mGRUE

minION Graphical Remote Unit Emulator

Design Document

Table Of Contents

Introduction

This project acts as an extension to the SMARTEn mobile DNA analytics system, which was created through a collaborative effort between the SCoRe group, the Networked Systems Research (NSR) group, Dr. Lauren Sassoubre, and Dr. Ian Bradley of the University at Buffalo. Dr. Jaroslaw Zola, the principal investigator (PI) of the SCoRE group, commissioned the minION Graphical Remote Unit Emulator (mGRUE) project in order to overcome a number of limitations that the SMARTEn system had when it came time to present his work. The first limitation of the SMARTEn system is its dependency on the MinION sequencer from Oxford Nanopore Technology, which has a minimum price of $1,000 USD per unit. The second limitation was the fact that the sequencer also takes an invariable amount of time to sequence genetic information to a host computer. This means that Dr. Zola could potentially keep an audience waiting upwards of an hour for the sequencer to finish a scan. Using MinION sequencers pose a high cost and time investment, which make it an unsuitable candidate to use during live software demos of the SMARTEn system.

The development of the mGRUE device along with its host device driver was intended to help solve these issues for Dr. Zola. The mGRUE device along with its embedded application works as an emulator for the MinION sequencer, specifically dealing with the serial communication transfer process of the genetic information. The mGRUE device itself is a Raspberry Pi 4 utilizing a 3.5 inch LCD touch screen display, and running the Raspbian operating system. The mGRUE host device driver serves to communicate with the mGRUE device and receive the genetic information the mGRUE transmits via serial. It will also run on the NVIDIA Jetson Nano, which is currently utilized by the SMARTEn system. The mGRUE implementation of the SMARTEn system will allow for a simulation of the SMARTEn system, without the cost and time bottleneck of the MinION device.

Basic Usage

The mGRUE desktop client is installed and runs on Dr. Zola's NVIDIA Jetson Nano device. The mGRUE embedded application is run on a Raspberry Pi 4 Model B device. The Raspberry Pi 4 is connected to the NVIDIA Jetson Nano via the serial (USB) port. On the touch screen panel of the Raspberry Pi 4, the user can select the specified file that will be transmitted to the desktop application via serial. Once the user hits the start button, the transmission will be in progress. The mGRUE desktop application will then parse the incoming DNA sequence strings to a folder specified by the application. The user will also be able to pause the transfer, increase the speed of the transfer, and also stop the transfer.

