User Manual For
Automated 3-Way Syringe Mixer

3-Way Syringe Mixing Team

Team Members:

Vincent Jencks vkj3@nau.edu
Andrew VanDenburgh amv336@nau.edu
Colton Smith css266@nau.edu
Yunchen Zhu yz246@nau.edu
Handi Xi hx25@nau.edu

EE486C – Spring 2018

Project Sponsor: Timothy Becker
Faculty Mentor: Ashwija Korenda
Instructor: Dr. Kyle Winfree

Introduction

In society today, most if not all medical procedures have some sort of automation to them, however some syringe mixing procedures are still done by hand. Due to the recent focus of treating aneurysms using a liquid embolic method for filling and removing the aneurysm within a vessel, there has been a demand for an automated mixing system. As of this report there is no recorded existence of an automated three-way mixer. While the process of understanding how the syringes are mixed is not complicated, the amount of factors that need to be monitored or regulated during the process is what makes the design cumbersome.

Due to how recent the development of the PPODA-QT liquid embolic treatment is, there is no known precise flow rate for the liquids to mix properly. With the development of an automated process to mix the liquid components of a PPODA-QT syringe, doctors will be able to treat aneurysms without worry of the current human error involved in the process. This development would allow hospital staff to have one less worry when attempting to isolate and treat the neurological condition of an aneurysm within a blood vessel.

We are pleased that you have chosen 3-Way syringe Mixer team for your needs.Our team designed an automated mixing system which will interface with syringe models already in existence. There is a strong need for this mixing system, as the evidenced by aneurysm remedy. We provide for you here a useful system for liquid medication mixing that has been user-designed to meet your needs. Some of the key highlights include: The user will set the T-jointed syringe configuration, holding 3 syringes, into our design. After the syringes are placed within the device, the user will be able to input a mixing strength that the device should take when mixing the liquids and set them mixing time. Then the user can get read the mixing data from LCD screen. After the mixing is done, the user will be able to safely extract the finished mixture which will be used in the liquid embolic treatment.

This manual contains important information for the use and maintenance of this device. It starts with preparing the use of the syringe mixer as well as the proper connections needed from the microcontroller to the PCB (Printed Circuit Board) and to off board components. The guide will also help the user understand where problems may occur and potential fixes to these problems. The purpose of this user manual is to help you, the client, reasonably use and maintain the 3-way mixer product in your actual business environment going forward. Our goal is to make sure that you are able to benefit from our product for many years to come!

Setting Up Device

Pin Configuration:

PCB PINS	
Motor 1	Output to Motor 1
Motor 2	Output to Motor 2
Motor 3	Output to Motor 3
JP5	Connect Motor 3 adjacent to Motor 2. Connect for three motor setup or leave open for two motor setup
SU1	Display PinHead
SU2	Display PinHead Continued
SU4	Button Pad Pin Head
Vin_Socket	12 Volts DC input : DC Power Jack type a connector

Display Pins	
VSS	Ground
VDD	5V
Vo	Display Brightness Control Use Potentiometer to Adjust
D0	Data
D1	Data
D2	Data

D3	Data
D4	Data
D5	Data
D6	Data
D7	Data
White	5V LED
Black	Ground LED

Button Pad Pins	
R1	Row 1 Input
R2	Row 2 Input
R3	Row 3 Input
R4	Row 4 Input
C1	Column 1 Output
C2	Column 2 Output
C3	Column 3 Output
C4	Column 4 Output

GPIO Pins	
GPIO_0	Motor_Select_1
GPIO_1	Motor_Select_2
GPIO_2	Direction_1
GPIO_3	Direction_2
GPIO_4	R1

GPIO_5	RW
GPIO_6	E
GPIO_7	RS
GPIO_8	BUTTON_1
GPIO_9	BUTTON_2
GPIO_10	BUTTON_3
GPIO_11	BUTTON_4
GPIO_12	BUTTON_5
GPIO_13	BUTTON_6
GPIO_14	BUTTON_7
GPIO_15	BUTTON_8
FPGA_GPIO_0	D0
FPGA_GPIO_1	D1
FPGA_GPIO_2	D2
FPGA_GPIO_3	D3
FPGA_GPIO_4	D4
FPGA_GPIO_5	D5
FPGA_GPIO_6	D6
FPGA_GPIO_7	D7
FPGA_GPIO_8	PWM

Hardware Installation and Setup Guide:

Connect SmartFusion SoC (System on a Chip) board to PCB as shown below in picture

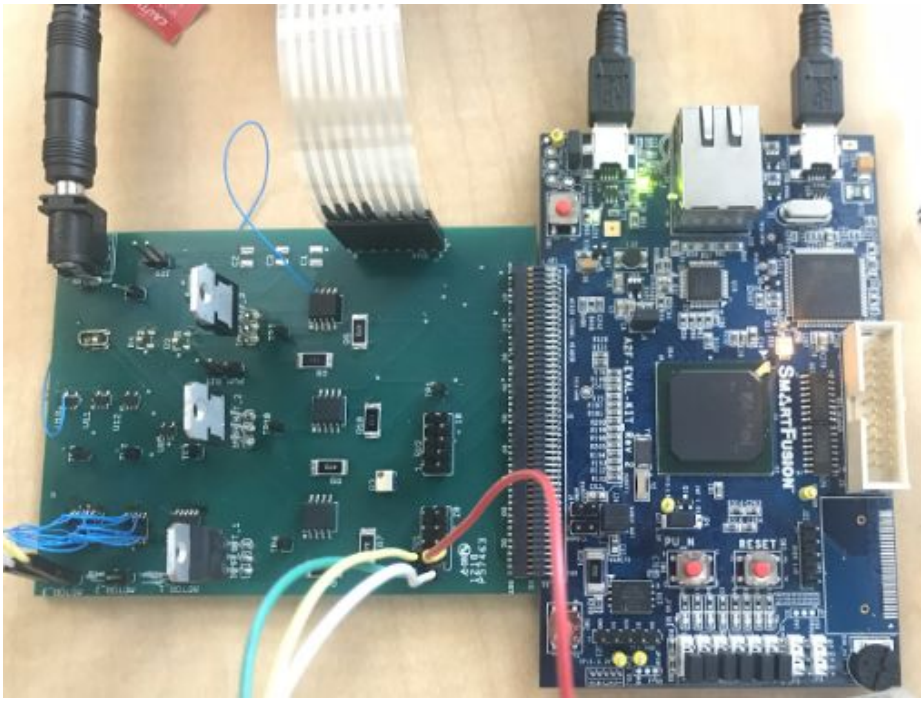


Figure 1: Header Connection

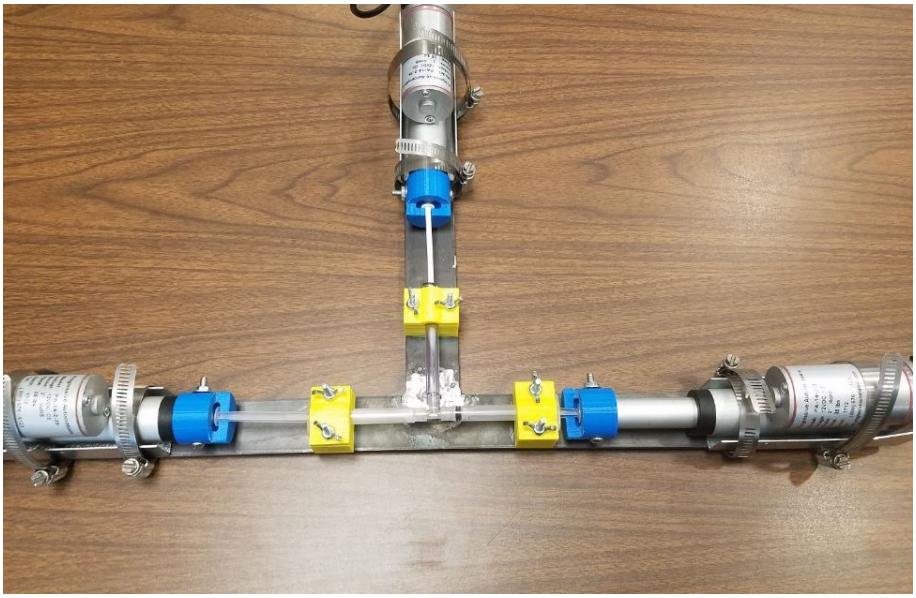


Figure 2: Base and Actuator Mount

As seen in Figure Y, there are some 3D printed parts attached to the steel T-Bar. The steel bar is layed out In the T-Shape as the joint connection puts the syringes into a T. The syringes need to be steady when the mixing is occuring, as bending a plunger of the syringe can cause the actuator to keep pushing and eventually snapping the plastic. To hold these syringes there are three 3D printed holders (yellow) that have divits for the syringe to fit into. Then a top is put through screws and clamped down with the use of wing nuts. These were chosen as they are fast to remove and put on.

