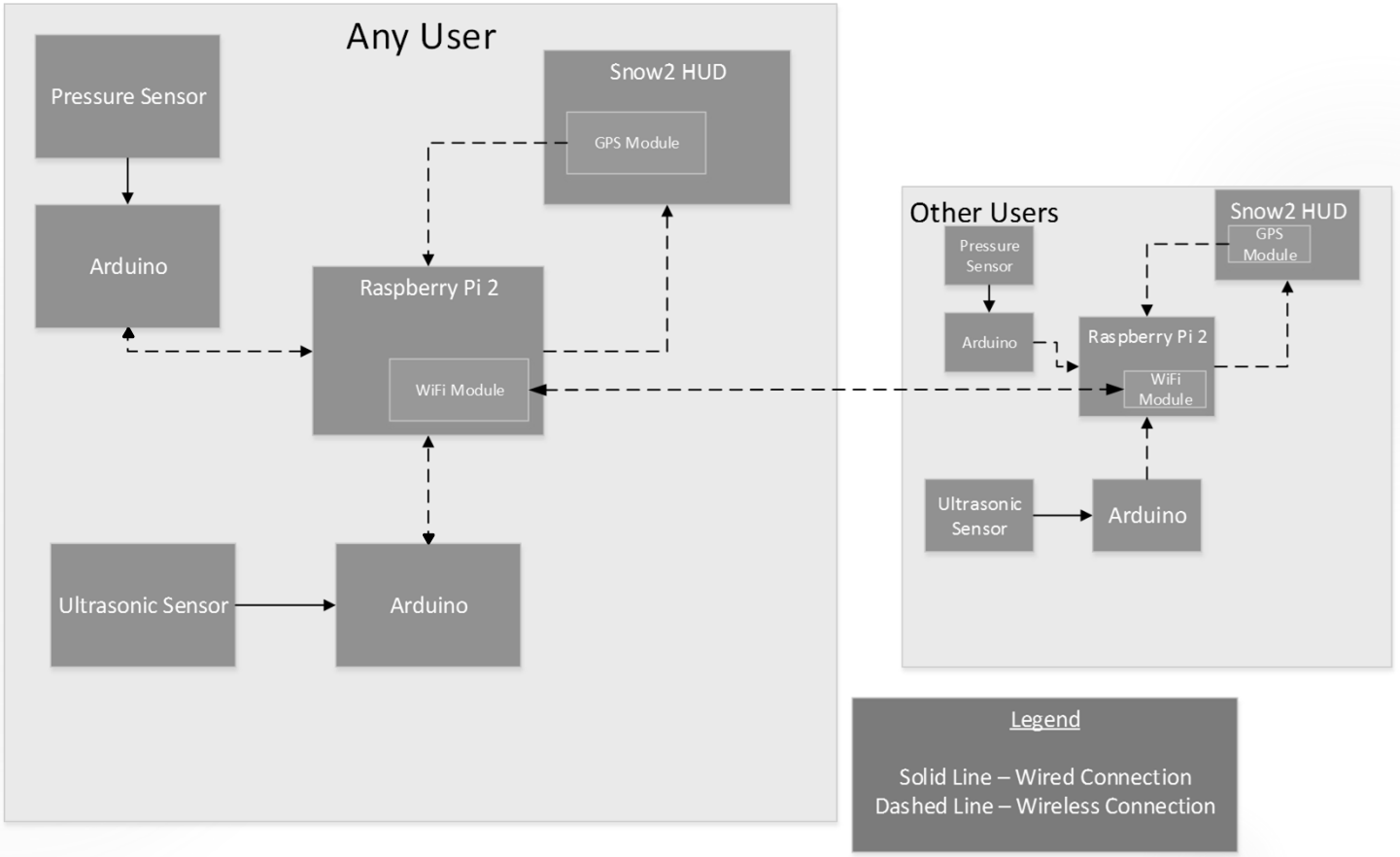
- Our research indicates that there is currently no single product available that addresses all of the most common distractions that a player faces while playing the game of paintball.
  - Teammate location
  - Air pressure remaining
  - Paintballs remaining
- We utilized various sensors, a display, and a communication network to develop a single system which alleviates these distractions.

Proposed Solution

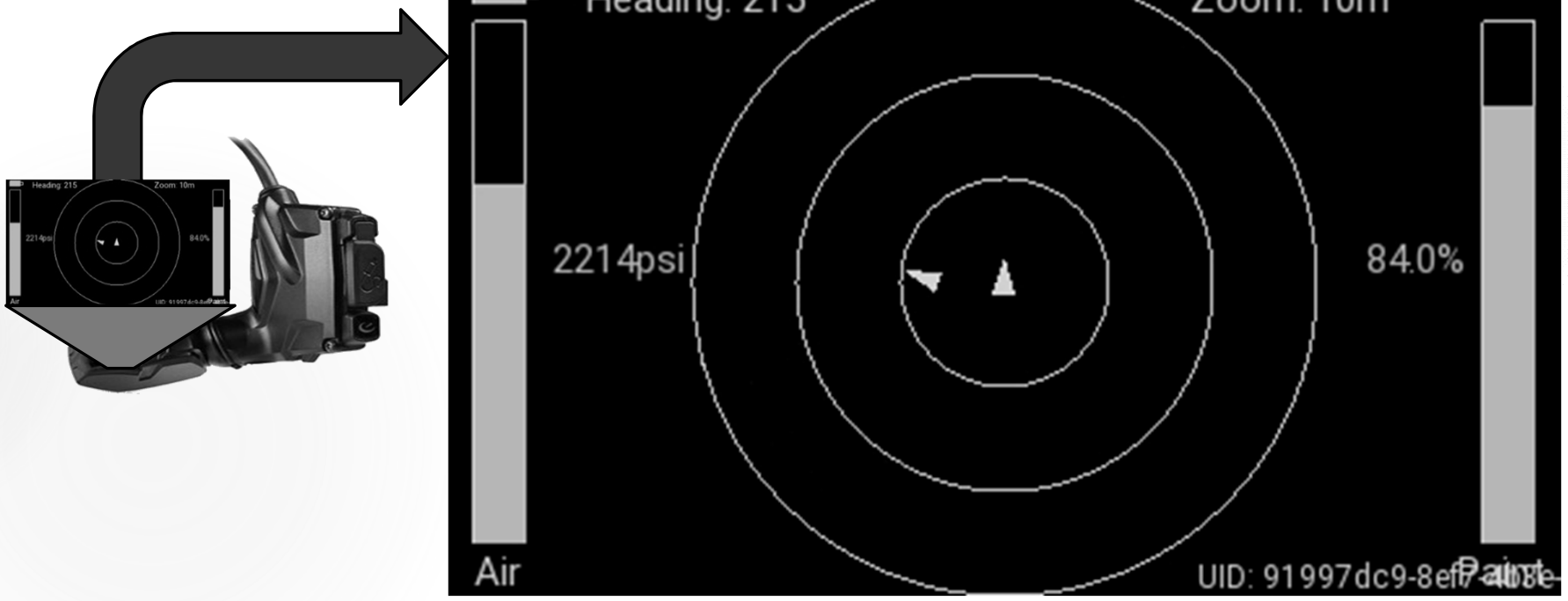
A system consisting of the following:

- Honeywell Pressure Sensor mounted to air tank regulator
  - Collects real time data on available pressure in tank
- Hopper with Ultrasonic Sensor System
  - Monitors available paint levels in hopper
- Arduino Pro Minis with XBee Dongles
  - Wirelessly transmit data from sensors to server
- Raspberry Pi 2 with Wi-Fi/XBee Dongles
  - Processes data
  - Share data between users
- Recon Snow2 Heads Up Display (HUD)
  - Displays Information to the user
  - Transmits user GPS data to the Raspberry Pi

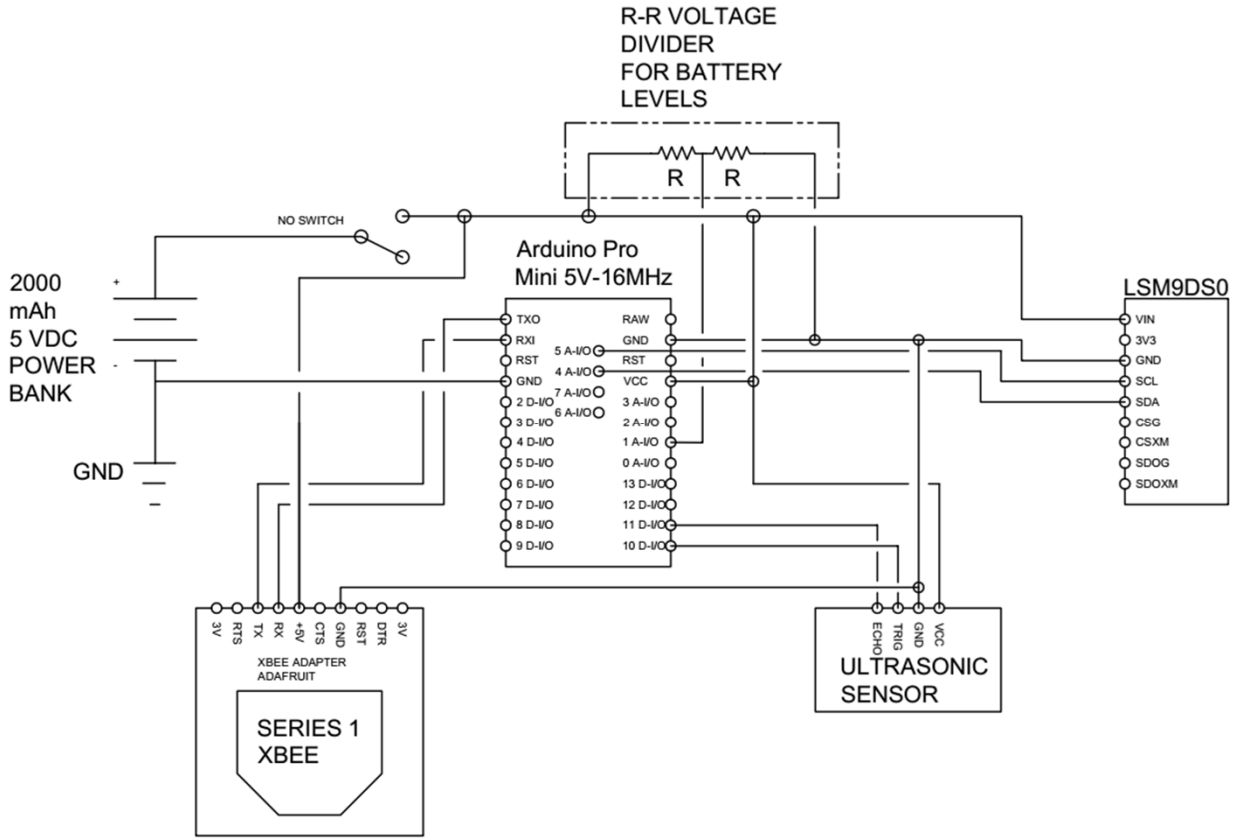
Proposed Solution – Block Diagram



Proposed Solution – HUD GUI

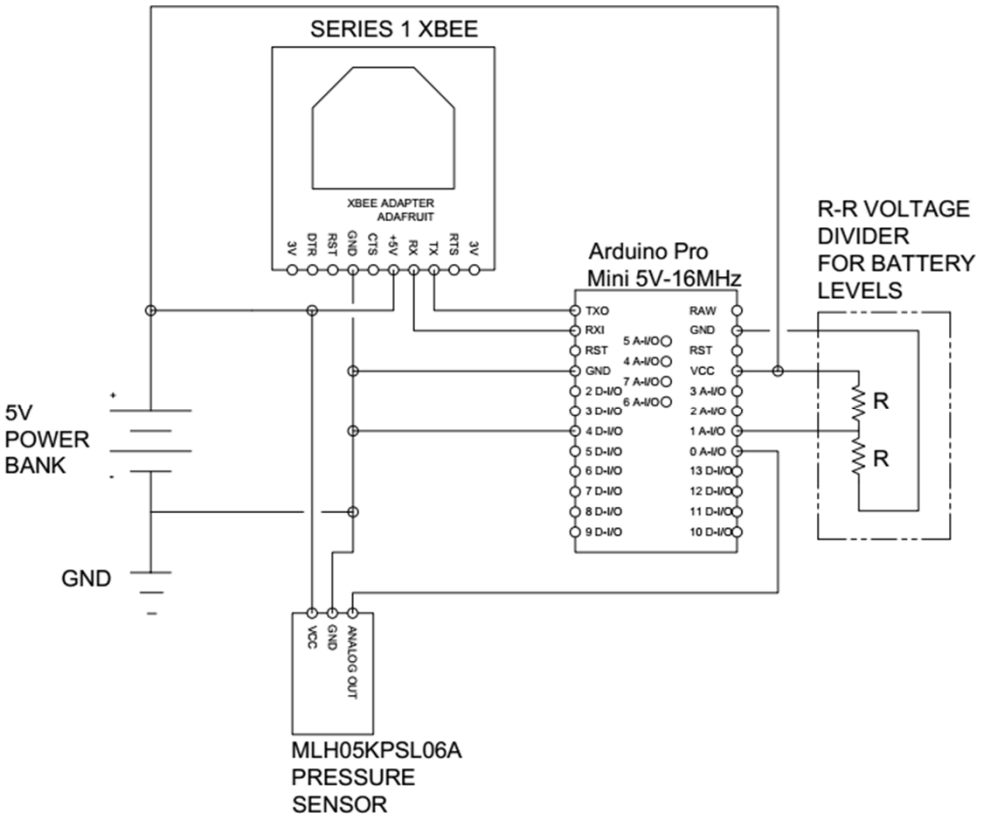


Electrical Schematics

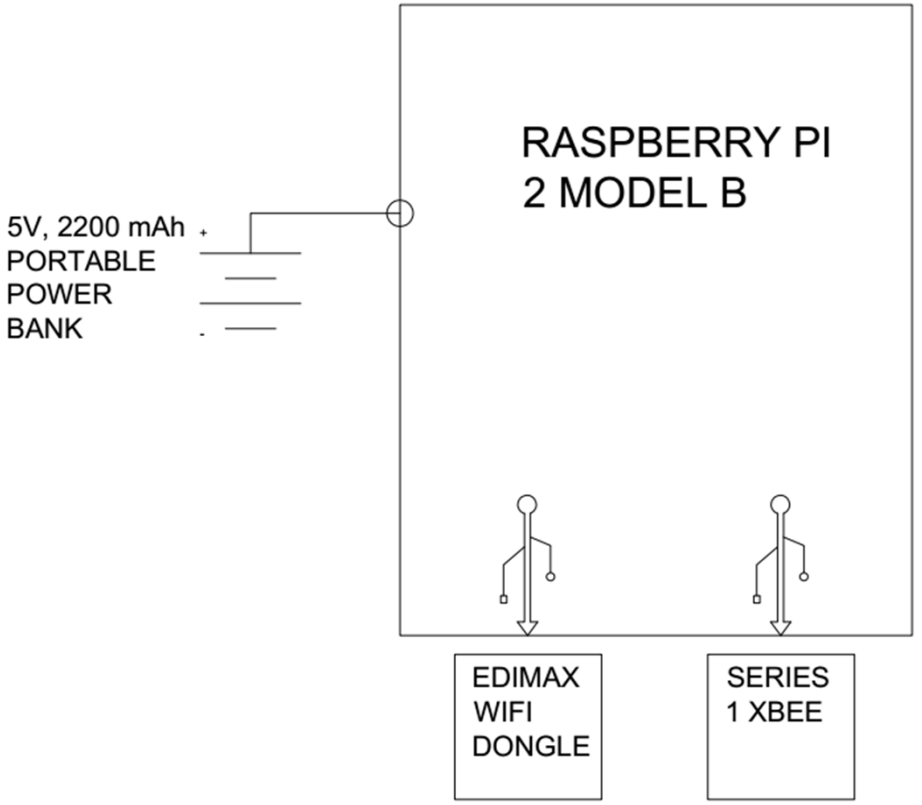


Paintball Hopper Schematic

Electrical Schematics



Compressed Air Tank Schematic



Raspberry PI Schematic

Background Research - Standards

- Standard ECMA-404 (*JSON Data Interchange Format)*
- Standard IEEE 802.11 – Wi-Fi
- Standard IEEE 802.13 - ZigBee