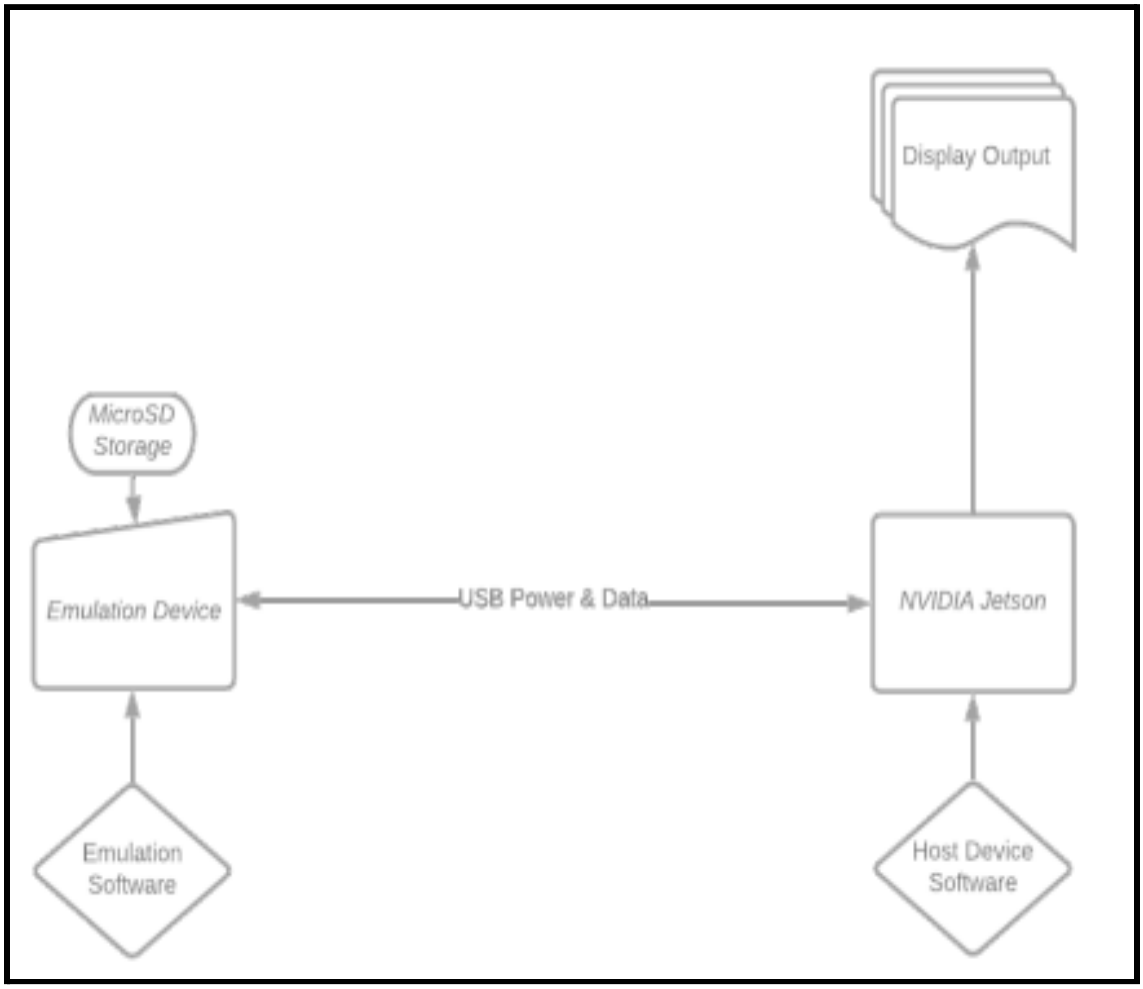
Bill of Materials

- Raspberry Pi 4B Kit
    - Raspberry Pi 4, utilized to run the mGRUE embedded application. GPIO ports also facilitate the usage of the LCD touchscreen as an optional human machine interface (HMI). Comes with a housing enclosure if the machine is run without the HMI.
- 32GB SD Card
    - Used to store the DNA sequencing data files and the installation of the Raspian operating system.
- USB-C 2.0 to USB-A Cable (3 Foot)
    - USB C Cable for serial communication and power of the device
- 3.5" LCD Touch Screen
    - Human machine interface that is compatible with the Raspberry Pi 4 GPIO. Comes with a housing enclosure to connect the Raspberry Pi 4 and the screen.

