Oregon State University
College of Engineering

CS Capstone Progress Report

May 6, 2018

100K Spaceport America Demonstration Rocket Project

Prepared for

School of MIME

Nancy Squires

Prepared by

Group 42

Michael Elliott
Sam Hudson
Glenn Upthagrove

Abstract

The following descries the progress of the OSU HART Computer Science team from the beginning of the academic year. Since the beggining of the academic year, the team has created a suite of software tools for tracking a rocket in flight. THe system has been feature complete since the end of the winter term. The first half of the spring term has been spent on testing, bug fixes, improvements to usability, and aid to other teams.

CONTENTS

1 Purpose and Goals

The purpose of the project is to design and build recovery and telemetry systems for a rocket that will reach a target altitude of 100,000 feet. The team has created telemetry interfaces for tramsmitting and receiving data and avionics for the recovery of the rocket. The goal of the project was to design a means of collecting, interpreting, and transmitting data on the rocket, in addition to receiving, interpreting, and displaying data on the ground. The team also needs to have a safe recovery of the rocket after it has completed its flight and come to rest. This software suite is intended to give reliable tracking of a rocket in flight, and graphically represent this data to the user. This is necessary as the rocket will go well outside of visible range, and the use of automated tracking will be essential to the recovery of the rocket.

The purpose of the Kalman filter is to remove the noise from the readings from the sensors, since the team will need to collect and utilize the data the team receives from the sensors even though it may be inaccurate. The goal of the Kalman filter is to get accurate enough readings to correctly judge the altitude of the rocket at apogee, as well as correctly time the deployment of the parachutes. Since our system is used in parallel with the Telemega system, the team will be able to compare our results with those from the professionally made board. The safe recovery of the rocket is also dependent on the effectiveness of the Kalman filter, for if the filter doesn't work properly, the accuracy of the reported location will suffer.

The purpose of the transmission code is to transmit the readings from the Kalman filter down to the ground station despite unfavourable conditions. This is especially important for meeting our goal of safe recovery of the rocket, since the GPS data sent from the rocket to the ground station will be used to locate it. The goal for the transmission code is to be able to get live updating speed and position data from the rocket. If the team can get live updates from the rocket as it reaches landmarks in its flight, our protocol works as intended. The purpose of the packet processing and interpolation code on the ground station is to supplement the transmission code and perform corrections. Another goal of the transmission code is to minimize the number of corrections and interpolations that the ground station has to make. The team aims to have as few dropped and corrupt packets as possible. The team also aims to be able to easily determine if a packet is corrupt, so as not to accidentally rely on corrupt data. Another goal of the transmission code is to minimize the impact of communication on the CPU so that the CPU time can be used for more important functionality. This means that a corollary goal for the transmission code is to minimize the packet size.

The goal of the packet processing and interpolation code is to ensure that all of the data received over the radio is correct. The other goal is to provide accurate estimations in between packets as well as predictions for the future. If our goal is successful, the visualizations will have a much easier time using the data provided. It will also minimize the potential impact of poor transmission performance on the other components of the ground station.

2 Retrospective

Positives	Deltas	Actions
There have been no merge conflicts this entire term, and team coordination has been much better. The team has stopped having regular team meetings, but meet more often now than in the past terms.	The team still could better coordinate work efforts, and have more solid weekly plans of action.	The team will reassess the Github issues still pending, and add more to them. We will then take the remaining weeks and divide them as evenly as possible and work on them week by week.

3 Michael Elliott

3.1 Goals for Michael Elliott

Michael Elliott was the author of the Kalman filter and interpolation.

3.2 Current Progress For Michael Elliott

The current status of the Kalman filter is that it is working as expected with a simplified prediction model for testing. The Kalman filter is already able to filter out randomly generated noise in testing. In the testing we have been using a simplified prediction model with static covariance in the inputs, so we still have more testing to do. We are currently waiting on the simulation data in order to base our prediction model on that. We are also waiting on the hardware so we can test the Kalman filter on the final host system. The filter is programmed as flexibly as possible in generic C without the use of any external libraries or system calls, so as long as there is a working C compiler for the processor the filter should run well. Ultimately the success of the Kalman filter is partially dependent on the hardware and the firmware, but as soon as that is complete we will just need to do more testing.