- Standard IEEE 802.15 - Bluetooth
- EPA regulation 40 CFR 273.13(a) (Waste management requirements for small quantity handlers of universal waste batteries)

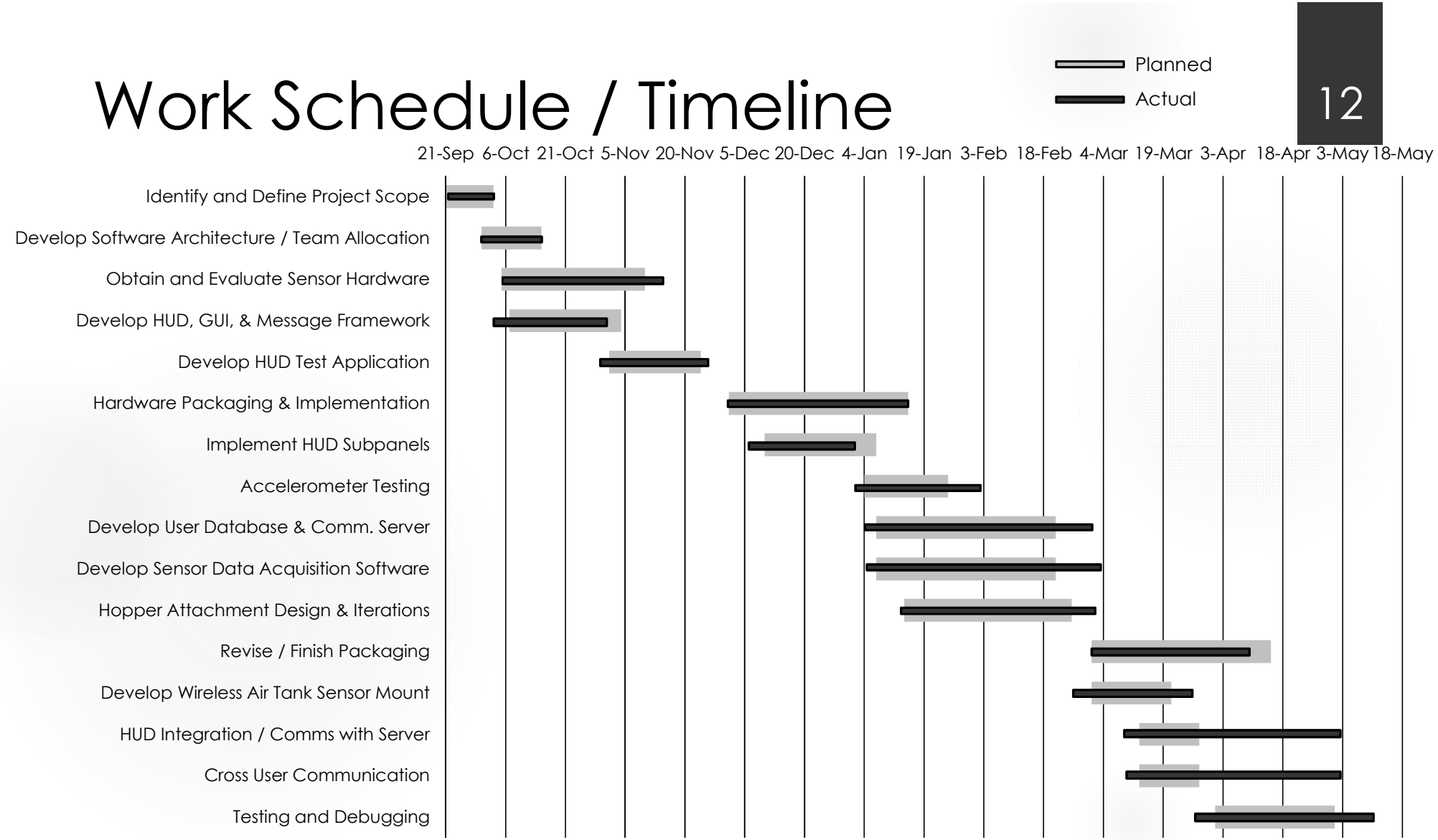
Team Member Responsibility

Member	Individual Contributions to the Project	
Richard Taylor	• HUD GUI application code • Xbee communication	• Database and communication server code
Brett Reich	• Database and communication server code • Sensor array data gathering/processing code • HUD GUI application code	
Kenneth Hale	• Sensor array data gathering/processing code Accelerometer filtering • Assist in 3D modeling	• Arduino to PI communication • Prototype manufacturing
AJ Schmidt	• 3D modeling for prototype • XBee communication testing • Prototype manufacturing	• Ultrasonic sensor feasibility analysis • Sensor data collection coding
Antonio Foster	• 3D modeling for prototype • Sensor data filtering • Prototype manufacturing	• Ultrasonic sensor feasibility analysis • Sensor data collection coding

