BlueDentist

LGS/CACI Capstone Fall 2019 Presentation

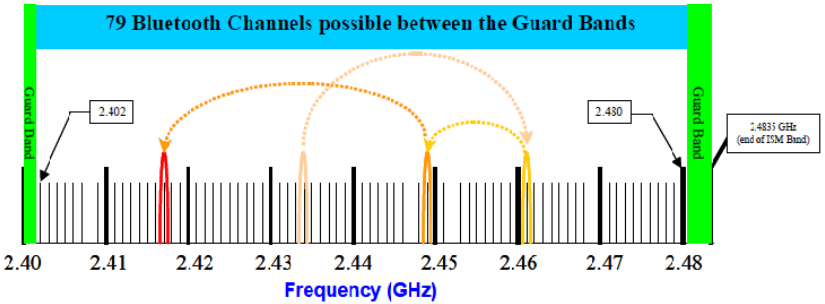
By Zachary Battles, Chris Chan, Griffin Danninger, and Jeff Longo

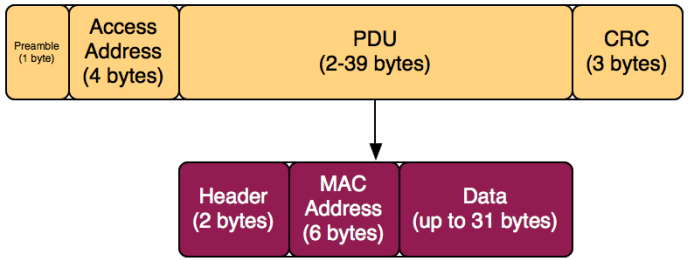
What is BlueDentist?

Bluetooth monitoring using wide band software-defined radios and GPU processing

A Bit About Bluetooth

- Limited to ~10 m
- Spread spectrum frequency hopping on 79 channels
- Device announces presence when in discoverable mode
- 48 bit unique addresses





Problems / Solutions

Bluetooth not advertised	→	BlueDentist identifies and records all activity
SDRs are labor intensive to reprogram	→	GPU allows for protocol flexibility
SDRs output large amounts of data	→	GPU parallelizes computation

Team

Jeff Longo - Project Lead, Hardware design: board implementation/layout

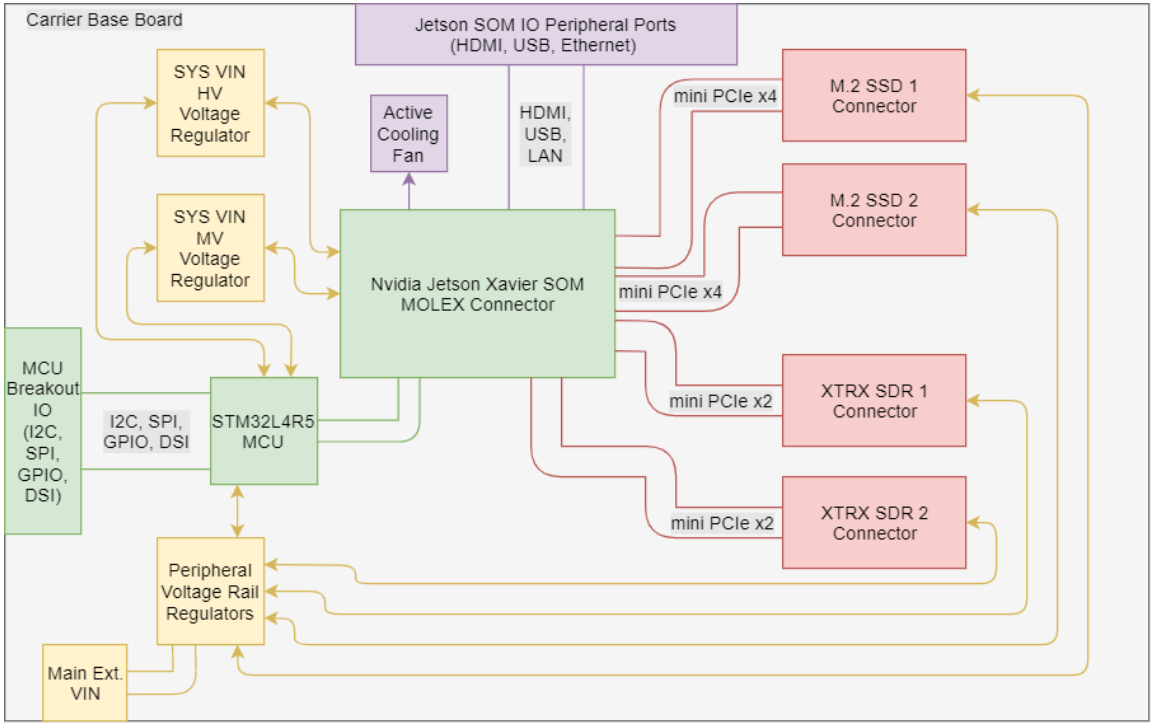
Chris Chan - Hardware design: part selection, schematic