The transmission code worked as of the last test on the old hardware. The new hardware is going to use the same radio, so the code should not be much different (if at all) on the new board. We are waiting on the hardware to do further testing. Since the architecture is going to be different on the new hardware, the communication between the Kalman filter and the radio may need to be updated. This is mostly dependent on firmware.

The interpolation python code can successfully interface with the C program used to read in from the radio on the ground station and verifies the checksum on each packet. It can also successfully generate a best fit curve to match the data received. There is still a bit of work to be done to allow the other modules to interface with it more easily.

3.3 Unfinished Components for Michael Elliott

For the Kalman Filter we need the final prediction model as well as the hardware in order to proceed with testing. Since the host system will have less random noise in the inputs as well as dynamic covariance of the inputs, we will need to do more rigorous testing once we have the hardware. Also the filter may require some slight modifications in order to make it more modular in case we wish to change the prediction model more easily in the future. This would be nice if we get updated simulation data and wish to update the filter to reflect the new simulation.

For the transmission code we are just waiting on hardware in order to finish it. We still need to test it with the new board, but hopefully it works similarly to the old one since the radios are the same.

For the interpolation code we still have a bit of work to do to make it play nicely with the other modules in the system, but besides that it is done.

4 Samuel Hudson

4.1 Goals for Samuel Hudson

Samuel Hudson was primary developer for the data base, API, and web application.

4.2 Current Progress For Samuel Hudson

This term has mainly been focused around supporting the electrical engineering team with the design and implementation of the firmware. One thing specifically that Sam has worked on is the extraction of GPS and altitude telemetry from NMEA data. For this Sam has written a parser in C. The parser reads from an input stream and adds to structures for storage in the EPROM and for transmission over the radio frequency to the ground station. Included below is the gps_data_extract_nmea function that is responsible for carrying out said functionality.

```
struct gps_data extract_nmea(char data[])
{
    const char s1[2] = "\r";
    const char s2[2] = ",";
    char *token1, *token2;
    char *saveptr1, *saveptr2;
    token1 = strtok_r(data, s1, &saveptr1);
    char des[80];
    struct gps_data r;
    while (token1 != NULL)
    {
      strcpy(des, token1);
      token2 = strtok_r(des, s2, &saveptr2);
      const char *t[15];
      int i = 0;
      if (strncmp(token2, "$GNRMC", 7) == 0)
      {
        while (token2)
        {
          t[i] = token2;
          token2 = strtok_r(NULL, s2, &saveptr2);
          i++;
        }
        i = 0;
        if (strncmp(t[2], "A", 2) == 0)
        {
          r.lat = atof(t[3]);
          r.lon = atof(t[5]);
        }
        else
        {
          r.lat = 0.0;
          r.lon = 0.0;
```

```
        }
      }
      else if (strncmp(token2, "$GNGGA", 7) == 0)
      {
        while (token2)
        {
          t[i] = token2;
          token2 = strtok_r(NULL, s2, &saveptr2);
          i++;
        }
        i = 0;
        r.alt = atof(t[9]);
      }
      token1 = strtok_r(NULL, s1, &saveptr1);
    }
    return r;
}
```

In addition to developments to firmware Sam has made improvements to the API. The timer that exists on the application was having issues syncing with launch time. Now the clock uses the launch time set by the start_launch function defined in the API as the source of truth for the time value rather than the application start up time. The code below shows the improved launch data get function that returns start_time and end_time.

```
@app.route('/api/v1.0/launch/', methods=['GET'])
def get_launch():
    launch = db.launch
    if launch.count() == 0:
        return jsonify({'result': 'no data'})
    output = []
    cursor = launch.find().sort([("_id", pymongo.DESCENDING)])
    output.append({
        'start_time': cursor[0]['start_time'],
        'end_time': cursor[0]['end_time']
    })
    return jsonify({'result': output})
```

GPS data for both the sustainer and the booster have also been added to the web application to make determining the location of the rocket trivial. This data is extracted from an existing poll telemetry function and displayed on the main page of the application.

4.3 Unfinished Components for Samuel Hudson

The usability study is currently in progress. We are currently in the process of interviewing different members of the class. The aim of the usability study is to determine whether the rocket tracker application is to the standard the team expects, and information is easily interpretable. Some of the questions include How easy was it to determine the state of the rocket at one point in time?, Can you find the altitude of the booster and sustainer?, Can you find the velocity of the booster and sustainer? and How is your overall user experience?. The team is hoping to receive responses to these questions by May 11th.

