University of Nevada, Reno

Serial Logger for Offshore Data Collection

A thesis submitted in partial fulfillment
of the requirements for the degree of
BACHELOR OF ENGINEERING, COMPUTER
SCIENCE AND ENGINEERING

By
Brian Goga

Sergiu Dascalu, Ph.D, Internal Thesis Advisor
Trevor Kavanaugh, BS, External Thesis Advisor

May, 2016

UNIVERSITY

OF NEVADA **THE HONORS PROGRAM**
RENO

We recommend that the thesis
prepared under our supervision by

BRIAN GOGA

entitled

SLODC: Serial Logger for Offshore Data Collection

be accepted in partial fulfillment of the
requirements for the degree of

BACHELORS OF ENGINEERING

Dr. Sergiu Dascalu, Computer Science and Engineering

Tamara Valentine, Ph.D., Director, Honors Program

May 2016

Table of Contents

Abstract

Mechanical parts in machinery are prone to vibration damage. Software is often used to keep track of data regarding the frequency and magnitude of these vibrations. General Electric currently has to bring out expensive equipment to log and convert the data from the software into relevant and readable data, risking the equipment in the process.

The goal of this project was to use a Renesas Embedded GUI board to create a device that no longer requires bringing out expensive equipment to read the data. The board logs the data and makes it possible for the information to be saved onto a USB stick. The data then is sent to a remote computer and converted into usable data.

Introduction

One of the biggest problems with any machinery that moves, is vibrations. Vibrations can be very dangerous since they can cause cracks in the machinery and can cause parts of the machine to come loose. These parts may cause serious damage to the structure that houses the machine or to personal nearby. In order to prevent these things from happening measurements of the vibrations can be made; but this has its challenges. Equipment size, environmental factors, etc. can make it difficult to get an accurate reading, or a reading at all. One place this is very apparent is on an offshore oil rigs.

The Serial Logger for Offshore Data Collection (SLODC) measures the vibrations and saves the data so it can be examined later. It is secure and works on most of the General Electric machines; while also making sure the data is in the correct format so employees can use it. The reason the SLODC is better than what is currently used to measure the vibrations is because it is small, inexpensive, and portable; it is easily moved from one machine to another, if conditions get too severe it can be stored, and it can be replaced if it were to become damaged or lost.

Over the course of this project several changes and choices were made to produce SLODC as a product. First, only one of the serial ports on the device will be used. It has also been determined that modifications to the board may be necessary to fully connect the top half of the board's serial port to the bottom half of the board's USB port. Second, it has been decided that the data will be stored on the board's local memory via bulk transfer methods instead of control or interrupt based methods. Next, the project will use a program to generate serial serial communication for the purpose of debugging and demoing. Lastly, Renesas has advised alternate compilation techniques be used in order to get around licensing issues that have occurred when using their IDE which prevent compilation and debugging.

Requirements

Requirements Workshop Summary

The entire team met with Trevor Kavanaugh on the 2nd of March in the Knowledge center at UNR to capture requirements for the project. This brainstorming session lasted for a little over an hour and had Tarrayna acting as an engineering technician who would be used to operate the device. The first half of the brainstorming session focussed on what the device should or could do, while the second half focused on ideas for the embedded device's user interface. From this session, the most important points discussed were that the device could read and categorize serial data information and that the device was as easy to use as possible. Secondary concerns from the session included showing users the progress of SLODC as it was working and possible extensions towards increased connectivity through bluetooth and an app or through ethernet. The least important requirements generally involved follow up work for the project which included security features, code style (convert to a more readable C++ style of coding as opposed to C for readability), and automatically update time through an internet connection.

The operational requirements agreed upon are as follows. Serial data should be labeled as being incoming or outgoing and have a timestamp for easier referencing. The timestamped data can be stored either byte by byte or through packets. SLODC should also be relatively configurable; mainly for different baud rates of data transfer and for different handshaking requirements. That would help the device to be as one-size-fits-all as possible.

The user interface requirements agreed upon are as follows. Each main module should have a separate screen for the user. Limit each screen to have only a handful of options due to the size of the screen and legibility of the device. Display data progress through a progress bar as data is being spooled or read in so that the user knows the machine is still working. The device should also notify the user when it is complete or an error has occurred. (This was probably the biggest point since errors could develop from reading, writing, or failure to detect USB or serial devices.)

Software Requirements

Functional Requirements:

Priority 1 Requirement

F01	SLODC shall warn users when overriding previous data
F02	SLODC shall allow users to set/update date and time
F03	SLODC shall let the user know when importing/exporting data is complete
F04	SLODC shall timestamp data that is imported
F05	SLODC shall allow users to select features using touch screen
F06	SLODC shall show the user progression of importing/exporting data
F07	SLODC shall show the user when an error has occurred

Priority 2 Requirements

F8	SLODC shall allow users to configure serial ports

F9	SLODC shall show the user instructions on how to use system
F10	SLODC shall allow the user to stop/restart importing/exporting data
F11	SLODC shall identify the direction data is traveling as incoming or outgoing data

Priority 3 Requirements

F12	SLODC shall protect data by having users log in to use the system
F13	SLODC shall allow user to show importing data on screen
F14	SLODC shall create an error correction system to handle noise in the imported data

Non-Functional Requirements:

Priority 1 Requirements

NF01	SLODC shall be written in C
NF02	SLODC shall save data to USB stick
NF03	SLODC shall have interrupt priorities. First for cancel. Second for getting info to/from the server.Third for display
NF05	SLODC shall have a simple UI
NF06	SLODC shall use the Renesas Embedded GUI Board
NF07	SLODC shall use the Micrium real time operating system

Priority 2 Requirements

NF08	SLODC shall format data to CSV
NF09	SLODC shall respond to user inputs within 0.5 seconds

Priority 3 Requirements

NF09	SLODC shall send data over ethernet
NF10	SLODC shall send data over bluetooth-to-phone

Hardware Requirements

Functional Requirements:

Priority 1 Requirement

F01	SLODC shall warn use the Micrum operating system
F02	SLODC shall allow use e^2 Studios

Non-Functional Requirements:

Priority 1 Requirements

NF01	SLODC shall use the USB port to export data
NF02	SLODC shall use the touch screen to get input from the user
NF03	SLODC shall use the serial ports to read data from the machines

