Overview of project goals

Your project is a **resume item**. Its goal is to help maximize your candidacy for potential future employment.

Students are allowed to work either by themselves or in teams, up to teams of three. For some students, it may actually be optimal to work in teams of two (rather than solo) to potentially allow for a more advanced project, and therefore a more impressive final result that students can show off to future employers.

In addition, as one student asked during class, it can be useful (if not graduating this semester), to pick a project that could be worked on further in the future (either over the summer and / or next semester). I have had students do this in the past, and some of the best final projects were the result of multiple semester projects (for example sometimes the first iteration ends up being essentially an initial proof of concept, and then the second is a more proper implementation utilizing lessons learned).

Coding Projects

Students are allowed to use whichever programming language(s) they are comfortable, including but not limited to C/C++, java, python, HTML / javascript, ruby, perl (redflag…), R, matlab, etc. Generally, I would recommend C++ (and possibly assembly) for performance related, Depending on the particular project and its target goals, however, certain languages may be much more appropriate (especially if students' are attempting to maximize the outcome and results of the project, and thus maximizing its utility as a resume reference item for seeking employment post-graduation).

Project Requirements

In order to provide you with a more realistic coding environment (of the type you will likely encounter post-graduation) and to allow me to best monitor the progress of your projects and help you along the way, the following will be required:

- Git (version control)
- Task management system to plan, track task progress, etc.
- Automated test scripts (unit testing, etc)
- Well documented code
- Final report and final presentation given by the group

Due Date

Grades are due May 17th, so the project will be due before then. That said, once it is finalized just how many students are working individually vs in groups (and therefore how many presentations and papers need to be read) the final due date will be provided. In addition, I will offer the ability for students to provide work after the due date, at the penalty of a slightly reduced grade (for those who really the need time). The grade penalty is to motivate that most students submit their projects earlier (just as in electronic trading, there is a cost for being slower…)

Separately, since none of you are graduating this semester, it is also likely possible for grades to be updated later. Assuming the University would allow it, I have no problems with students to continue working on their project and resubmitting them for an improved grade later as well.

Notice

Remember: the list of projects below are **only examples**. You are not limited to selecting from the below list, and even if you do, we can discuss any possible modifications or variants of the proposed projects. I actively encourage you to consider coming up with your own project ideas as well, focusing on the areas that are of most interest to you both personally and in regards to future employment, industries of interest or demand, etc.

Example Types of Public Market Data Sources

The ability to perform specific projects relies on an ability to gain access to the required data. Below are examples (but certainly not the only) publicly available data related to finance, electronic trading, etc.

Market Data Sources

NASDAQ ITCH: ftp://emi.nasdaq.com/ITCH/Nasdaq_ITCH/ only seems to now have 12/28/2018, 01/30/2019, and 3/27/2019. This is unfortunate as it used to have much more data.

IEX (both TOPS and DEEP): https://iextrading.com/trading/market-data/

Deutsche Börse Public Data Set: https://registry.opendata.aws/deutsche-boerse-pds/

Bloomberg Terminal (only some markets, for example SIP feed containing all US exchanges, but not the direct feed, also contains tools for tracking ocean-related vessels and structures such as oil platforms, shipping containers, etc); there are Bloomberg terminals located in the lab at the MSFE office at UIUC, also in the library at UIC.

Many of the various bitcoin APIs (at least for real-time if not historical data)

Regulatory and Firm Data

FCC Universal Licensing System: http://wireless.fcc.gov/uls/index.htm?job=transaction&page=weekly

Millimeter wave database: http://www.micronetcommunications.com/LinkRegistration/Query.aspx

SEC Registered Broker Dealer List: https://www.sec.gov/help/foiadocsbdfoiahtm.html

FAA Aircraft Registration:
https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download/

SEC EDGAR database: https://www.sec.gov/edgar/searchedgar/accessing-edgar-data.htm

IRS 990 filings (non-profits which may disclose hedge fund and other investment holdings):
https://registry.opendata.aws/irs990/

Correlations between NASDAQ and IEX market data

Analyze (for those few days available) depth of book data from both NASDAQ and IEX (many days available). Note this could result in HUNDREDS of potential different research projects, investigating various potential phenomena or patterns. A few examples of the types of phenomena that could be looked at:

MPID-Marked NASDAQ Market Data vs IEX price level-aggregated market data.