Zachary Battles - System integration and software design: Radio/SSD integration

Griffin Danninger - Software design: Bluetooth parsing, CUDA processing

Design Overview

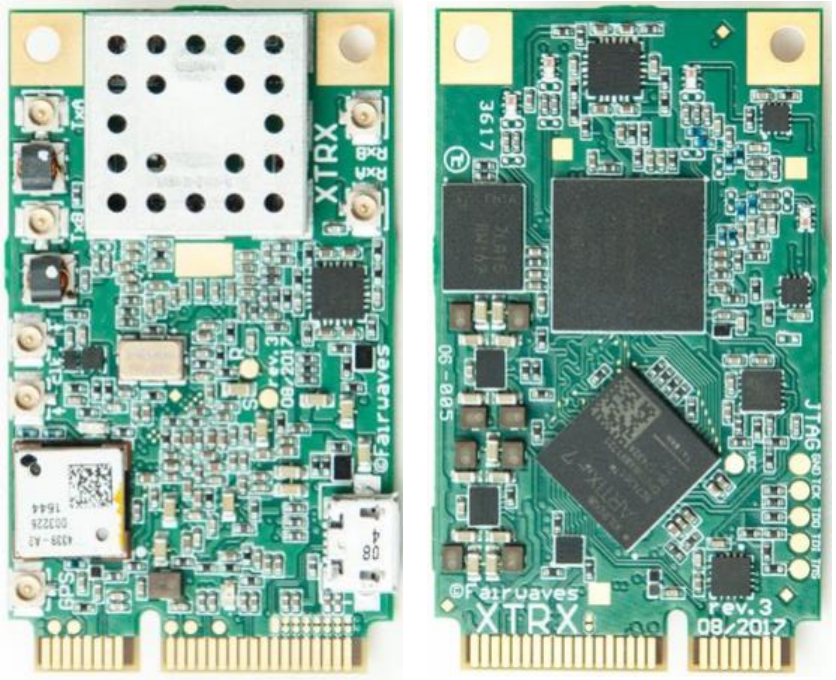
Hardware Block Diagram



Key Hardware Components

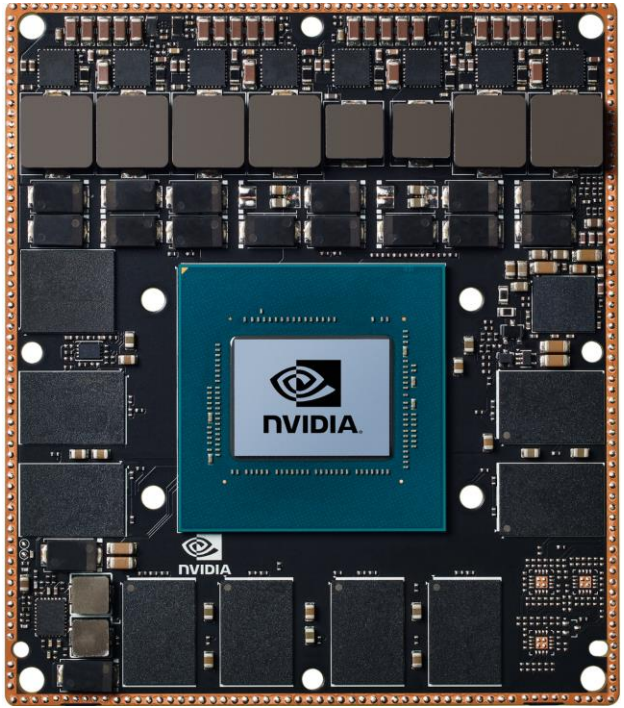
Bluetooth Radios: XTRX

- Mini PCIe 2.0
  - Max Throughput: 7000 Mbps,
    292 Msps (Limiting factor)
- 12 bit DAC/ADC Resolution
- Bandwidth: 30 MHz to 3.7 GHz



System GPU: Nvidia Jetson Xavier

- 512-Core Volta GPU with 64 Tensor cores
  - 11 TFLOPS (FP16)
  - 22 TOPS (INT8)
- 8-core Carmel ARM v8.2 64-Bit CPU
- 16GB 256-bit LPDDR4x
- Supports HDMI, USB, PCIe, and Gigabit Ethernet