Once this is done the end of the plunger will be placed inside of the lips in the blue 3D printed parts. These are there to allow the actuators to not only push but pull the actuators to help relieve the load for other actuators while mixing as well as having the ability to pull liquid into one syringe.

The actuators themselves are clamped to the steel bar with the use of hose clamps, one right before the piston and one around the largest point. With the use of hose clamps, the actuators can be adjusted in location based on how filled the syringes are as well as varying syringes. While this process is slightly time consuming it allows for variable syringes and well as different cc's of fluid. If similar conditions are set, the actuators should not have to be adjusted.

Software Setup Guide

There is no software needed to set up for normal use of the product. However, the team has not been able to fully finish the product. This tutorial on Libero, and Softconsole will teach future engineers or enthusiasts on how to further develop the Software used in this project.

Libero SoC setup guide

Libero is a microsemi program for programming the SmartFusion boards as well as other boards owned by them. Libero is necessary if the user wishes to modify the programming in the future. This program creates a GUI interface that will allow user to enable or disable peripherals on the SmartFusion board, as well as define their features. Peripherals are all the other gadgets hooked up to the microprocessor. The peripherals used in this project are the Fabric Interface (FPGA), GPIO (General purpose input/output) , and Clock Management. Note that every peripheral is well documented by microsemi and you may search these manual at microsemi's website

To download Libero go to "www.microsemi.com" navigate to FPGA SoC in products menu, then go to design tools, and then libero. After downloading Libero you will need a license to run the program. You may request a silver license which is free for one year and may be renewed. Visit the microsemi website for more information.

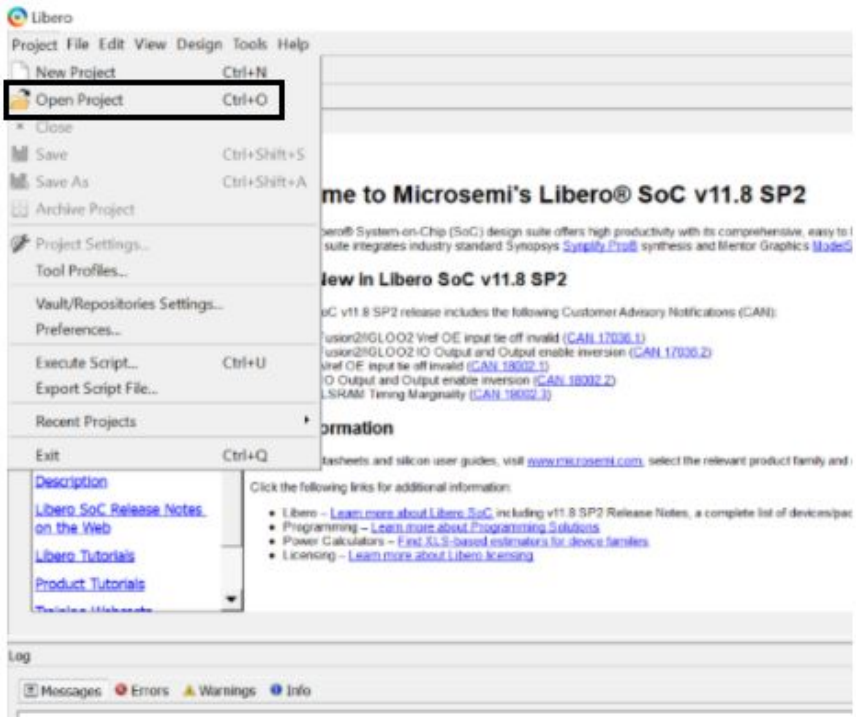
SoftConsole setup guide

SoftConsole is a C# based language used to write to the microcontroller. After peripherals have been configured in Libero the next step is to write code to the MPU (Micro-processing unit). The MPU used in the SmartFusion board is an ARM M4 which is a efficient yet powerful MPU made for embedded systems.

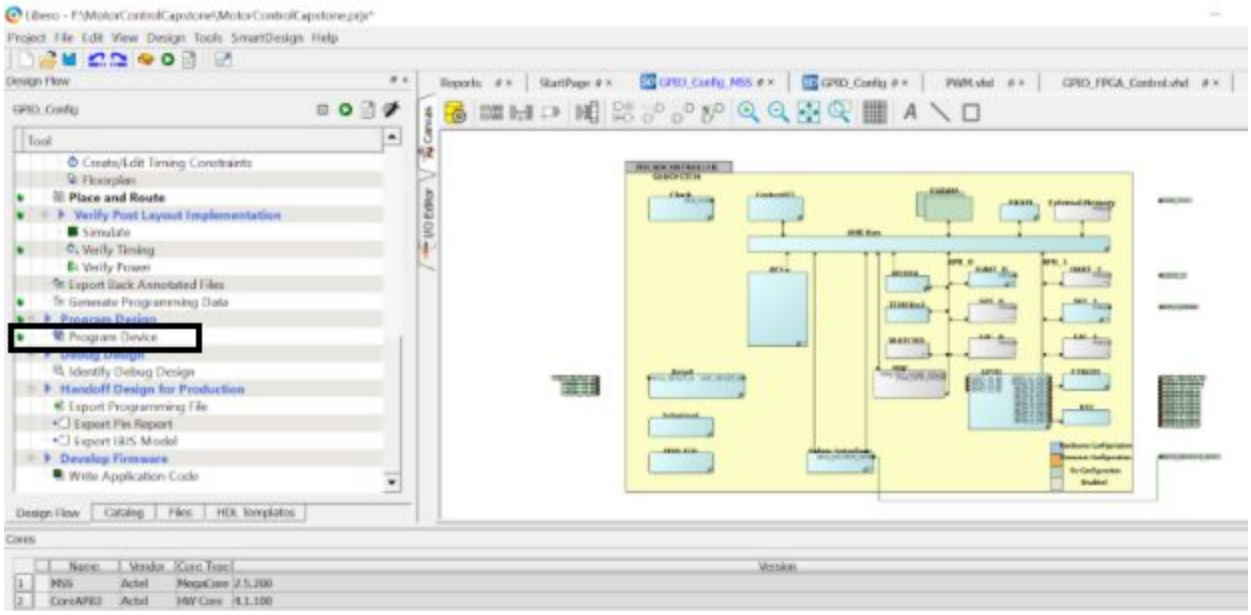
SoftConsole should install with Libero. However to download SoftConsole manually go to "www.microsemi.com" navigate to Partners, then to accelerate ecosystems and design tools.

Getting to the Code

After opening Libero navigate to Project → Open Project. Then navigate to the project that has already been set up. The project should be a .prjx file.

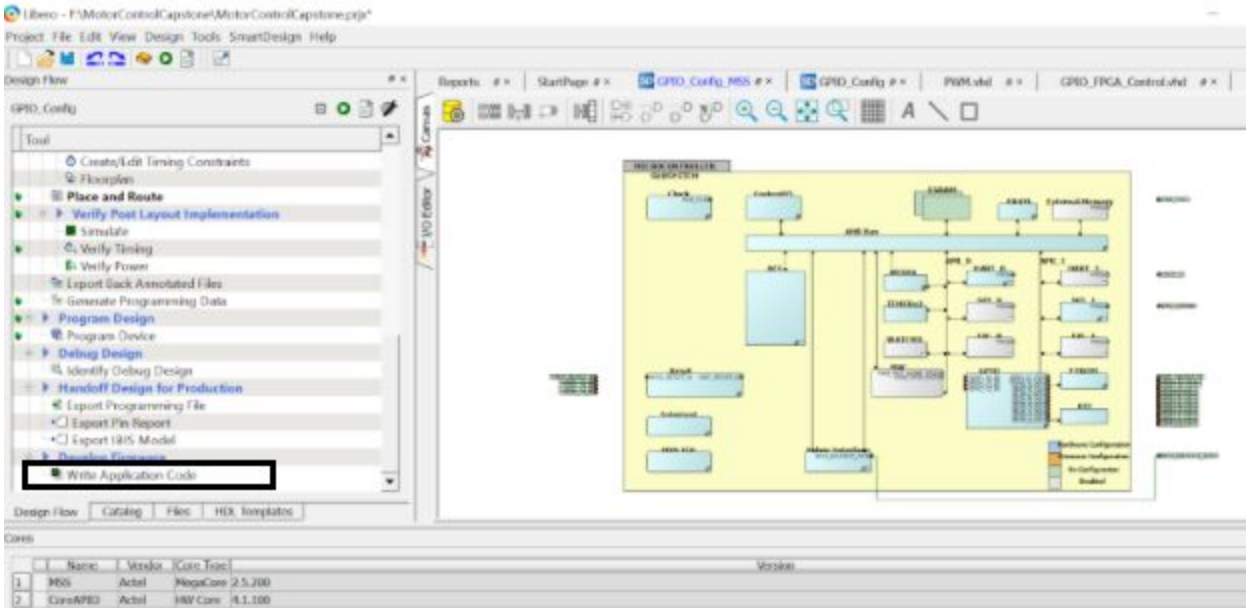


After navigating to the project it is important to program the device after hooking up the SmartFusion board to the computer.

