Remote Control

Final Report

Anson Kwan

Shawn Jordan
EGR 304
Arizona State University

December 7, 2020

Problem Definition

The purpose of our project is to create a twenty or more step chain reaction machine (CRM) that utilizes wireless communication to help to entertain and teach space travel to the population during this pandemic. Table 1 as shown below is our project specifications and constraint setting.

Table 1.
Project Performance Table

#	Metric	Unit	Marginal Value	Ideal Value
1	Number of "steps"	#	5 / student	# justified by design
2	Product Cost	$	$25 / student	< $25 / student
3	Power supply	#	≥ 1	# justified by design
4	Cypress PSoC microcontroller	#	≥ 1	# justified by design
5	Different analog sensor(s) read by a microcontroller to control actuator(s)	#	≥ 2	# justified by design
6	Actuator(s) controlled by a microcontroller based on sensor data	#	≥ 2	# justified by design
7	Wireless communications to other students' projects	#	≥ 2	# justified by design
8	Serial communications to communicate with another chip or device	#	≥ 1	# justified by design
9	Functioning custom printed circuit board	#	≥ 1	4
10	Programming language	Programmed in C/C++		
11	Functioning during final demonstration	%	50	100
12	Wireless response time	sec	5	1
13	Voltage of power supply	VDC	>3.3	⩽12
14	Runtime of CRM	sec	>60	>120
15	Dimensions of CRM	ft^2	4	< 4
16	Weight of CRM	lbs	<30	<20
17	Upper noise limit	dB	<70	<60
18	Lower noise limit	dB	>30	>50
19	Life of power supply	hrs	>10	>30

Design Concept

Figure 1-4 depicts the storyboard which is the basis for the design of the chain reaction machine with the purpose of entertaining and educating the customer. Figure 5-8 is the realization of this storyboard in a chain reaction machine format that will be our final design.