NASDAQ has two different scenarios where it publishes order quotes via the ITCH packet format, 1) those orders which are publicly advertised by who placed the order to publicly identify itself (it is literally advertised as being submitted by a specific investment bank, electronic trading firm, etc) and 2) those orders which are submitted by firms requesting that NASDAQ keep anonymous who submitted the order (this is normally the default case in finance). One could do a study analyzing patterns of identifiable firms' batches of orders on NASDAQ with possibly correlated material on the IEX market data (remember, on IEX's DEEP data feed, individual orders are **not** identified by individual firm ever, and even more so, individual orders (even anonymized) are not actually directly displayed, only the aggregated price levels; that said, depending on how IEX chose to do or not do message aggregation, it is possible to sometimes detect individual orders being placed based on monitoring certain variables).

One example research question in doing the above: it is possible to identify specific market data updates (trader-submitted orders) on IEX that (with some confidence level) may have been submitted by a specific firm, based on 1) that specific firm having submitted other related orders to NASDAQ and 2) that those submitted orders the firm specifically submitted telling NASDAQ to identify its orders. Secondly, if it is possible to detect specific broker-associated orders on another venue, is it then possible to estimate that firm's potential latency between the two (or more) trading venues. As discussed in class, this can be a non-trivial question due to the potentially **massive** effect of queueing of packets in the network and software applications and the resultant distortions in timestamping.

Study of differences in depth of book between NASDAQ vs IEX

Examine for various types of stocks the relative depth of book on NASDAQ vs IEX. Somewhat plainly: does one exchange "on average" seem to offer either / both better advertised prices than another exchange or deeper order books. In addition, are there some types of stocks for which there is a difference but not others (based on factors such as market capitalization of the company, price of the stock, volume, type of company or stock, etc).

Performance Analysis of IEX and / or NASDAQ

This project is essentially researching the question “how fast is an exchange and what is the distribution of its performance”? By analyzing the market data, embedded within it are the rates of order matching, book updates, etc. An interesting project (that students have in fact done in the past quite successfully) was to compare the relative performance of NASDAQ and IEX matching processes. Note also with IEX, given the data is actual network packet captures, there are additional timestamps (when a message was generated in software before actually being transmitted vs when that message payload and enclosing packet was actually received over the network) that can be used for additional performance analysis (ticker plant performance and processing characteristics).

Estimation of SIP order book based strictly on IEX publicly available feed

This project would involve downloading SIP-based aggregated market data from Bloomberg (remember the “consolidated” feeds offered by at least NASDAQ and NYSE are the special aggregated feeds which combine direct feeds from all the exchanges that a stock trades on).

The context of this research project is that many trading firms are growing increasingly concerned about the continued cost and cost increases of market data feeds from various exchanges (this has become especially contentious in cash US equities space). While an ultra-high frequency firm would never attempt to rely strictly on IEX market data, it is an interesting question with regards to many, especially human, traders as to whether the free real-time offered IEX data feed is “good enough” for some purposes and instruments. If so, this could have continued impacts on the ecosystem, as fewer traders may continue paying for the other fees direct feeds (for example to their brokers), and begin to rely more so on IEX’s free public feed, which would reduce the number of subscribers for exchanges’ paid market data, possibly pressuring exchanges to raise fees even more (causing those who few firms left who have to pay even more outrage).

Analysis of FCC and millimeter wave data

There are an enormous number of potential projects related to analyzing various FCC and other radio link registration data. Subsets of these projects may be particularly attractive as potential resume items to show possible mployers because many possible projects could involve very attractive visuals (graphs of network paths, 3D renderings of buildings, radio towers, microwave signal propagation, etc).