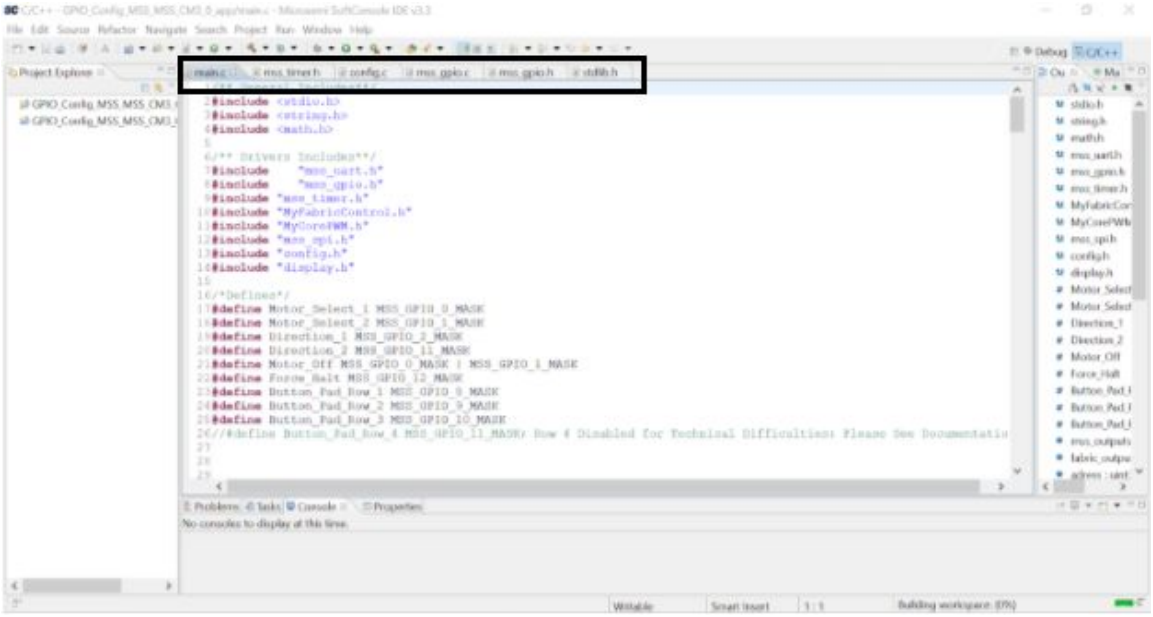


Following the programming of the device there should be check marks along the left column next to many of the different tool names. If an X comes up then there is an error that will need to be fixed.

When the device has been programmed it is then possible to pull up the code by clicking on the Write Application Code Section.

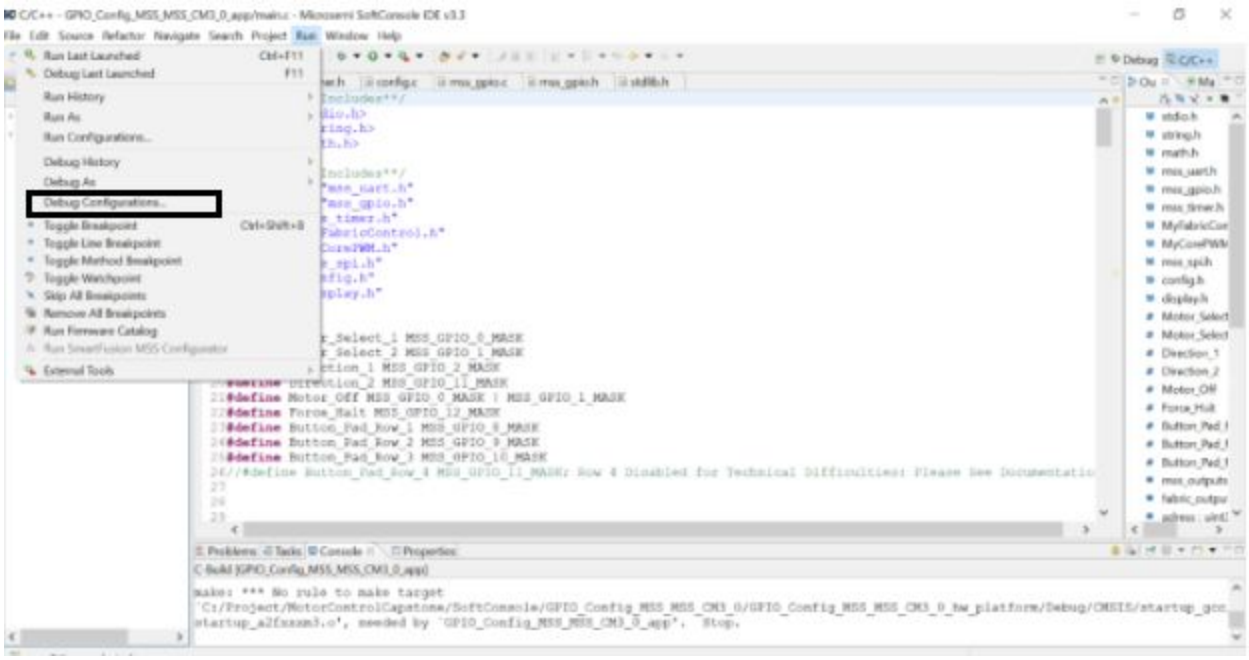


A new window for SoftConsole will then pop up showing the many different tabs of code that can be accessed. Each one plays a role in the construction of the code, by being referenced in the main code which can be seen when SoftConsole is initially opened up.



To run the code, it is important to first go to Project → Clean… which will setup the code to be able to be run. When another window pops up make sure to hit clean all. This step should be executed each time the code is changed after being saved.

After cleaning the program, go to Run → Debug Configurations… to have the code initialized to be able to run.



The code is then ready to be run by pressing the Continue button on the control bar.

If at any time the code needs to be manually stopped, the terminate button can be pressed. When this happens though, it means that the Debug Configurations… will once again need to be called.

Button Setup

Simply insert the button pad into the 8 pins that are linearly connected at the bottom right of the PCB which is right next to the header which connects the PCB to the SmartFusion board. When inserting it is important that the button pad when falling away from the PCB will be button side down.

Configuration & Use

Now that everything is hooked up all that should be left to run the device is to hit a few buttons on the button pad. Hitting the 1 button on the pad should cause the system to stop at any point in time during the process. When pushed, it will be needed to run debug configurations once again because the code will exit the program altogether. By hitting button 2 the code should run autonomously for the specified amount times written within the code. The points of the code that control the timing and mix number are explained within the software configuration section below.

Software Configuration:

This section is meant to show how different pieces of code are used within the whole code.

Main.c

Int StartScreen(void)

This function is used to Start the device. Its purpose is to take in inputs from the user to define preconditions in the mixing process.

Condition Enter: When enter is passed the StartScreen() function exists and begins mixing.
Condition Reset: When reset is pased the StartScreen() function turns off Relays, Exits Code, and safely shuts down.

To Do Condition Power: When power is passed it will set the precondition of the PWM duty cycle. This function is uncomplete and needs to be implemented by future engineers or modifiers of this project.

Example:

```
uint8_t neg_Edge = 0x00;
setNegEdgeMyCorePWM(neg_Edge);        <- see MyCorePWM for more information
setPeriodMyCorePWM(0x08);             <- see MyCorePWM for more information

Int main(void)
{
neg_Edge StartScreen();
}
```

Config.c

Void setGPIO(void)

This function sets up parameters needed to use GPIO ports. This function must be called before anything to setup preconditions for the PCB circuit.

Example:

```
setGPIO();
```

Void setSPI(void)

This function set up parameters needed to use SPI 1. External SPI devices may interface with project by using this interface. This function must be called before calling any other SPI functions in driver.

Example:

```
setSPI();
```

MyFabricControl.c

void init(void)

Does nothing

void user_data(uint8_t value)

This function passes in the corresponding GPIO pins to be set on/off for the fabric gpio pins.

Ex:

```
user_data(0x03);          <- bring fabric gpio pins 1 and 2 high and others low
```

MyCorePWM.c

void setPeriodMyCorePWM(uint8_t value)

This function sets the period trigger of the PWM pin. Combined with setNegEdgeMyCorePWM this function is used to create a PWM signal at users specifications.

Note: Takes in clocks then number passed in the amount of clock cycles before trigger event

Math:

Time for Period (s) = Fabric Frequency (1/s) * value

Ex:

```
# This code sets the pwm from off to 50% duty cycle at 8 clock cycle period
setPeriodMyCorePWM(0x00);       <-set period 0 so signal low all times
setNegEdgeMyCorePWM(0x00);      <-set negative edge trigger 0
setNegEdgeMyCorePWM(0x04);      <-sets negative edge to 4 clock cycles
setPeriodMyCorePWM(0x08);       <-raises period trigger to 8 clock cycles
```

Void setNegEdgeMyCorePWM(uint8_t value)

This function sets negative edge trigger of the PWM pin. Combined with setPeriodMyCorePWM this function is used to create a PWM signal at users specifications.