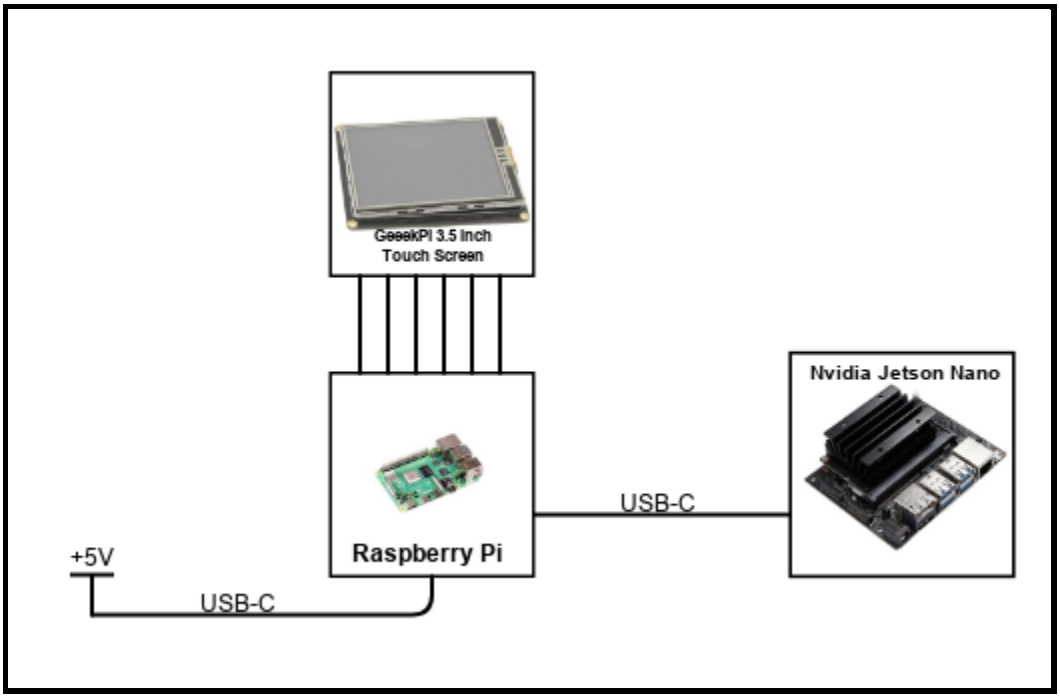
Material Purchase Links

- Raspberry Pi 4B Kit
    - https://www.pishop.us/product/raspberry-pi-4b-budget-kit/
- 32GB SD Card
    - https://www.amazon.com/SanDisk-128GB-MicroSDXC-Memory-Adapter/dp/B08GY9NYRM/ref=sr_1_3?crid=2HHBPVRCOTNQO&keywords=micro%2Bsd%2Bcard&qid=1646065792&sprefix=micro%2Bsd%2Bcard%2Caps%2C88&sr=8-3&th=1
- USB-C 2.0 to USB-A Cable (3 Foot)
    - https://www.amazon.com/AmazonBasics-USB-C-Cable-USB-IF-Certified/dp/B085SBKFJR/ref=sr_1_7_sspa?keywords=usb%2Bc%2Bto%2Busb%2Ba%2Bcable&qid=1653066346&s=electronics&sprefix=usb%2Bc%2Bto%2Busb%2Ba%2Bca%2Celectronics%2C79&sr=1-7-spons&th=1
- 3.5" LCD Touch Screen
    - https://www.amazon.com/GeeekPi-Raspberry-Touchscreen-Heatsinks-320x480/dp/B083C12N57/ref=sr_1_4?crid=I5WK0DHTUC28&keywords=pi+

Functionality Diagram



System Diagram



- Please refer to the section titled **Hardware Architecture** for more details regarding Raspberry PI GPIO ports and the **3.5” LCD Touch Screen Panel** pinout.

Hardware Architecture

The mGRUE team selected the Raspberry Pi as the embedded device for the emulation system, due to the Raspberry Pi's low-form factor, availability of touch screen LCD peripherals, which also function as an human machine interface. The LCD touch screen peripherals are connected via the GPIO ports of the Raspberry Pi, indicated in the aforementioned table.

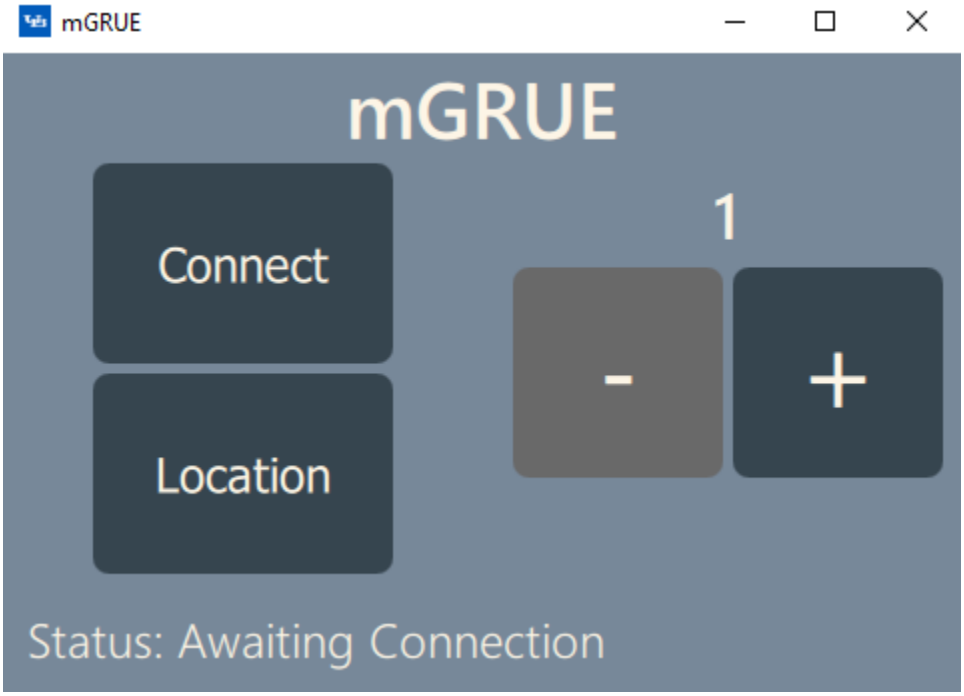
Pin Number(s)	Symbol	Description
1, 17	3.3V	3.3V Power Input
2,4	5V	5V Power Input
3,4,7,8,10,12,13,15,16	NC	NC
6,9,14,20,25	GRND	Ground
11	TP_IRQ	Touch Panel Interrupt
18	LCD_RS	LCD Register Selection
19	LCD_SI	SPI data input of LCD
21	TP_SO	SPI data output of LCD
22	RST	Reset
23	LCD_SCK	SPI clock of LCD
24	LCD_CS	LCD chip select
26	TP_CS	Touch Panel chip select

