By: Ernest Duckworth, Samuel White, and Sam McKay

Date: 4/5/2022

D2: Crypto Bot

**Proof of Materials:**

* Raspberry Pi 4 4GB
  + Price: $180 (entire kit)
  + 
* Fingerprint Scanner
  + Price: $204
  + 

**Summary of Project**

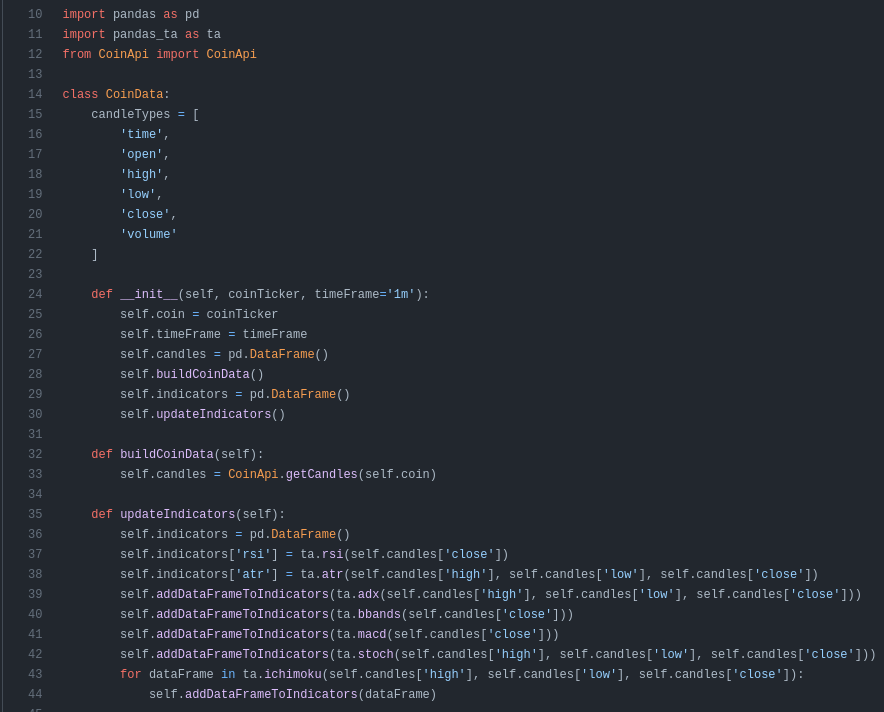
* This development project is an IOT device run on a raspberry pi that will evaluate different cryptos and automatically buy/sell if possible
* Will have the ability to connect to people's accounts in CoinBase
  + With prior setup from your account (must create API keys)
  + Using the fingerprint scanner to authenticate and save people's accounts/keys
* It will have a GUI so that you can easily track and control the bot.
* Will utilize 3-4 threads
  + GUI (main)
  + Scanner (producer)
  + Analyzer (consumer)
  + buyer/seller/notifier (market access & logger)
* Files in project
* CoinAPI.py
  + This is the API connection to retrieve new data about Cryptos
* CoinData.py
  + This is the model that will hold all the data relevant to a single coin
* CoinScanner.py
  + This is the controller for a list of CoinData (list of cryptos trading/watching)
* CoinUI.py
  + This is responsible for the creation and maintenance of the GUI
* Analyzer.py
  + This is the controller for the analysis of the CoinData
* Trader.py
  + This is the controller for the process of buying/selling coins
* CryptoBot.py
  + This is the summed controller for all data necessary to trade independently from the user
* Main.py
  + This starts the CryptoBot after the authentication process

**Progress**

Currently

***CoinAPI.py***

* This is the API connection to retrieve new data about Cryptos
* Can read x amount of candles for y coin ticker in z timeframe
* Can get the current price of coin x
* Can change the ticker timeframe that is received
  + Ex: 1m, 5m, 30m, etc
* All Candles are returned as pandas.series or pandas.dataframe