Design Models

Architecture Design

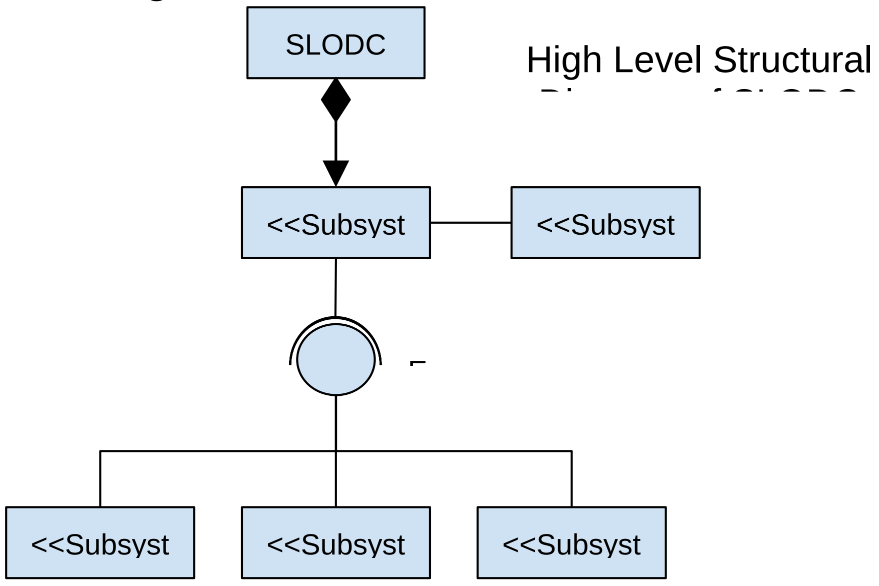


Fig. 1 Structural Diagram for SLODC

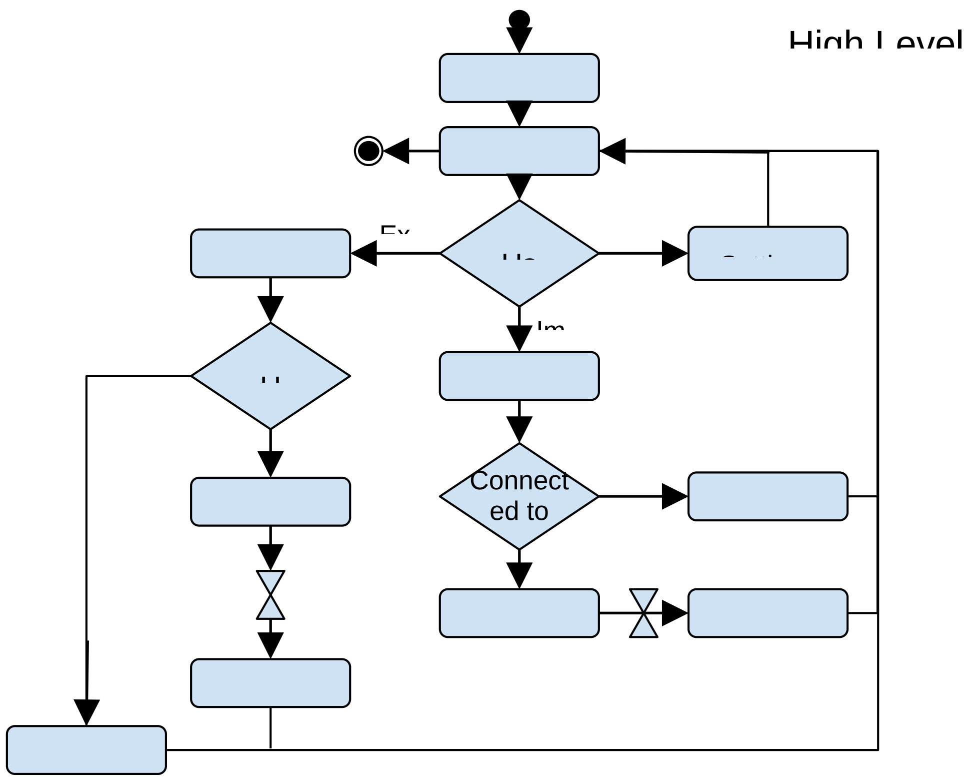


Fig. 2 Behavior Diagram for SLODC

Class Diagrams

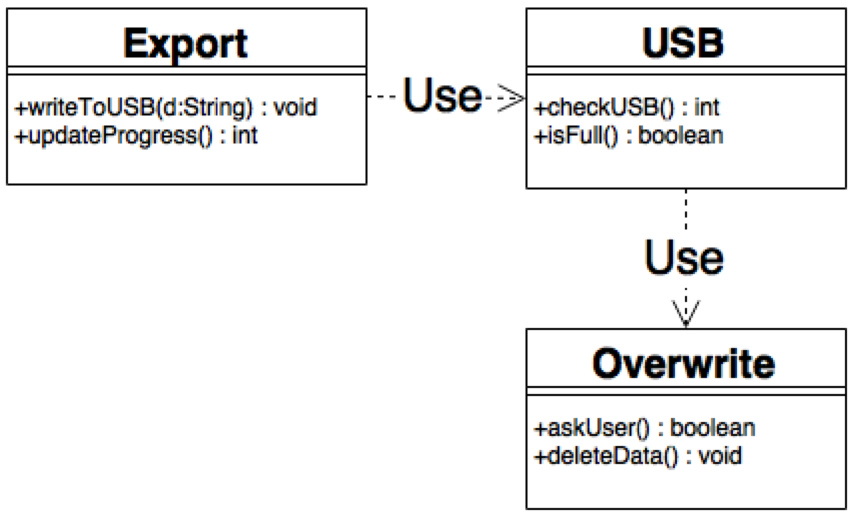


Fig.3 Class diagram for the USB subsystem

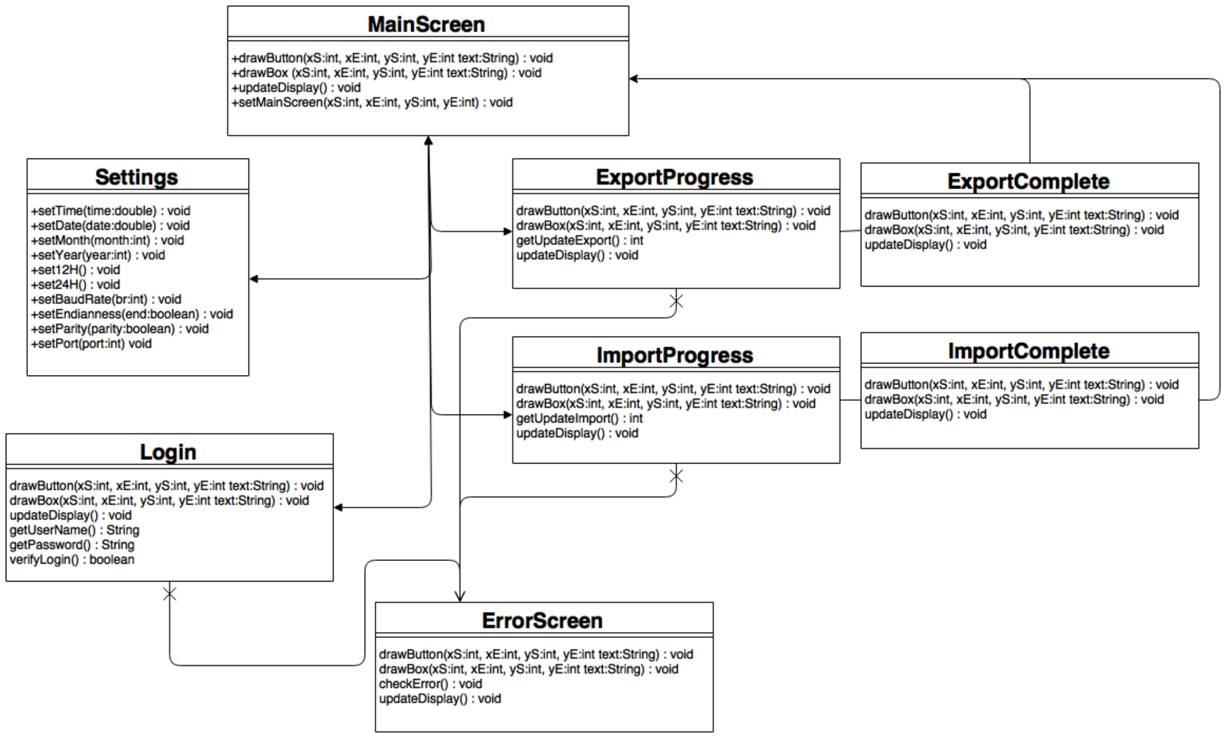

MainScreen
+drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
+drawBox (xS:int, xE:int, yS:int, yE:int text:String) : void
+updateDisplay() : void
+setMainScreen(xS:int, xE:int, yS:int, yE:int) : void
Settings
+setTime(time:double) : void
+setDate(date:double) : void
+setMonth(month:int) : void
+setYear(year:int) : void
+set12H() : void
+set24H() : void
+setBaudRate(br:int) : void
+setEndianness(end:boolean) : void
+setParity(parity:boolean) : void
+setPort(port:int) void
ExportProgress
drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
drawBox(xS:int, xE:int, yS:int, yE:int text:String) : void
getUpdateExport() : int
updateDisplay() : void
ExportComplete
drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
drawBox(xS:int, xE:int, yS:int, yE:int text:String) : void
updateDisplay() : void
ImportProgress
drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
drawBox(xS:int, xE:int, yS:int, yE:int text:String) : void
getUpdateImport() : int
updateDisplay() : void
ImportComplete
drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
drawBox(xS:int, xE:int, yS:int, yE:int text:String) : void
updateDisplay() : void
Login
drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
drawBox(xS:int, xE:int, yS:int, yE:int text:String) : void
updateDisplay() : void
getUserName() : String
getPassword() : String
verifyLogin() : boolean
ErrorScreen
drawButton(xS:int, xE:int, yS:int, yE:int text:String) : void
drawBox(xS:int, xE:int, yS:int, yE:int text:String) : void
checkError() : void
updateDisplay() : void

Fig.4 Class diagram for the User Interface subsystem

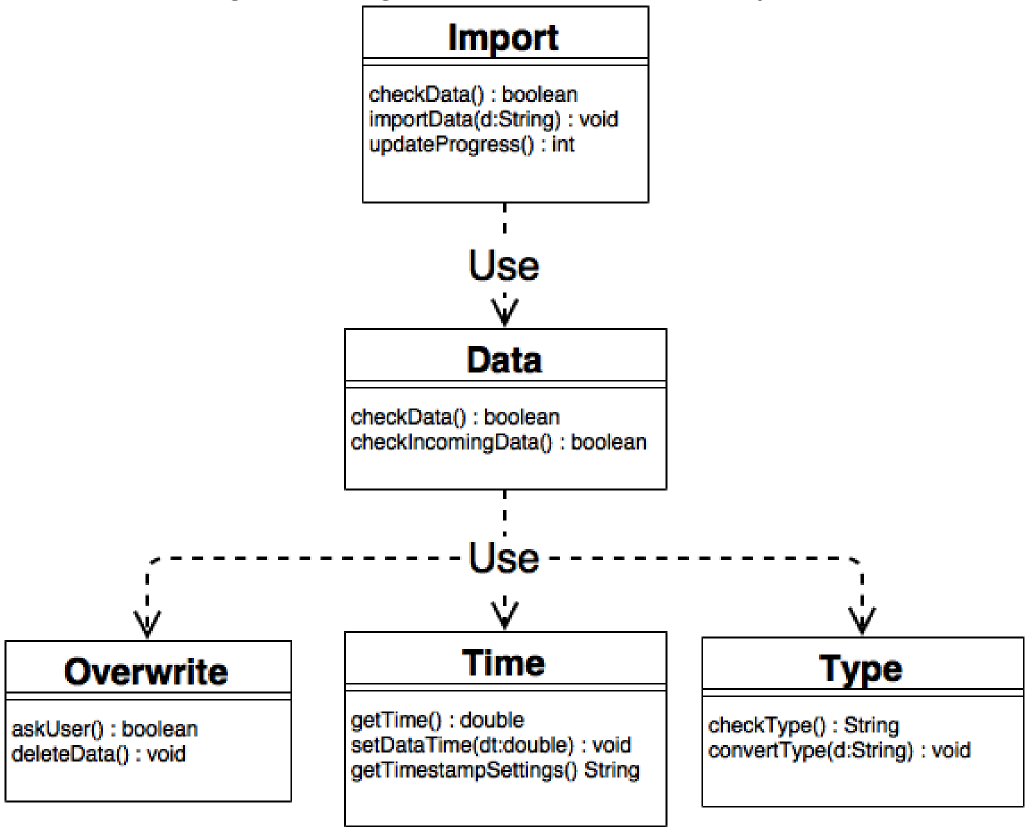


Fig.5 Class diagram for the Data subsystem

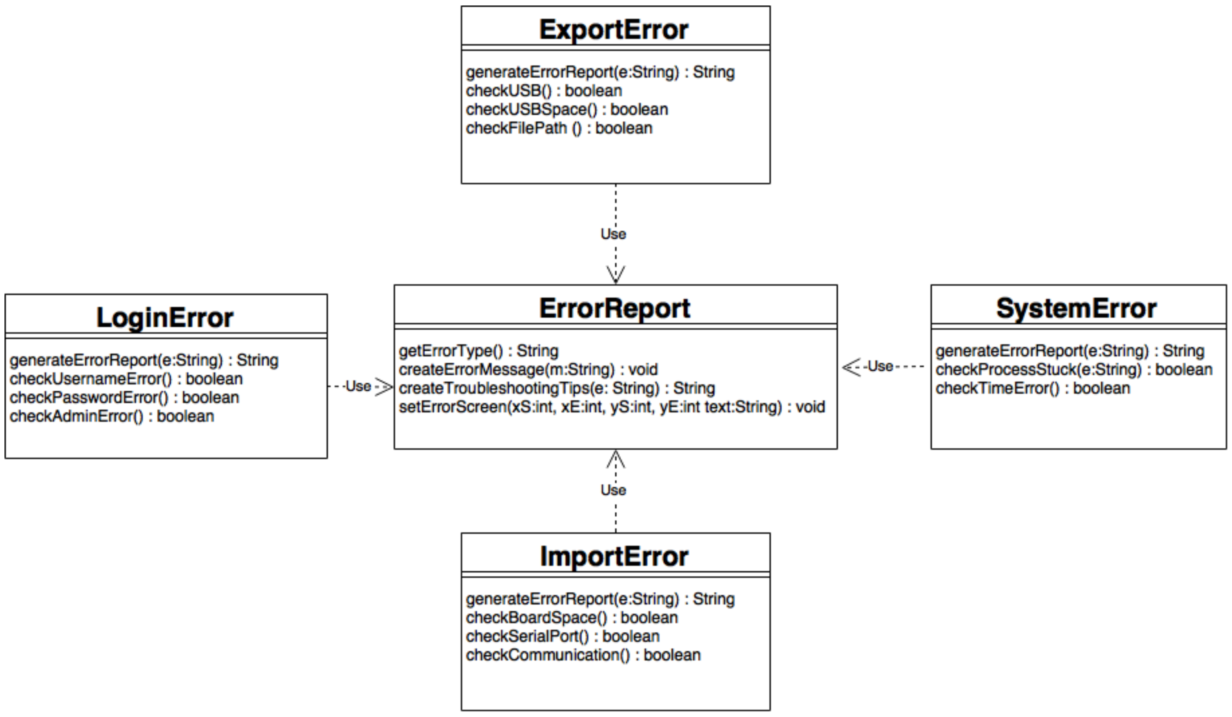

ExportError
generateErrorReport(e:String) : String
checkUSB() : boolean
checkUSBSpace() : boolean
checkFilePath () : boolean
Use
LoginError
generateErrorReport(e:String) : String
checkUsernameError() : boolean
checkPasswordError() : boolean
checkAdminError() : boolean
Use
ErrorReport
getErrorType() : String
createErrorMessage(m:String) : void
createTroubleshootingTips(e: String) : String
setErrorScreen(xS:int, xE:int, yS:int, yE:int text:String) : void
Use
SystemError
generateErrorReport(e:String) : String
checkProcessStuck(e:String) : boolean
checkTimeError() : boolean
Use
ImportError
generateErrorReport(e:String) : String
checkBoardSpace() : boolean
checkSerialPort() : boolean
checkCommunication() : boolean

Fig.6 Class diagram for the Error subsystem

Program Units

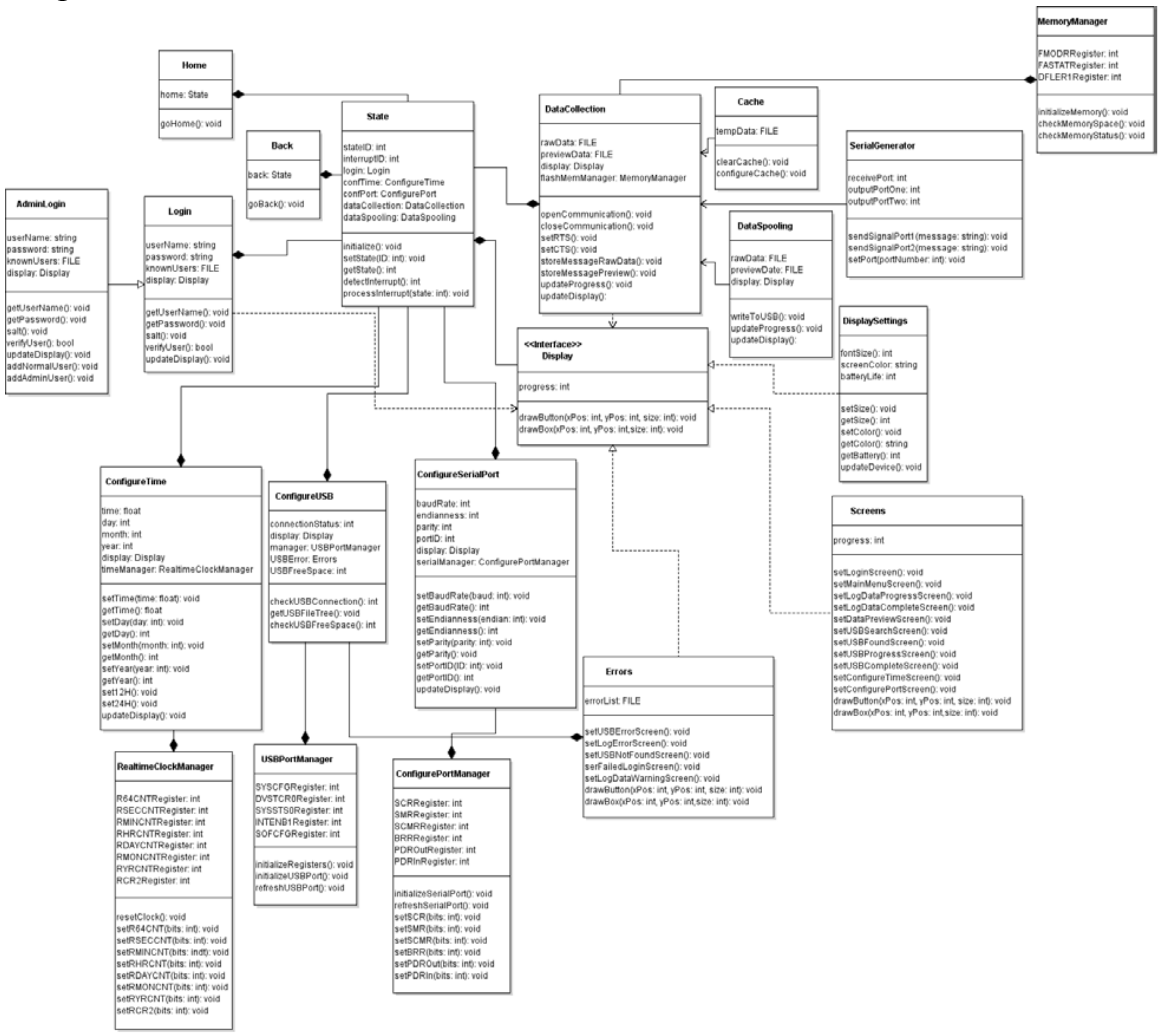


Fig 7. Program Units of SLODC

Methods:

Method	**Description**
getUserName	Used to acquire the username the user has input into the device.
getPassword	Used to acquire the password the user has input into the device.
verifyUser	Verifies the user based on the username and

	password that was acquired from getUserName and getPassword.
detectInterrupt	This will be constantly checking for an interrupt from the user.
processInterrupt	This will determine the priority of the interrupt that was detected, and execute the interrupt based on its priority.
setMonth	This will set the month so that it can be used for the timestamp for the collected data.
setRTS	This will send an RTS signal between SLODC and the machine to verify that data transfer can occur
setCTS	This will send a CTS signal between SLODC and the machine to begin data transfer once both parties are ready
storeMessageRawData	This will store the raw data gathered.
setBaudRate	This will allow serial ports to be configured to communicate across different baud rates.
salt	This will help us further encrypt the passwords, keeping our system secure.
setEndianness	This will allow serial ports to be configured to accept data packets as either big endian or little endian.
setParity	This will allow serial ports to be configured to accept either even or odd parity bits for data transfer error detection.
setPortID	This will set the serial port that is used to communicate with the machine.
writeToUSB	This will write the stored data onto the USB.
setState	This will update the SLODC system state based on user input. Each state will only allow certain actions to prevent race conditions.
updateDisplay	This will update the display as new information needs to be displayed.
openCommunications	This will open communications to be able to

	sniff transmitting data.
closeCommunications	This will close the communications so the device doesn’t try to sniff data that it doesn’t have the ability to.
updateProgress	This will update the progress display when transferring data to the USB.
drawButton	This will draw a button on the display with a given location and size.
drawBox	This will draw a Box on the display with a given location and size.
setMainMenuScreen	This will set the Main Menu Screen on the display
setLogErrorScreen	This will set the Error that will be displayed when there is a log error
setUSBErrorScreen	This will set the Error that will be displayed when there is an USB error
setYear	This will set the year so that it can be used for the timestamp for the collected data.
goHome	Returns the user to the main menu screen.
getState	Determines what state the program is in.
storeMessagePreview	Displays information colled from data collection process.
clearCache	Clears the cache before data collection to create space for future data storage.
configCache	Configures the memory space used to store data
setSize	Sets the dimensions of the objects being displayed on the screen.
setColor	Sets the color of the objects being displayed on the screen.
setTime	This will set the time so that it can be used for the timestamp for the collected data.
setDate	This will set the date so that it can be used for the timestamp for the collected data.

setUSBNotFoundScreen	Displays warning to user that USB is not connected.
setFailedLoginScreen	Displays warning to user that login credentials were incorrect.
setLogDataWarningScreen	Displays warning to user that data logging is taking place and that devices need to remained plugged in.
setLogCompleteScreen	Displays success screen after data logging is complete.
setUSBCompleteScreen	Displays success screen after data export is complete.

Detailed Design

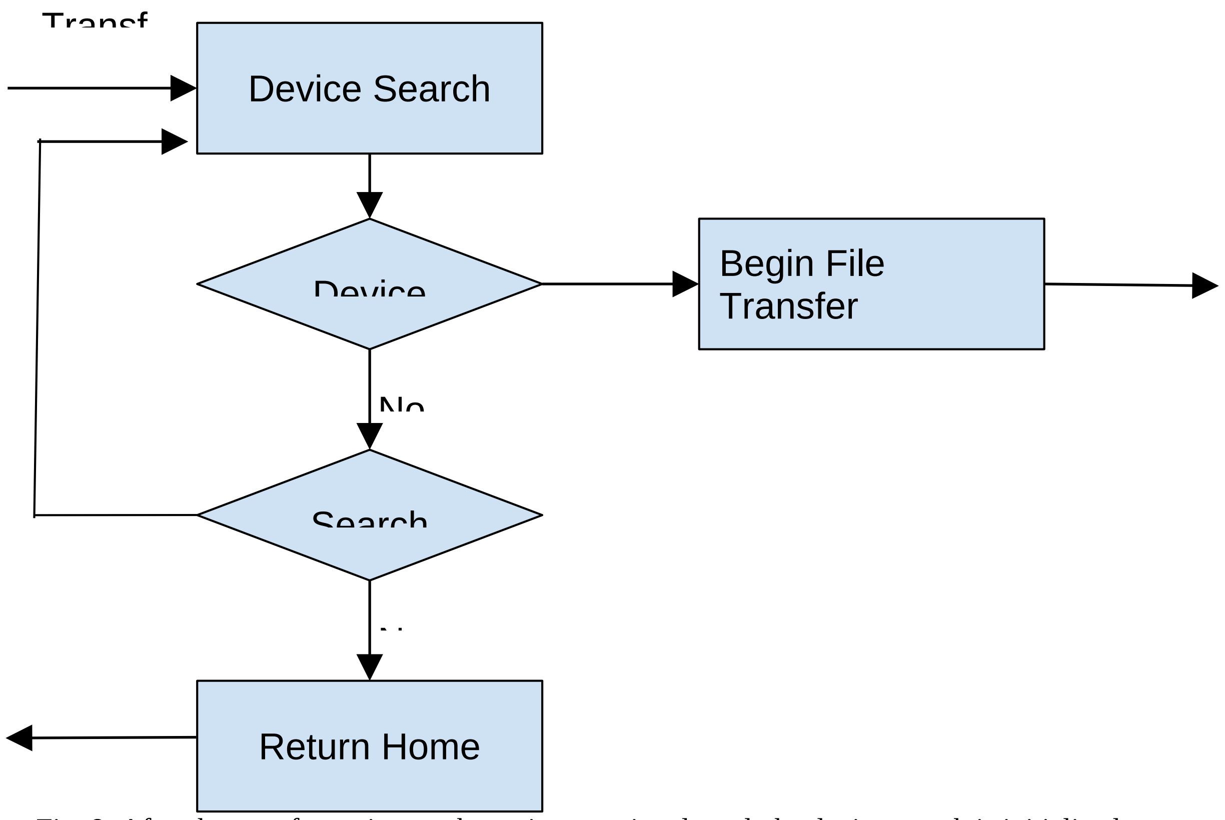


Fig. 8: After the transfer option on the main menu is selected, the device search is initialized.

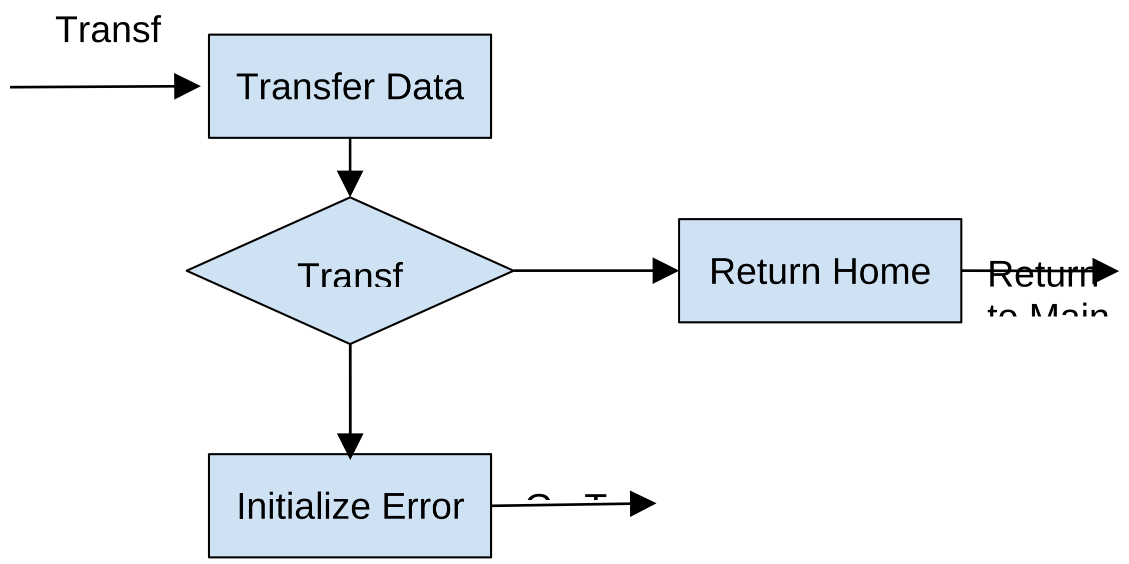


Fig. 9: When a device is successfully found after the search. The Transfer protocol is initialized.

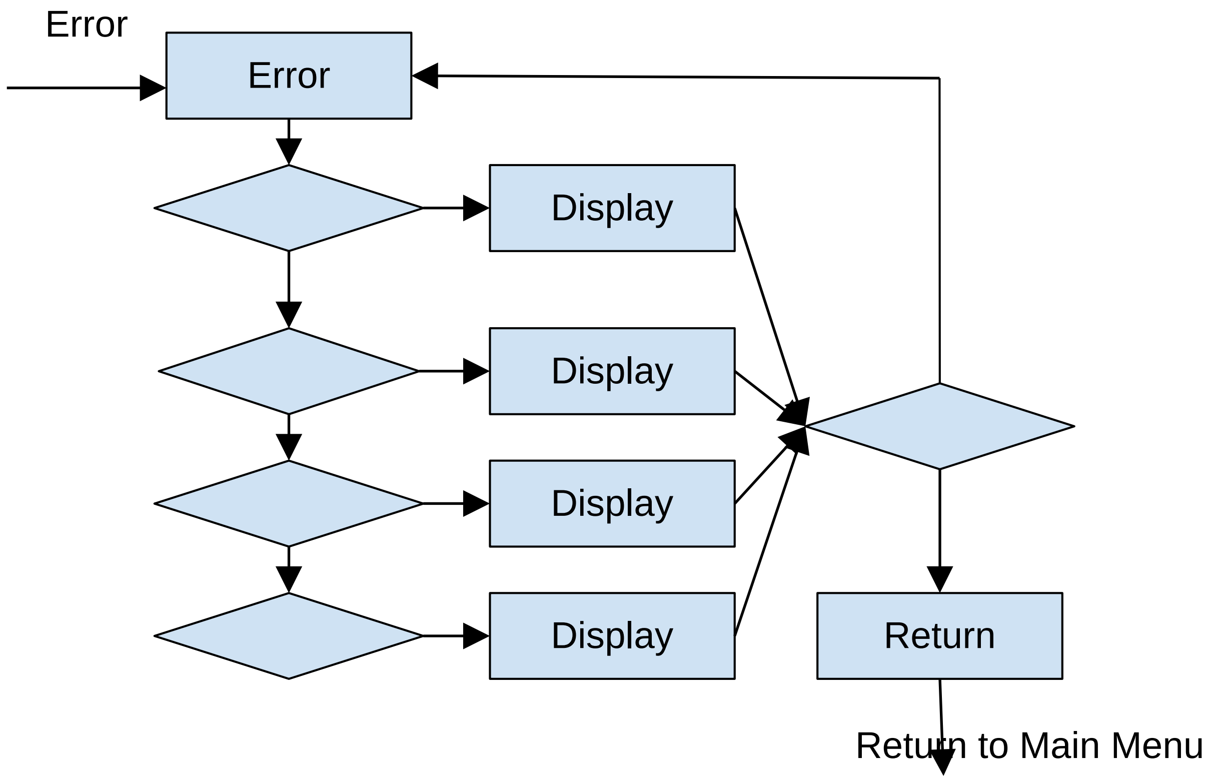


Fig. 10: Shows the process the device follows when an error is detected during file transfer.