It was also imperative that testing on the mGRUE desktop client was done on a system that most resembles the Nvidia Jetson Nano during testing. This required the testing to be done on the Ubuntu operating system, specifically, to ensure that the software works seamlessly with the specified host device.

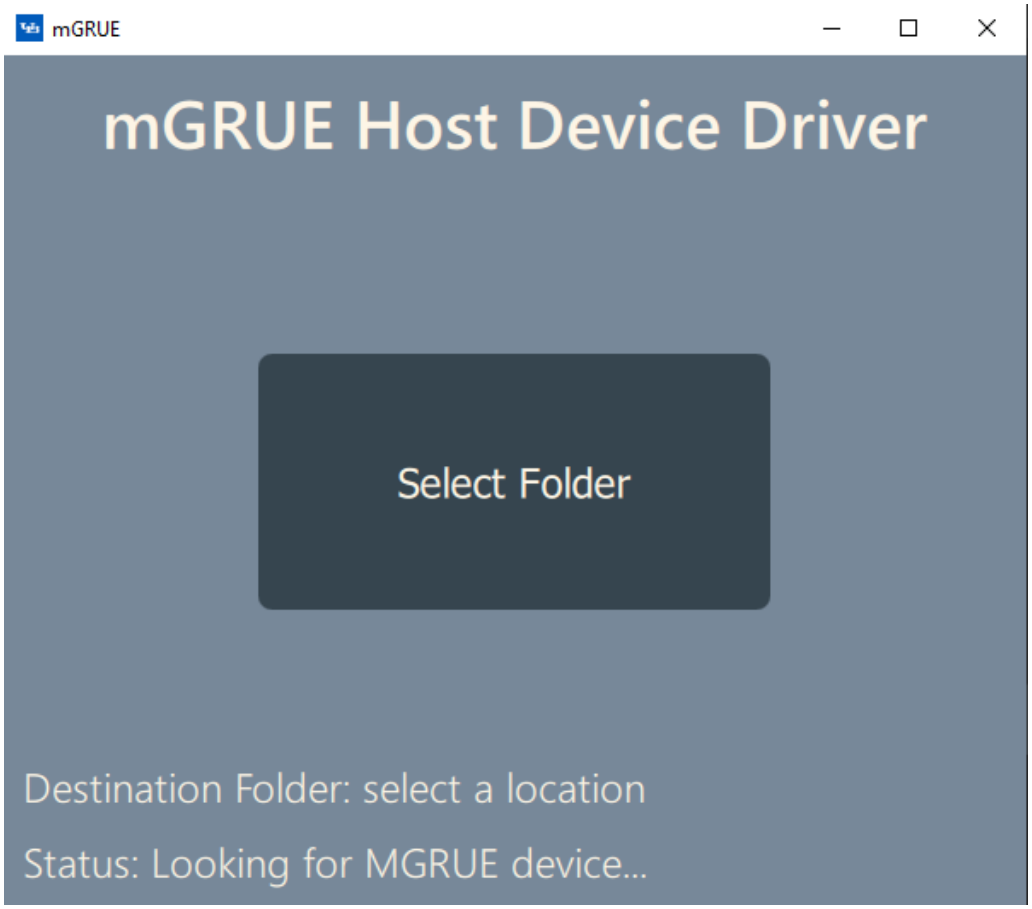
Software Architecture

The mGRUE embedded emulation software and the host device driver software were written using the PySide2 Qt graphical user interface and the pySerial library. Using PySide allows for a decreased development time, as Python abstracts the cross-platform build process of the user. Utilizing Python on the emulation hardware also allows for the ease-of-use for the serial communication protocols, as both utilize pySerial. This eliminates the need for redundant bug testing due to language specific nuances. For example, POSIX implementations of the serial communication protocol add NULL byte stripping and CR-LF translation. Although this implementation can be mitigated, it is both easier and safer to utilize the same serial communication library from both the emulation hardware and desktop software standpoint.