*Code Snippet of Coin Data*

***CoinData.py***

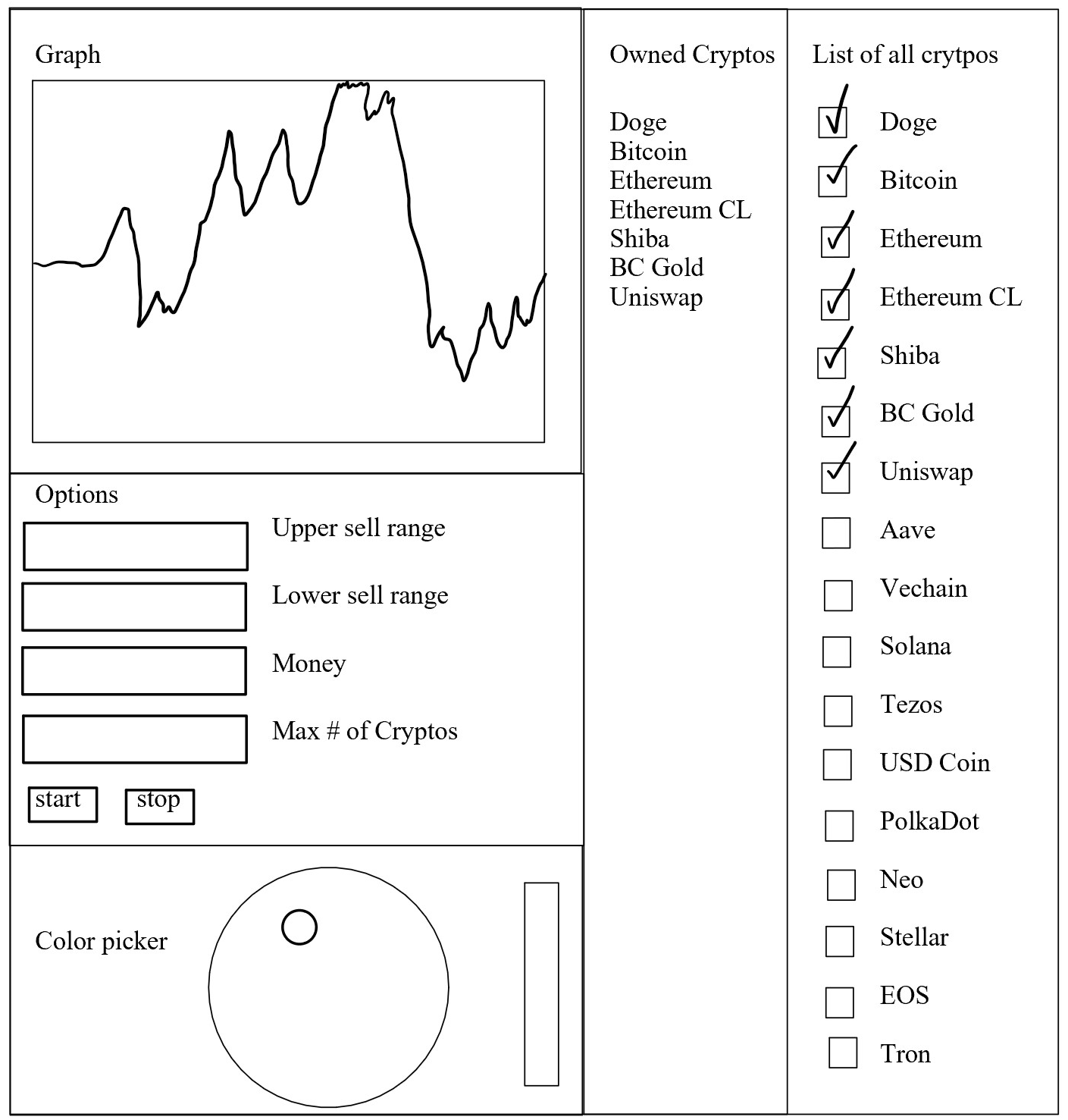
* Can store data in various time frames candles
  + Ex 1m, 5m, 30m, etc
  + Stored as pandas.dataFrame
* Can add a new candle(s) to coin data
  + Adds as long as the times are different and correspond to the correct timeFrame
  + Can add multiple candles if for some reason it is timeFrame(s) behind more than one unit
* After candle(s) are added it reevaluates all indicators for that CoinData