Note: Takes in clocks then number passed in the amount of clock cycles before trigger event

Math:

Time for Negative Edge Trigger (s) = Fabric Frequency (1/s) * value

Ex:

```
# This code sets the pwm from off to 50% duty cycle at 8 clock cycle period
setPeriodMyCorePWM(0x00);       <-set period 0 so signal low all times
setNegEdgeMyCorePWM(0x00);      <-set negative edge trigger 0
setNegEdgeMyCorePWM(0x04);      <-sets negative edge to 4 clock cycles
setPeriodMyCorePWM(0x08);       <-raises period trigger to 8 clock cycles
```

Display.c

void clearDisplay(void);

Clears the display

void returnHome(void);

Return Home is cursor return home instruction. Set DDRAM address to "00H" into the address counter
Return cursor to its original site and return display to its original status, if shifted. Contents of DRAM does not change

void entryModeSet(uint8_t ID);

I/D Increment /decrement of DDRAM address (cursor or blink). When I/D = "High", cursor/blink moves to right and DDRAM address is incremented by 1. When I/D = Low, cursor/blink moves to left and DDRAM address i decreased by 1. CGRAM operates the same as DDRAM, when read from or write to CGRAM

void displayOnOff(uint8_t ID);

This function controls diplay's on/off state as well as the cursor on/off state
Parameter ID: (0bXXXXXDCB)

D: Display On OFF control bit - set high to turn on display or, bring low to turn off display
C: Cursor On Off control bit - set high to turn on cursor
B: Cursor Blink control bit - set cursor to blink
Ex:

```
diplayOnOff(0x00);          <-turns display off
diplayOnOff(0x09)           <-sets display on with cursor and cursor blink
```

void cursorShift(uint8_t ID);

Shifts the cursor left or right based on the parameter passed in to ID
Parameter ID: (0bXXXXXABX)

A: Shift Left: Set high to shift cursor left
B: Shift Right: Set high to shit cursor right

Ex:

```
cursorShift(0x04);          <-shift cursor left
cursorShit(0x02);           <-shift cursor right
```

void writeDisplay(uint8_t data);

This function write to display. See Technical document on display 1602A to see all write commands for data.

Parameter data: Code to write to display. Each data code has linked character to write to display.

Ex:

writeDisplay(0x03) <- write ‘0’ to display
#see 1602A display document for more data commands

Mixing Process

User confirms that they want to start a mix
User defines the mix length desired
The device starts by mixing the linear syringes together
Then the hardening agent is added into the mixture
The device mixes the linear syringes once more
After mixing the fluids are retracted from the initial syringe holding the hardening agent

Finally, the mixed fluids are able to be used for aneurysm treatment

Maintenance

The most significant thing for maintenance is to keep the syringe mixer clean. Reasonable usage may extend the service life. Use disposable syringes and keep equipment in shady and cool environment without sunlight and moisture. After use, clean the device and make sure there is no residue around. When not in use, put the equipment in a confined place to protect it. Besides, properly lubricated and periodic check on components are needed based on the operating frequency. The microcontroller board may be more likely to have a long lifetime. Finally, keep the equipment away from children and out of reach.

Troubleshooting Operation

When troubleshooting, it is important to remain calm and constantly test to see what may be going wrong. Below are some of the possible things that could go wrong.

PCB Troubleshooting:

First, before following the advice below, double check that all of the components are on the board and that any wires that connect to the PCB are properly attached. It is assumed that even if the new PCB layout is used, it is possible that modifications will be made using wires. A multimeter is required for the steps to come. Also, make sure that the voltage source is NOT plugged in at the start of your troubleshooting procedure.

Error Loose Connection:

Step 1: The most important thing to do when troubleshooting is to check connections which occur between the SmartFusion and the PCB. Be extra careful not to directly connect the voltages coming out of the SmartFusion to the grounds of the SmartFusion.
First, connect the SmartFusion to the PCB if it is not already. Then, using the test points on the PCB, make sure that each part of your circuit that should receive power are receiving power from the SmartFusion.

If, everything is as expected, then you can move to step 2. If, you find out that the voltages are not within an acceptable range, it is likely that the circuit has bridging, there is a wire connected incorrectly, or there are wires touching that shouldn't be. Bridging is when solder connects multiple pins, that are next to each other, that should not be connected. At this stage the multimeter won't help out much, and you will need to rely on your eyes. Luckily however, you can look at the Eagle schematic which should help you see how each component on the PCB is supposed to be connected.

Test Point Voltage Check:

Step 2: After making sure that the initial connections are fine, you should now use the test points given to check other pieces of the circuit. The reason we do this before plugging in the external power source, is that the voltage and current from the SmartFusion will be much lower. Meaning that if anything does go wrong, there is less of a chance that components will be damaged. While going through these procedures it is likely a good idea to have the Eagle schematic of the PCB up, to verify connections. If the Eagle schematic and board are up at the same time, you can click the eyeball button on each, which allows you to see the interaction between the two. Additionally, if any of the components require SmartFusion input to function, you can step through the code to create different test scenarios, allowing you to have better insight as to what is happening.

When testing a specific component, put the 5V line or an active output line to the input of the component, using the test points. The test points are the rods which stick up all over the PCB to help analyze what is happening at that spot in the design. Don't forget to factor in the amount of voltage and current that each component is rated for. Then after considering these factors you should be able to deduce if that component is working as intended by measuring the output pins of the component or any test points connected with the output.

If everything is working then move onto another component, until you are sure that the components are working appropriately. However, if there is an issue then you should first double check the schematic of the component to make sure that it's pins are in the correct configuration. Additionally, if the component being looked at has multiple pins, which are to be working and/or giving an output, it is crucial that each one is checked for appropriate values. If the component isn't working at all, then it could be a defective component or there may be a bad connections. Bad connections come in many forms ranging from a poor soldering (too little) connection to the PCB, to wires or pins being connected together which shouldn't be. In the case of too little connection there will likely be little to no voltage or current where it needs to be. In the case of extra connection there will be voltage on places where there shouldn't be voltage, and can often be found on the pins of the same component or nearby components. Either way, this will mostly require careful examination.

Check Current Output

Step 3: Now that we know that the components are correct we can plug in the power supply to the PCB, AFTER making sure to unplug the previous testing. Note, that if jumpers between the test points are still in, something is likely to catch fire or explode (trust us).

With the power source attached you can now use the multimeter to test the connections along the path that requires the external voltage. Similar to the previous step you will be checking the components to make sure that each one is working as intended, with the possibility of stepping through the code to create different cases and looking at the Eagle schematic. If you are at this step however, it is more likely that the problem lies in the components that are only hooked up to the power supply or the connections themselves. If the components are all working as intended then it is possible that any wires on the board are causing the issue, by connecting things that shouldn't be connected or just connected incorrectly.

PCB Malfunction:

Step 4: Given that nothing was overlooked when following the previous steps, then the problem is likely to be the PCB itself. Either the amount of current and/or voltage was too high and the PCB couldn't handle it or the PCB's design was faulty from the start.
There isn't much to be done when the PCB has been overvolted, due to the fact that this will likely cause bridging between lines inside of the PCB. If it is unknown where that bridging has happened it is extremely difficult to proceed and may be best to order a new PCB with modifications. Increasing the width of the internal lines of the PCB should allow for more current and voltage to run through the lines. If it is known where the internal bridging has

happened you can cut the line between the components. Following this, wires can be soldered to connect the components directly.

Similarly, when the PCB design is incorrectly setup, it is possible to cut the internal lines without needing to order a new PCB. To cut the line, you will need a small sharp blade (box cutters and scalpels are sufficient). With the blade you should make a cut where there are few other lines close by. When cutting be sure to scrape along the line's direction until you see the copper. After being revealed you will need to apply a little extra force on the copper to scrape it out of the PCB. It is important not to cut too far because it is eventually possible to affect the lines on the other side of the board. Finally after the cut apply wires to any connections which now need to be made.

Code Troubleshooting:

When troubleshooting the code it is important to reference the previous section which describes how the different pieces of code function. Understanding how the pieces of code work together may help to solve the issue occurring.

Step 1: Double check that any naming conventions are accurate, as even the best coder can at times misspell proper variable or function names. Alternatively, it is possible that similar function names can cause for confusion within this process.

Step 2: If the code seems unclear then it is possible to test different portions of the code. This can be done by adding breakpoints around the area that wishes to be tested. A breakpoint makes it so that the code is paused on that point, but does not execute the code on that line. A breakpoint can be made by double clicking to the left of one of the numbered lines. Breakpoints will show up as a circle next to the number of the line, and can be made to disappear by double clicking them again. It is recommended that test points be set after the outputs have been set, so as not to confuse you on what is happening with the code. Make sure that when done testing the code, that there are no more breakpoints, which could cause your code to halt at inopportune times.