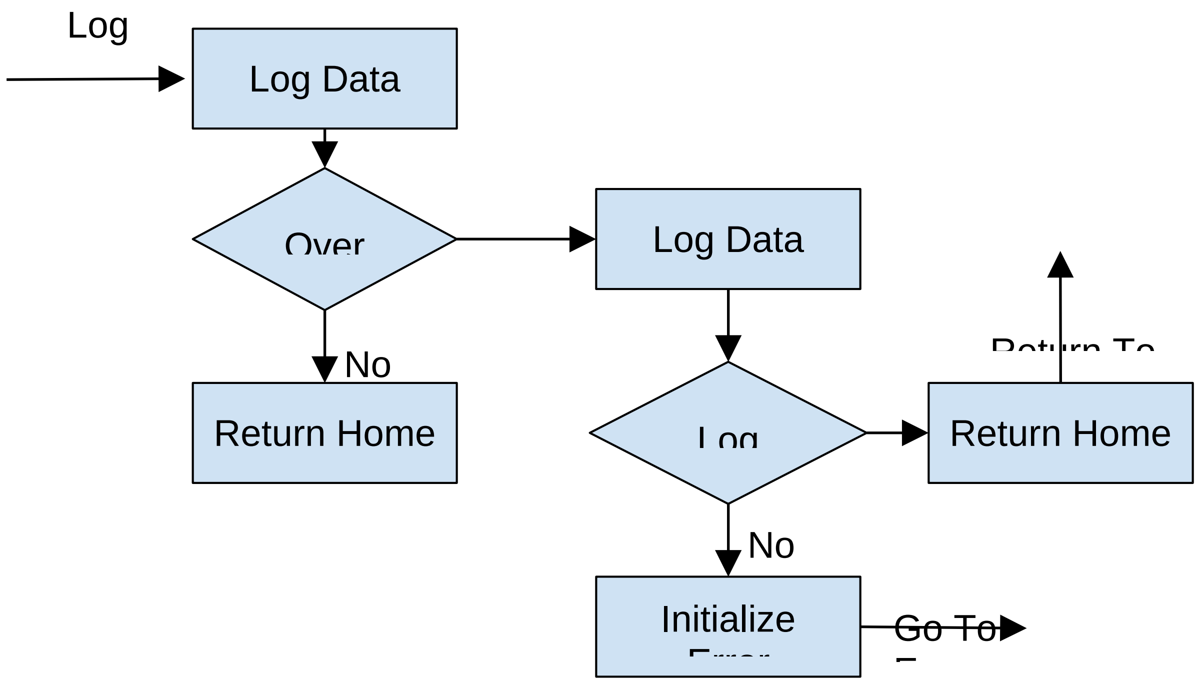


Fig. 11: Shows the process the program follows when Log is selected at the main menu.

Data Design

Data Structure:	**Purpose:**
Back	This class allows for the user to return the device to the previous page. For example the user want to return to the previous menu after finishing data logging.
Home	This class allows users to return to the home screen.
State	This class allows for the device to be put into a specific state. This can vary from login, data transfer to data import.
Data Collection	This class allows for the data collection to take place. Data is stored in the cache.
Cache	This class allows for the data storage to occur in the cache. It contains operations to modify saved data.
Data Spooling	This class allows for the data to be collected and stored into the cache.
Login	This class allows for users to login to the system for security purposes.
Admin Login	This class allows for admins to add other normal users and admins to the

	system
Display	This class allows for objects to be added to the screen for display purposes.
Display Settings	This class allows for manipulation of the Display so that objects are more readable.
Configure Time	This class allows for there to be a time used by the the device for timestamping data imports.
Configure USB	This class allows for the USB device to be used and for the USB’s file path to be displayed
Configure Serial Port	This class allows for the ports that will be used by the device to be configured.
Screens	This class takes advantage of the Display and determines what will be shown during different states.
Errors	This class allows for errors to be displayed to the user as well as methods to correct errors.
Memory Manager	This class initializes the flash memory of the board for read/write use and provides information about the free memory available.
Realtime Clock Manager	This class is used to reset the clock (if necessary) and keep track of all the different timekeeping registers.
USB Port Manager	This class initializes the USB port to allow SLODC to transfer data to a USB memory device. It also contains the means to refresh the connection to check for devices that may have been plugged in before SLODC was turned on.
Serial Port Manager	This class initializes the serial port. It also updates the registers to allow for different serial port configurations.
Serial Generator	This class contains functions to send data from a PC’s USB ports to the embedded device via the serial port.

User Interface Design

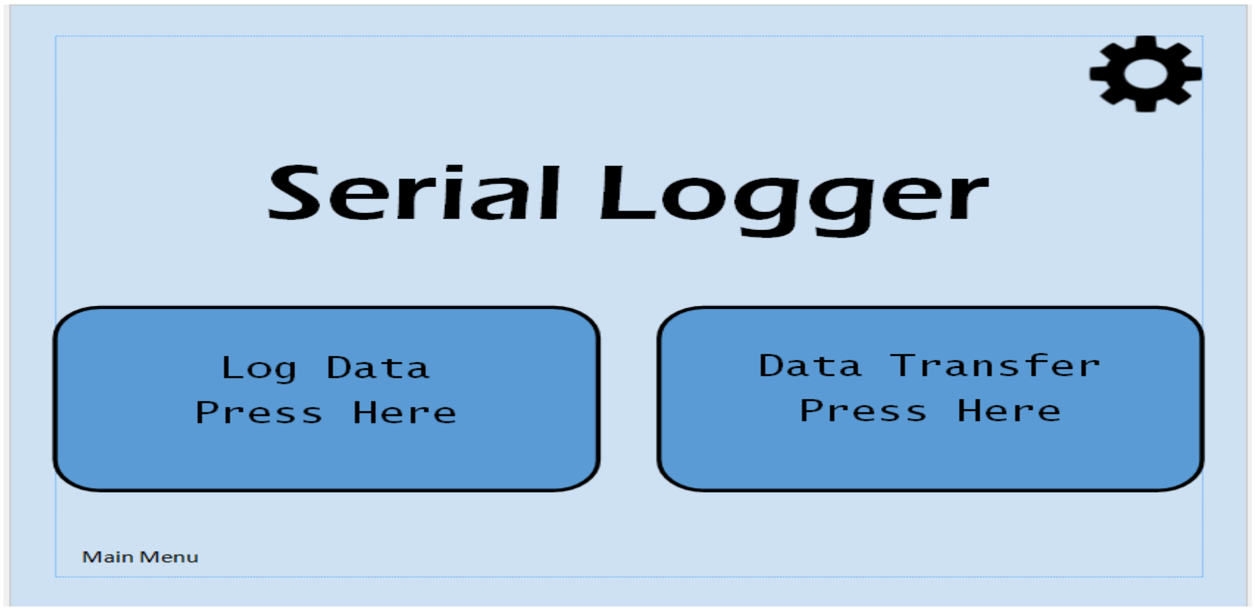


Fig. 12: This will be what the menu screen for the serial logger will look like. The gear will give the user access to settings for this application.The left rectangle will initiate the logging phase when pressed. The right rectangle will initiate the data transfer phase when pressed.

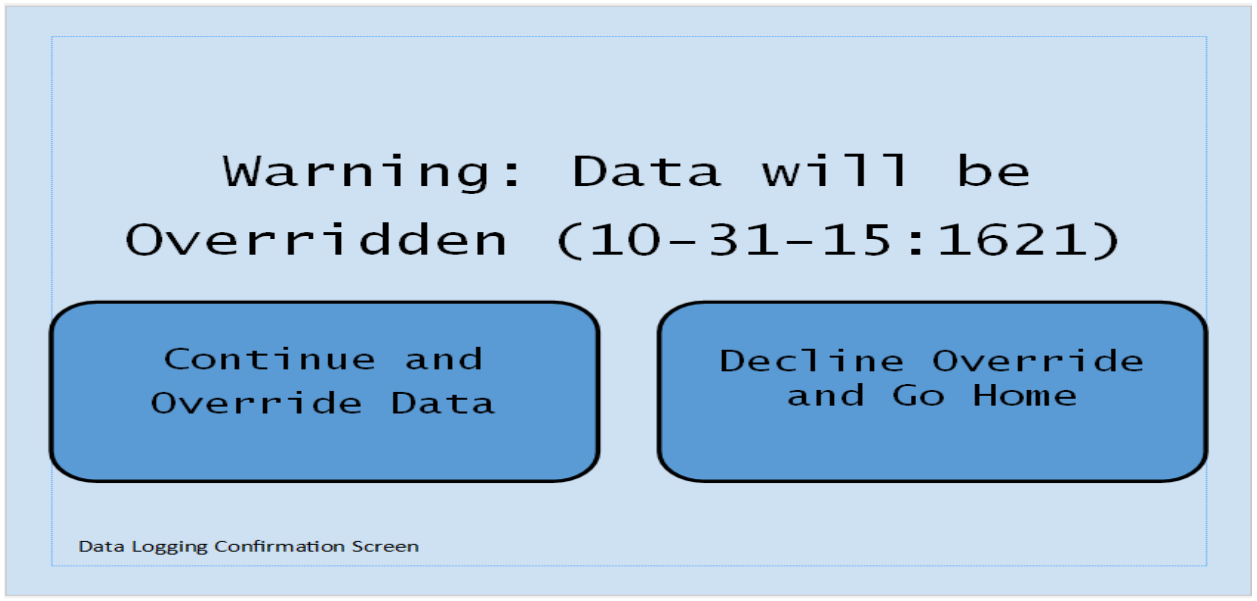

Warning: Data will be Overridden (10-31-15:1621)
Continue and Override Data
Decline Override and Go Home
Data Logging Confirmation Screen

Fig. 13: This screen will appear when the user chooses to log new data. The date and time of the most recent log will be displayed and warn users it will be overridden. This is because the device will only hold a single log at a time. The left rectangle will override the data and start logging when pressed. The right rectangle will send the user back to the menu screen.

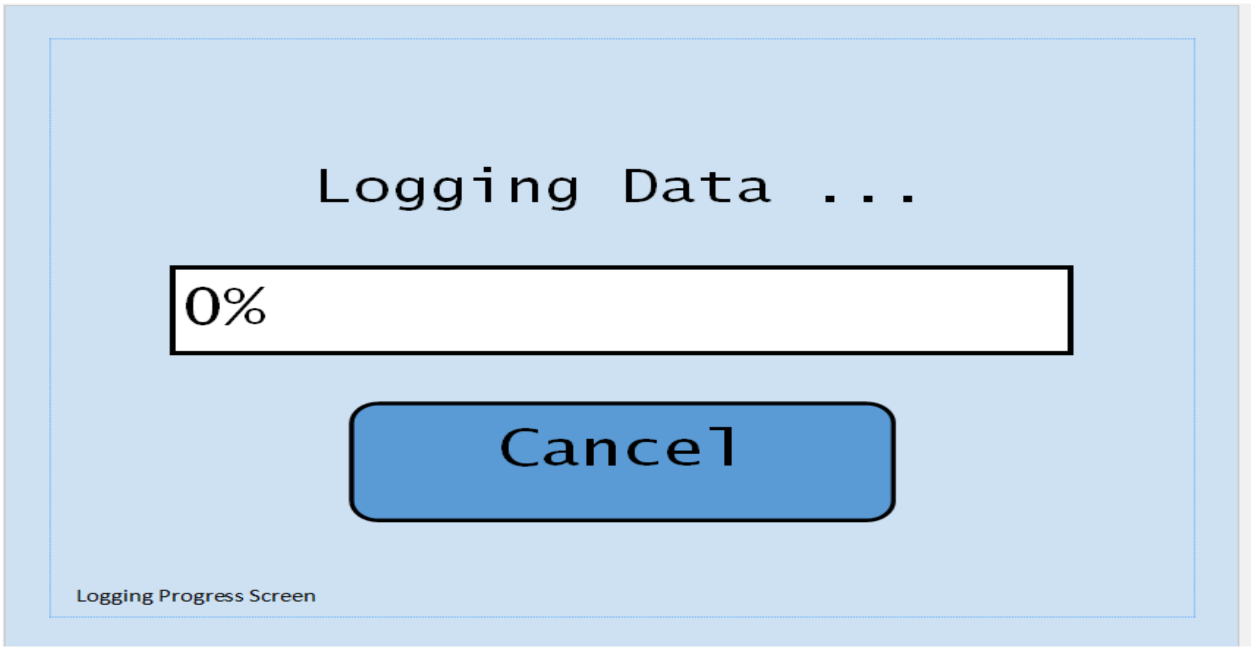


Fig. 14: This screen will show the progress of the log. The goal is to display both a progress bar and the percentage of the logging process that has been completed. The cancel button will allow the user to cancel the logging process and return them to the menu screen

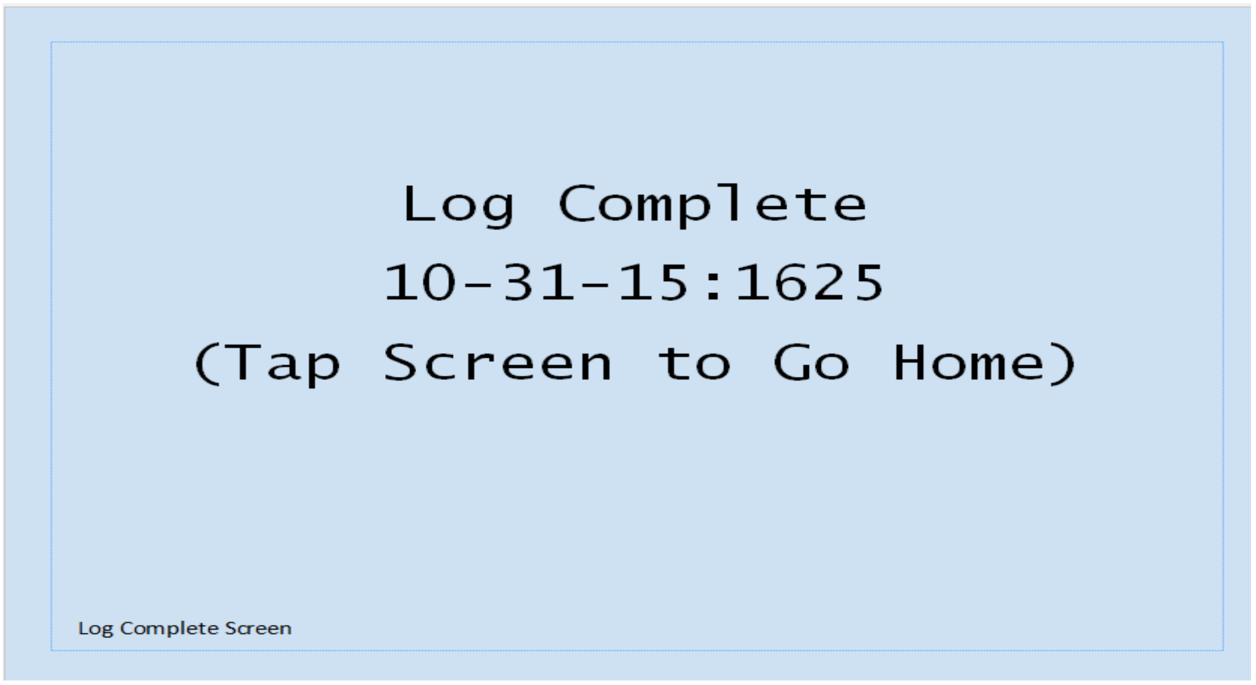


Fig. 15: This screen will display when the log is complete. The data and time of the completed log will be displayed. Tapping on the screen will bring the user back to the menu.