Supervisor MCU: STM32L4R5ZIT6

- 120 MHz, 2 MB flash, 640 kB RAM
- 4x I2C, 3x SPI, 6x USART, USB OTG
- Readily available Nucleo development board targeting chosen MCU

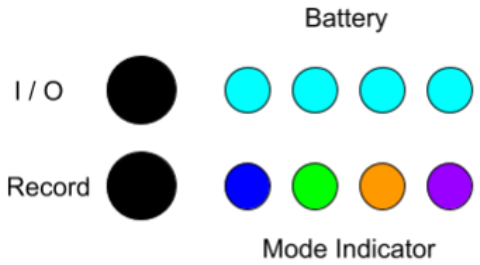
Key Software Processes

Software Processes

- Data ingestion and processing
- Data storage
- Post-processing and analysis
- Off-device monitoring program (interface over IP)

User Interaction

- Power button
- Record Start/Stop button
- Battery status indicator
- Operating mode indicator
- Interfaceable over IP for:
  - Live monitoring
  - Data analysis and downloading



Progress So Far and Roadmap

Hardware

Fall 2019

Transition to Altium, Finish Iteration 1 Schematic

- STM32 power sequencing,
- HDMI/USB/Ethernet
- 1 M.2 slot, 1 PCIe lane

Winter 2020

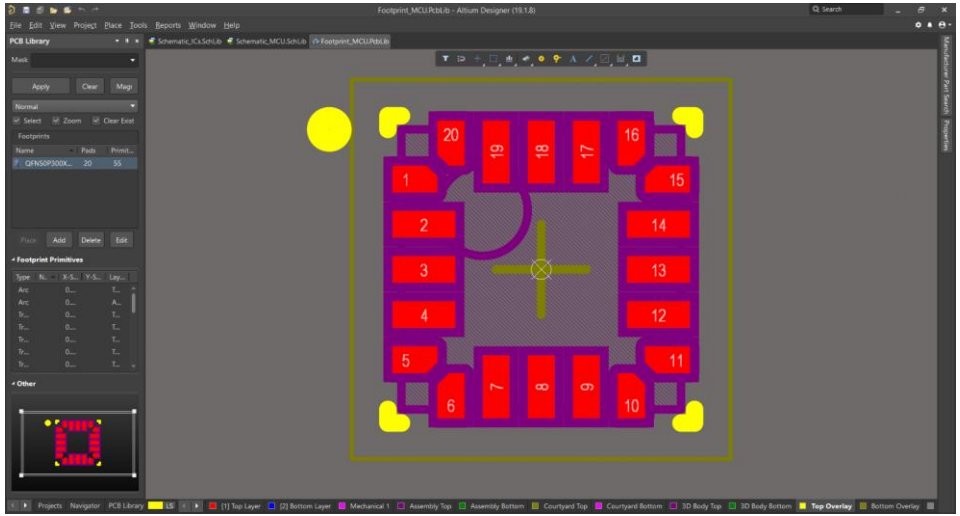
Layout Iteration 1 PCB, Test

- High speed, 6 layer design
- Test Jetson/peripheral functionality

Spring 2020

Layout Iteration 2 PCB, Test

- Route additional M.2/PCIe lanes
- Address any concerns from iteration 1



Software

Fall 2019

Set up Jetson Xavier for CUDA development

- Learn basics of CUDA
- Set up Jetson dev kit
- Interface with XTRX SDR

Winter 2020

Algorithm Development

- Develop data and processing pipeline
- Run proof of concept on dev kit

Spring 2020

Hardware Integration / Optimization

- Optimize for embedded hardware
- Expand for use with 2 SDRs and SSDs

```
// For the CUDA runtime routines (prefixed with "cuda_")
#include <cuda_runtime.h>

#include <helper_cuda.h>
/**
 * CUDA Kernel Device code
 *
 * Computes the vector addition of A and B into C. The 3 vectors have the same
 * number of elements numElements.
 */
__global__ void
vectorAdd(const float *A, const float *B, float *C, int numElements)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;

    if (i < numElements)
    {
        C[i] = A[i] + B[i];
    }
}

/**
 * Host main routine
 */
int
main(void)
{
    // Error code to check return values for CUDA calls
    cudaError_t err = cudaSuccess;

    // Print the vector length to be used, and compute its size
    int numElements = 50000;
    size_t size = numElements * sizeof(float);
    printf("[Vector addition of %d elements]\n", numElements);

    // Allocate the host input vector A
    float *h_A = (float *)malloc(size);

    // Allocate the host input vector B
    float *h_B = (float *)malloc(size);

```

Questions?