***CoinScanner.py***

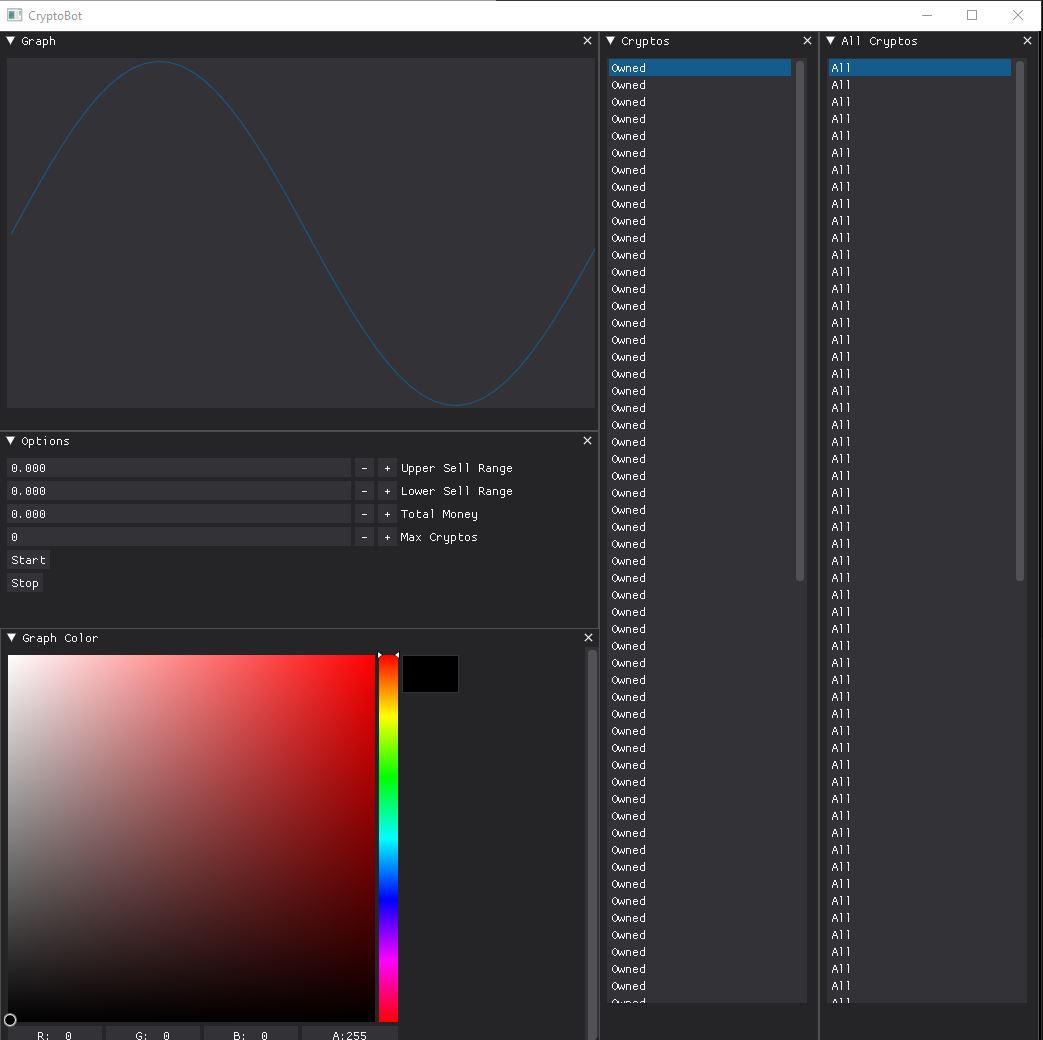
* Can recognize when a timeFrame (1m, 5m, etc) has passed and calls each CoinData to add a new candle(s)
  + Adding to the coin is done from the coin itself so the scanner holds zero calls to CoinAPI

***CoinUI.py***

* Responsible for creating and maintaining the GUI. Used to start and stop the threads that analyze and purchase coins for the user.
  + The goal for UI currently



* + Prototype of UI



***Analyzer.py***

* Nothing

***Trader.py***

* Nothing

***CryptoBot.py***

* Nothing

***Main.py***

* Nothing

***Threads***

* Nothing
  + Everything currently is run on a single thread

Planned

***CoinAPI.py***

* Currently, nothing is planned but could involve adding functions for
  + CoinData.py
  + Trader.py

***CoinData.py***

* Currently, nothing is planned but could involve adding functions for
  + CoinScanner.py
  + Analyzer.py

***CoinScanner.py***

* Needs to be run on its own thread constantly
* Needs to notify the analyzer thread that a coin is now ready for analysis
  + Can be accomplished in a number of ways and we are still determining the best way to do this

***Analyzer.py***

* Needs to have a way to communicate with CoinScanner.py so that it can determine when and what coin to evaluate and rank
* Needs to have a way to communicate with Trader.py if the current analysis leads to a buy/sell case
* Needs to have an algorithm for several different types of analysis
  + Indicator analysis
  + Fundamental analysis
  + Other trading strategies that we find useful
  + The return value should be a weight to how much we believe it is going to go up or down

***Trader.py***

* Needs access to personal account
  + From API keys to CoinBase
* Needs to be able to make purchases on the market
  + If we still have funds
* Needs to be able to sell (part of) coins on the market
  + If we own any of that coin
* Possibly connect to Discord or another notification service to log that x coin just hit buy/sell case
  + And if we want to log how much the bot bought/sold

***CryptoBot.py***

* Needs to have user account loaded
* Needs to start all threads
* Needs to begin bot trading process in the background of GUI
* Needs GUI (or can be in main)

***CoinUI.py***

* Graph of % change in assets
* Color picker to change the plotline
* List of owned cryptos
* List of cryptos to choose from
* Needs connection with backend
* Needs to be able to update the graph every minute or so