5 Glenn Upthagrove

5.1 Goals For Glenn Upthagrove

Glenn Upthagrove claimed primary responsibilty for several pieces of the pipeline. His main responsibiites were data logging, 3D trace, and most of data handling. This team member has also become the author of the test data generator, the graphical user interface, and has become primary architect of interprocess communication.

5.2 Current Progress For Glenn Upthagrove

As of the end of winter term, the computer science team is feature complete on all required features listed in earlier documents. There are some features that were implemented that were beyond the requirements listed before. The data that is received can be formatted into JSON strings, and logged to text files. The data can be stored in a database and then read by the web application and presented to the user in a clean 2D representation from the work of Samuel Hudson. . The data can also be read from a text file and presented in a 3D space. The 3D trace is also capable of doing this in near real time, and given an Internet connection can get a map to texture the ground plane with centered at any point on the globe via the Google Static Maps API. The several pieces are all held together by a data handler that will spawn all necessary parts and link them together. These links are inter-process communication allowed by pipes and FIFOs. There is a data generator that allows for the data to be faked for both testing and simulation purposes. The configuration scripts also has been improved over this past term to make the installation of the software suite far easier for the average user. A graphical user interface, or GUI, was also written using Pythons tkinter module. The GUI makes the software more accessible to the average user, who may find command line arguments complicated, especially with many arguments. The majority of this term has been spent, however, working alongside the electrical engineering team to write firmware for the backup avionics board.

5.3 Unfinished Components for Glenn Upthagrove

The only part that is technically incomplete for Glenn Upthagrove is to receive data from hardware. This is however outside the control of the computer science team. The avionics board has been the responsibility of the electrical engineering team, and different boards differ in result with the same code. This alongside firmware that has yet to yield results has caused the electrical engineering team to move to whole new processor on a new board. These hardware issues and the recent change in hardware has made data retrieval impossible, as there is no functional hardware to retrieve data from. Aiding in the development of working firmware is the highest priority for the coming weeks. There is also always room for more testing of code, though it still has yet to fail under synthetic loads which are expected to be much more difficult than real ones. There is also always room for improvement on usability, and thusly the GUI, trace, and configuration scripts can be polished as time permits.

5.4 Importnat Code

The following are piecs of importnat code that were written by Glenn Upthagrove.

5.4.1 Configuration Script

Below is the BASH script that configures a machine running Linux to run the HART software suite.

```
echo "installing apt-file"
echo
sudo apt-get install file -y
sudo apt-get install apt-file -y
echo
echo "updating apt-file"
echo
sudo apt-file update -y
#docker
echo
echo "installing docker"
echo
sudo apt-get install docker.io -y
sudo apt-get install docker-compose -y
#C/C++
echo
echo "installing gcc"
echo
sudo apt-get install gcc -y
echo
echo "installing g++"
echo
sudo apt-get install g++ -y
#Make/CMake
echo
echo "installing make"
echo
sudo apt-get install make -y
echo
echo "installing cmake"
echo
sudo apt-get install g++ cmake -y
#OpenGL
echo
```

```
echo "updating"
echo
sudo apt-get update -y
echo
echo "upgrading"
echo
sudo apt-get upgrade -y
echo
echo "installing mesa-utils"
echo
sudo apt-get install mesa-utils -y
echo
echo "installing mesa-common-dev"
echo
sudo apt install mesa-common-dev -y
echo
echo "installing freeglut"
echo
sudo apt-get install freeglut3 -y
sudo apt-get install freeglut3-dev -y
sudo apt install libglu1-mesa-dev freeglut3-dev -y
echo
echo "copying libglut.so to /usr/lib64"
echo
sudo mkdir /usr/lib64
sudo cp ./3d_trace/libglut.so /usr/lib64/
sudo chmod o=rwx /usr/lib64/libglut.so
sudo chmod u=rwx /usr/lib64/libglut.so
echo
echo "installing binutils-gold"
echo
sudo apt-get install binutils-gold -y
echo
echo "installing libglew"
echo
sudo apt-get install libglew-dev -y
echo
echo "installing build-esssential"
echo
```

```
sudo apt-get install build-essential -y
echo
echo "installing libglew 1.5 dev"
echo
sudo apt-get install libglew1.5-dev libglm-dev -y
#Python and required modules
echo
echo "installing python"
echo
sudo apt-get install python -y
echo
echo "installing pip"
echo
sudo  apt-get install python-pip python-dev build-essential -y
sudo  pip install --upgrade pip
sudo  pip install --upgrade virtualenv
echo
echo "installing Pillow module for python"
echo
sudo pip install Pillow
echo
echo "installing requests module for python"
echo
sudo pip install requests
echo
echo "installing pyUSB"
echo
sudo pip install pyusb
echo
echo "updating"
echo
#update and upgrade
sudo apt-get update -y
echo
echo "upgrading"
echo
sudo apt-get upgrade -y
echo
echo "Done"
```

```
echo
#make
echo
echo "Making programs"
echo
cd 3d_trace
make
cd ..
cd logging
make
cd data_handling
make
cd ..
cd data_generator
make
cd ..
cd conversion
make
cd ..
echo
echo "done"
echo
#make desktop icon
echo
echo "making desktop icon"
echo
cp ./HART.desktop ~/Desktop
echo
echo "Done"
echo
echo
echo "Creating Directory File"
echo
pwd > HART_DIR.txt
sudo mv HART_DIR.txt /HART_DIR.txt
echo
echo "Done"
echo
```