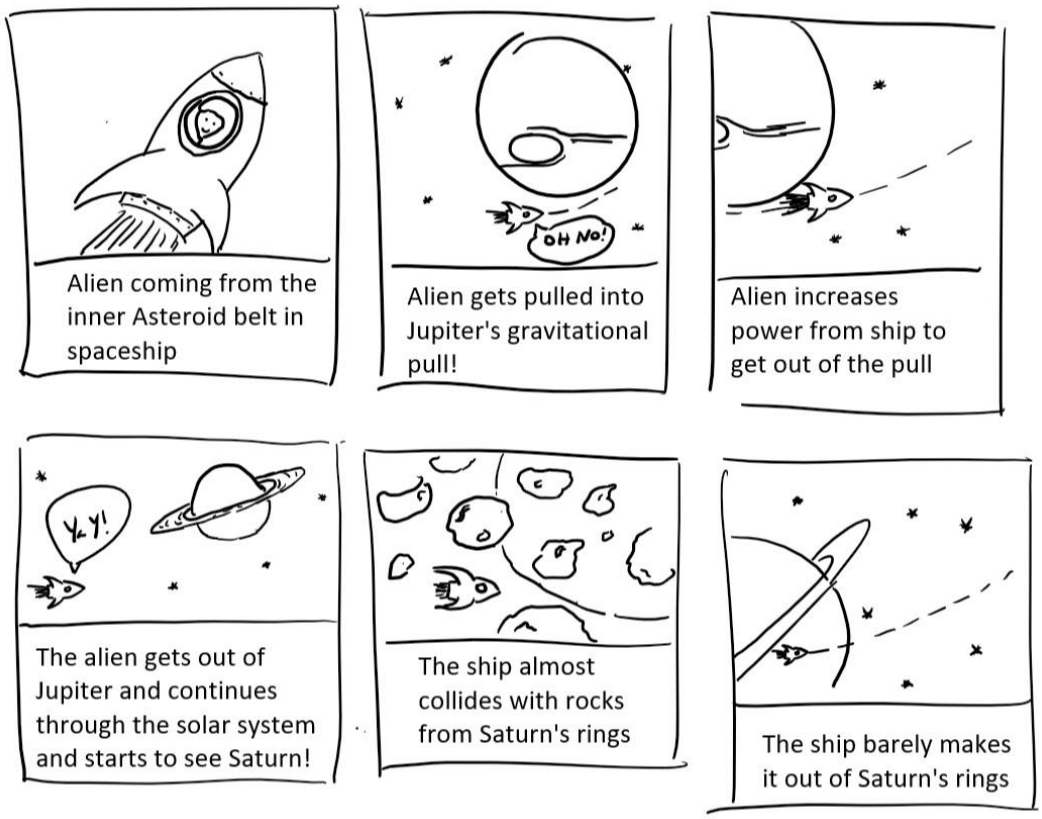


Figure 1. Anson's Storyboard

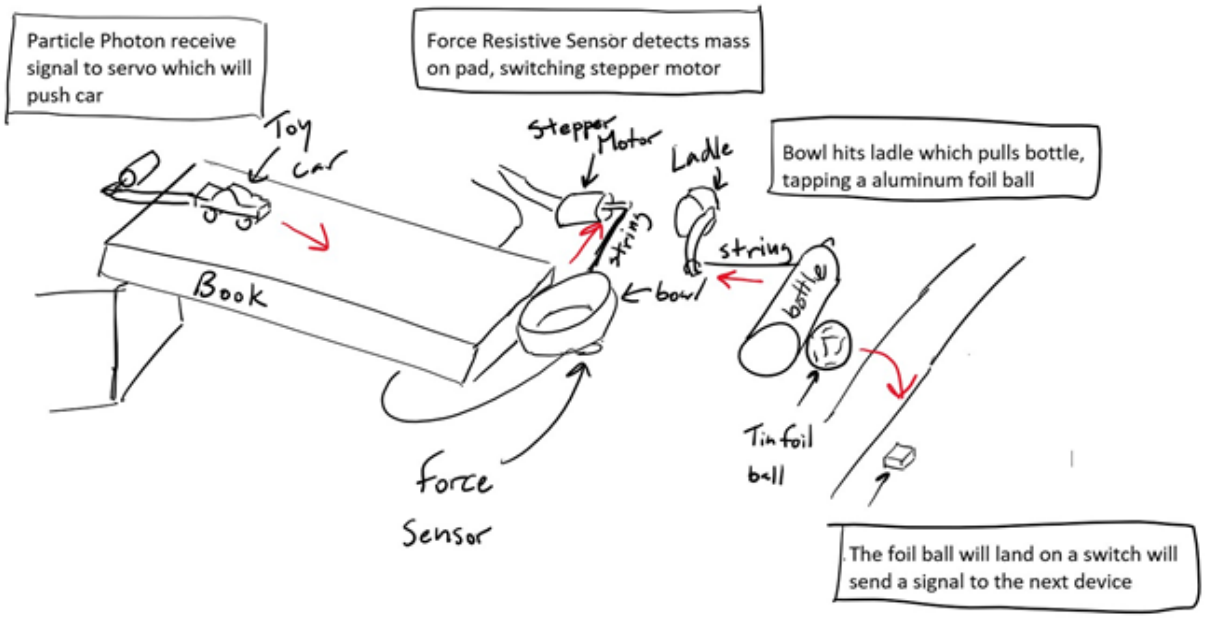


Figure 2. Anson's visual design concept

Final Electrical Block Diagram

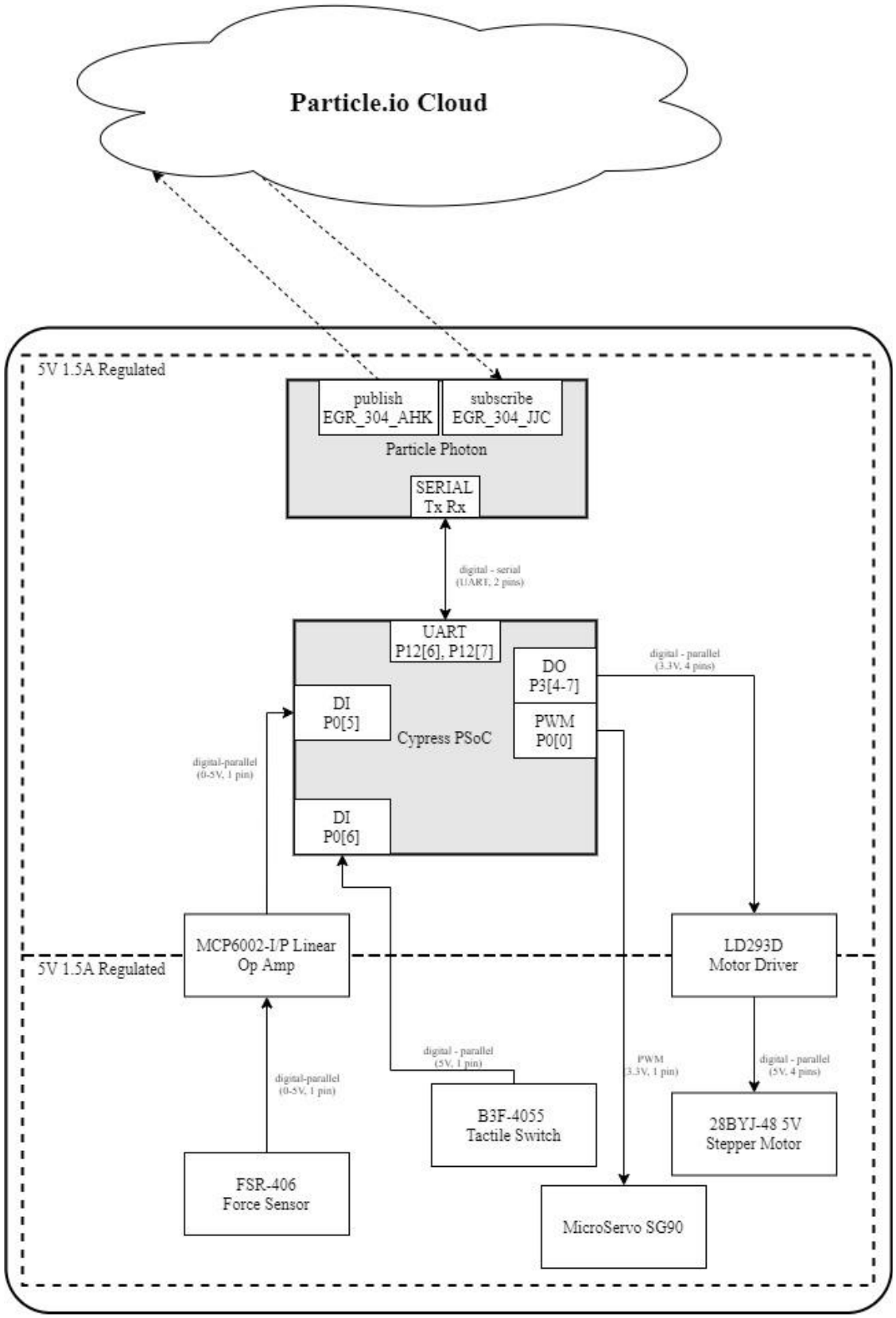
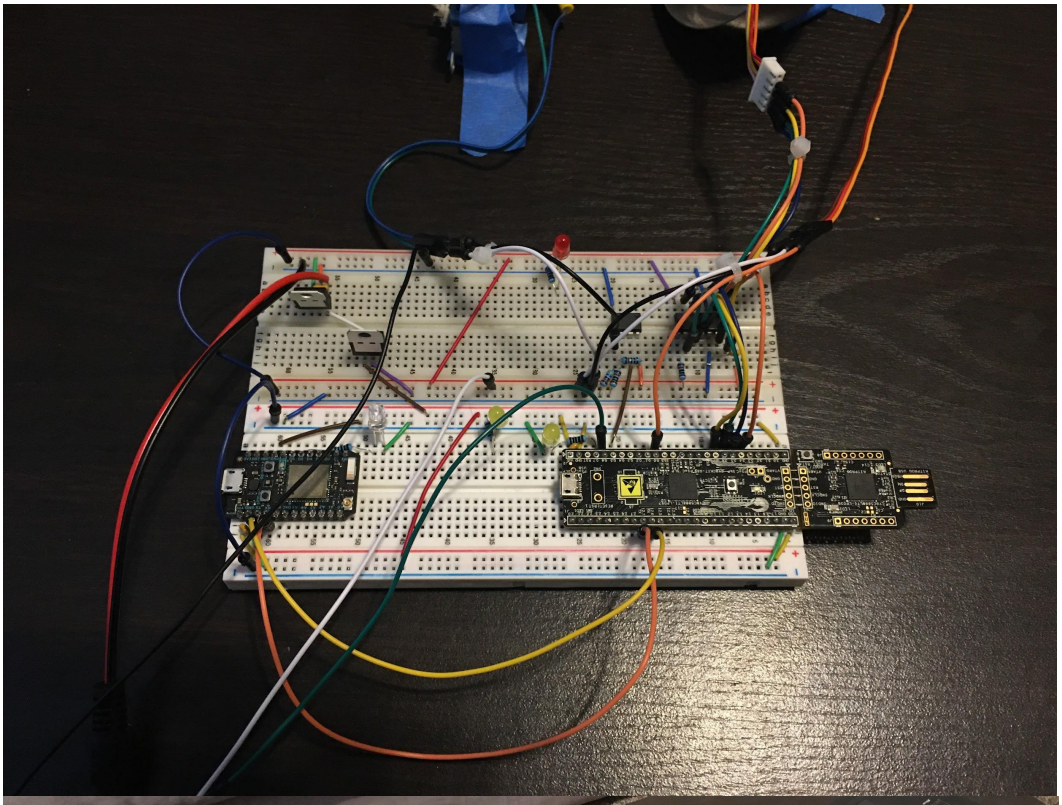
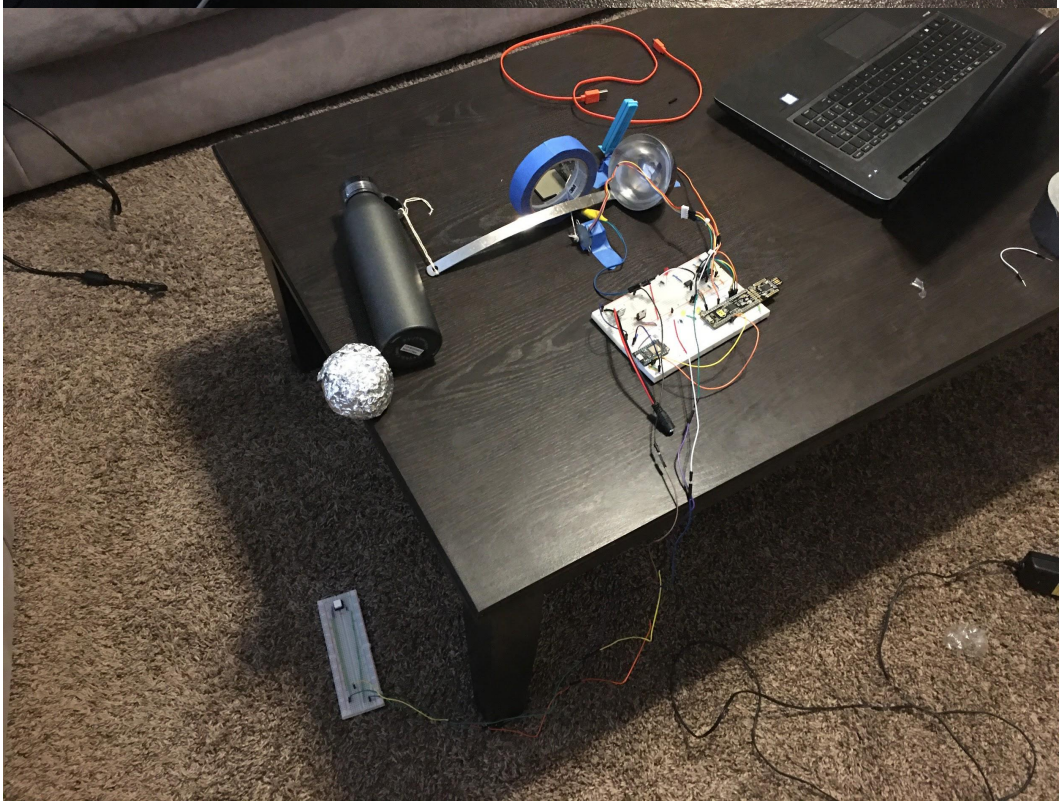


Figure 3. Anson's block diagram.

Progress, Lessons Learned, Version 2.0+ - Anson Kwan

a. **Project Photo(s)**





b. **Hardware Design**

i. Final Schematic

[See the next page for my schematic diagram]

ii. Changes

1. The main issue of my schematic was that by powering the PSoC and Photon via a 3.3V was unreliable for the Photon and was having trouble connecting to the internet. From Design Review, an external reviewer suggested that running your microcontroller at a lower voltage than the possible inputs to the microcontroller is not good practice. In order to fix these issues, I switched the 3.3V regulator to a 5V regulator which fixed the issue of my photon connecting to WiFi as well as enforced better schematic design habits.

2. The circuit for the L293D motor driver had two capacitors at the two enable pins which were unnecessary. Therefore, two resistors were added instead to act as pull up resistors in the circuit. I debated whether or not to connect the enable pins to the PSoC for flexibility but since I am unfamiliar with programming the PSoC, I decided to write the pins high in hardware to make the software more simple.

3. Initially, the test LED pin that I selected on the PSoC was P[1]_3, but after looking in PSoC Creator, this pin is assigned as a debugging pin that can not be rewritten. I therefore switched the test LED pin to P[0]_7 which was easily accessible on the breadboard circuit used for my final project.

