Spectre Battlezone

Laser Tag Without Lasers

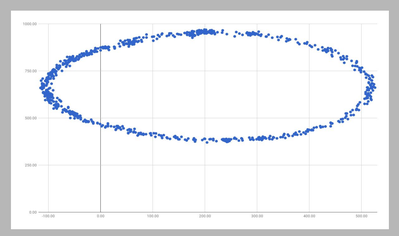
Abstract

Spectre Battlezone is a twist on traditional laser tag without using lasers. Each player is provided a rifle configured for gameplay. The rifle is equipped with a digital display showing a real-time map and current player statistics along with location of teammates, opponents, and in-game upgrades.

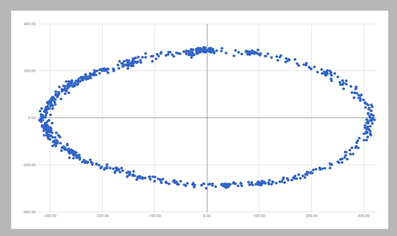
The goal of the project was to design a system comprised of smaller embedded systems. The design was to include embedded linux development tools, modern communication protocols, and to make laser-tag more like an Arena Shooter!

Magnetometer Offset

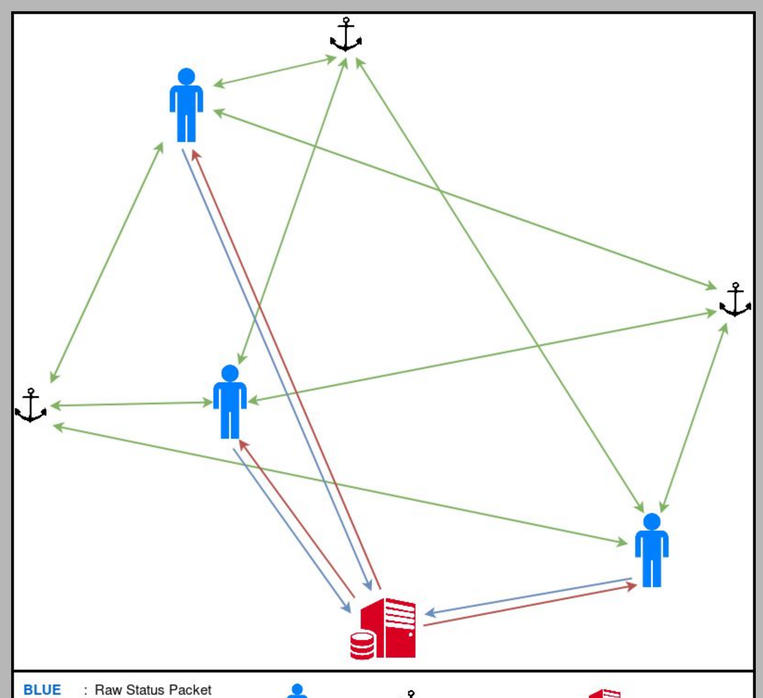
The raw magnetometer data was sampled and forced to be centered at the origin. This provided a global heading reference for gameplay.



Raw Data



Forced To X-Y Axis



The rifles were configured to send raw status at the end of one full location refresh over a WiFi connection to the gameserver. The gameserver would then perform a global update and push game data back to the rifles through a javascript browser.

Methods & Materials

Protocols Used

SPI, I2C, MQTT, TCP, Socket.IO, UWB (decawave custom logic), Linux System Calls (ioctl, fcntl)

Programming Languages

C, Javascript, CSS, HTML

Gameserver

PC with Arch Linux

Laser Rifles (Tags)

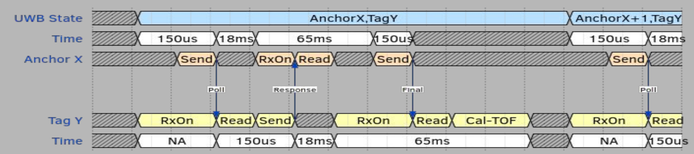
Raspberry Pi 3
Decawave DWM1000 (UWB)
LSM9DS1 (IMU)
Battery Pack
3.5” HDMI Display

Locationing Anchors

Raspberry Pi Zero
Decawave DWM1000 (UWB)

UWB 2-Way Ranging

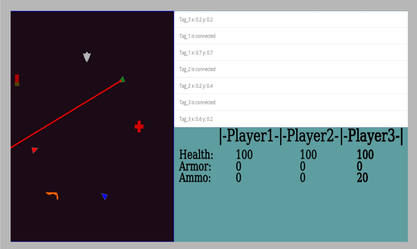
Ranging protocols consist of handshakes between modules, with 2-way implying that both modules have the chance to initiate a handshake. The distance is extracted by timestamping the Rx and Tx actions, performing a Time of Flight algorithm and correcting clock skew between modules.



All timings listed are worst case execution time.

Gameplay GUI

User interfaces showing the location and status of all players involved in a current game.

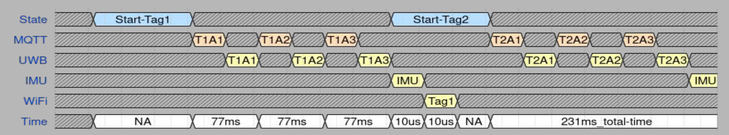


Server GUI

Player GUI

System Timing Diagram

UWB dictates that only one connection can be had at any given time. As such, system timing was required to be very precise and finely tuned. Each module in the system was synchronized using MQTT signals.



On average, the raw status of one player could be polled in 231ms. This brings a maximum system refresh time to 1.44 Hz with a prior goal of 4Hz.

Results

The end system was synchronous and produced input usable for a scaled down version of laser-tag. Although the system would not be adequate for large scale applications, it successfully demonstrates the enhanced features described.

Several lessons learned include:

- Test driven development
- Optimization through refactoring
  - Optimal performance requires out of the box thinking
  - Early system-level debugging
- Timing analysis and system synchronization

Evan Boren: e.boren@aggiemail.usu.edu
Devin Gillespie: devingillespie612@gmail.com
Taylor Spencer: tayspen72@gmail.com

Utah State University
ELECTRICAL AND COMPUTER ENGINEERING

Special Thanks To: Dr. Don Cripps USU IEEE