5.4.2 Graphical user Interface

Below is the code for Grpahical User Interface for the end user to interact with. this code is written for Python 2 using the Tkinter module.

```
# for Python2

import Tkinter as tk   ## notice capitalized T in Tkinter

import tkMessageBox as tkm

from PIL import ImageTk,Image

import sys

import os

import subprocess


#print current Python Version

print(sys.version)


#define sim flight callback function

def simcallback():

        os.chdir('./data_handling')

        subprocess.call(['./datahandle', '-sim'])

#define Tracking callback function

def monitorccallback():

        os.chdir('./data_handling')

        subprocess.call(['./datahandle'])
```

```
#define Logging callback function

def logscallback():

        dirfile = open('/HART_DIR.txt', 'r')

        string1 = dirfile.read()

        string2 = string1[:-1]

        os.chdir(string2)

        root2 = tk.Tk()

        root2.title("View Logs")

        root2.minsize(width=480, height=480)

        root2.maxsize(width=1920, height=1080)

        root2.resizable(width=True, height=True)

        text1 = tk.Text(root2)

        for filename in os.listdir("./logs"):

                text1.insert('end', filename)

                text1.insert('end', '\n')

                myfile = open("./logs/"+filename, 'r')

                string = myfile.read()

                text1.insert('end', string)

        text1.pack()

        root2.mainloop()
```

```
#change directory

dirfile = open('/HART_DIR.txt', 'r')

string1 = dirfile.read()

string2 = string1[:-1]

os.chdir(string2)

#define gui

root = tk.Tk()

root.title("HART Avionics")

root.minsize(width=480, height=480)

root.maxsize(width=1920, height=1080)

root.resizable(width=True, height=True)

#sim button

simbtn = tk.Button(root, text="Sim Flight", command=simcallback, bd=4)

simbtn.pack()

simbtn.place(relx=0.5, rely=0.4, width=128, x=-64)

#monitor button

monbtn = tk.Button(root, text="Monitor Flight", command=monitorccallback, bd=4)

monbtn.pack()

monbtn.place(relx=0.5, rely=0.5, width=128, x=-64)
```

```
#view logs

logbtn = tk.Button(root, text="View logs", command=logscallback, bd=4)

logbtn.pack()

logbtn.place(relx=0.5, rely=0.6, width=128, x=-64)

#start main UI loop

root.mainloop()
```