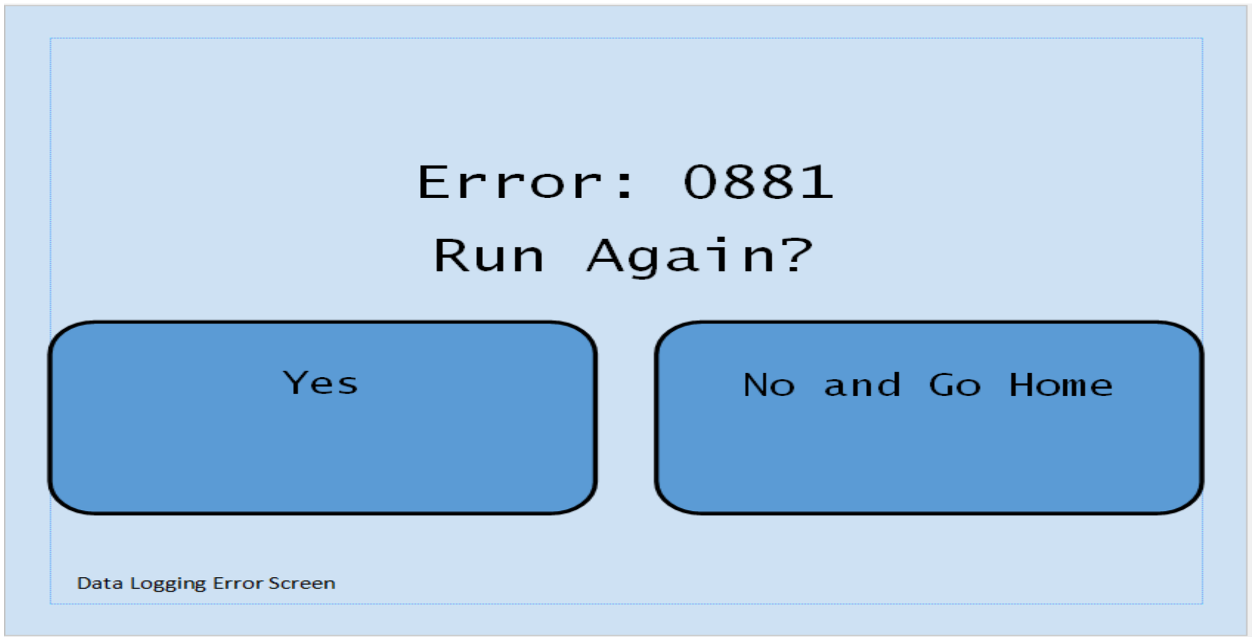


Fig. 16: If there is an error while the data is being logged this screen will display (The error screen while transferring will be identical). The error number associated with the problem will be displayed. The left rectangle will send the user back to the logging screen when pressed. The right rectangle will send the user back to the menu screen when pressed.

Fig. 17: This screen will display when the user selects the transfer rectangle at the menu screen. The goal is to add an indicator so the user isn't just looking at text when waiting for a device to be found.

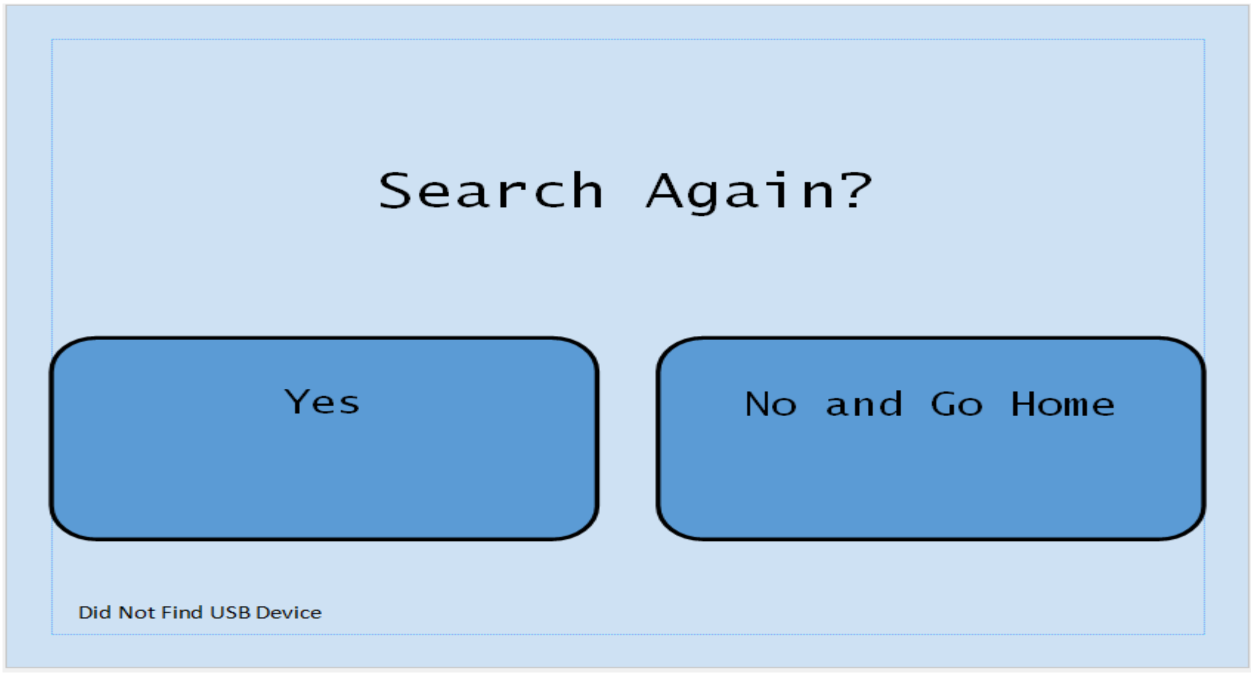


Fig. 18: If a device is not found for the data transfer the program will ask the user if they would like to search again. Pressing yes will send the user back to the search screen. Pressing no will send the user back to the menu screen.

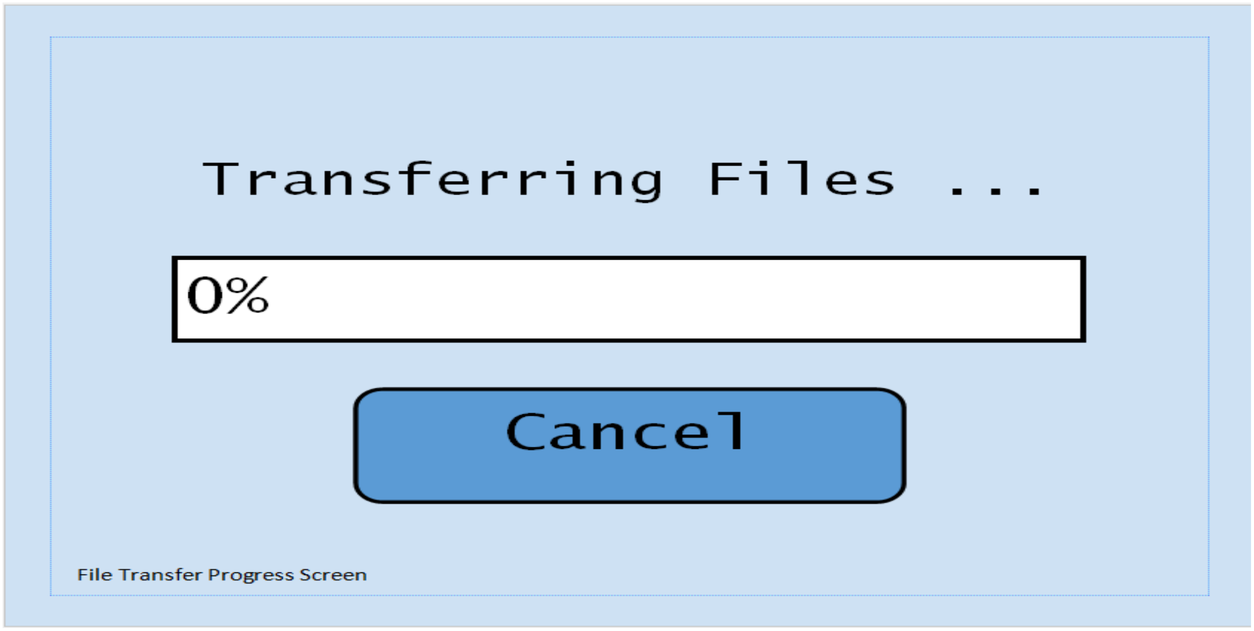


Fig. 19: This is the screen that will display after a transfer device has been successfully located. Similar to the logging screen, a progress bar and percentage is what will be displayed while the data is being transferred. The cancel button will allow the user to cancel the transfer process and return them to the menu screen

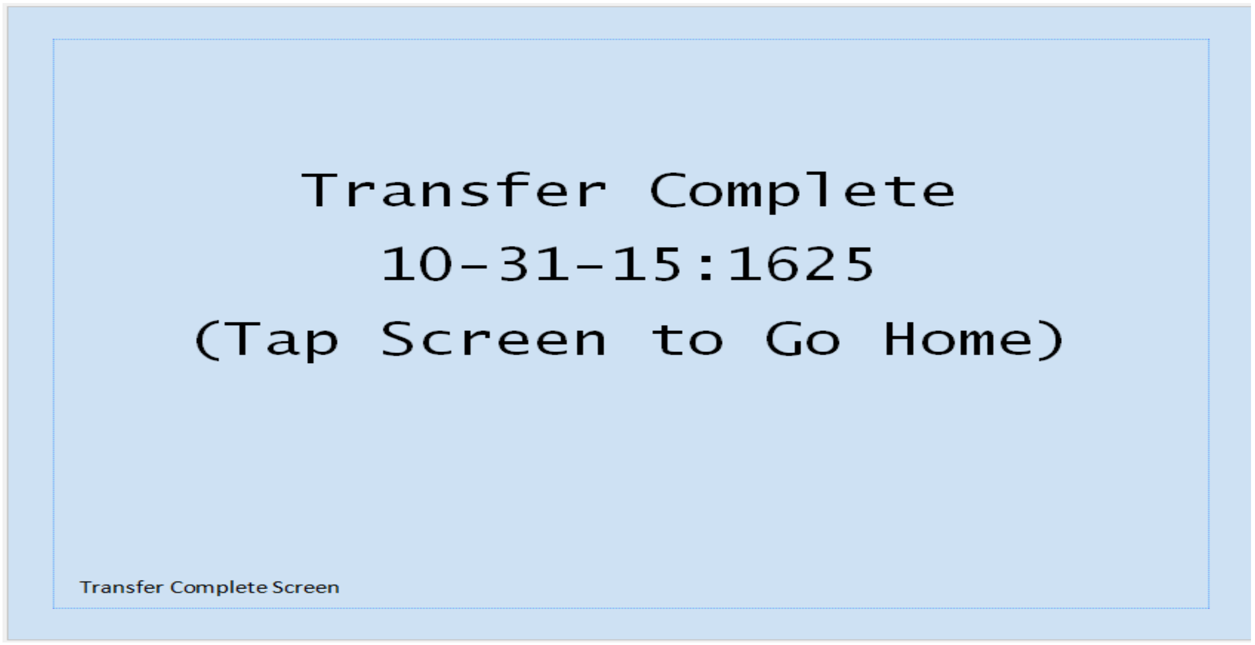


Fig. 20: This screen will display when the transfer is complete. The data and time of the completed transfer will be displayed. Tapping on the screen will bring the user back to the menu.