4. The reason that the switch circuit was changed to pull down resistor from a pull up resistor was a matter of convenience and compatibility with my downstream neighbor for my chain reaction machine. This is because the code provided initialized the compare value used to start our machine with zero and since a button with a pull up resistor would signal an active low (zero) when pressed distinguishing between the initial value and the button pressed would require more additional logic on the receiving ends side. To avoid this, switching the button with a pull down resistor means that the signal sent when the button is pressed is a one instead which is different from the initialized value.

5. When initially designing my schematic, it was my understanding that the stepper motor had to be powered independently from the motor driver to supply enough voltage to the motor. However, during testing and verification of each subsystem the motor was heating up very quickly as well as the 5V regulator which meant that there was a short circuit. After some testing, it was found that the stepper motor that was used did not need an additional power so by removing the 5V supply directly to the motor it fixed the short circuit.

iii. If I were to create a version two of my hardware design the first thing that I would change would be the components that simplify the software portions of the project. For example, the force sensitive resistor is connected to an op amp that is configured as a comparator which made the software much more simple. In a second version, I would change the configuration of the op amp to a non-inverting amplifier so the system has more

flexibility in other aspects of the device. Additionally, it is better practice to set parameters and logic in software instead of in hardware to allow the maximum flexibility for the design process. I would also change the tactile switch usd to another button with lower resistance to being pushed such as a microswitch or limit switch. The reason for this is the switch used requires that some amount of weight with a certain amount of force is needed to press the switch. In the case of my device, a ball was dropped to press the switch which was hard to line up and difficult to find an object with enough mass to press the button. If a lower resistance switch was used, the design could have been much more compact as well as reliable since pressing the button would require less force. Finally, the last major change that would be made is that instead of using a unipolar stepper motor, a standard DC motor would have sufficed. Initially, a stepper motor was used since I was unsure if I would want to change the direction of the motor but as the project progressed, it was determined that the motor would be only going one direction. Changing motors to a DC motor would simplify the software and runtime of the device. Since stepper motors are slow, this was a major hold up for the rest of the device. If a DC motor was used, this would have greatly decreased the runtime of the system. Additionally, the code to turn on a motor only requires a PWM signal opposed to writing to each drive pin of the stepper motor and requiring delays in between each sequential step. In an attempt to design a more efficient state machine, a DC motor would have been a better choice for this design.

c. Software Design

i. UML Diagram

[See the next page for my UML diagram]

ii. Changes

1. Before design review, it was planned that a signal was constantly going to be published to the photon and updated in the main loop of the PSoC code. However, due to the constrictions of the MQTT server, it became clear it was a better idea to only send a signal when the button was pressed. This was achieved by having an if statement with the condition of if my button state was high it would publish a message to my photon. Additionally, there was a counter to ensure that the message was not sent too many times.

2. The stepper motor was initially only going to run for a short duration of time. After building my mechanical system, it was clear that the motor was going to run for a much longer duration of time so it required increasing the counter of the stepper motor to 1500 cycles. This was accomplished by incrementing a counter variable within an if statement.

3. Sending just a single character or integer to the cloud was what I had initially planned to execute but after the example code, adding a starting and ending character to my message was necessary. Additionally, using the sensor data to send the message also was important to accomplish. To do this, I used the sprintf() function in C to create a character array from my sensor data and then published that string to the photon.

4. Another software component that was added to the system was a reset for the Servo. Since my design only required the servo to actuate in one direction, this required that I constantly write to the PWM block so the sero moved for the entire duration of my chain reaction machine or I reset the position of the Servo each time I turned on the system. For the ease of testing my design, resetting the position of the servo made it much easier for the reliability of the system.

5. Due to not being familiar with C/C++ syntax, I had planned on using boolean statements to determine states of my device but since there are no boolean statements in C without adding any libraries, this had to be changed to ones and zeros to define point in my state machine.

iii. For a second version of the software for my project, I would try to understand more about programming in C to potentially better organize and optimize my software design. I would organize all my actuators and sensor code into separate functions in order to minimize the amount of code in my main loop to make debugging more efficient since it would be easier to locate each portion of my code. Additionally, I would also try to learn more about interrupts and incorporate them into my state machine. During debugging, I ran into an issue when I was trying to run my servo and stepper motor at the same time but both require delays after writing to the pins. In this case, the delay for the servo motor was much longer than the delay for my stepper motor which caused some issues that

resulted in my stepper motor not working properly. In order to fix this, I set conditions and counters to get my stepper motor to run at a different point and duration for my project but that required additional variables and logic that were not strictly necessary. In another iteration of this device, using interrupts would optimize the code by decreasing the number of delays I would have to use in the code. This would additionally fix the issue that the time it took for me to receive the code to the start of my machine was delayed due to the way that I wrote my program. Finally, adding an LED screen to my hardware and software would be useful to display messages about the progress of the device would be helpful as a debugging tool but also as a visual aid to the progression of the chain reaction. In the beginning phase of software development for this project, I used the serial communication with PuTTY to read the inputs and outputs of my sensors which was helpful. However when running the hardware via a power supply the signals being received and sent was very much a black box. If I had added a subsystem for a display, this would help bring some transparency to what is the state of the software. Additionally, this would allow for some text to explain more of the storyline of the device whereas the device I created was abstractly correlated with the storyline that was initially planned. Overall the software that I created for this iteration sufficed for the scope of this project but in further iterations I would hope to bring the quality of work that is more reliable and have an overall quicker response time, so that the software does not constrain any portion of the design of the device.

d. Lessons Learned

1. The basic understanding of different electrical components and how they interact with each other as well as sensors and actuators

2. The bridge between software and hardware and how different circuits affect the software which is very helpful to my understanding of programming and hardware design

3. Being able to use datasheets to determine if a part will be applicable in the application that it will be used by discerning what information is necessary and what isn't

4. Desigining and formatting a schematic using software, specifically Capture CIS to create a circuit

5. Creating footprints and converting a schematic into a PCB layout and then prepare that PCB for manufacturing

6. Understanding that sensors are basic electrical components that are applied for specific environments which is helpful to my understanding of how to utilize these sensors

7. Learning more of how to code in C/C++ and getting more familiar with pointer based logic in programming

8. Serial communication with USB and how to communicate from a microcontroller to computer

9. Learning how to write microcontroller communication via Tx and Rx pin to another microcontroller to communicate data back and forth

10. Publishing and subscribing to an MQTT server and how to receive and process data from the server

e. Recommendations for Future Students

1. It is very important that you take the time to test out the individual subsystems of your circuit as well as running the whole system on a breadboard before manufacturing your circuit on a PCB. Even if the datasheet says that certain configurations are possible, it is worth double checking these assumptions before creating permanent traces on the PCB.