Societal Impacts

- Targets existing players and large gaming community
- Potential to get thousands of people out of the house and be more active
- Possible Integration into Civil Service/Military Applications

Work Schedule / Timeline

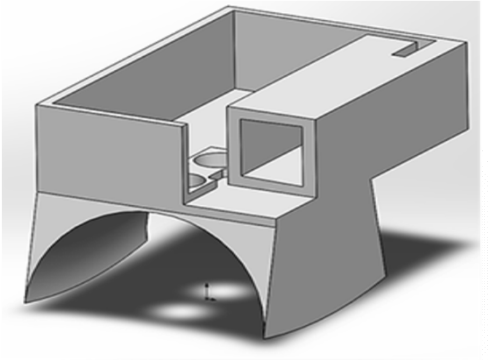


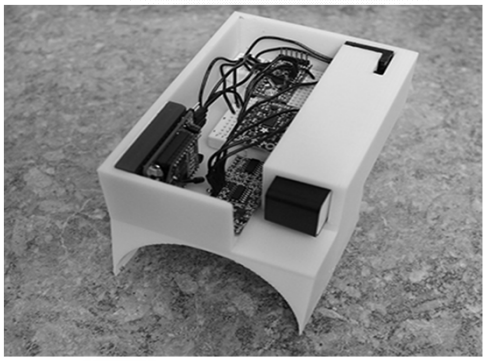
Budget – Out of Pocket

Model Name	Unit Cost	Units	Sub-total
AA Battery Pack	$19.49	2	$38.98
Micro-Bluetooth 4.0 LE	$8.99	2	$17.98
Micro-SD 16GB (Sony 70Mb/s)	$8.99	2	$17.98
CanaKit Raspberry Pi 2 + case	$46.99	2	$93.98
SNOW2 (HUD only)	$399.00	2	$798.00
MLH05KPSL06A – Pressure Sensor	$146.14	2	$292.28
Photo-Resistor (20pcs)	$4.69	1	$4.69
SNOW2 (HUD + Goggles)	$549.00	1	$549.00
Laser Diode (5 pcs)	$3.54	1	$3.54
Edimax EW-7811Un Wi-Fi Adapter	$9.99	2	$19.98
Female / Male / Male 1/8th	$36.47	1	$36.47
Shipping (for goggles)	$54.99	1	$54.99
Arduino Pro Mini	$20.00	2	$40.00
Adafruit 9-DOF LSM9DS0	$20.49	1	$20.49
		Total	**$1,988.36**
		Per Person	**$397.67**

Testing & Validation – Unit Testing

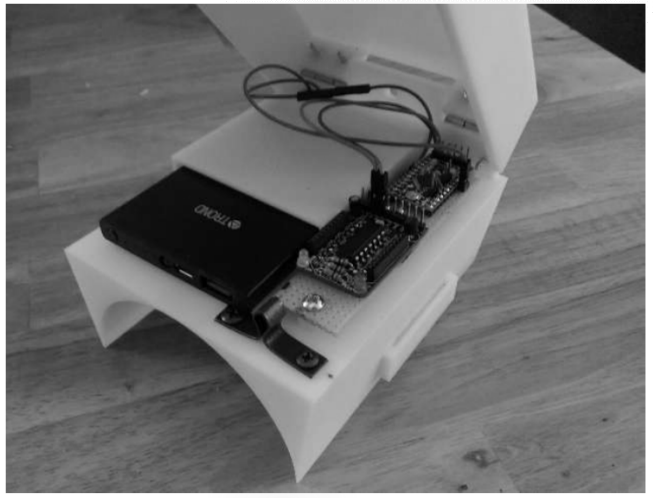
- Hopper Mount Assembly
  - Ultrasonic distance sensor
    - Calibrated sensor outside hopper
    - Calibrated in hopper with varying paint levels
    - Smoothed sensor data
  - Accelerometer
  - Battery drain testing
    - 16.6 hours processing/transmitting
    - 21.2 hours processing/ NO transmitting





Testing & Validation – Unit Testing

- Air Tank Mount Pressure Sensor Assembly
  - Analog readings vs Digital readings
  - PSI/Voltage testing
  - Battery drain testing – 37.2 hours processing/transmitting
  - Circuitry validation
    - Battery drain issue



Testing & Validation – Unit Testing

- Heads-Up Display (HUD) Application
- Determined type/resolution of data returned from GPS
  - Simulated data processing/rendering with ‘virtual’ users
  - Is everything displayed as it should be?
- Message handling/dispatching
  - Correct handling of incoming messages and associated data
  - Correct message dispatching with appropriate data
  - Position update contention testing
- User interaction
  - Zoom level
  - User ID selection

Testing & Validation – Communication Testing

- Server socket contention resolution
- Automatic reconnection verification
- Next-server fallback testing
- Invalid input handling
- Timeout handling
- Self-testing tool suite for network messaging primitives

Testing & Validation – System Testing

- Field testing
  - Repetition to verify data consistency
  - Air pressure and paint level data
- Screen capture utility (Android Debug Bridge & tablet)
  - Allowed us to reference screen captures
- Player position verification
  - Logged player position
    - Walking
    - Running
    - Driving
  - Compared logged latitudes/longitudes with real-world locations

Testing & Validation – Results

- All data successfully displayed on the HUD module
- Air data and paint level data are accurate
  - Air data is direct tank reading
  - Paint level data sufficiently processed
- Positional data is partially reliable
  - User GPS data verified to be within 1 to 5 square meters of actual real world position
  - Heading data from GPS module is UNUSABLE for this application
    - Relies totally on motion to update
    - Extremely slow to update
  - This system will require a functioning INU for proper user location determination

Demo Video

Questions

Supplemental Slides To Follow

Typical User Setup

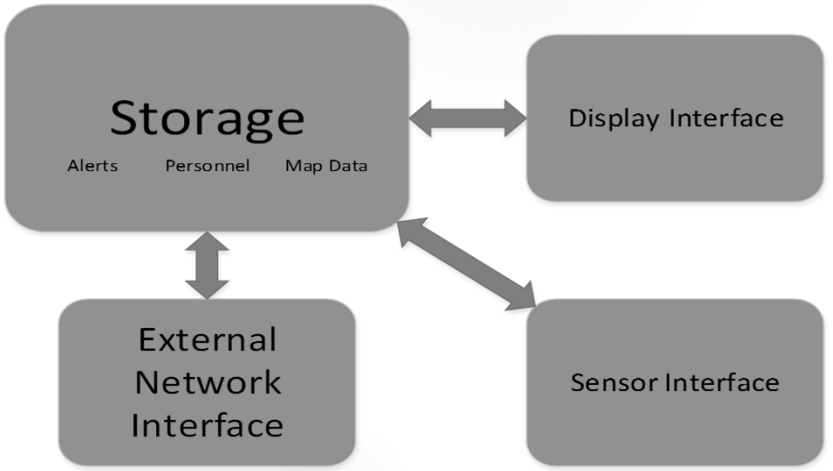
Fall Progress – Hardware

- All equipment purchased and acquired
- Honeywell Pressure sensor mounted and tested in 2 locations
- Gathered important data using Arduino
- Preliminary Break Beam Feasibility Test Completed