Additionally, when testing the code with the test points it is possible advance from one test point to the next by simply hitting the continue button mentioned previously. However, if it is desired to step through single lines of code at a time, it is important to use the step function which allows the user to do just that with each click.

A Component Blew Up:

Whenever, this happens it is important to rapidly remove power sources from the devices. Hopefully, this was done before reading this in the manual.

Confirm that all power is disconnected from the circuit. After this has been checked proceed to discover where the problem has occured. This should be able to be done by simply identifying the connections associated with the component which ‘blew up’. Most often in these cases, something was wired incorrectly resulting in the overloading of a component past its tolerable threshold. In the case that the connections seen to the naked eye are correct, then the problem lies within the the previously made circuit connections. For assistance on finding out what is wrong with the PCB, please see the PCB troubleshooting section above. Before consulting the section, it is important to take the component that ‘blew up’ off the board by desoldering it. Once off the board, it is advisable to not put another component on the board, considering that the same outcome will come to fruition. While following the above troubleshooting guide try testing the component pads for correct values instead of the component itself. Additionally, in the case that the component’s function was crucial, you can always simulate it by connecting wires from one pad to another, by hand or through a soldered connection.

How the Control Circuit Works:

Figure D shows the current layout of the PCB. At the top right is where the power supply from the wall plug connects to. At the very bottom is the header that connects from the PCB to the SmartFusion board. Parts were removed as they were unneeded and to avoid open circuits, wires jumping from location to other location can be seen. The pins in the bottom right are what connects to the button inputs for the design. The pins on the bottom left are what connect to the display. The Pins at the top left are what go out to the motors. In the set of two pins, the bottom pin connects to the black wire of the motor while the top pin connects to the red wire of the motor.

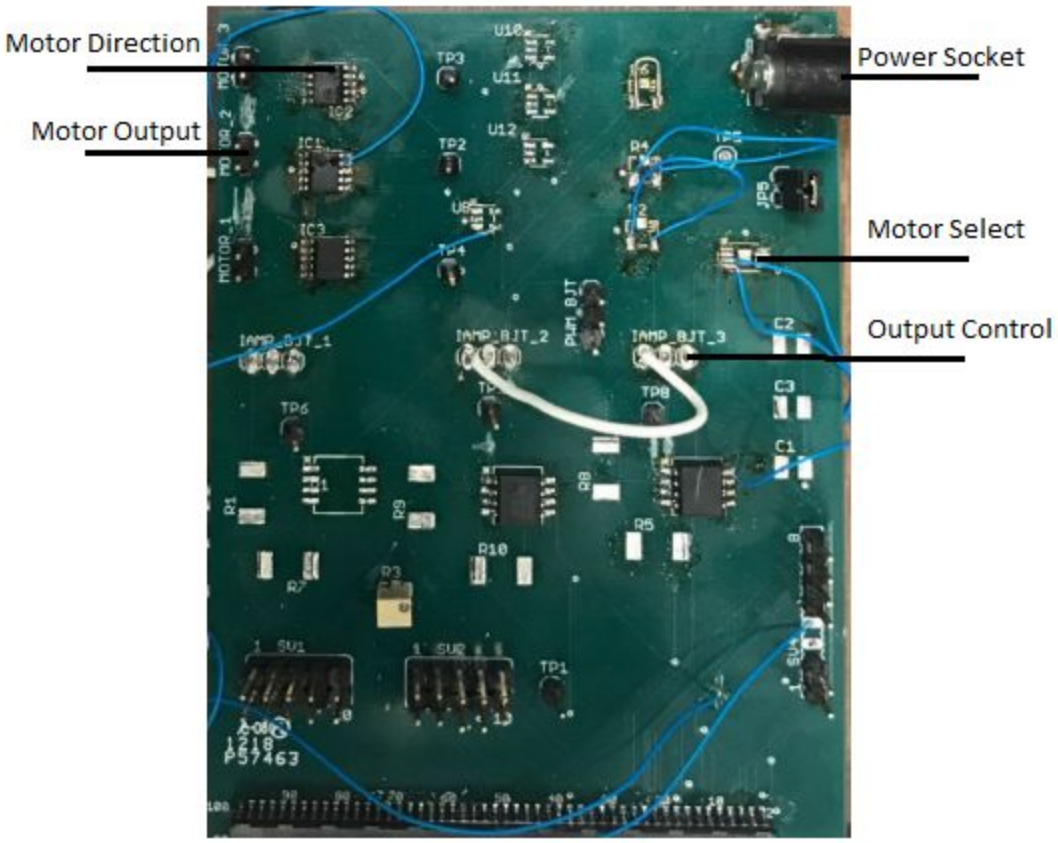


Figure 3: PCB Layout

Voltage to the circuit starts from the power socket, and then immediately runs to the the MUX component which selects which motor is to be on at which times. This was duty was shifted to a proto board off of the schematic, due to the positioning of the MUX on the board being unable to handle the current load of the power supply. After reaching the motor select the circuit would lead into the control of the output via BJTs. These can be thought of as switches which require an external voltage to turn them on. The BJTs were taken off of the board entirely, due to the fact that they were limited by the voltage that was supplied to turn them on. To remedy this issue, relays were used to replace these on the same proto board used for the MUX. A relay is also similar to a switch but requires more inputs and is more bulky and allows for better current to flow through the circuit. This portion of the circuit only has the switch on when the signal controlling the switch is high, which means that with a pulse width modulation controlling the input, the duty cycle being sent to the output can be adjusted. Following the output control is the output's direction control. This is controlled by an H-bridge connected to the SmartFusion, giving it the ability to determine which motor output is considered ground or power. Many believe that they can simply make an H-bridge themselves with the correct components, but it is a complex system which requires specific devices. The H-bridges currently being used on the design can only handle 1 amp of current, before they begin to fry. Finally after running through

these important components the output can be hooked up to the motors, in the configuration previously mentioned.

Status of Planned Features (WBS)

Chassis

The three syringes are mixed in a T formation. Thus, a T-formation base is required to hold the whole device.The holders and the base can be designed on SolidWorks. The team got the design paper with some help from mechanical students. Our team use 3D-printer to create the holders. This part is almost the mechanical part in our project. The final step is to adjust the distance between each actuators.

Circuit

It contains the parts shipped, parts received, Eagle's using, simulation, breadboard connecting, and soldering. Design the circuitry to control the actuator selection, actuator direction selection, and vary the actuator speed, the circuit part is important in the project, and it contained two subsystem to drive input and display. At the beginning, it is necessary to build a text-circuit first. After we finished the circuit layout in Eagle, we made a PCB board to achieve its function. Soldering is based on breadboarding, components soldered from DIP packages onto PCB.

Software

It contains created PWM, relay, H-bridge, timers, and the trouble shooting. Eagle and all codes work is the main necessity for this part. It controls the microprocessor to drive the whole system automatically. Our team choose SmartFusion Eval2, it has peripherals already mounted in the top.

Hardware

It contains actuators, button pad, and display screen. The actuators are used for driving syringes with linear motion. The button pad is used for users input information, and the screen can show the final information when the equipment finished mix.

The microprocessor used in this design is called the SmartFusion. SmartFusion is an System on Chip (SoC) microcontroller. This means it comes with all the tools necessary to develop the system. It does an excellent job at sending and reading signals, generating PWM signals (Pulse Width Modulation, which allows digital outputs to act like analog), and has a full FPGA right on the board. It is coded in C# the goto to language for professional developers in embedded systems. C has a bit of a learning curve however, as engineers we are already experienced in the language. Libero is used to program the controller giving it an easy graphical user interface. Thus the SmartFusion is an easy controller to use as well.