2. Learning C/C++ early on in the semester or learning it before taking the class will be very helpful to grasping example code as well as implementing your own logic into example code. Lots of code will be provided for you to use but it does not mean anything if you are having trouble understanding the logic flow of the code, so getting comfortable with reading code will be a very useful skill.

3. Starting early on assignments, especially later on in the semester this will save you from having to scramble at the end of the semester along with your other classes to try and finish your project. Generally speaking in this class, if you are able to ask questions early about an assignment, the better off you are at understanding the material.

4. Don't be afraid to ask multiple people for feedback on your schematic and PCB design. Even if one person might say that your circuit looks good, it is always good to show at least one more person because everyone is checking for different things in schematics.

5. Keeping your project as simple as you can will help tremendously with following correct procedures and the success of your project. This class forces you to absorb a lot of information so by minimizing the complexity of your design will allow you to better implement the information you are learning since the dots to connect between the information you learn and how to apply it is a much smaller step.

Appendix I - Ansow Kwan

a. Specifications

Project Performance Specifications

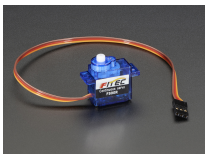
#	Metric	Unit	Marginal Value	Ideal Value	Actual Value	Evidence and Discussion
1	Number of "steps"	#	5	# justified by design	5	**Hardware:** N/A **Software:** N/A **Mechanical:** Determined by the storyboard and means of denoting story using minimum steps
2	Product Cost	$	$25	< $25	$12	**Hardware:** The cost of the sensor and extra passive components and op-amp **Software:** N/A **Mechanical:** N/A
3	Power supply	#	≥ 1	# justified by design	1	**Hardware:** a 3.3V and 5V rail will be used determined by used components **Software:** N/A **Mechanical:** N/A
4	Cypress PSoC microcontroller	#	≥ 1	# justified by design	1	**Hardware:** Used to communicate with the sensor/actuator subsystem and with Particle Photon **Software:** N/A **Mechanical:** N/A
5	Analog sensor read by microcontroller to control actuator	#	≥ 1	# justified by design	1	**Hardware:** A Force Sensitive Resistor with an op-amp circuit will provide input to a microcontroller **Software:** PSoC will read input of op-amp and signify start of actuator **Mechanical:** N/A
6	Actuator controlled by a microcontroller based on sensor data	#	≥ 1	# justified by design	1	**Hardware:** Stepper motor and motor driver circuit will be connected to PSoC **Software:** C code will be written to communicate from PSoC to motor driver after conditions met from sensor **Mechanical:** N/A
7	Wireless communications to other students' projects	#	≥ 2	# justified by design	2	**Hardware:** N/A **Software:** Photon and PSoC program will be written to relieve signal and send signal over Cloud **Mechanical:** N/A
8	Serial communications to communicate with another IC	#	≥ 1	# justified by design	2	**Hardware:** PSoC will be connected to op-amp and motor driver to communicate with sensor and motor **Software:** Pins will be connected to communicate with IC **Mechanical:** N/A
9	Functioning custom PCB	#	≥ 1	1	1	**Hardware:** Components have been chosen to work on power rails and rated for current output **Software:** N/A **Mechanical:** N/A
10	Programming language	Programmed in C/C++				**Hardware:** N/A **Software:** PSoC and Photon will be used **Mechanical:** N/A
11	Functioning during final demonstration	%	50	100	50% +	**Hardware:** N/A **Software:** N/A **Mechanical:** N/A

Team Performance Specifications

#	Metric	Unit	Marginal Value	Ideal Value	Actual Value	Evidence and Discussion
1	Wireless response time	sec	5	1	1	**Hardware:** Particle Photon will be used to create both Client and Server classes. **Software:** Minimizing the size of data being sent between client and server to decrease lag. **Mechanical:** N/A
2	Voltage of power supply	VDC	>3.3	<12	5V	**Hardware:** A 12V 3A AC Power Adaptor will be used and will power a 3.3V and a 5V power rail **Software:** N/A **Mechanical:** N/A
3	Runtime	sec	>30	>60	25	**Hardware:** Hardware and circuits were selected to minimize the amount of programming **Software:** Particle Photon will be programmed to relay signals as quickly as possible to maintain fluidity for device **Mechanical:** At least 5 steps to ensure that the runtime of the device is around thirty seconds
4	Dimensions	ft^3	8	< 8	4 ft^3 +	**Hardware:** N/A **Software:** N/A **Mechanical:** Components of the device need to be spread out a minimum distance to be viewed and properly function.
5	Weight	lbs	<30	<20	5 lbs	**Hardware:** Components were selected to be as light as possible **Software:** N/A **Mechanical:** The device is going to use a minimum of PSoC, Photon, PCB. Including all other components, the weight will suffice.
6	Upper noise limit	dB	<70	<60	50	**Hardware:** N/A **Software:** N/A **Mechanical:** Portions of CRM will be designed with low impact and noise mechanisms.
7	Lower noise limit	dB	>30	>50	35	**Hardware:** Stepper Motor was produces low level noise when used **Software:** N/A **Mechanical:** Steps in CRM will include minimum movement and interaction between components.
8	Life of power supply	hrs	>10	>30	30+	**Hardware:** Selection of voltage regulator and AC Power Supply have undetermined lifespan but are in the years range **Software:** N/A **Mechanical:** N/A

b. Major Component Selection Rationale

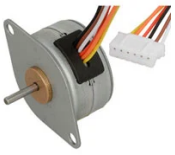
Servo Motor Selection

Solutions	Pros	Cons
Micro Servo SG90 $2.99/each link to product 	• Inexpensive • Mounting holes easy for installation • No additional circuitry to interface with PSoC	• Rotation is limited to 180 degrees • Higher dead bandwidth
Continuous Servo FS90R $7.50/each link to product 	• No angle limit of servo • Mounting holes easy for installation • Small housing profile	• More expensive • Lower torque of output shaft