5.4.3 Conversion

Below is a header file for C code that can convert a struct telem_data into a JSON string and visa versa.

```
#ifndef __CONVERSION
#define __CONVERSION



#define _XOPEN_SOURCE 500 //makes usleep work
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <math.h>
#include "./telemetry.h"
int debug;
//void convert(char* buff, float vel, float lat, float lon, float alt, float time){

/*******************************************
*Title: convert
*Params: a string pointer, a telem_data*
*Description: Converts a telem_data struct into
*a JSON packet
*******************************************/
void convert(char** buff, struct telem_data* in){
        char str1[9] = "velocity";
        char str2[9] = "latitude";
        char str3[10] = "longitude";
```

```
        char str4[9] = "altitude";
        char str5[5] = "time";
        char str6[5] = "type";
        if(buff == NULL){
                *buff = (char*)malloc(256);
        }
        memset(*buff, '\0', 256);
        sprintf(*buff,"'{\"%s\":\"%f\", \"%s\":\"%f\", \"%s\":\"%f\", \"%s\":\"%f\", \"%s\":\"%f\"
        if(debug){
                printf("%s\n", *buff);
        }
}

/*********************************************
*Title: structure
*Params: a string pointer
*Description: Converts a JSON packet into a
*telem_data structure
*********************************************/
struct telem_data structure(char** buff){
        char* messagecpy;
        char* token;
        struct telem_data data;


        token = (char*)malloc(sizeof(char)*256);
        messagecpy = (char*)malloc(sizeof(char)*256);
        memset(token, '\0', 256);
        memset(messagecpy, '\0', 256);
        strcpy(messagecpy, *buff);
        token = strtok(messagecpy, ":");
        token = strtok(NULL, "\"");
        data.vel = atof(token);
        token = strtok(NULL, ":");
        token = strtok(NULL, "\"");
        data.lat = atof(token);
        token = strtok(NULL, ":");
        token = strtok(NULL, "\"");
        data.lon = atof(token);
```

```
        token = strtok(NULL, ":");
        token = strtok(NULL, "\"");
        data.alt = atof(token);
        token = strtok(NULL, ":");
        token = strtok(NULL, "\"");
        data.time = atof(token);
        token = strtok(NULL, ":");
        token = strtok(NULL, "\"");
        data.type = token[0];
        return data;
}

#endif
```

6 Problems and Solutions

The biggest problem the tema has encountered so far this term is with the avionics hardware. The processor was changed on us just last week, so the team basically has to restart the process of integrating with the hardware again. This also means teh team has more work to do for the firmware for the bord which, although not the responsibility of this team, this team has found itself assisting with quite frequently. As stated earlier, there is still some work to be done with making the interpolation code work nicely with the rest of the system, but it should be a relatively easy fix and the module it is not actually a part of the requirements but rather a quality of life improvement.

7 Detailed Development

7.1 Week 1

7.1.1 Michael Elliott

- **Summary:** Had three meetings with the entire team, with the ECE subteam, and within our own subteam to discuss progress over spring break and plans for this term.

7.1.2 Samuel Hudson

- **Plans:** This was the first week back from Spring break. This week consisted of mapping out what work was left to do.
- **Problems:** Getting an understanding of where everyone was after break.
- **Progress:** We were able to make a plan and determine how to taken the work for the rest of the term.
- **Summary:** This week we made a concrete plan as to how to tackle the remaining work of the term. We have a better understanding of how we can progress and everyone seems to have made reasonable progress over the spring break.

7.1.3 Glenn Upthagrove

- **Plans:** Get back in touch with the groups and form a plan.
- **Problems:** Catching up and figuring out what we have all been done takes time.
- **Progress:** I missed a meeting, but got back in touch with everyone as time permitted.
- **Summary:** This week I focused on getting the rest of the term in motion. I missed the AIAA meeting due to travel delays, but I met with everyone who was in town at a later time in the week. Sam is in Europe still and will not be available to work for a week or more.

8 Week 2

8.0.1 Michael Elliott

- **Summary:** Met with the entire team to discuss progress from the past week and plans for the coming week. Worked on firmware with the ECE team. Met with the Aero and Recovery team to discuss the rocket simulations and incorporating the data into the Kalman filter Had our subteam meeting to talk about plans.

8.0.2 Samuel Hudson

- **Plans:** This week we were determining which team members are responsible for developing firmware code.
- **Problems:** The ECE were having issues getting the board to work correctly. Making debugging difficult.
- **Progress:** By the end of the week we were able to get the LEDs to work on the board to signify some operations.
- **Summary:** This week we started allocating work for the firmware development portion of the project. I was assigned the task of making improvements to Baro and building a parser to extract NMEA data from the data stream.