***Main.py***

* Needs to have the fingerprint authentication service implemented
  + After authenticated user needs to either enter their keys or load their keys from an encrypted file
* Needs to create the CryptoBot and begin its process
  + If GUI is in crypto bot we’ll basically never enter back into main again
    - If GUI is in main then it will start the bot and begin building its GUI

***Threads***

* GUI (parent)
  + Need to create all of the GUI processes
  + Need to create how Trader notifies GUI when buy/sells happen
* Scanner
  + Need to create when the scanner thread is created and how it communicates with the Analyzer thread
* Analyzer
  + Need to create when the analyzer thread is created and how it communicates with the Scanner thread (when to analyze which coin)
* Trader
  + Need to create when the Trader thread is created and how it communicates with the Analyzer thread

**Potential Security Risks:**

There are a few security risks that may arise during this project. As we are dealing with people's trading accounts that are linked to government-issued currency this has a possibility of losing 100% of the money it was given. This also has the risk of having people's keys stored locally and others somehow being able to access others' accounts still.

Let's first look at the first issue of being able to lose all of someone's money they put into it. As with investing this is always a possibility, there is no guarantee to make money on any single investment. Anyone who is putting money into this project should understand that there is always a possibility of losing every single penny that they invested. Actually preventing this possibility would be to set a stop loss. Where the user could input either a percent value or a monetary value of how much they are willing to lose from the initial start. This way we can ensure that the bot is still able to trade but only within that stop-loss limit. Then if we ever hit or go below that limit we can send a message to the GUI that buys/sells have been stopped and they can either decide to change the stop loss or just let the bot run without the capabilities of actually buying/selling and only notifying when a stock has entered buy/sell zones.

Dealing with the second issue of protecting someone's keys on the local system. There are a few ways that we can go about protecting this information. We can use the hash value that is returned from the fingerprint scanner to encrypt a file that stores their personal keys. This way there is no way to view the owner's keys without first decrypting them, that way anyone with access to the system is unable to look at anyone's keys except their own (given they have the hash value of their fingerprint). The next option we could do to remove this issue is that the fingerprint scanner could remember people's names/usernames but not actually store their keys. This way their keys must be entered every time they start the bot. This way there is absolutely zero possibility for someone's keys to be stolen from the app other than the fact of them being stolen when they were entered.

**Limitations like resolution, accuracy, or response time:**

This project has a few limitations that could significantly affect accuracy and response time. As we are running this entire project on a raspberry pi it has enough hardware support to run this at a basic level trying to trade on a few coins but as we move into a larger “watchlist” of coins to analyze and trade we could begin to have more data than we can analyze and read from the 2 threads (scanner and analyzer). We also need to be very careful about how we store the data used in the program and filter out “old” data that is no longer relevant to our trades.

Examining the first issue we see that as our watchlist grows the more we’ll be counting on the scanner and analyzer to run as quickly as possible. For example, if the list of coins takes more than the timeframe of candles it's examining (1m candles by default) then we would eventually begin to fall minutes behind the actual candles being fetched. So as the list of coins to scan/analyze increases our response time increases. Since we are limited to a single producer(scanner) and consumer(analyzer) there is not much we can do to improve this limitation other than ensuring that the semaphore for communication between producer and consumer has no effect on what data is being analyzed.

The second issue stems from how we get rid of old data. As this program is meant to run for extended periods of time we need to be careful about how we filter out data that is no longer relevant. The main concern is how we are going to maintain the size of the data that the GUI consumes. The secondary concern is keeping a limit on the size of the data frames for candles and indicators. This could easily be accomplished by removing a row from the data frames as we add a new one.

**Cost and Marketability of a Device:**

This type of project and device is geared towards those who want to get involved in day trading but do not have the time to spend looking at the stock market all day. This device is intended to help alleviate some of those challenges by having a Rasberry Pi Day trade for you. The overall cost thus far has been under $200. With the hopes that this product brings in some money. It can almost be said that this cost is negligible since the product will eventually pay for itself, that is if the market continues to go up.

Due to the low cost and low power of such a device, the crypto bot is highly marketable and obtainable by those who might not have much money. However, the marketability and success of this product are dependent on how well the stock market does. During COVID the Dow Jones stock exchange dropped 10%, the sixth-worst drop in history. At a time like that, this product has almost no use. In the end, it is how well we can write some predictions that determine how well this product turns out in the long run.

**Conclusion:**

At this current time, we are making good progress towards the end goal. Where the code is right now is not enough and it is not where we want it to be, but we hope that within the next week or so to have a large portion of the project done. Some of the areas that we are planning on tackling next are the analyzer of the candles and then connecting the data from the analyzer to the front-end. One of the biggest goals we have is to get a nice, interactive GUI that is easy to use so the user can easily manage his or her cryptocurrency. In summary, we like the progress that’s been made but would like to make some progress in the coming future.