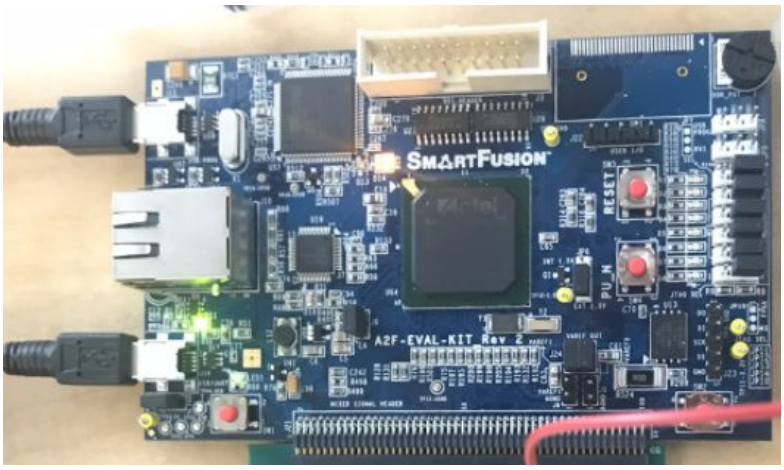


Figure 4: Smartfusion Board

Vincent:

Person Primarily Responsible (Vincent Jencks)					
ID	**Activity**	**Description**	**Deliverables**	**Other People**	**Progress**
0.0	**Purchase Components**				
0.1	Components Purchased	Purchased most components. More components will be added as needed	· Need very Basic Schematic · May need different parts if specifications are not adequate	Andrew	complete
1.0	**Circuitry**				
1.1	PWM	PWM signal and circuitry	-	-	complete
1.1.1	Design PWM Circuit	Design PWM circuit to limit current to motors	· Simulate Circuit · Test circuit with bare parts (no code PWM generated with Arduino)	-	complete
1.1.2	Program PWM Signal	In "Smart Fusion Evaluation Kit 2" microprocessor program PWM signal	· Write VHDL program to send GPIO from high to low synchronous with clock · Create register to store clock period and negative edge · Write to VHDL component though ARM Cortex in C#	-	complete
1.1.3	Test PWM With Motors	Test the full circuit with the motors. Observe speed vs torque characteristics.	· Bread Board Circuit Without Motor · Bread Board Circuit with motor, test for Motor Speed and Torque · Note as speed decreases, torque increases optimal speed will be set to push syringe	-	incomplete
1.1.4	Synchronize Motors	Use PWM clock signal to time motors spin	· Find Amount of clock periods required to fully extend actuator	-	complete
2.0	**Software**				
2.1	Display	Display and communication protocol	-	-	complete

2.1.1	Get SPI protocol working	SPI (Serial Peripheral Interface) used to communicate with external devices	· Signals to program CLK, DO, DI, SS · SS tied to low state due to only one SPI slave	-	Dropped from project display uses parallel signal on GPIO
2.1.2	Connect Display	Talk to Display over SPI line	· Connect the display and send basic information	-	Complete
2.1.3	Program Display to show Text	Display strings and numbers with Display	· Start GUI by programing simple text	-	Incomplete
2.1.4	Program Display to show graphics	Display Graphics and Charts with display	· If Text successful try getting graphs	-	Dropped form project display can only show text
2.0	**Software**				
2.2	Timers	Tracks mixing time	-	-	complete
2.2.1	Enable Timers	Enable the Timers built into Smart Fusion Board Will be used to track mixing time	· Enable Timers to send information to Microcontroller for Displaying information	-	complete
1.0	**Circuitry**				
1.2	Soldering	soldering	-	-	
1.2.2	Solder Components	Solder to Perf-Board	· **All** breadboard and EAGLE layout must be done to begin soldering	Andrew	complete

Components

The parts that were initially ordered were based on the specification of a maximum of 1 amp draw. This ended up becoming a problem later, the actuators that were needed by the end design pulled up to 6 amps of current. Because of this, the redesigned circuit is going to use parts that run a minimum of 20 amps of current through them. The new circuit submission would be for another team to work on.

Circuitry

One the motor selection portion, the main job was assisting in breadboarding the mux and finding current and voltage tolerances.

A current supply was added by taking advantage of a BJT (Bipolar Junction Transistor) current gain. There was a current buffer circuit online that was tested, however it was dropped it due to the convenience of buying a more powerful supply. Current buffer was eventually desoldered from PCB.

The MUX (Multiplexer) was replaced a with relay and made a quick gate driver with inverters to step up GPIO logic to 5V

Assisted with H-Bridges. Worked through the document for the Hbridges, pin configurations, tolerances, and went through the process of wiring it. Then proceeded to test if Voltage flipped when activated successfully.

Designed the PWM by using the FPGA on the SmartFusion board. This fabric component had registers to take in period and negative edge trigger from the microprocessing unit via the APB3 bus. It then outputted corresponding signal to selected pin. This may then go into the gate of a transistor to modulate power to motors. However in order to do so logic signal must be stepped up to 12V to fully turn on transistor and did not have the resources to do such. Resources would be an optocoupler, so 12 volts may be passed in and the 3.3V PWM signal could turn on and off device.

The next step was to synchronize the motors by using on board timers then calibrating it to switch motor directions once countdown stopped. To implement this, an interrupt was created that when triggered would handle by switching motor direction.

Software

Software includes button pad, display, pwm, inputs and outputs, and motor mixing process. Unfortunately things kept getting pushed and the Button Pad and Display functions are not fully developed. However the skeleton for the software is written. The display code however was not able to be completed, although it was done on breadboard. As the circuit kept failing due to current limitations the software for the display and button pad never got picked up again.

Display

The 1602a display for the project was used with parallel asynchronous signals rather than SPI. This was because it was found there were enough GPIO pins to support it. This made implementing the display easier as to program it simply turned signals high per the data sheet. However, as mentioned before after writing the skeleton functions for the code, it never was able to fully implement it into the project as hoped. By skeleton meaning the basic functions such as

clear and write were written. However the display never gave user feedback due to time constraints.

PCB and Soldering

There was also the responsibility for designing the PCB. The schematic for the MUX was done so all that needed to be done is copy it over. Next was setting up the schematic, this included laying out all the components on the board and wiring them, and finally exporting CAM files and made sure PCB had no problems at manufacturing.

The board was modified it to allow more current. Modifications that were made include replacing the mux with a relay, taking out current amplifier, and trying to wire muxes in parallel to achieve 2A current rating. However the muxes were to small and the wires were causing bridging so this was canceled.

Andrew:

Person Primarily Responsible: Andrew VanDenburgh				
ID	**Activity/Task**	Description	Deliverable(s)	Other People
1	**Motor Direction Circuit**			
1.1	Circuit Setup	Test and design the H-bridge and MUXs in a circuit.	• Breadboard circuit • Test data	-
1.2	Integrate	Connect this circuit with the already built motor select circuit on the breadboard.	• Breadboard circuit • Test data verifying functionality	-
2	**Controlling I/O**			
2.1	Simulate	Verify understanding of the coding language and practice with sending and receiving appropriate value types.	• Test data • Structured code	-
2.2	**Setup**	Establish connections		
2.2.1	Button Communication	Translation of what was simulated to being able to read and store data from the user input button pad.	• Test data • Verification of communication • Translation of communication	-
2.2.2	Motor Communication	Translation of what was simulated to being able to write and store data to the motor circuit via the control circuit.	• Test data • Verification of communication • Translation of communication	-
2.3	Integrate 3.2.1 and 3.2.2	Allow button pad input to influence how the motors will perform.	• Test data • Verification of communication	-
3	**Chassis**			
3.1	Design	Establish constraints and determine best configuration	• Working model • Hierarchy of needs/constraints	Working with Colton
3.2	Prototype	Use 3D printing or other objects to test for flaws in design.	• Stability and requirement verification	Working with Colton
3.3	Create and Integrate	3D print final configuration and connect with finished parts	• Working layout • Compact and stable design	-

H-bridge and MUX Setup and Integration

In the PCB design, the H-bridge and MUX were some of the more crucial design components to make the actuators move in different directions and at different times. Testing the dip package components before ordering them or putting them onto the PCB schematic, was the first priority. The testing was completed, however it turned out that the dip package H-bridge on hand was a half H-bridge which meant it was unusable. Due to self-imposed deadlines the order was confirmed for an h-bridge component, before ordering a dip package replacement.

The implementation of the MUX and H-bridge were fully integrated into the Eagle schematic. After being ordered however, the connections of the H-bridges on the PCB were found to be incorrect due to an error in naming convention. To remedy this mistake, cuts and soldered wire connections were made(detailed more in the troubleshooting section), to allow for the circuit to work as intended. Additionally, the MUX was able to be implemented correctly.

Unfortunately, specifications were changed after the implementation of these components. Currently, the MUX and H-bridge are not rated for the new specifications given. With the specifications of up to 5 amps and 12 volts, it is believed that the VTS7960 H-bridge would be a viable option when stepping up the current and voltage. This is necessary considering that the current design can only handle 5 volts and 1 amp, while using the .

Button and Motor Communication

Motor communication, which has since changed to actuator communication, was a great success. The motors have the ability to move back and forth, with the two linear actuators moving at the same time and the third being able to move independently. The basic essential codes to run these operations has been written, with more complex codes unaccomplished. Originally it was envisioned that the code would take the user parameters and adjust the speed of the actuators and timing based on the parameters. These codes were not written.

Button communication followed a similar vein, in that the button pad used was connected to the device and could be read to. However, due to the involvement and interconnectedness with the motor's more complex communication codes, the implementation of multiple buttons went unachieved. The design did manage to implement a start and stop button. The start button can be activated so long as the mixing process is not in progress, and the stop button has the ability to stop the process at any point in time along the process. The stop button, will actually exit the code entirely, which may not always be desired if wanting to be run again.

Chassis

The working chassis for this design was built to meet most of its specifications. While, not fully complete, it was able to function as a sturdy base for mixing, which was the main requirement. An encasement for the circuitry was not created nor was there a stand for the button pad to be held for user ease. All other specifications were met, with the ability to account for syringe size adjustments using clamps to change the actuator position. The only revisions that could be made to the current design, beyond those previously mentioned would be to make the system more easily portable and make it easier for the syringe holders to be removed following a mixing procedure.