8.0.3 Glenn Upthagrove

- **Plans:** Get to work on pyUSB and firmware.
- **Problems:** pyUSB is not well known to anyone I have contact with. Firmware still has trouble loading.
- **Progress:** I have started with pyUSB, but have much to do. The firmware can now be loaded and we have gotten blinking lights.
- **Summary:** This week the ECE team found how to flash the board correctly and how to set LED lights so we can begin doing real development and testing of firmware. I have a Telemega and Teledongle so I can work on interfacing with that as well.

9 Week 3

9.0.1 Michael Elliott

- **Summary:** Met with the entire team to discuss progress from the past week and plans for the coming week. Work on the Kalman filter to more easily accept the data from the simulations. Met again with Aero and Recovery to go over the simulations and figure out how to most easily extract the necessary data from them Talked to my interviewee about his project for my WIRED! style article Met with our subteam to talk about the expo poster.

9.0.2 Samuel Hudson

- **Plans:** This week we were determining what else has to been done to improve the web application and firmware.
- **Problems:** The data handle init script is having issue starting docker.
- **Progress:** We fixed issues with docker start script and also I made some progress on the parser.
- **Summary:** This week I was able to write some more code to support the development of the NMEA parser. Also an issue with the docker start script has been resolved.

9.0.3 Glenn Upthagrove

- **Plans:** Firmware and usability study.
- **Problems:** Sam has not started his usability study and firmware constantly gives a BUSY signal over SPI and I2C channels.
- **Progress:** I fixed the BUSY issue because we were using the wrong handles, but data is still no good, no word on Sam's study as of yet.
- **Summary:** This week and next week are all firmware. I want to start getting data in. We can blink LEDs and we can transmit data over the radio, so there is a start, but we need actual data. The SPI interface to the barometer was giving BUSY all the time, same with the MPU over I2C, but I found that issue, and I fixed read and write by replacing our old quick read and writes with the HAL functions that are more robust. The data is still all garbage though and I have only just reached that point so at this stage there are any number of things it could be.

10 WEEK 4

10.0.1 Michael Elliott

- **Summary:** Met with the entire team to discuss progress from the past week and plans for the coming week. Finished my article Worked on the poster for the expo Talked with the ECE team about their plan to roll out new hardware.

10.0.2 Samuel Hudson

- **Plans:** This week we were planning out what question would be useful to ask our class members to improve the overall user experience of the web application.
- **Problems:** It was hard to find times for peers to carry out interview for the usability study.
- **Progress:** We finally found that we could have a complete usability by the end of week 6.
- **Summary:** The week I completed the development of the NMEA parser and tested it with the ECE team and we also planned out the usability study and found times to work carry out interviews.

10.0.3 Glenn Upthagrove

- **Plans:** Firmware.
- **Problems:** We are still having many issues with the board from ECE.
- **Progress:** none on firmware. We have discovered differing results from different boards, indicating hardware issuers.
- **Summary:** This week I gave helped with firmware again, though we have had no success. I had a long conversation with Dr. Winters, to ensure we will be safe if the board still has issues in the future. We flashed

the second board, and discovered that the same code on different boards causes different results, indicating that perhaps the boards have been improperly manufactured.

11 WEEK 5

11.0.1 Michael Elliott

- **Summary:** All team meeting Final touches on the expo poster; sent the poster off for printing. Work on the progress presentation Work on the progress report Met with our subteam a few times to discuss the expo etc. Made a few tweaks to the interpolation code.

11.0.2 Samuel Hudson

- **Plans:** We determined different areas to work on for the development of the firmware for the new board.
- **Problems:** This week we found out that the ECE team have decided to change board which effects our development process.
- **Progress:** We were able to meet up a few times to discuss the development of the board.
- **Summary:** This week we determined that the ECE are changing the board and that we have to make significant contributions to support their development efforts.

11.0.3 Glenn Upthagrove

- **Plans:** Quality of life and progress report.
- **Problems:** We have a progress report to write, which is hard considering so much time has been devoted to firmware this term.
- **Progress:** I have completed a lot of the quality of life goals, and we have a plan for doing the report this weekend.
- **Summary:** This week we have a report to do. This will be difficult as almost all our time has been on firmware. We also have not done our usability study, but we can present the plan. I have done some important maintenance to the repository of late, and fixed a bug with my Docker instance. I also made a few improvements to the usability of the software, by adding the GUI officially, as well as better installation scripts and a script that downloads for the user, and a desktop icon that opens the GUI.