A – Pressure Sensor Mounted on Marker
B – Pressure Sensor Mounted on Tank

Fall Progress - Software

- Base classes for GUI display framework
  - Moving Map Overlay
    - Utility classes (i.e. Latitude/Longitude to Pixel Position)
    - Supplemental classes (i.e. overlay constructors, icons, alert formatting, etc.)
  - Air Tank Readout and Paint Level Overlay
- Python Application Packaging for Android
- Basic network framework
  - Communication of JSON formatted messages
- Data Collecting Utility
  - Gathered important data from Pressure Sensor
  - Paint detection algorithm



Winter Progress – Hardware

- Paint Level Detection System (Completed)
  - Components:
    - Accelerometer/Gyroscope – Track orientation of marker so only meaningful data is used
    - Ultrasonic Sensor – Measures current paint level in user's hopper
    - Arduino Pro Mini – Microcontroller to collect data from sensors
    - XBee – Wireless serial communication with server
    - Battery – Removable and rechargeable USB 5V power bank
- Compressed Air Level Detection System (Model nearing completion)
  - Components:
    - Honeywell Pressure Sensor – Measures current air pressure remaining in user's tank
    - Arduino Pro Mini – Microcontroller to collect data from sensor
    - XBee – Wireless serial communication with server
    - Battery – Removable, rechargeable USB 5V power bank

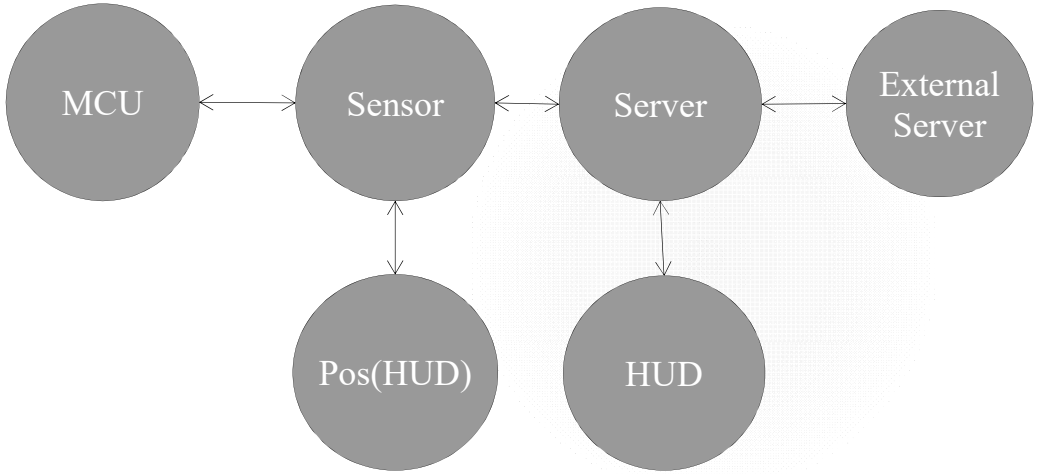
Winter Progress - Software

Estimated Data Rate Requirements:

MCU	->Sensor:	s * [148, 15]
Sensor	->Server:	[735, 12]
HUD	->Server:	[18 + n*18, n*738]
Pos (HUD)	->Sensor:	[220, 17]
Each Server	->Server:	[18, 735]

Significant Numbers:

Localhost Combined: 747
Node Internal Serial: s * [163]
Node Internal Socket: [984] + n * [753]
All External Socket: n * (n-1) * [753]
Total: n * 984 + (n*n) * [753] + n * s * [163]



Key:

Rate Units:
[Up/Down] (Bytes/Sec) * (Refresh Rate)

Expansion Scaling:
s → Number of Serial Sensor MCU Node Per Node (Normally s=1)
n → Number of Nodes Total (Normally n > 1)

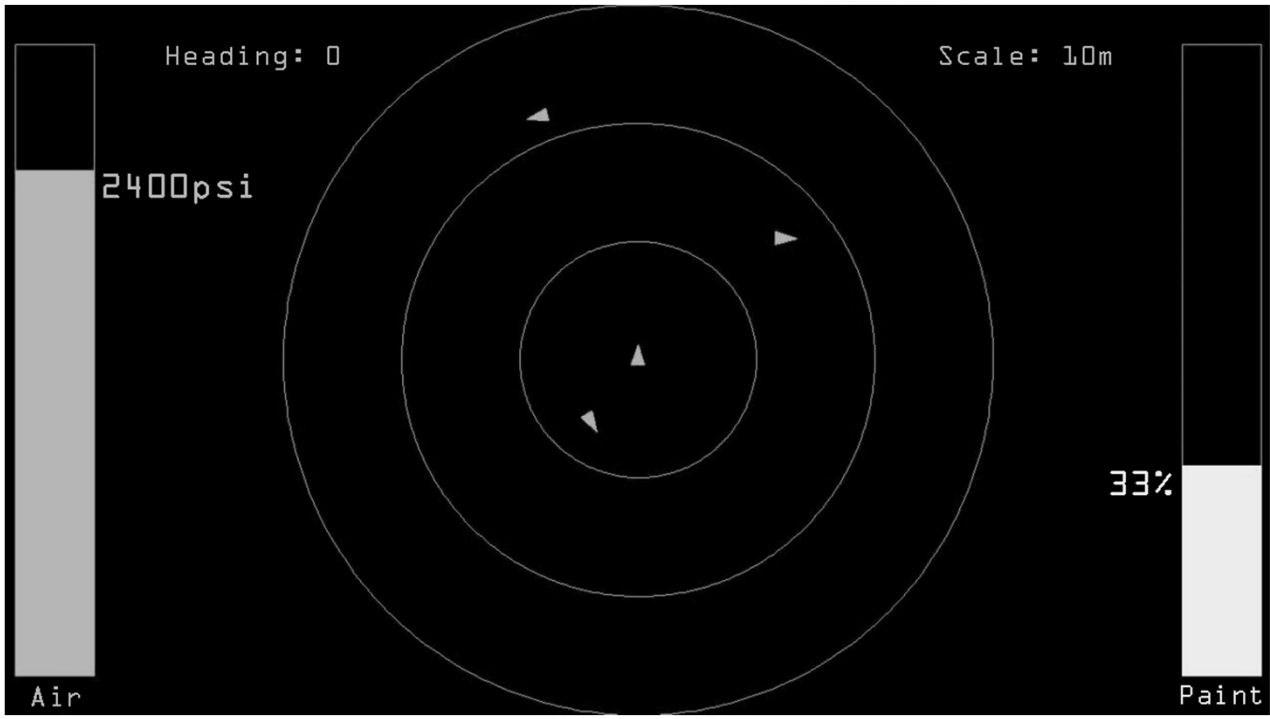
For Example:

n=5, s=1 Total: 23,810 bps

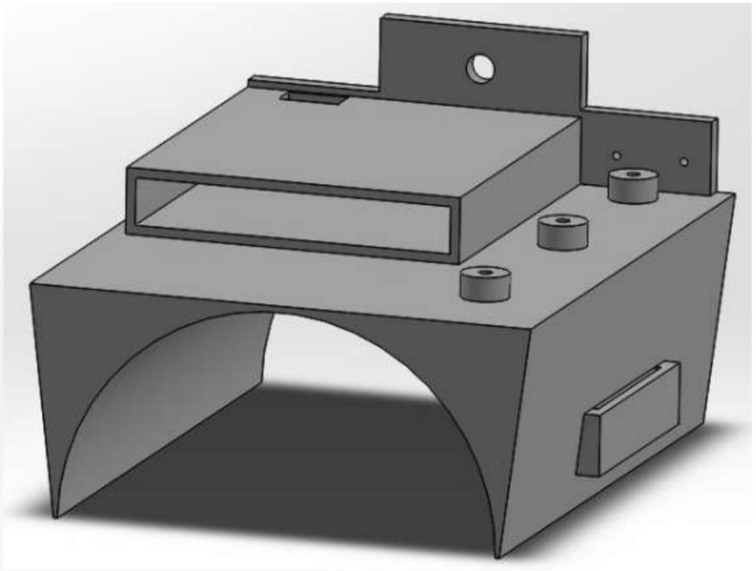
n=10, s=1 Total: 86,770 bps

Winter Progress - Software

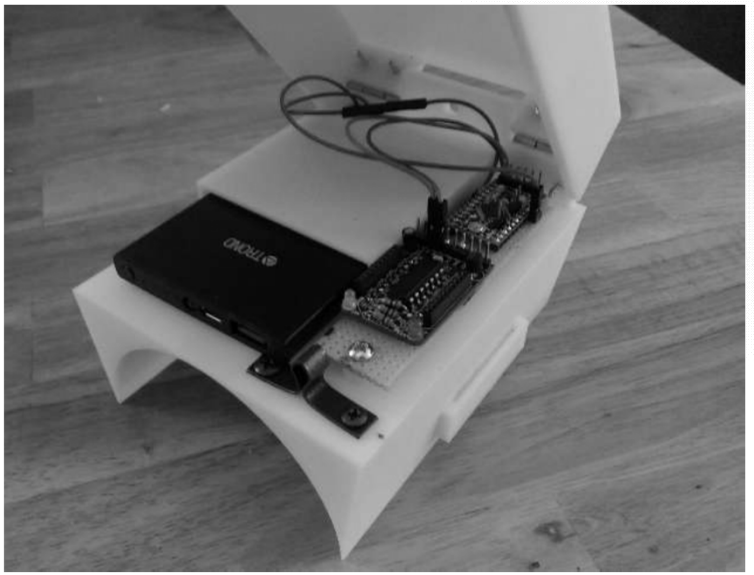
- Break from previous design
  - Less-‘busy’ display
  - Tree cover
  - Implementation Time
- Server Communications
  - WiFi
- Air / Paint level
- Teammate position indicator
  - Lat/Longs and headings
  - Haversine Formula
- User’s heading and scope scale
- GUI work is complete
- In process of executing on HUD
  - Configure App permissions properly
  - HUD API Interfacing