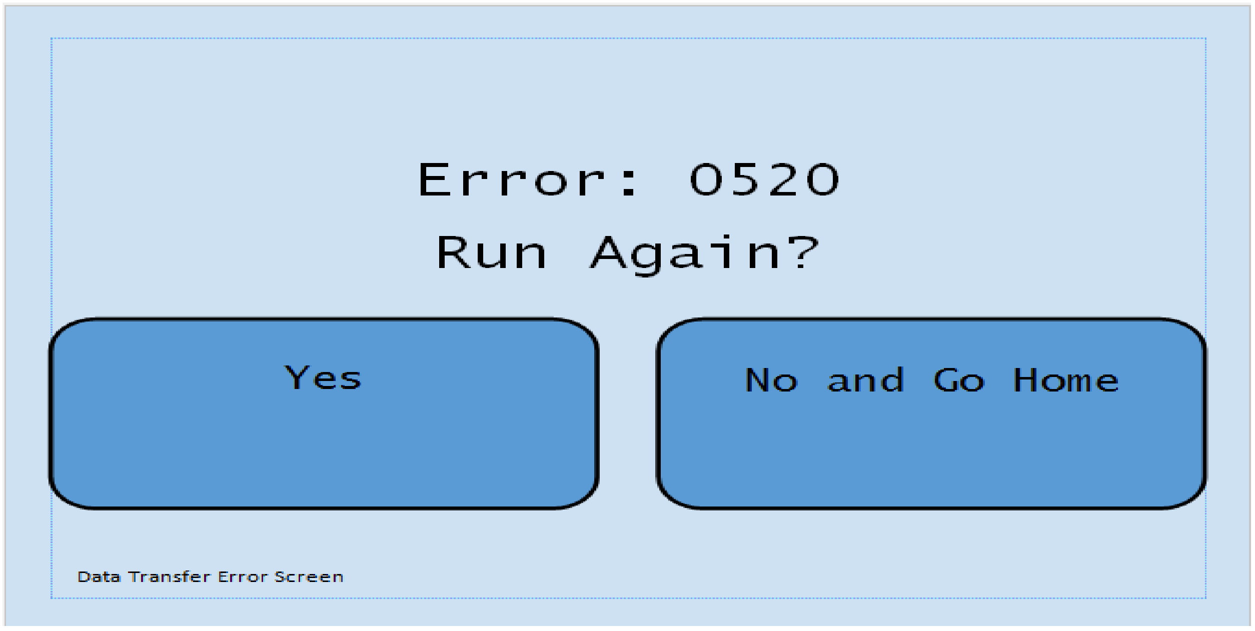
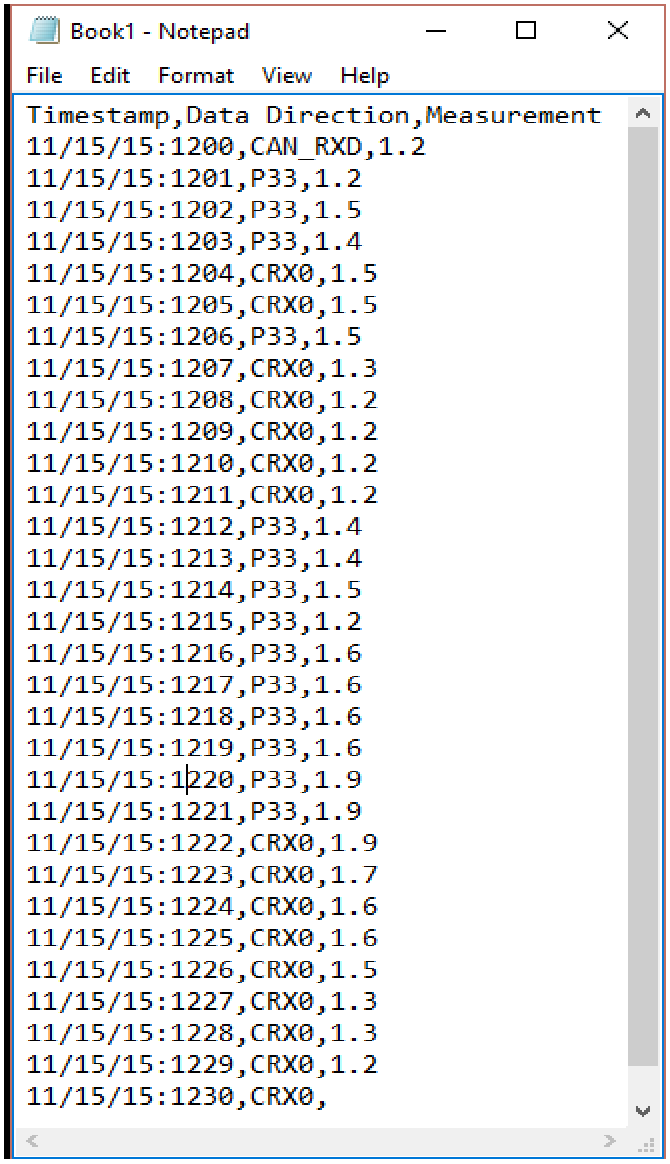
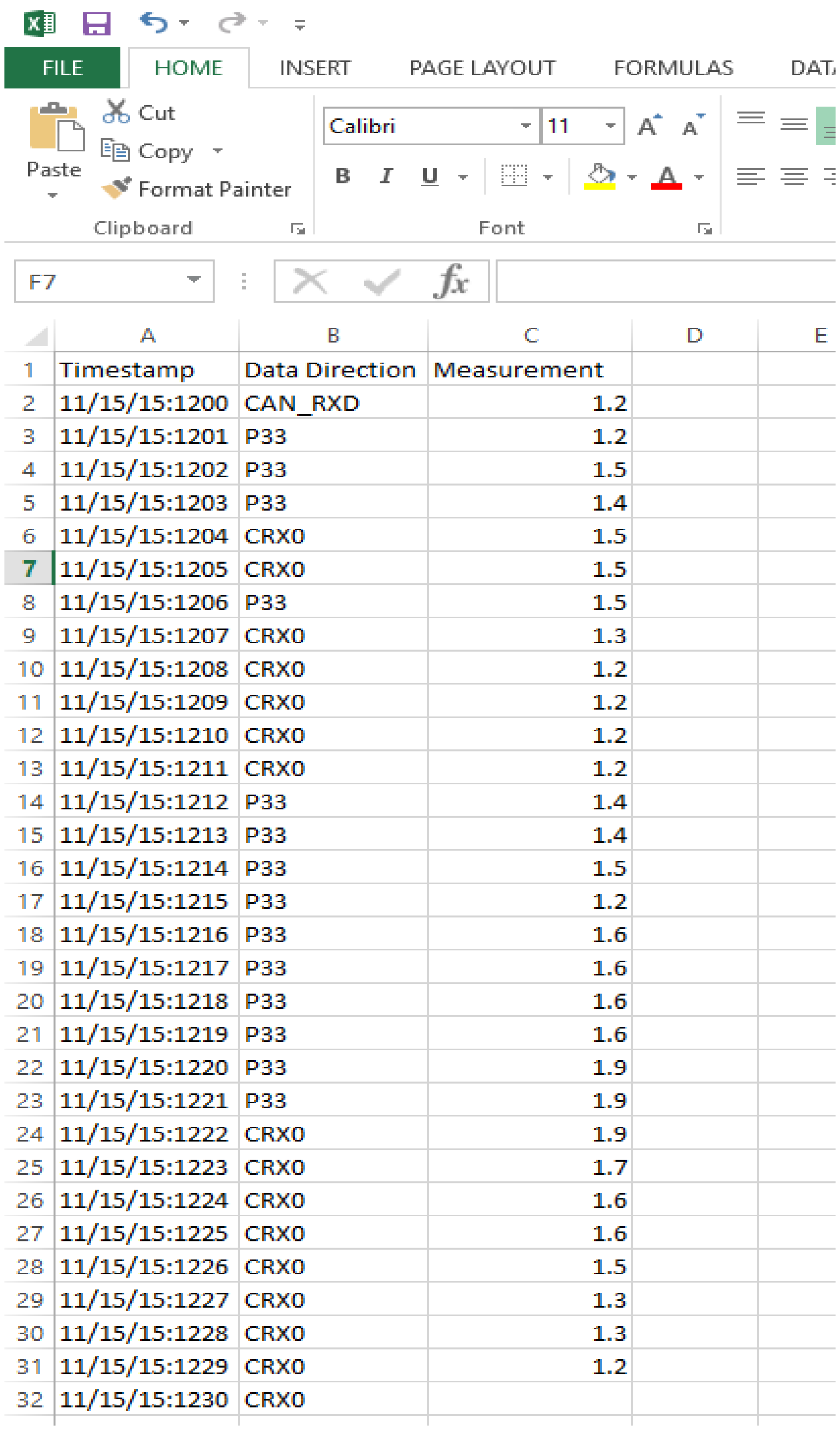


Fig. 21: If there is an error while the data is being transferred this screen will display (The error screen while transferring will be identical). The error number associated with the problem will be displayed. The left rectangle will send the user back to the transfer screen when pressed. The right rectangle will send the user back to the menu screen when pressed.



```
Timestamp,Data Direction,Measurement
11/15/15:1200,CAN_RXD,1.2
11/15/15:1201,P33,1.2
11/15/15:1202,P33,1.5
11/15/15:1203,P33,1.4
11/15/15:1204,CRX0,1.5
11/15/15:1205,CRX0,1.5
11/15/15:1206,P33,1.5
11/15/15:1207,CRX0,1.3
11/15/15:1208,CRX0,1.2
11/15/15:1209,CRX0,1.2
11/15/15:1210,CRX0,1.2
11/15/15:1211,CRX0,1.2
11/15/15:1212,P33,1.4
11/15/15:1213,P33,1.4
11/15/15:1214,P33,1.5
11/15/15:1215,P33,1.2
11/15/15:1216,P33,1.6
11/15/15:1217,P33,1.6
11/15/15:1218,P33,1.6
11/15/15:1219,P33,1.6
11/15/15:1220,P33,1.9
11/15/15:1221,P33,1.9
11/15/15:1222,CRX0,1.9
11/15/15:1223,CRX0,1.7
11/15/15:1224,CRX0,1.6
11/15/15:1225,CRX0,1.6
11/15/15:1226,CRX0,1.5
11/15/15:1227,CRX0,1.3
11/15/15:1228,CRX0,1.3
11/15/15:1229,CRX0,1.2
11/15/15:1230,CRX0,
```

Fig. 22: This is an example of what the expected program output will look like. The data includes the a date/time slot, the port the data was received to, and the measurement of the data. More columns are expected to be added later as the limits of this project become better understood.



	A	B	C	D	E
1	Timestamp	Data Direction	Measurement		
2	11/15/15:1200	CAN_RXD	1.2		
3	11/15/15:1201	P33	1.2		
4	11/15/15:1202	P33	1.5		
5	11/15/15:1203	P33	1.4		
6	11/15/15:1204	CRX0	1.5		
7	11/15/15:1205	CRX0	1.5		
8	11/15/15:1206	P33	1.5		
9	11/15/15:1207	CRX0	1.3		
10	11/15/15:1208	CRX0	1.2		
11	11/15/15:1209	CRX0	1.2		
12	11/15/15:1210	CRX0	1.2		
13	11/15/15:1211	CRX0	1.2		
14	11/15/15:1212	P33	1.4		
15	11/15/15:1213	P33	1.4		
16	11/15/15:1214	P33	1.5		
17	11/15/15:1215	P33	1.2		
18	11/15/15:1216	P33	1.6		
19	11/15/15:1217	P33	1.6		
20	11/15/15:1218	P33	1.6		
21	11/15/15:1219	P33	1.6		
22	11/15/15:1220	P33	1.9		
23	11/15/15:1221	P33	1.9		
24	11/15/15:1222	CRX0	1.9		
25	11/15/15:1223	CRX0	1.7		
26	11/15/15:1224	CRX0	1.6		
27	11/15/15:1225	CRX0	1.6		
28	11/15/15:1226	CRX0	1.5		
29	11/15/15:1227	CRX0	1.3		
30	11/15/15:1228	CRX0	1.3		
31	11/15/15:1229	CRX0	1.2		
32	11/15/15:1230	CRX0			

Fig. 23: The data is expected to be easily converted into a excel file. The original final format is CSV.