Choice: MicroServo SG90

Rationale: The MicroServo SG90 is much less expensive than the other option. Additionally, the limiting factors of the servo are easily overcome by the design of the device. The only purpose that the servo will hold is to apply some kinetic energy to the next object in the chain reaction machine. So the cons of this component will likely never be reached.

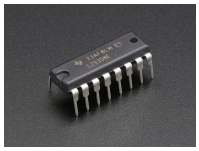
Stepper Motor Selection

Solutions	Pros	Cons
Unipolar Stepper Motor 28BYJ-48 5V-DC $4.95/each link to product 	• Inexpensive • Double D shaft for ease of mounting • Large diameter shaft	• Low resolution (64 steps/rev) • Loud during operation
Unipolar Stepper Motor 25BY4801 4-Phase 5VDC $8.95/each link to product 	• Smaller motor footprint • Easier access to mounting holes	• More Expensive • Small diameter of shaft • Low resolution (48 steps/rev)

Choice: 28BYJ-48 5V-DC

Rationale: The 28BYJ-48 5V-DC is much less expensive than the other option. Additionally, the stepper motor will have an output shaft that is easier to interface with, allowing better application of the motor as well as transfer of torque from the motor. The loudness of the motor can be overcome by housing the motor to prevent interference with other aspects of the device.

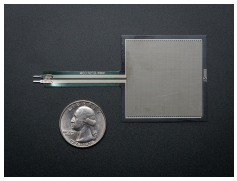
Motor Driver Selection

Solutions	Pros	Cons
L293D Quad Half H-bridge Motor Driver $2.95/each link to product 	• Lower continuous current • Low power dissipation	• More Expensive • Lower peak current
SN754410NEE4 Quad Half H-bridge Motor Driver $2.52/each link to product 	• Inexpensive • Easy to solder	• Higher continuous current • Higher power dissipation in air

Choice: L293D Quad Half H-bridge Motor Driver

Rationale: The L293D Quad Half H-bridge Motor Driver will have a lower power footprint when in operation. This is incredibly important as that power conservation for this project will be vital to the success of this device. Although it is more expensive, this trade off is worth the benefits. Additionally, although the maximum peak current is lower than the other option, it is still high enough to be used within the context of this device.

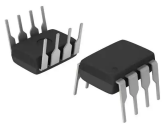
Force Sensor Selection

Solutions	Pros	Cons
FSR-406 $8.95/each link to product 	• Low Drift • Good documentation • Sensitive pad • Large surface area	• More expensive • Recommended crimps to solder tabs
FSR077BE $6.52/each link to product 	• Inexpensive • Through hole, direct solder • Large sensitivity range	• Documentation is less through • Low sensitivity for small applied forces

Choice: FSR-402

Rationale: The FSR-402 has better documentation and reference for applications that will be useful during designing circuits for this sensor. Additionally, the pad of this Force Sensitive Resistor has a lower cutoff force and is more sensitive which will allow more flexibility during design. The higher cost and need for additional methods of attaching the sensor can be accounted for and is worth the cost for the ease of implementation of the larger surface area of the pad.

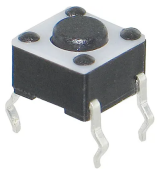
Operational Amplifier Selection

Solutions	Pros	Cons
MCP6002-I/P Linear Op Amp $0.33/each link to product 	• Large operating temperature range • Low voltage output swing	• Higher slew rate • Higher input voltage offset
LM358PE4- GP 2 Circuit Op Amp $0.36/each link to product 	• Recommended for using with FSR-402 • Low slew rate	• Higher voltage output swing • Lower gain bandwidth product

Choice: MCP6002-I/P Linear Op Amp

Rationale: MCP6002-I/P Linear Op Amp and the other options are very similar and only have very minimal differences. The higher slew rate will have a more drastic effect if a square signal is used and since the component will not be used in this manner it will have minimal effect. Additionally, the lower voltage output swing helps with the efficiency of the system. Also, there is already an inventory of this op-amp which is crucial to finalizing the custom circuit.

Pushbutton Selection

Solutions	Pros	Cons
B3F-4055 Tactile Switch $0.44/each link to product 	• Projected Plunger for easier use • Higher contact resistance	• More Expensive • Larger dimensions and area
FSM2JH Tactile Switch $0.11/each link to product 	• Inexpensive • High voltage rating	• Difficult to solder due to size • Small button interface

Choice: B3F-4055 Tactile Switch

Rationale: Tactile switch B3F-4055 will be used because due to the projected plunger, a cap can be added to the button that will make it easier to interact with. Additionally, due to the higher contact resistance, the switch will be less likely to display the effect of creep. Despite it being more expensive and a larger area, these factors can be accounted for when final positioning and design of the switch.

5V Regulator Selection

Solutions	Pros	Cons
L7805CV $0.50/each link to product 	• Supplies enough current to 5V rail • Internal current limiter	• Requires heat sink to increase max current • Lower max operating temperature
TL780-05CKC $0.51/each link to product 	• Supplies enough current to 5V rail • Internal current limiter	• Higher shipping cost • No longer manufactured

Choice: L7805CV

Rationale: L7805CV regulator will be used due to the fact it has a sufficient current rating which will allow for more margin of error when designing my printed circuit board. Additionally, there is more flexibility in the amperage if a heat sink is added and the max current increases. The max operating temperature is lower than the other option but it is still high enough for the environment and load that is being pulled by the circuit.