Some of the many possible options (though you and your potential teammates could come up with your own) are listed in various sub projects below.

Generate an optimized SQL schema for the database

The FCC database and format is currently **highly** redundant and at times the schema is somewhat confusing (especially with regards to which column(s) are to be used for joining data from different tables).

A potential project would be to create a set of programs / scripts and SQL schema that is more efficient and can copy the existing FCC database into that new schema. For example, if in the existing database there are, for each row, the exact same large set of fields (name, address, city, state, zipcode, telephone number, etc) which are repeated for every row, it is possible to instead generate a new table (for example "entity"), and simply have all rows with the repeated information have a new column ("entity_id"). This could dramatically reduce the size of the database, and thus make numerous analysis and advanced information synthesis more efficient. Slightly more advanced, as discussed in class, would be to add various common heuristics to automatically identify when two slightly different entered entities are actually the same entity (for example, a person would recognize, and a computer program could be to coded to determine that "Acme Corp, 999 N Main St, Chicago, IL" is in fact the same as "ACME Corporation, 999 North Main Street, CHICAGO, IL", using relatively simple rules around common abbreviations, case-insensitivity, etc).

Usage of Geospatial frameworks and the associated potential research queries

Another potential area (with applicability far beyond just electronic trading or finance) is the addition and use of "geospatial" extensions that are commonly offered on many databases and geographic APIs. Several types of FCC (and other government) data contain GPS coordinates, addresses, etc. Geospatial frameworks could potentially allow for querying that would otherwise be time consuming to manually implement from scratch.

For example, suppose you know the GPS coordinates of all four corners of a particular rectangular data center (actually often auto-mineable from opensource map databases). Geospatial extensions may allow queries such as "select all microwave transmitters that are within X meters of any point on this building". An even more advanced query, potentially yielding additional interesting information, "select all entities / firms that have a certain percentage of their total links that are all within some distance of the major financial data centers or the straight path arcs between those data centers".

Note that historically, in public discussions and papers on electronic trading and microwave, data centers are treated as point-like objects. Obviously, they are not, and as the marginal importance of ever tinier latency differences continues to increase as markets become more deterministic, the importance of more properly modeling the exact geometry and physics of network paths increases as well. Building upon the prior example, one would actually want to factor in not just the four corners of the building in 2D, but the entire 3D geometry of the building. This type of analysis can be essential when analyzing questions such as path-planning for potential future radio links (is there a building, mountain, etc, blocking the 3D path between two potential radio links?).

Automated techniques for tagging potential electronic trading radio assets

There are many potential routes by which one could accomplish this (one example was somewhat inferred in prior project proposal, mentioning a query of using physical locations of data centers and radio registrations).

A potential project could investigate the usage of various ML / AI / statistical techniques for identifying potential electronic-trading related radio assets (microwave and millimeter wave links, towers, buildings, etc).

A simple example of a first start to this problem: download the list of SEC registered broker-dealers and look for firms that have mailing addresses (including suite number, not just street address) that are associated with both broker-dealers and with radio links. Given the lengths that firms will go to (essentially reduce length), it would not be inconceivable to some day find, for example, an electronic trading firm registering AM / FM / TV stations. Given what we discussed last lecture with regards to encoding HD audio onto the existing FM specification, can you think of how this could conceivably be used for transmitting **other** data? 😉

Another more straightforward project would be augmenting the database and information to identify not just individual radio **links** (a single vertex in the graph) but grouping the various links into their entire **paths** (essentially all the hops from one data center to another) and groups of paths (it would be understandable if, in addition to building the straightest path, firms also built slightly "out of the way" redundant links running along that straightest path, for redundancy in case the absolute straightest path were blocked).