Hardware Design

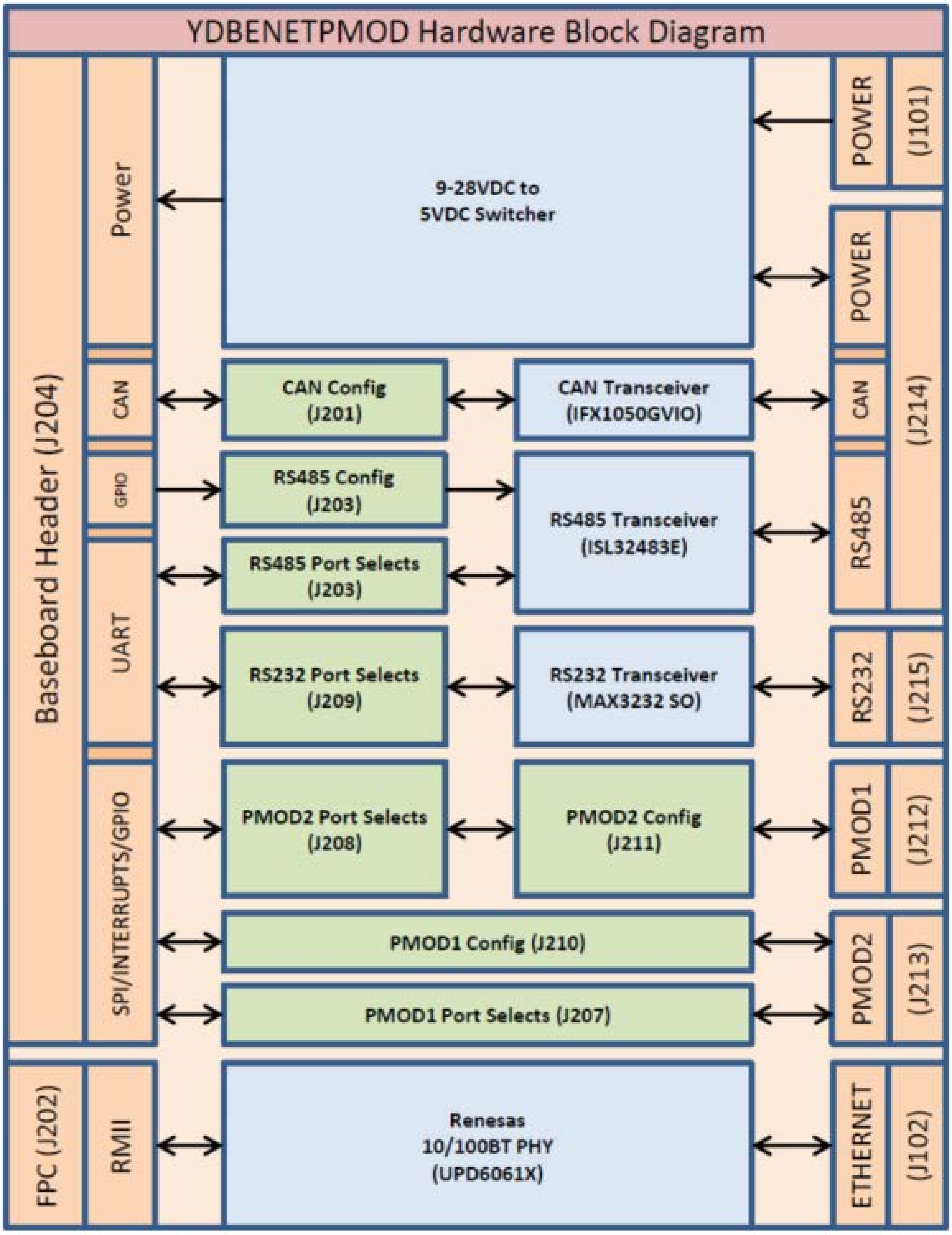

YDBENETPMOD Hardware Block Diagram
Baseboard Header (J204)
Power
CAN
GPIO
UART
SPI/INTERRUPTS/GPIO
9-28VDC to 5VDC Switcher
CAN Config (J201)
CAN Transceiver (IFX1050GVIO)
RS485 Config (J203)
RS485 Port Selects (J203)
RS485 Transceiver (ISL32483E)
RS232 Port Selects (J209)
RS232 Transceiver (MAX3232 SO)
PMOD2 Port Selects (J208)
PMOD2 Config (J211)
PMOD1 Config (J210)
PMOD1 Port Selects (J207)
POWER
(J101)
POWER
CAN
RS485
(J214)
RS232
(J215)
PMOD1
(J212)
PMOD2
(J213)
FPC (J202)
RMII
Renesas 10/100BT PHY (UPD6061X)
ETHERNET
(J102)

Fig. 24: This is a diagram of the hardware block diagram for the Renesas Embedded GUI and Communication Solution Kit.

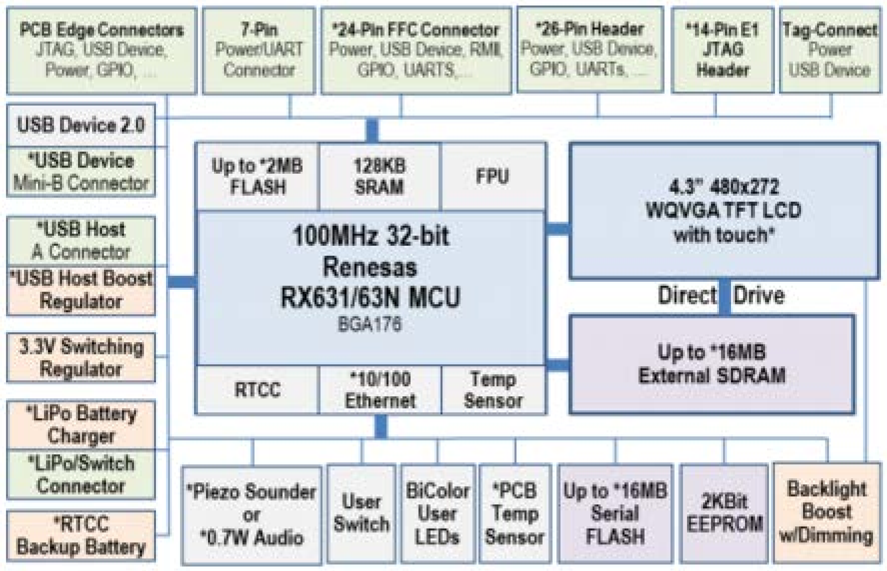

PCB Edge Connectors JTAG, USB Device, Power, GPIO, ...
7-Pin Power/UART Connector
*24-Pin FFC Connector Power, USB Device, RMII, GPIO, UARTs,...
*26-Pin Header Power, USB Device, GPIO, UARTs, ...
*14-Pin E1 JTAG Header
Tag-Connect Power USB Device
USB Device 2.0
*USB Device Mini-B Connector
*USB Host A Connector
*USB Host Boost Regulator
3.3V Switching Regulator
*LiPo Battery Charger
*LiPo/Switch Connector
*RTCC Backup Battery
Up to *2MB FLASH
128KB SRAM
FPU
100MHz 32-bit Renesas RX631/63N MCU
BGA176
RTCC
*10/100 Ethernet
Temp Sensor
4.3" 480x272 WQVGA TFT LCD with touch*
Direct Drive
Up to *16MB External SDRAM
*Piezo Sounder or *0.7W Audio
User Switch
BiColor User LEDs
*PCB Temp Sensor
Up to *16MB Serial FLASH
2KBit EEPROM
Backlight Boost w/Dimming

Fig. 25: This is the hardware block diagram for the SIM 225, Renesas Embedded GUI Solution Kit.

Components

- 4.3” WQVGA 480 x 272 color TFT display - Touch panel with color display.
- RS232 Serial Port - Operates as a UART transceiver. It has high speed serial port capability and requires no CTS/RTS hardware handshaking support.
- RS485 Serial Port - Operates as a UART transceiver. It can operate at half or full duplex and is capable of working in point-to-point and multi-drop networks.
- USB Mini-B Device connector/USB A Host connector - Used to export collected data. USB 2.0 full speed/Embedded host port capable of supplying up to 150mA
- RoHS TTL-232RG serial Cables - Allows the embedded device to communicate to a third party device.

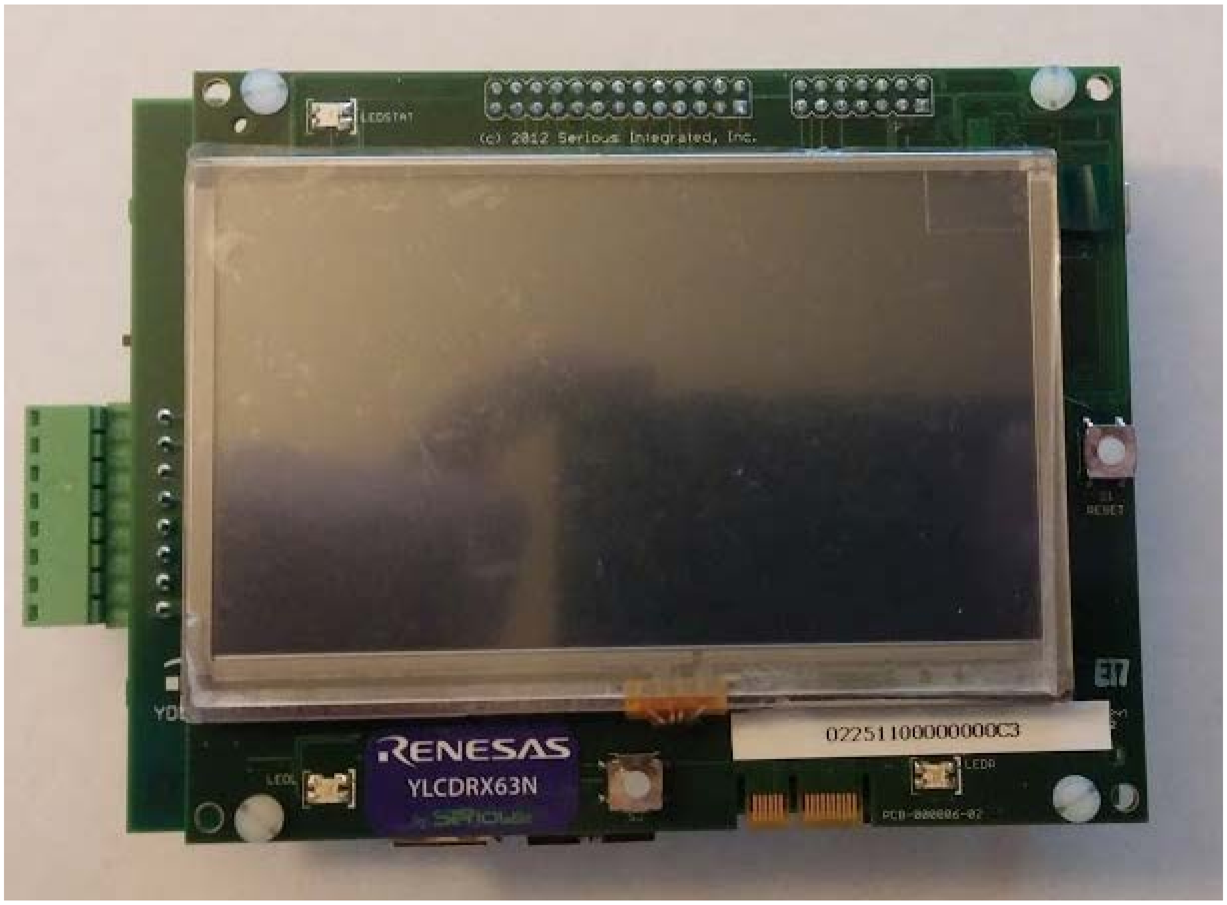


Fig. 26: This is the touch screen of the Renesas Embedded GUI and Communication Solution Kit. This is the main feature that the user will use to input and receive information.

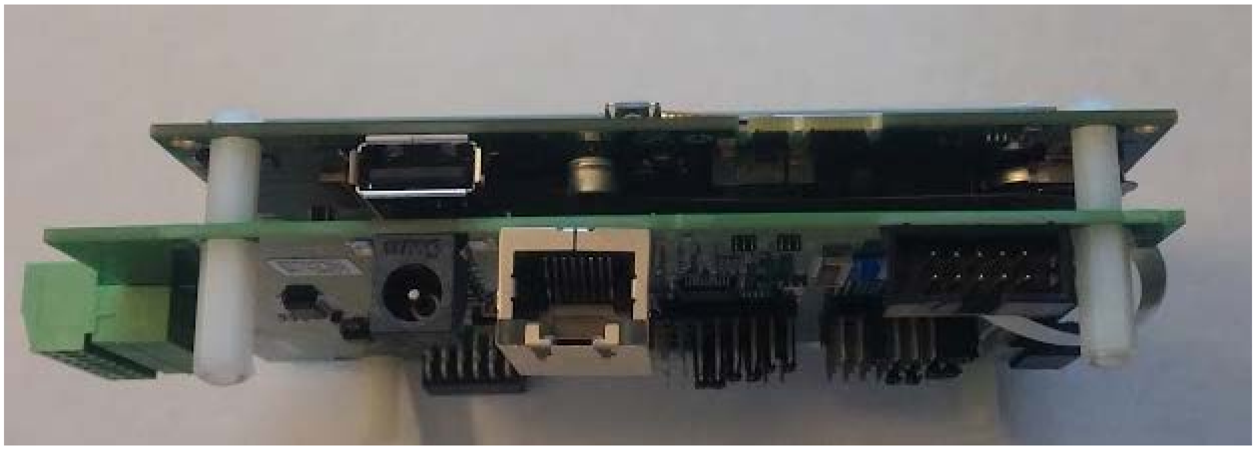


Fig. 27: This is a side shot of the board. The top board has the USB port that our group will be using the most to save the transmitted data. The bottom board has the power barrel jack and the serial ports. It also contains the ethernet port, which our group may use later in the project, to transmit data over the internet.