Spring Progress - Hardware



Solidworks 3D Model of Air Tank Attachment

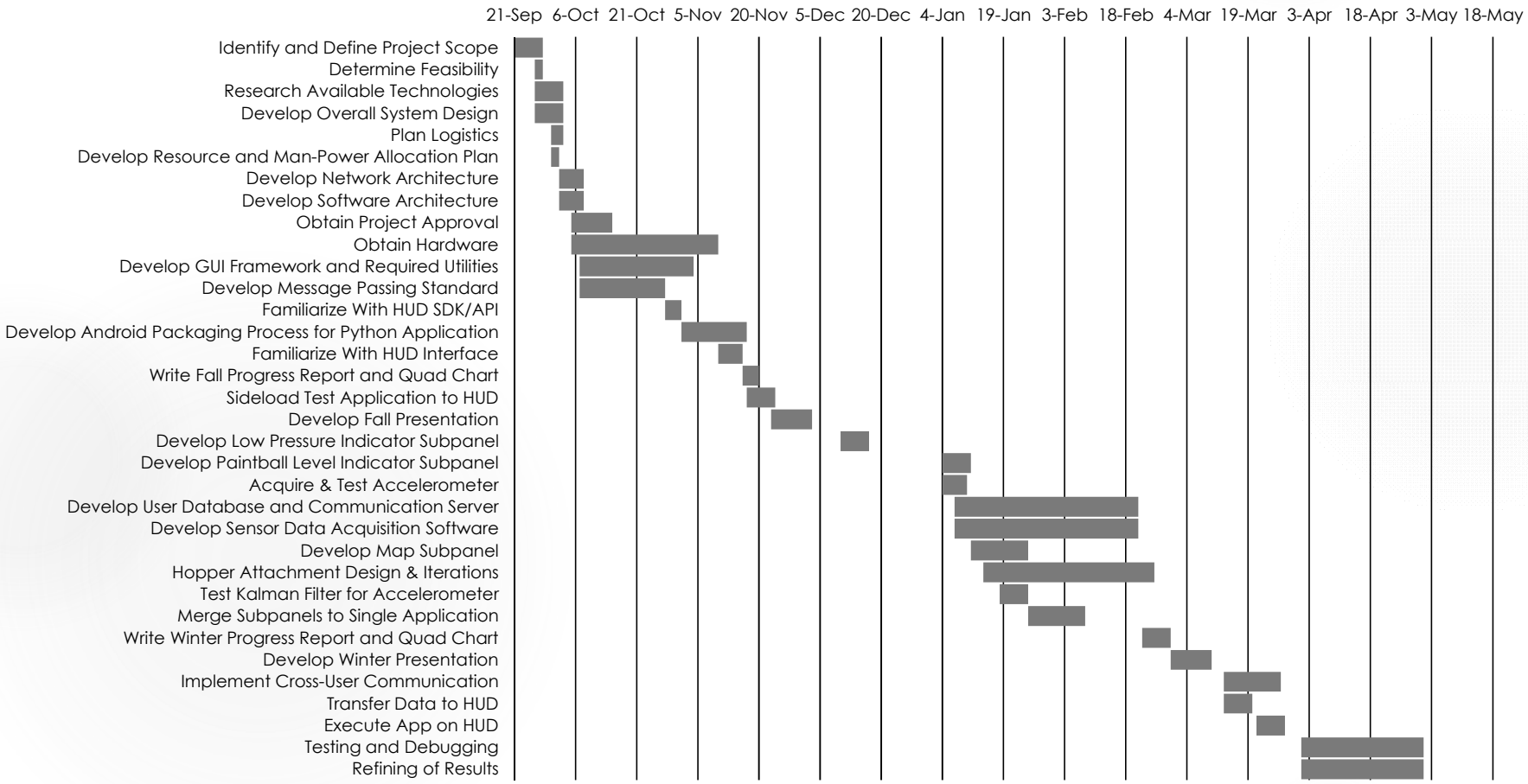


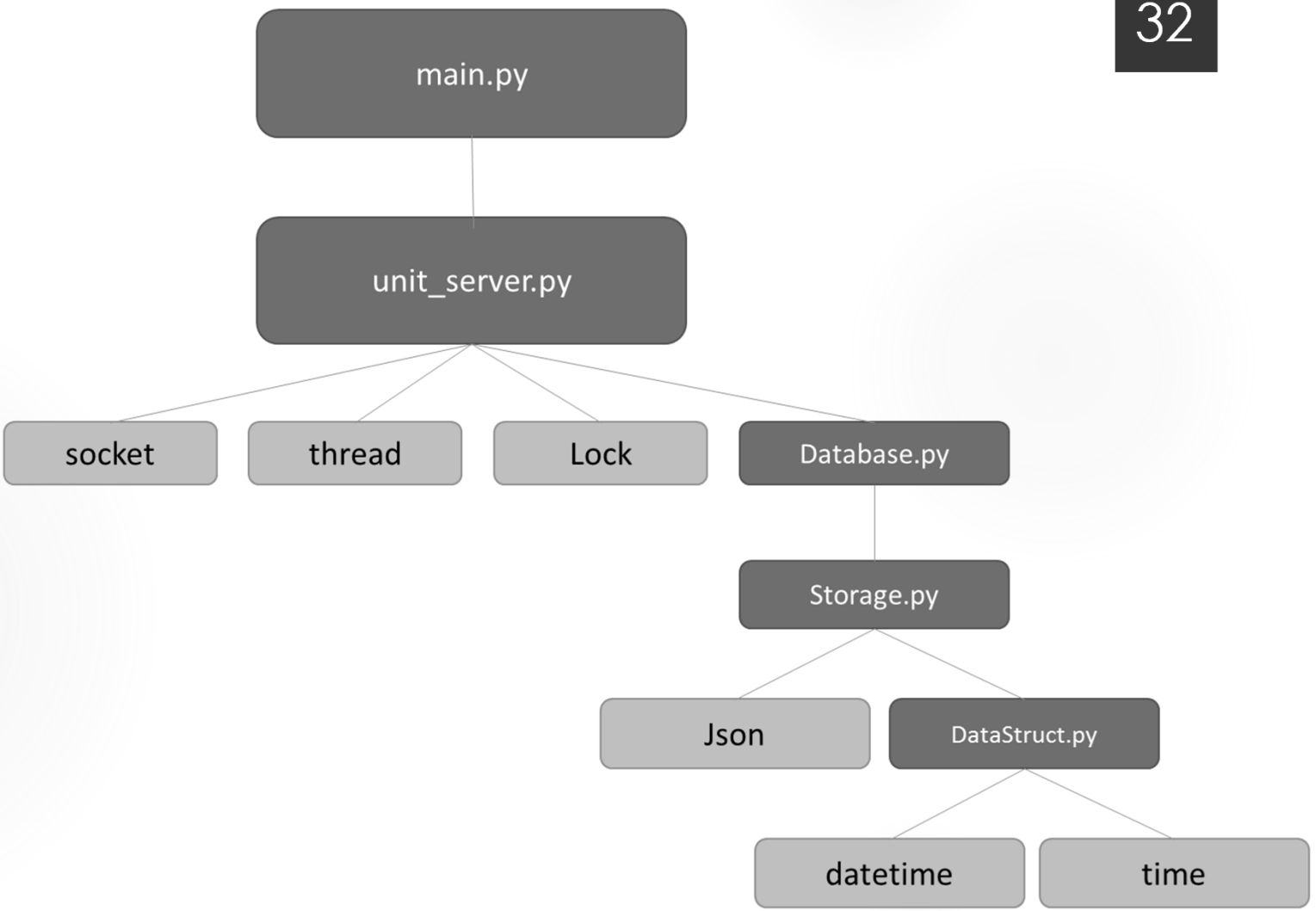
Realized Air Tank Attachment with Circuitry