US Federal Government Data Fusion Project

Note this section could result in countless different combinations of possible projects. Build a data warehouse joining data from multiple publicly available American (and / /or other countries') government-provided public data sources, including possibly some interesting subset of for example:

- SEC: Securities Exchange Commission is the United States regulators for various securities related to investment (for example stocks, options on stocks, etc).
- CFTC: Commodities Futures Trading Commission is the United States regulator for futures contracts and the commodities markets upon which they are based
- FCC: Federal Communications Commission is responsible for regulating the electromagnetic spectrum of the United States
- H1B: Department of Labor publishes a list of every single H1B visa application filed, including employer, salary range, position title, etc. (very useful)
- FAA: Federal Aviation Administration regulates air transportation (including maintaining a database of all structures 200 feet or higher, and maintaining databases of pilot and plane registrations).
- https://www.usa.gov/statistics
- https://catalog.data.gov/dataset?groups=finance3432#topic=finance_navigation

Depending on which data sources are fused, there are many interesting questions and queries one could then perform. Some examples are below.

Develop method of identifying likely electronic trading firms and / or identify other interesting trends, correlations, etc. Simple example usage: "Generate a list of all firms which have matching addresses at both SEC registered-broker dealers and FCC microwave related registrations". "Generate a list of all radio manufactures used by HFT firms, summed by number of instances of each manufacturer". "Generate a list of all Chicago and NYC-located firms employing both radio engineers and either quants or traders", etc.

To understand context, see https://sniperinmahwah.wordpress.com/2018/07/16/shortwave-trading-part-iv-sleuthing-examples-research-tools-techniques-deputies-wanted/

Real-time Weather Impacts on Electronic Markets

There have been various academic papers in the past analyzing the potential weather impact on electronic trading wireless radio microwave links. Weather can disrupt microwave links, and therefore weather can alter the dynamics of electronic trading markets. See https://www.eurofidai.org/sites/default/files/pdf/parismeeting/2016/Shkilko_2016.pdf .

One interesting potential data fusion project would be to analyze more granular weather data and attempt to show at a much tighter granularity the instantaneous impact on financial markets. For example, instead of simply relying on the detection of rain over a 15 minute interval, it may be possible for certain weather formations (a very sharply defined rain system, for example) to detect the exact moment when that weather system reaches a particular microwave link and the subsequent impact on other markets.

This could also result in an enormously impressive visual project if one were to use various 3D weather rendering platforms combined with a tick by tick or second by second snapshot of an impacted market.

As mentioned to one student, a simple potential method of detecting possible changes resulting from the weather would be sudden changes in the number of orders and / or amount of liquidity posted to the order book right as a storm front suddenly passed over a particular link.

Software Developer / Quant Employer Database Mining

The intent of this project is to utilize various public databases already mentioned to generate a list of potential employers in the financial space and their contact info that may be interested in those who took this course. A simple example could be to utilize the H1B disclosures, search out those firms which in the past have hired or attempted to hire for job titles relevant to the industry, run web queries searching for their website / careers page, etc. Additional information could be analyzing features such as salary distributions across specific employers, geographic regions, adjusting for cost of living, etc.

Microwave radio path latency estimation

An interesting viable project would be to build a framework to estimate the expected latency of various electronic trading firm's microwave path lengths given: GPS coordinates of each hop in the link, height of the tower, assumption of latency of repeater (in all but the terminal nodes, in most cases a radio tower has been a receiving and transmitting radio in order to relay or repeat the signal on to the next hop), real-time temperature between the links (alters index of refraction, etc). Note this is also an example that could be fed data from the prior mentioned project of using various public Federal government sources (in this case the FCC and the other public millimeter wave databases).

Realtime physics-based network packet simulator

The intent of this project is to more accurately calculate and ideally visualize the real-time instantaneous state of packets as they physically traverse a network. As we've discussed at length in class, packets are not instantaneous transmissions, but have tangible length in physical space and take known amounts of time to actually be transmitted over a specific medium.