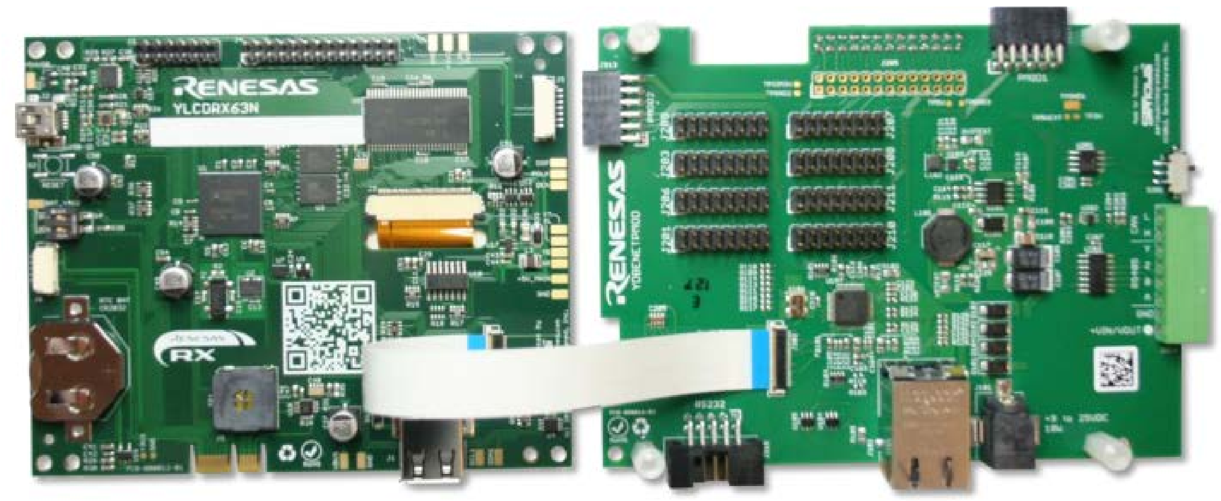


Fig. 28: This is how the two boards will be connected together. They will use a 26 pin board to board (shown at the top middle of the left board). It will also use a 24 pin flex cable that will be connected into the FPC connector(White flat cord shown connecting both boards together).

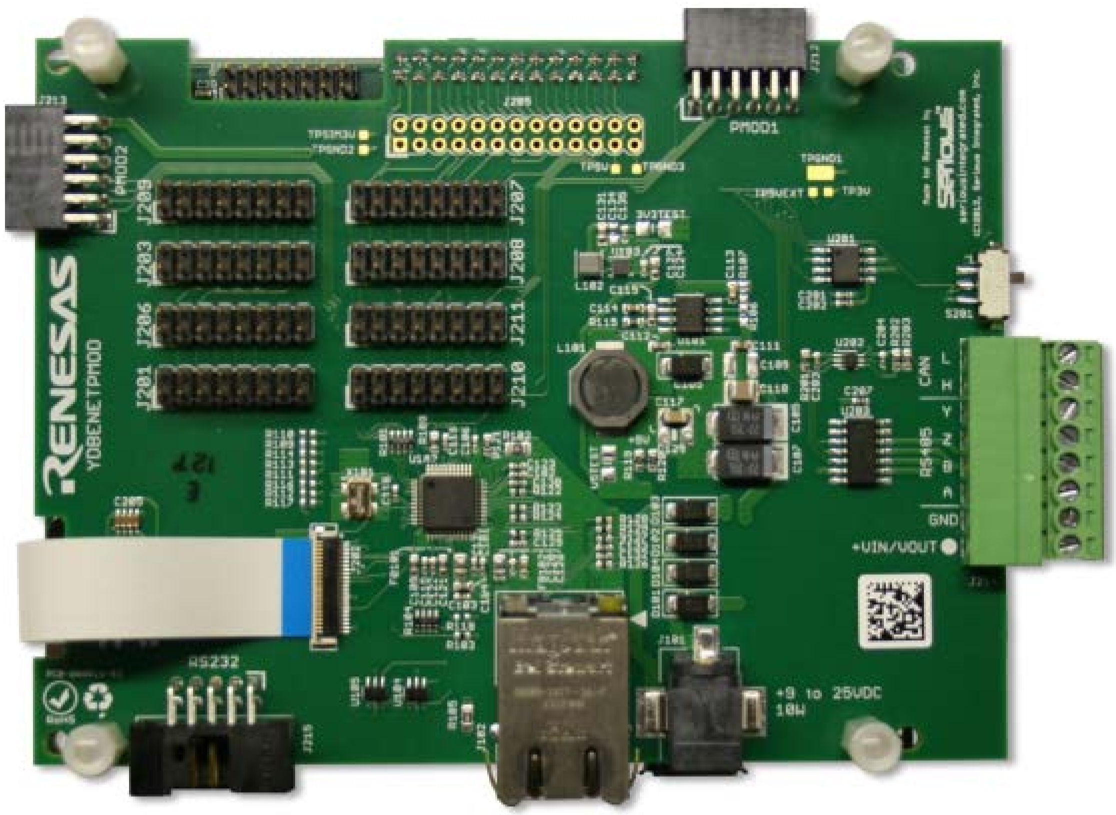


Fig. 29: Here is another shot of the bottom board. The black box at the bottom, center right, is the power barrel jack. When both boards are connected, they both cannot be powered with a USB so we must use a 9-25VDC 1W power supply connected to the power barrel jack

Glossary of Terms

- ADC - Analog to Digital Converter; a device that converts an inputted physical quantity (generally voltage signals) to a digital representation of that quantity
- Baud Rate - rate at which data is exported in a communication channel (represented as bits per second)
- Byte-By-Byte Timestamping - timestamping method in which each byte is individually processed and labeled as it comes; can be faster, but is less readable than packet timestamping
- CTS - Clear To Send; a signal between computers that indicates that the transmission can proceed
- Data Logger - an electronic device that records data over time through built in instruments, programs, or sensors
- Duplex System - a system in which two devices can communicate with each other; a full duplex allows both devices to communicate with each other simultaneously while a half duplex only allows communication in one direction at a time
- E2 Studio - the IDE that will be used to develop the SLODC project; the IDE is based off of the eclipse IDE, but is specialized for debugging and testing on Renesas boards
- Embedded System - computer system with dedicated functionality within a larger mechanical or electrical system
- Endianness - a type of data transmission style in which dictates where the most significant byte of data is located within a transmission; the sender and receiver need to be in agreement
- Error Detection - techniques that are used to ensure reliable data flow over channels that may be subject to noise or other data transmission problems
- Flow Control - the process of managing the transmission data rate in order to prevent a fast sender from overwhelming a slow receiver and creating a bottleneck in their communications
- Interrupt - signal to the processor that an event has occurred that requires immediate action
- Micrium - a C based RTOS that supports user interface interrupts, timers, and serial port monitoring
- MicroEJ - a software tool used to emulate the Renesas board used in the project; it allows for a java-based development of the embedded system
- Output Port - the port in which data is sent during serial communication
- Packet - a small piece of a larger data item that is designed for more efficient transmission
- Packet timestamping - grouping and then labeling a series of incoming data; tends to be more readable than byte-by-byte timestamping, but may be subject to performance issues
- Parity Bit - a simple type of error detection in which a bit is added to transmissions to create an even or odd count of 1's in order to ensure that a message is valid

- Pin - a part of an embedded system that exists either as a physical connection to a board or as a register; pins can be used to determine the states of an embedded system
- Polling - process of actively sampling the status of an external device by a client program
- Receive Port- the port of incoming data from transferring data over a serial connection
- RTOS - Real Time Operating System; an operating system that serves data as it is entered
- RTS - Request To Send; a signal sent by a communicator to verify that the other device is ready for data transmission
- Salting - a security feature in which additional input is added to a password to aid in its encryption
- SCI - Serial Communications Interface; controls the configuration settings and operation of the serial ports
- SCMR - Smart Card Mode Register; used to identify the direction of data travelling on the serial port
- Serial Port - a physical interface through which data enters or exits one bit at a time
- SMR - Serial Mode Register; register controlling the configuration of the serial port's parity, stop bits, communication mode and message length
- Spooling - process in which data is sent from one device to another device for intermediate storage
- Stop Bit - a pattern of bits that is used to indicate the end of a transmission
- Timestamping - adding a sequence of characters to data logs to identify the time that the data was processed as well as identifying whether the data was incoming or exitting
- Time Sharing - the act of splitting up the computer resources through multitasking; a service may request to use computing power, but only be given a limited amount of time to use its allocated resources
- UART - Universal Asynchronous Receiver/Transmitter; device that translates between serial and parallel data forms
- Widget - object used in the Micrium Operating System to handle user interface features; widgets include drop down menus, buttons, file trees, etc.

References

Russel, David. Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment. Morgan and Claypool Publishers, 2010.
This book contains the basics of embedded systems and can function as a good reference for basic functionalities of data analysis and port configuration while also serving as a launching point for advanced features. The book is especially useful in its sections related to USART management and memory addressing.This book will be helpful in the project for determining how to configure the board that will used in this project. The biggest challenge in this project revolves around using embedded systems and this book acts a guide for overcoming this challenge.

Fan, Xiaocong. Real Time Embedded Systems. Elsevier 2015.

Fan's embedded systems book has a more detailed look into embedded systems than Russel's. It examines a lot of elements more closely tied into SLODC such as the use and implementation of RTOS and the execution of real time task execution and management. It also has a useful sections on embedded system architectures and thread usage. Like the book by Russel, David; this text helps with learning how to program and work with embedded systems. The SLODC and RTOS methods mentioned in this book will be the most valuable for this project.

Asrodia, Pallavi, Vishal Sharma. "Network Monitoring and Analysis by Packet Sniffing Method." IJETT. Vol. 4, No. 5, May 2013.
This paper focuses on the impact of packet sniffing. It identifies the core modules required for packet sniffing as well as some of the impacts of packet discovery. The paper identifies packet sniffing as a way to troubleshoot problems within a network and to identify where problematic signals are coming from.This paper helps with the project because it gives examples and techniques for packet sniffing.The core modules required for packet sniffing are essential for grasping the methods to be used in this project.

Reiger, Robert, Yan-Ru Huang. "A Custom-Design Data Logger Core for Physiological Signal Recording." IEEE Transactions on Instrumentation and Measurement. Vol. 60, No. 2, February 2011.
This paper shows a lot of methods and solutions to approaches based on nonfunctional requirements and challenges. It explains why certain hardware features were selected. The paper also goes into some detail about how the specialized serial logger operated with particular attention paid towards the operation of the ADC. This paper is relevant to the project in that it helps develop techniques for determining what features in this project are the most essential. By identifying the nonfunctional requirements and possible challenges, resources can allocated to resolve them.

Nhivekar, G.S., R.R. Mudholker. "Data Logger and Remote Monitoring System for Multiple Parameter Measurement Applications." e-JST 2011.
This paper discusses methods used in the design and implementation of an embedded serial logger that measures temperature and humidity. The article goes into specifics related to the use and importance of timers as a necessity to get meaningful data from the analog to digital conversions.

Fonseca, Rodrigo, Prabal Dutta, Phillip Levis, Ion Stoica. "Quanto: Tracking Energy in Networked Embedded Systems."
This paper focuses on the promotion of the Quanto system which analyzes the energy consumption and system of embedded systems. Gaining knowledge of what processes are hogging up resources and power of embedded systems can allow for optimization which will improve the runtime and performance of a system. This is incredibly useful for battery operated embedded systems.
Kocher, Paul, Ruby Lee, Gary McGraw, Anand Raghunathan, Srivaths Ravi. "Security as a New Dimension in Embedded System Design." DAC 2004.

This conference paper addresses some of the issues related to securing data managed by embedded systems. The paper discusses multiple types of attacks that an embedded device can be subjected to and discusses architectures that add resistances to these attacks. It also examines some of the mechanisms and tricks that can be used to help lock down an embedded system from unwanted attention and interaction.

Tsai, Wei-Tek, Lian Yu, Feng Zhu, Ray Paul. “Rapid Embedded System Testing Using Verification Patterns.” Software, IEEE 22.4 (2005).
This article discusses the difficulties of testing embedded systems and offers some solutions for improvement. The article recommends the development of verification patterns to test different scenarios that typically arise from the requirements specification. Verification patterns can be used at a basic level to test out various use cases and at more advanced levels to ensure that timing-based operations work correctly.

Badhiye, Sagarkumar, Chatur Wakode. “Data Logger System: A Survey.”
This paper gives an overview on how data loggers are developed and work. It defines and differentiates different categories of data loggers and gives a general overview on the operating flow of these devices. The article discusses and brings up the idea of using different channels to assist in gathering data more quickly. While parallelism would be great to use on the SLODC project, it won’t be feasible due to board and time limitations.

Popa, M., A.S. Popa, V. Cretu, M. Micea. “Monitoring Serial Communications in Microcontroller Based embedded Systems.”
This paper outlines some different techniques to monitor serial data. The paper primarily discusses the use of RS232, LIN, and SPI systems to monitor a CAN connector. The paper goes into detail on the advantages and disadvantages of the use of all of these systems over the application, data, and physical layers. It was helpful for determining whether or not to utilize the CAN connectors on the embedded device as well as give an insight into how to monitor the incoming and outgoing data.