Additional Work

Due to an unexpected parameter shift, caused by ignorance of mechanical properties, many of the assignments within the project required extra attention. Mainly this dealt with the verification and modification of the PCB circuit, at all points.

Colton:

Colton Smith					
ID	**Activity/Task**	**Description**	**Deliverable(s)**	**Other People**	**Status**
1	**Analog**				
1.1	**Charge Pump**	Simulate and Breadboard		-	Completed
1.11	Charge Pump Simulation	Create a charge pump capable of taking in a 5V source and output various output voltages	• Working LT Spice simulation of 12V, -12V, -5V charge pump	-	Completed
1.12	Charge Pump Breadboarding	Apply values and knowledge from simulation with actual parts	• Functioning charge pump able to provide rail voltage to Op Amp	-	Completed
1.2	Buffer	Creation of buffer to convert voltage from a high impedance to a low impedance	• Device capable of helping drive motors despite potential voltage divider qualities	Handi	Completed
2	**Digital**				
2.1	EAGLE	Working with Eagle to connect all circuit components into on schematic	• Eagle schematic for PCB board layout	Yunchen	Completed
3	**Hardware**				
3.1	Chassis Creation	Creation of case to hold circuitry and motor for the final design	• Single containment unit to hold all components	Andrew	Completed

Charge Pump

As stated in the WBS, all of these tasks were completed as planned. The first task is the charge pump. This was then broken down into two parts, the simulation and the breadboarding. Under the first iteration of our project, the supply was going to be a 5 volt source so to step up the voltage if twelve volt motors were going to be used, the supply would need some kind of rails for the op amp (operational amplifier) to get the voltage to work for 12 volts. The initial plan was to have multiple charge pumps that fit multiple situations. After after proving concepts, the next step was to use physical components to test. Both inverting the voltage for the negative rail while also doubling the input voltage to bring it up and a positive rail, functioned as planned. After this design though, the use of a different supply would would not need to use an op amp of any sort. A different supply was used, as such this part was not need. While dropped from the final

design, the team wishes they had a 12 volt low current signal for turning on MOSFETS without having the saturation current limit.

Buffer

The goal of the buffer was to avoid a potential problem in the design. The power supply would have some sort of inherent resistance inside of it. If it were to attach the motors, which would also have some resistance inside of them as well, a voltage divider may be created unintentionally. In the case of a voltage divider, the motor would not be receiving the proper voltage. So this buffer circuit was used both to combat that problem while also helping step up the current going to the motor. In the end this too was pushed to the side, the better supply would not need to step up the current. As for the voltage divider problem, it seemed to be a non-problem. As such this circuit was not necessary.

EAGLE

The job was to help provide the schematics that had been simulated to EAGLE so that the PCB could be set up. As the previous section stated, both of the above devices were not necessary to have, so they were not included inside of the PCB. As such, the EAGLE portion of this WBS amounted to nothing, but nevertheless complete.

Chassis

There needed some form of frame to be holding the design so that it would stay steading while mixing the syringes. This meant that both the motors would need to be locked in as well as the syringes. This did end up being completed but only for the actuators and the syringes, there was not time to put the smartfusion or the PCB into some protective casing like initially hoped for. The project went under a massive change which will addressed next.

The Redesign

The plan was to order the PCB before March, 19 so that it would come the week following when the team was available to work again to effectively use the campus wide break. Because of this, the PCB ended up being rushed out to meet the planned goal. The following week the client informed that the design would not work even remotely for mixing the syringes. The circuitry had been prepared expecting up to 12 volts and half of an amp of current. Instead, the client and the mechanical engineers stated a pushing force around 20 pounds (25 pounds including a safety factor) for pushing the syringes. The motors we had been testing with would only be able to do a fraction of that. As such, almost all of the components purchased were not rated for these operations. Because of this, the next step was looking through actuators. While looking for solutions it was realized the PCB had errors in the wires connecting the parts. From there it was a matter of cutting lines or soldering on wires for all of the errors on the board while also having create a new circuit with the parts that currently owned, as recreating a PCB and getting parts would stretch the budget as well as the time available. In the end this caused other plans to be

pushed while working on a functional solution. In the end, the project did not work due to the parts not being rated for the actuators as well as running out of time.

During this whole process, since the previous activities were completed, new activities included helping with the soldering and finding the errors in connections in the board. It was a processes of taking three steps forward and then two steps back. The main process that took up a majority of the time is that soldering onto a PCB takes time, and if solution did not work it was a matter of desoldering and trying another potential solution.

Yunchen:

Person Primarily Responsible: Yunchen Zhu					
ID	**Activity/ Task**	**Description**	**Deliverable(s)**	**Other People**	**Status**
1	**Digital Circuitry**				
1.1	Design Circuitry	Design the control circuit between MUX and microprocessor	• Circuit schematic • Know datasheet of MUX well	-	Completed
1.2	Motor Selection	Write code for Console to generate the logic outputs as switches of MUX	• Set MSS GPIO ports • Use PuTTy to test outputs	-	Completed
1.3	**Construct and Test Circuits**	Build and test.			
1.3.1	Multiplexer	Construct and test circuit with multimeter.	• Breadboard circuit • Test data • Verify functionality	-	Completed
1.3.2	Microprocessor control circuit	Construct and test circuit with LEDs and buttons.	• Breadboard circuit • Test data • Verify functionality	-Handi Xi	Completed
1.3.3	Integrate 1.3.1 and 1.3.2 circuits	Integrate two breadboard circuits into one.	• Single breadboard circuit • Test data verifying functionality and linearity	Testing help from Vincent Jencks	Completed
2	**PCB design**				
2.1	**Build schematic**	Create a clean final schematic with labeled parts	• Add parts to a schematic • Wire up the schematic • Making named, labeled net stubs		Completed
2.2	**Lay out the PCB**	Turn from the schematic editor to the related board and make the layout.			
2.2.1	Arrange the Board	Arrange the parts and minimize the area of our PCB dimension outline.	• Create a board from schematic • Move parts • Adjust the dimension layer	-Colton Smith	Completed
2.2.2	Routing the Board	Turn each of gold airwires into top or bottom copper traces and check them.	• Place vias • Route clearance • Rip up traces	-Colton Smith	Completed

Two activities are involved in this part: Digital Circuitry and PCB design. As the WBS chart shows, all the tasks were completed successfully without delay.

Motor/Actuator Selection

The first task is to design and partially build the control circuit. In order to mix the syringes, the team needs a control circuit to select which actuator is needed at the moment. The actuator should be controlled by peripheral software like SmartFusion Console. To begin with, multiplexer is functional to switch and select signals.The team constructed and breadboarded it with two buttons and two LEDs to test the function. After the function is verified, the team hooked up the MUX with the SmartFusion. The only thing that can be considered as a challenge was the code part. The team had difficulty writing the whole part but finally solved it. The code on Console for our microprocessor to generate the logic outputs as switches of MUX were completely written. In the process, PuTTy is a useful tool to test the results after debugging the code. The combined circuit can select output signals with Console.

PCB Design

The PCB part is the one of the most important tasks of the whole project. It also can be broken down into two sub-tasks: Schematic design and PCB layout.

The team tested the H-bridge circuit and created the library for the whole schematic design. First of all, the list of the parts was confirmed and the team ordered them. Based on a draft of the whole design. All the major components that were likely to be used have the libraries from Digi-key except the female header. For the components whose libraries were provided by Digi-key (like amplifier and H-bridges), the team downloaded the EAGLE CAD models and imported them on the schematic. The header model which was another challenge for the team because there was not a ready-made model for the connection of boards. Then all the components were placed after the libraries were finished. The team wired them based on the circuit design.

The second part was to lay out PCB. After the schematic was build, the team can be able to create a board from it and optimally moved and rearranged the parts.Then the team routed the board with automatic check. After everything was done, the PCB files was sent to the factory for manufacturing and printing.

Follow-up work

After all the assignments were finished, the team started with the work that kept the whole project moving. We arranged the syringe position on the base and found the optimal structure for combining all parts. Almost all the follow-up work involved PCB alterations.

Handi:

Person Primarily Responsible: Handi Xi					
ID	**Activity / Task**	**Description**	**Deliverable(s)**	**Other People**	**status**
1	**Digital Circuitry**	leave this line blank, for the highest level of activity			
1.1	Design Circuitry	Distinguish all part, draw the schematic how to build the circuit.	• Circuit schematic • Simulation	-	completed
1.2	Construct and test circuits	On the basis of the simulation, build the circuit for driving the motor.	• Connection circuit • Simulation	Yunchen	completed
1.2.1	Motor selection	Programed to select which motor drives.	• Code • Circuit schematic		completed
1.2.2	Microprocessor control circuit	Use SmartFusion, constructed test of circuit with LEDs and buttons.	• Code • Test data • Breadboard circuit		completed
2	**Analog Circuitry**				
2.1	Breadboard buffer	Transfer voltage from high impedance to low impedance to drive motors.	• Single breadboard circuit • Test data • simulation	Colton	completed
2.2	Current source	Transfer current by simulating, integrate two breadboards into one	• Soldered circuit • Test data • simulation	-	Not need

This is the breakdown structure, it contains two parts, the digital circuit design and the analog circuitry.