A possible simulation engine would involve using cable lengths, ethernet switch latency times, packet lengths, to then model and visualize packets as they actually move through a network every one hundred picoseconds (both cables and switches). Note that the math on this is actually fairly simple (I'd provide you a few fixed numbers for those that are non-obvious). The interesting aspect revolves building the simulator and performing the iterative step by step updates of the system.

One non-trivial example is that when multiple packets arrive at a switch at the same time, destined to leave on the same output port, one has to model that the switch needs to queue the second packet, and will not begin transmitting it until the first packet has finished transmitting (its transmission time proportional to packet length). Another example is that switches can be configured either in "cut-through" or "store-or-forward", and depending on which, the switch will either start sending out the packet before the whole packet has arrived (cut-through), or alternatively it will hold off transmitting until it has fully received the entire packet (store-and-forward). If cut-through, the latency is often fixed. If store-and-forward, however, the latency of the switch relaying the packet is actually variable (again proportional to the length of the packet). Factoring in these types of details can provide much more detailed analysis and modeling of expected network performance (and therefore trading performance).

As a side note, if any students are interested in future employment in the scientific modeling fields (particle physics, etc), this could be a useful project for gaining experience to modeling of complex interconnected systems.

Bitcoin market data recording project

This project involves creating a simple market data recording proof of concept, and targeting various bitcoin exchanges since they often provide free easily accessible APIs.

Establish connections to multiple bitcoin exchanges (ideally collocated in the same AWS data center), record their market data, possibly from multiple machines. Depending on the exact nature and end goal of the project, this could be done of your personal machine or in the cloud.

If done sufficiently well, the recording portion alone could constitute a sufficient project. That said, having the data could also allow for some very interesting research projects and analysis built on top of the data, such as analyzing timestamp disparities, correlations between different bitcoin exchanges or various cryptocurrencies, etc.

As one simple example, most bitcoin exchanges do not (and cannot-being cloud based) use multicast to send market data, but rather are forced to rely on TCP. The implication of this is that the exchanges must individually send data to each person listening, and not everyone will receive the data at the same time. One interesting (and highly important for traders) question: how much of an advantage or disadvantage could one TCP connection receiving bitcoin data from a particular exchange be relative to another customer listening on a different TCP connection. Having a sense of this number can be very important when attempting back testing for example, or any academic or regulatory analysis of the bitcoin market.

(Paper) Trading algorithms

There are countless possible projects that could be done for developing simple algorithms to attempt to paper trade virtually any financial asset that is of interest to you or your group. Note by "paper trading", it is meant not **real** trading, but in a simulated environment in which fake money is used to attempt to simulate the behavior of how actual markets function.

However, this, would rely on the student having access to a backtesting framework and / or real-time sandbox / paper trading account. This may be possible using interactive brokers. See https://www.interactivebrokers.com/en/index.php?f=5041 and https://www.quantstart.com/articles/Using-Python-IBPy-and-the-Interactive-Brokers-API-to-Automate-Trades . There are also other platforms and frameworks for doing this, and students are encouraged, if interested, to research them and select another if of interest.

Implement a "mini exchange"

This project would involve implementing (probably using your own very simple basic protocols for order entry and market data) a mini exchange. It would consist primarily of three applications: 1) a simple gateway (which "customers" connect to directly, and then send orders to and receive updates from), 2) a simple matching engine (which receives the orders, determines if the order matches with any resting orders on the book, and then adds that order to the order book), and 3) a simple market data ticker plant (which takes the output messages sent by the matching engine, and generates an anonymized market data book update message).

In addition to the three apps, you would also write basic test app harnesses (at a minimum something that makes multiple connections to the gateway, sends in multiple orders, and then validates that orders that should match do, orders that should rest in the order book do, and then that appropriate fill messages are received via the gateways, and that the correct market data updates are received from the ticker plant).

Note there is a large range of "scalability" and "complexity" that can be added or removed to this particular project. This is very useful both for ensuring the ability to deliver something relatively simple but also iteratively adding features and complexity. As an example, initially one could target only supporting simple limit orders (no market orders, hidden, icebergs, FAK, or any other order modifiers). Your chosen custom "order entry" protocol could be an extremely simple UDP based format with only a few fields. Your chosen market data book update protocol could choose to only output "top of book" (best bid and ask), etc.