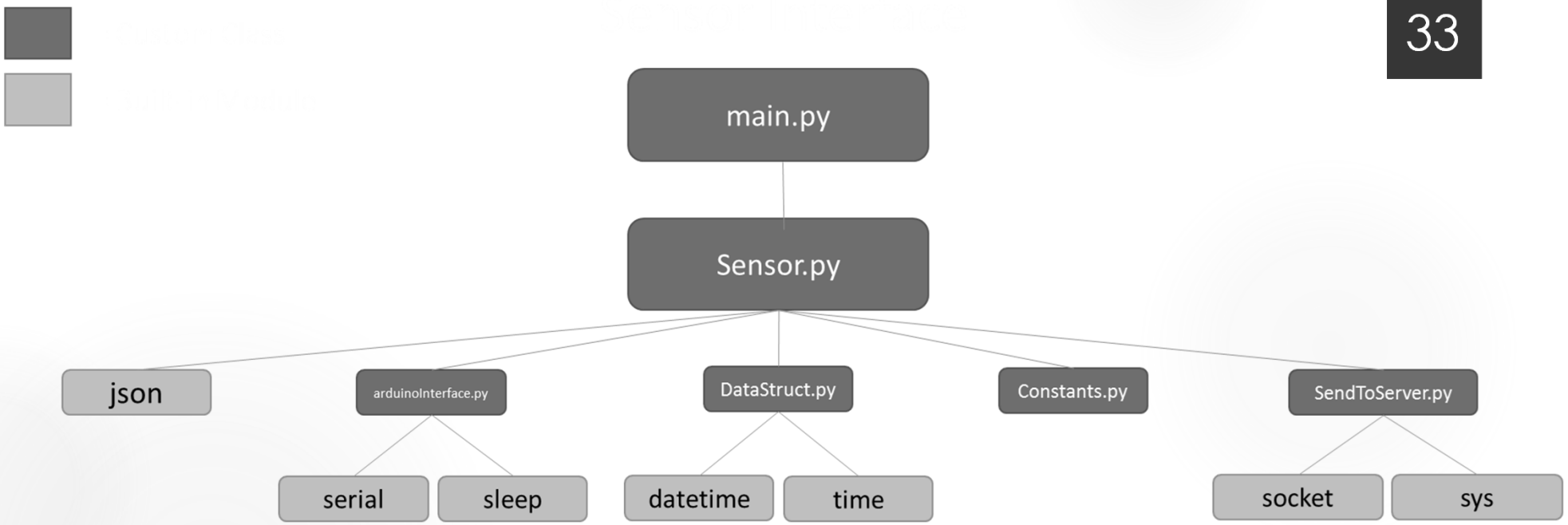
Addressing Comments

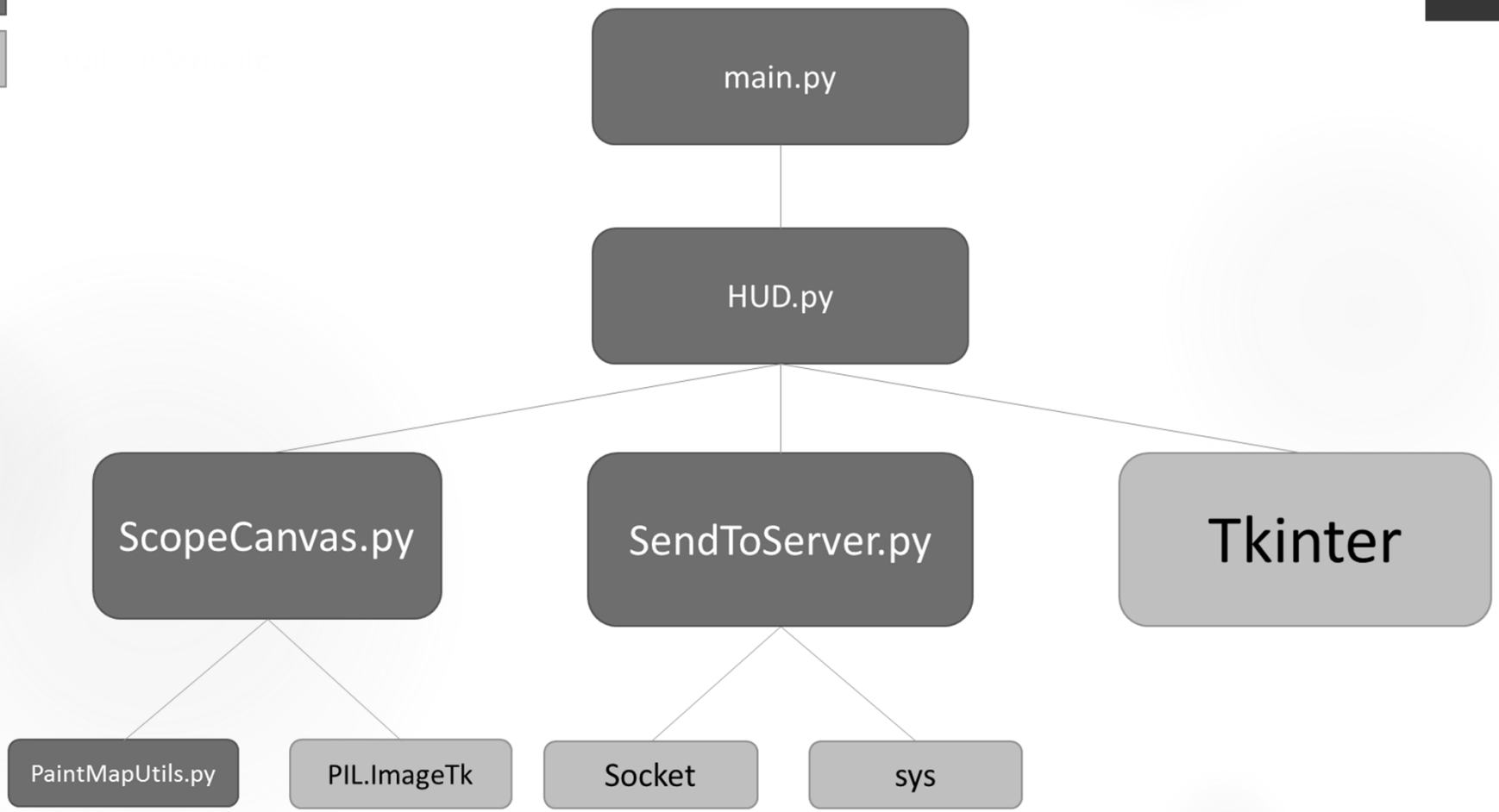
- "Is it possible that the P.E.T.E.R.S. could be utilized for malicious purposes?" (Paraphrased)
  - Yes, but the same could be said for any piece of communication equipment
  - System would be useless to that end in its current state
    - Utilizes unencrypted transmission
    - Communication effected via a jammable, consumer WiFi network
- "Design content of E.E. vs Design of Software should be explicitly stated"
  - Hardware design and circuit construction was simplified by heavy inclusion of COTS
  - Various electrical design considerations, nonetheless
  - Software to link all elements and display data is all custom designed and implemented

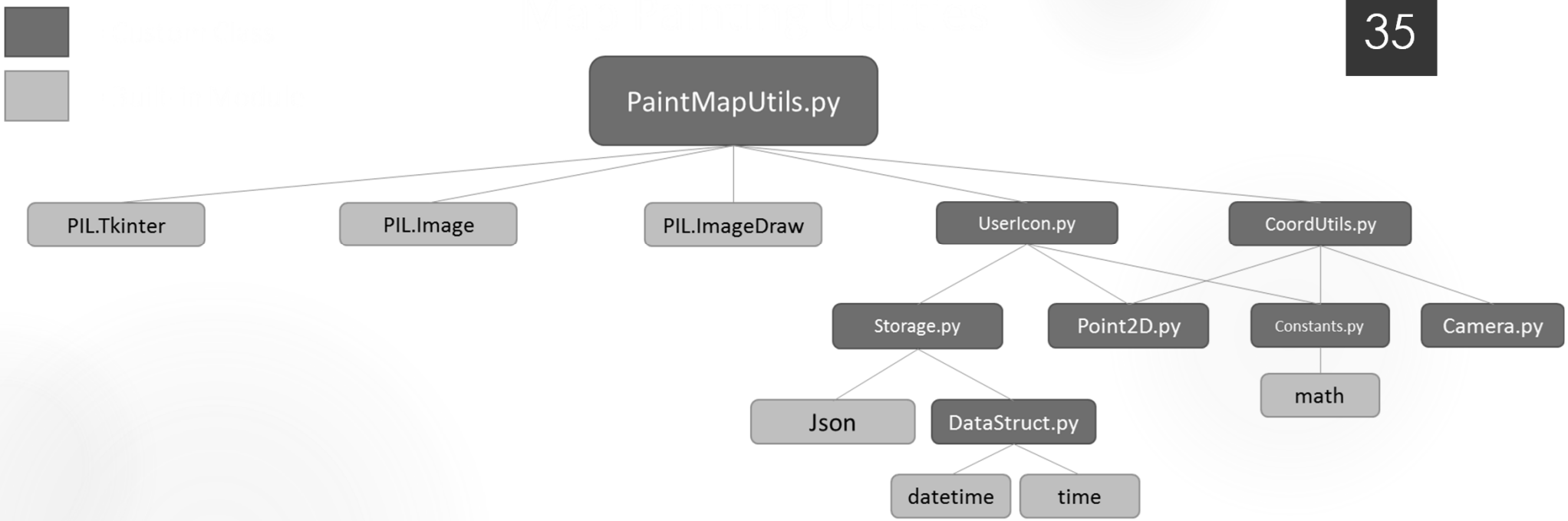
Work Schedule / Timeline




main.py
unit_server.py
socket
thread
Lock
Database.py
Storage.py
Json
DataStruct.py
datetime
time


Custom Class
Built-in Module
main.py
Sensor.py
json
arduinoInterface.py
DataStruct.py
Constants.py
SendToServer.py
serial
sleep
datetime
time
socket
sys


main.py
HUD.py
ScopeCanvas.py
SendToServer.py
Tkinter
PaintMapUtils.py
PIL.ImageTk
Socket
sys


Custom Class
Built-in Module
PaintMapUtils.py
PIL.Tkinter
PIL.Image
PIL.ImageDraw
UserIcon.py
CoordUtils.py
Storage.py
Point2D.py
Constants.py
Camera.py
math
Json
DataStruct.py
datetime
time