Power Source Selection

Solutions	Pros	Cons
NBSA36120300HU 	• Included with kit • Provides enough current for PCB	• Limited data about power source • Power cable length is short
12VDC 5A Power Supply $10.95/each link to product 	• Supplies lots of excess current to PCB • Datasheet is available	• More expensive • Larger dimensions and weight

Choice: NBSA36120300HU

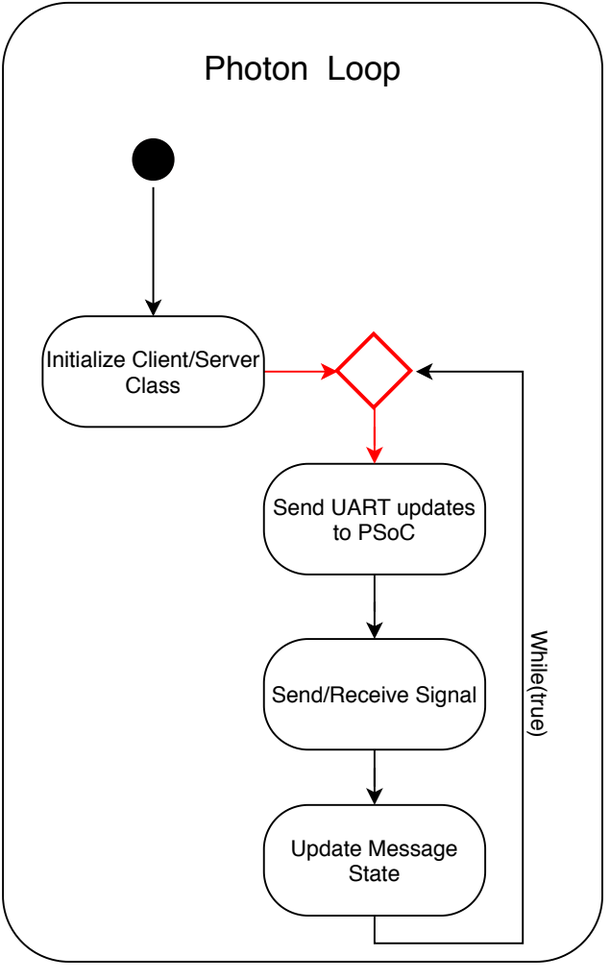
Rationale: For the power source, NBSA36120300HU was chosen because it provides enough current to the PCB design and is included in the kit so it is substantially less expensive. By using this power supply, the project budget can be spent on other components. Additionally, the information needed within the scope of this project can be found on the power supply itself, so the lack of documentation should not be an issue.