In addition, this could be implemented to all run on a single machine. However, it would be more impressive and useful for your experience if you implemented it to run on multiple machines using a "dev ops" framework for bringing up multiple VMs on your computer. This actually involves very little relatively increased work but provides a much more impressive demo (with multiple VMs for the "gateway" VM, "ticker plant" VM, etc), each with their own IP addresses, etc. I can provide you with the basic framework for this using vagrant based VMs.

Lastly, if you wanted to work on a larger group especially, one person's project could in fact be to capture the traffic from your mini exchange and study the distribution of the performance, latency, processing rates, etc.

Book building Visualizations of Market Depth

As has been covered throughout the course, the ability to visualize and succinctly understand an enormous amount of numeric timeseries data is essential in analyzing modern electronic markets. There are countless common methods of visualization, graphing, etc, that are common throughout the industry. You have seen some in class just as displayed via the Bloomberg terminal for example.

The purpose of this project would be to utilize, for example IEX DEEP depth of book market data, and develop methods of visualizing that data. As a simple example see below.

See http://www.nanex.net/aqck2/4700/20150428.GC.M15.12.gif

This is also an interesting project because it could be expanded upon in the future, and also because it holds the opportunity (for those students that are interested) to experiment with 3D or even VR-related APIs and technology (though certainly not necessary).

Experiment with AWS OpenCL / SDAccel FPGA development for packet parsing

FPGAs (Field Programmable Gate Arrays) have become enormously popular and arguably essential in some aspects of electronic trading. Historically this customizable hardware was programmed in various hardware description languages (VHDL, Verilog, SystemVerilog, etc).

In recent years, there have been numerous attempts at developing tools and specifications to allow for programming such devices using higher level software languages (for example OpenCL, C/C++).

One potential project would be to utilize Amazon Web Services F1 (FPGA), and C++ for generating FPGA (custom hardware) for doing packet parsing (or something else of interest to you). The advantage of using AWS is that you would not have to buy an FPGA, setup a development environment, etc. The downside however is that it does cost money to rent the VMs.

To get a sense of what this looks like, see https://github.com/aws/aws-fpga/tree/master/SDAccel and https://github.com/aws/aws-fpga/tree/master/SDAccel/examples/aws/kernel_3ddr_bandwidth/src

Alternatively, students could use other FPGA boards (and even other frameworks), however, based on surveys of the class of prior experiences, this may be too ambitious to accomplish in one month's time for some students. That said, there are now even frameworks for generating hardware from python (see https://github.com/m-labs/migen) so a sufficiently ambitious student could do so.

Implement a raw TCP engine from scratch

TCP is at the heart of much of the global electronic trading system, especially for order entry. This project would involve implementing a functioning TCP engine (client and / or server) using raw packet API (you build the full contents of the payload rather than relying on the operating system to implement the TCP portion). This is often a "right of passage" in the UHFT developer community.

Note that you wouldn't be expected implement a full implementation of all possible scenarios of TCP, but even having students implement a sufficient portion that is able to successfully connect, send and receive data, and disconnect, can be an extremely valuable learning method for understanding some of the details of TCP.

For those interested in cybersecurity, note this project is also highly relevant as TCP can be a source of risk (TCP man in the middle or injection, etc).

"Wireshark-like" UI

Build a simple packet parsing (just ethernet, IP, UDP, and TCP) and visualization program (could be written in python, HTML5, etc). There are many possible extensions and options on what a potential project could entail for this. For example, are you focusing on the impressiveness of the UI, the speed of the decoding, the maximum file size you can handle, etc.

Wireshark decoder for IEX DEEP protocol

Implement a wireshark dissector for IEX's "DEEP" market data protocol (depth of book rather than just top of book): see https://github.com/iexg/iexdissectors (but that is for "TOPS", not DEEP).