mGRUE Emulator Application



mGRUE Desktop Application



The Qt software on the embedded system drives the human machine interface of the embedded device. The touch screen allows the user to interact with the GUI and the GUI offers receptive feedback to the user. This receptive feedback is exemplified by displaying error messages, rendering button animations to indicate button presses and system messages explaining the status of the transfer. The desktop software also has an option to be run via the command line. This eliminates the need to install the PySide2 library on a host device if a user does not want to use it.

Software Source Code Distribution

In order to effectively and easily distribute the mGRUE software to both Dr. Zola and the rest of the world, we created a github organization that contains the repositories for both the mGRUE embedded software and the Host Device Driver. The relevant Github pages are linked below:

- UB-mGRUE Github organization: https://github.com/UB-mGRUE
- mGRUE embedded software repository: https://github.com/UB-mGRUE/m-grue
- mGRUE host device driver repository: https://github.com/UB-mGRUE/m-grue-driver
- mGRUE project documentation: https://github.com/UB-mGRUE/m-grue-documentation

Human Machine Interface (HMI)

As previously stated, the mGRUE team depended on the **3.5” LCD Touch Screen Panel** to facilitate the interaction between the user and the embedded device. The embedded device is responsible for the function of the serial communication process. The user selects the specific DNA sequence file from its storage, the user is responsible for starting, stopping and pausing the serial communication. The low form size of the Raspberry Pi 4 makes this an adequate emulation substitution for the minION DNA sequencer device.

Design Constraints

In terms of design constraints, for both the mGRUE emulation software and the desktop application, we were mitigated to a specific OS. The first reason is because the operating system that the desktop application will be used by the current SMARTEn system will be run on the Ubuntu operating system. While being constrained by the OS, we are constrained by permissions of the system. For example, on the Nvidia Jetson, we will have access to the apt package manager command, but it will be difficult to build Qt for the system without having full restricted access to the OS.

On the side of the emulation device, we were constrained by the OS support of the **3.5” LCD Touch Screen Panel.** This is exemplified by the fact that drivers are only supported by specific operating systems and to use the touchscreen panel, it requires a touchscreen calibration settings which are OS specific as well. This substantially increases the time spent refactoring the calibration settings and drivers to work on the operating system of our choice. This is suboptimal, considering there is also a deadline for the system and time spent refactoring can better aid in user experience.

System Performance

The mGRUE system functions as previously described in the introduction. The mGRUE embedded software successfully runs on the device and successfully facilitates the serial communication. The mGRUE desktop application successfully reads the data sent over serial and parses the data onto its filesystem. The mGRUE embedded application allows the user to speed up, slow down, pause and stop the transmission process.

Transfer Speed Metrics		
Level 5	14.6 MB/s ~ 5500 rps	100k records in 18 seconds
Level 1	7.95 MB/s ~ 3000 rps	100k records in 33 seconds
Level 3	5.30 MB/s ~ 2000 rps	100k records in 50 seconds
Level 2	2.65 MB/s ~ 1000 rps	100k records in 100 seconds
Level 1	1.33 MB/s ~ 500 rps	100k records in 200 seconds

*rps = DNA sequence records per second

**Integration testing has revealed that the mGRUE device can only transfer DNA sequence data at about half the speed shown in the above table when connected to Dr. Zola's NVIDIA Jetson. The NVIDIA Jetson likely is not supplying enough power to the mGRUE which causes it to underperform.

Future Considerations

In the future, for a subsequent iteration of our mGRUE system, we would advocate getting a LCD with better driver and calibration support in the future. The constraint with using the driver is not detrimental to the experience of the user, however we have less control over the optimization strategies on the operating system level. For example, we are mitigated to using the Rasbian operating system, yet the operating system's installation size and pre-installed binaries are redundant for our mGRUE system.

Points of Contact

- Jonathan Baffo - jmbaffo@buffalo.edu
- Warren Green - wogreen@buffalo.edu
- Christian Palladino - cpalladi@buffalo.edu
- Brian Scorcia - briansco@buffalo.edu
- Aaron Siegel - aaronsie@buffalo.edu
- Lucas Simpson - lucassim@buffalo.edu