Motor/Actuator selection:

This part is used for the motor selection. Follow the client's needs, this project should mix three syringes, the team should choose which one to run. Team members choose motors at beginning, but finally found that actuator can provide a linear motion, so the team chose to use actuators

replace motors. The first step is design circuitry, put the components on breadboard and be familiar with their function. In this part the team used SmartFusion to control, and this microprocessor board was used for change signals and timers, in order to drive the whole system automatically. Team decided to use multiplexer to switch signals. The text part is to worked with a team member, and built a breadboard with two LEDs and two buttons to text the selection function. After the function is verified, team members hooked up MUX with SmartFusion board. For the microprocessor, the team decided to use programming on Console to create the logic outputs. When doing the code part, there comes some challenges, so members asked Vincent for help. Finally, this part was finished.

Breadboard Buffer:

This part was created by two team members. The current source has internal resistance, but the motors have small resistance, too. It is necessary to transfer voltage from high impedance circuit to low impedance circuit to ensure the team can drive the motors. Team members simulated it on LTSpice first, and created it on breadboard with op-amp. After those texting, our team connected buffer and charge pump. At the end, this part was finished finished, but the voltage drive problem did not as our team expected before. In our finished product, the breadboard buffer did not show in it.

Current Source:

When the team do the breadboard buffer part, there had a plan to create the current source at the same time. When team members simulated the buffer, the project was needed to build a current source possibly. Each motor had rated current, so before the team changed the current value, the current may be not the value that our team need. Our team chose to use LTSpice to simulate it. However, the current source does not need at end. So this part did not show in the finished product as well.

In conclusion, it is worth to do this project, this project contains almost all learning points. For myself, the inadequacies is lacking of experience about how to complete a project before, so always feel confused about how to do, what to do in the next step. Especially the programming, the team considered it as the biggest challenge, but team members were all kind to help, all team members are hardworking in their parts. Though the display part in the finished product is not perfect as our team expected, this project is still a great progress for the team.

Future Improvements

As previously stated, there were things that should be changed or adjusted and will addressed in this section.

In the appendix there will be included a new PCB that is crafted with the new knowledge of the project. Superfluous parts of our previous design will be removed and new parts will be suggested for replacement as they fit the new current and voltage levels that are for the actuator. The schematic for this part will be inside a flash drive with most other information that will be needed to start fixing the syringe mixer. Having another person to check over the schematic who is familiar with EAGLE will be a good idea as while the schematic will be created to work, minor mistakes may still exist. It is far more frugal to measure twice and cut once.

First part suggestion is for a new H-bridge. The BTS7960 H-bridge is a power motor controller capable of handling up to 24 volts and 43 A, well above what this syringe mixer will be doing. This suggestion is because the previous H-bridge that was used was only rated for 1 amp. With the actuators running possibly up to 4-5 amperes this part was not only non-functional but also a fire hazard.

Next is quality relay to be using. The G5LE relay is capable of handling up to 30 volts and 10 amperes. Again, these parameters fit well into the specifications for the actuators while the previous could handle up to 2 amps which in most cases the syringe mixer will never reach but pulling a part that close if never a safe idea.

Conclusion

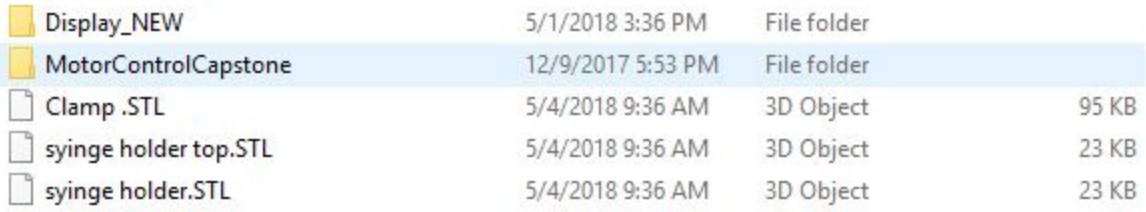
We would like to thank Dr. Timothy Becker and ATI for sponsoring this project. We hope that this device will fulfill their needs to automate the syringe mixing process to further develop and understand the PPODA-QT liquid embolic method to ultimately bring it to human testing and finally to the market. This has been wonderful learning experience for our team both with communication as well as technical experiences.

Although we are heading out to our professional careers if there questions on the project, the design, or any other questions please feel free to contact us at our emails on the cover page.

Best Wishes,

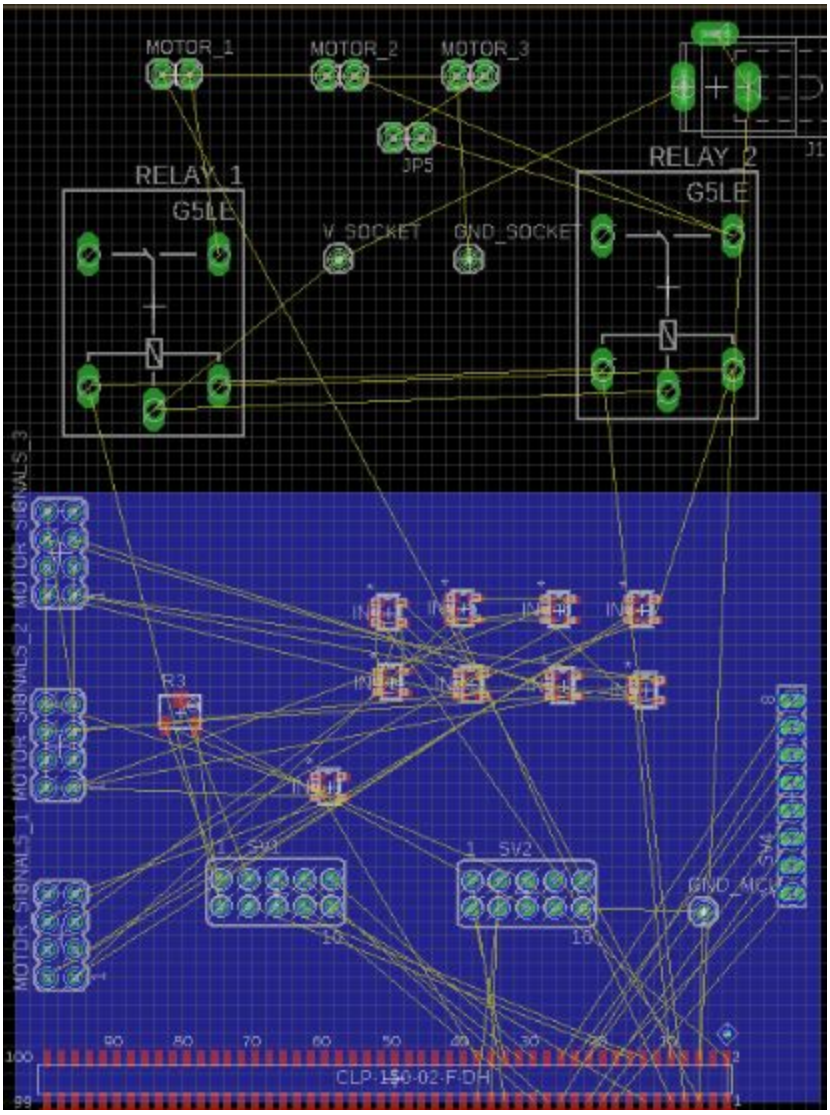
The 3-Way Syringe mixing team.

Appendices:



Display_NEW	5/1/2018 3:36 PM	File folder	
MotorControlCapstone	12/9/2017 5:53 PM	File folder	
Clamp .STL	5/4/2018 9:36 AM	3D Object	95 KB
syinge holder top.STL	5/4/2018 9:36 AM	3D Object	23 KB
syinge holder.STL	5/4/2018 9:36 AM	3D Object	23 KB

The Display_NEW folder contains the PCB board schematic. Again it is suggested to check connections on this new PCB to make sure everything is connected properly. The MotorControlCapstone contains all of the code that we used for the project inside of SoftConsole. The STL files contain the 3D parts that were attached to the base to hold the syringes and interface with the actuators. Additionally, there is a parts list document detailing the different parts to be used within the newer schematic.



This is the PCB board layout that is currently being worked on. This should include all changes that were done on the previous board as well as removing all unneeded parts.

References:

- Microsemi. (2017). SmartFusion Evaluation Kit. [online] Available at: https://www.microsemi.com/products/fpga-soc/design-resources/dev-kits/smartfusion/smartfusion-evaluation-kit#documents [Accessed 13 Oct. 2017]. (Picture of the SmartFusion)