As discussed in class, wireshark is a very common tool for viewing network packet capture files (pcap, etc). A dissector is the method that third party developers can use to add support to wireshark for new protocols (for example, wireshark will already know how to decode ethernet, IP, and UDP, but it will not know how to decode IEX's DEEP format, thus by default, the UDP payload data just appears as a raw binary blob).

Separately, note, if students tackle one of the "build your own" projects and create their own custom protocol for it (for example to have a custom gateway communicating with a simple custom matching engine), one way to improve your project would be to also add a simple dissector for your custom protocol. Specifically for doing DEEP, it is non-trivial because of the need to maintain state. For your custom protocol, however, it could be extremely simple (a few hours) for a simple few-field protocol that only contains (sequence number, stock, number of shares, price) for example.

Deployment and Analysis of Various Software-Defined Radio (SDR) GPS frameworks

SDR refers to capturing raw waveforms (radio transmissions) and then using software to actually decode and interpret the radio transmissions (compared to the historical method of simply building hardware and electrical circuits to do all receiving and decoding). It is a technology that holds enormous promise for a variety of industries.

GPS refers to the technology deployed by the United States Department of Defense offering a system of satellites broadcasting timing and other information which allow devices anywhere on earth to determine both their location and the time (this is the technology that everything from your cell phone to cruise missiles rely upon for location information).

Separately, note that, in part due to the defense / military nature of GPS, several other governments have since developed their own variants of GPS (Europe’s Galileo, Russia’s GLONASS, etc).

GPS can be extremely important to the global financial system because current best practice is often to utilize GPS antennas on the roof of financial data centers, and then feeding those signals to GPS receivers inside the datacenters which are then used to set the clocks and timing for all of the computers inside the data center. GPS powers the timing of the global financial system, and as we’ve covered in this course extensively, accurate timestamping is **very** important.

As discussed, it is now possible to utilize extremely low-cost generic USB SDR kits for receiving raw GPS waveforms on a computer. One potential project would be to utilize an SDR (such as an RTLSDR) to receive on your computer raw GPS waveform data and convert to GPS time, analyzing the performance of various GPS software frameworks, SDR receivers, etc.

(Advanced) HFT HF Hunt

The intent of this project is to attempt to detect transmissions from (publicly speculated) HF (HAM radio) transmitters (but legally, respecting property rights…).

Note that this could be done both by using directional antennas while in the vicinity, but also possibly using a publicly available set of SDRs tuned to the HAM radio. There was a particular set of SDRs that recently added support for triangulation (factoring in the time difference between when different receiving stations received the same single).

To provide one possible motivation for this project, see https://forums.qrz.com/index.php?threads/ham-radio-mentioned-prominently-in-high-frequency-trading-story.617581/ . Note that various HAM radio operators (civilians who use long distance radio to communicate directly) are concerned that some electronic trading firms may be broadcasting signals that are interfering.

(Advanced) Airplanes of interest Search

As mentioned in class, ADS-B is a protocol and technology whereby planes broadcast their identification, GPS coordinates, and other information. It is easily possible to receive this information and there are many public decoders for the data.

There are various public websites that rely on a fleet of recording stations around the globe to record this data and offer it to the public for tracking real time locations of planes. Individual planes, however, can request that their data be filtered from the public data.

One potential project could be to utilize and identify planes that are being filtered from the public data sources. The intent here is that those who may wish to obscure their location or flights may also be of interest to the market. An even more amusing project would be to search for planes that happen to be to be flying **extremely closely** along the straightest path between major market data centers (possibly repeating daily). Note this is less about the actual research result, so much as 1) the experience a student gains from implementing it, and 2) it could sound very impressive during interviews utilizing a host of advanced technologies across the technology stack.

Another viable project would be to simply utilize one of the publicly available decoder programs which output only the text values and then for your project implement a real-time GUI rendering the locations of various planes in real time on a map.