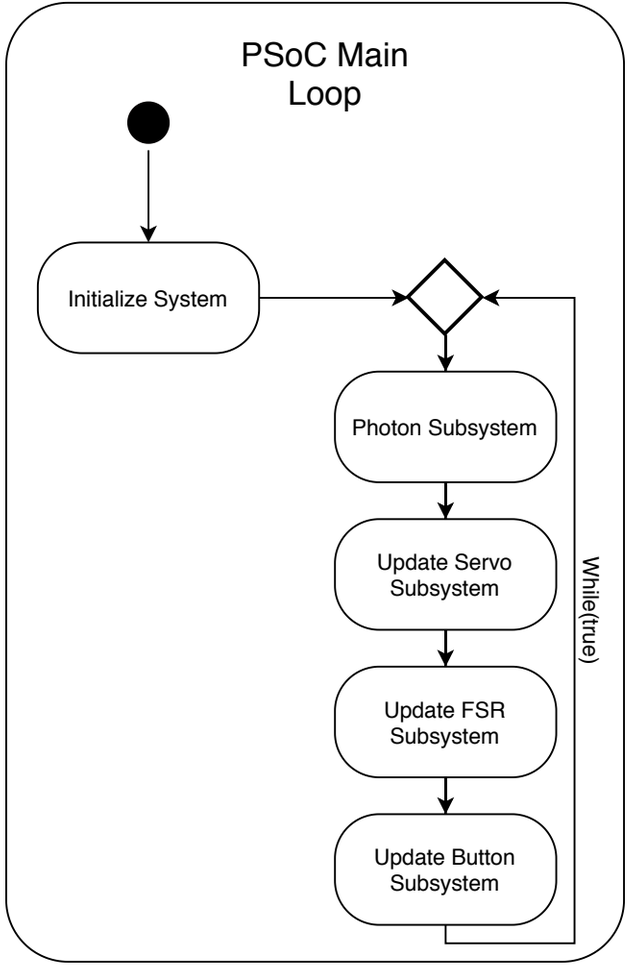
c. Power Budget

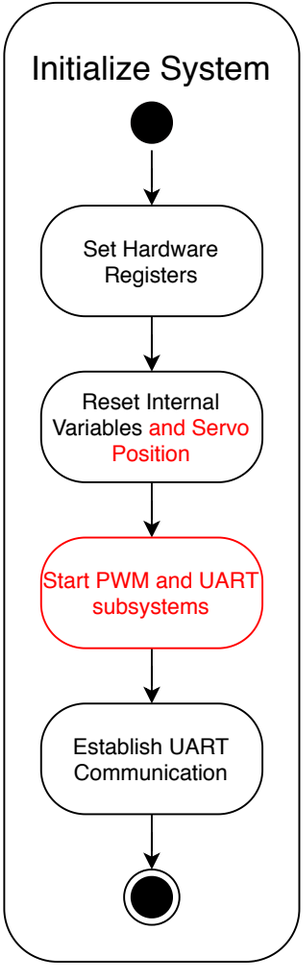
All Major Components	Component Name	Part Number	Supply Voltage	#	Ab. Max Current	Total Current	Unit
	PSoC™ 5LP	CY8CKIT-059	-0.5-6V	1	200	200	mA
	Particle Photon	PHOTONNOH	3.3-5.5V	1	430	430	mA
	Stepper motor	28BYJ-48	5-12V	1	240	240	mA
	H-Bridge	L293D	4.5-36V	1	60	60	mA
	Opamp	MCP6002-I/P	1.8-6V	1	30	30	mA
	Tactile Switch	B3F-4055	3-24V	1	50	50	mA
	Servo	SG90	4.8-7.2V	1	270	270	mA
	Force Sensor	FSR-402	5V	1	10	10	mA
+5V Power Rail	**Component Name**	**Part Number**	**Supply Voltage**	**#**	**Ab. Max Current**	**Total Current**	**Unit**
	Stepper motor	28BYJ-48	5-12V	1	240	240	mA
	H-Bridge	L293D	4.5-36V	1	60	60	mA
	Servo	SG90	4.8-7.2V	1	270	270	mA
	Tactile Switch	B3F-4055	3V-24V	1	50	50	mA
	Force Sensor	FSR-402	5V	1	10	10	mA
					Subtotal	630	mA
					Safety Margin	25%	
					Total Current Required on +5V Rail	787.5	mA
Regulator	+5V Regulator	L7805CV	8-35V	1	1500	1500	mA
	Total Remaining Current Available on +5V Rail					712.5	mA
+5V Power Rail	**Component Name**	**Part Number**	**Supply Voltage**	**#**	**Ab. Max Current**	**Total Current**	**Unit**
	PSoC™ 5LP	CY8CKIT-059	-0.5-6V	1	200	200	mA
	Particle Photon	PHOTONNOH	3.3-5.5V	1	430	430	mA
	H-Bridge	L293D	4.5-36V	1	24	24	mA
					Subtotal	654	mA
					Safety Margin	25%	
					Total Current Required on +3.3V Rail	817.5	mA
Regulator	+5V Regulator	L7805CV	8-35V	1	1500	1500	mA
	Total Remaining Current Available on +5V Rail					682.5	mA
External Power Source	**Component Name**	**Part Number**	**Supply Voltage**	**Output Voltage**	**Ab. Max Current**	**Total Current**	**Unit**
Power Source	Plug-in Wall Supply	NBSA36120300HU	100-120VAC	+12V	3000	3000	mA
Power Rails Connected to External Power	+5V Regulator	L7805CV	8-35V	1	1500	1500	mA
	+5V Regulator	L7805CV	8-35V	1	1500	1500	mA
	Total Remaining Current Available on External Power Source					0	mA

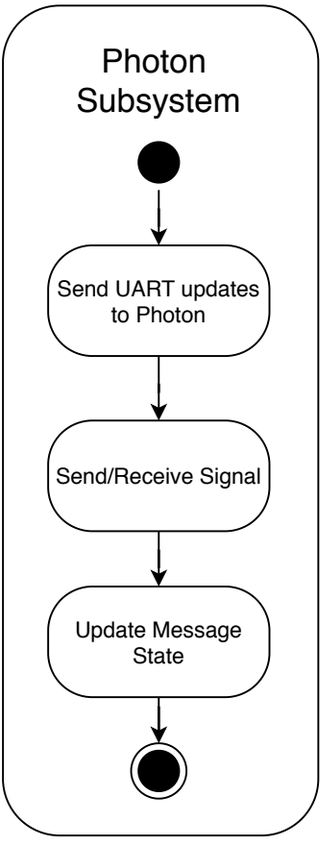
Anson Kwan's Bill of Materials (BOM)

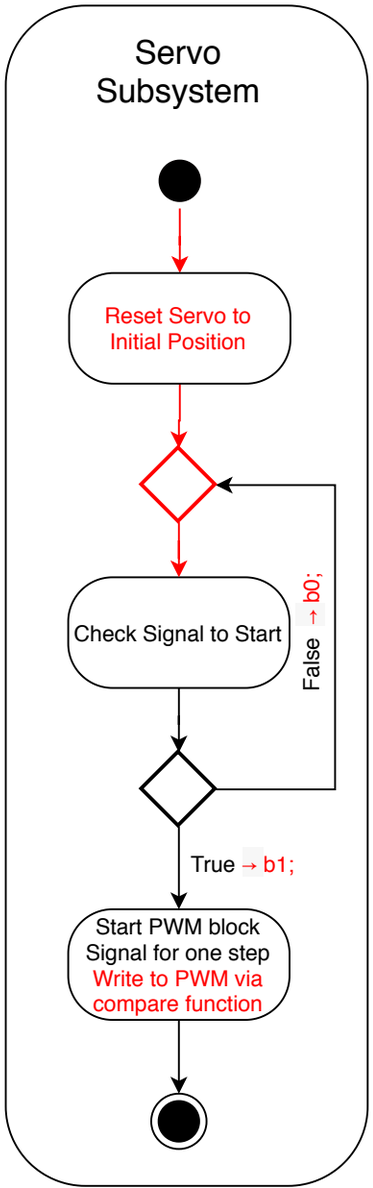
Part Name/Description	Unit Quantity	Unit Prototype Cost	Total Prototype Cost	Unit Production Cost	Total Production Cost	Manufacturer	Manufacturer Part #	Supplier	Supplier Part #	# Ordered	Date Ordered	# Recieved	Surplus	Schematic Reference Designators	Links
12V 3A Arris AC Adaptor	1	$15.77	$15.77	$15.77	$15.77	Arris	NBSA36120300 HU	N/A	N/A	0		1	0	J31	link to product
5V 1.5A Regulator	1	$0.50	$0.50	$0.15	$0.15	STMicroelectronics	L7805CV	Mouser	511-L7805CV	0		1	0	U5	link to product
5V 1.5A Regulator	1	$0.50	$0.50	$0.15	$0.15	STMicroelectronics	L7805CV	Mouser	511-L7805CV	0		1	0	U6	link to product
PSoC 5LP CY8C58LP Microcontroller	1	$15.11	$15.11	$15.11	$15.11	Cypress	CY8KIT-059	DigiKey	428-3390-ND	0		1	0	CY8C58LP1	link to product
Particle Photon	1	$19.65	$19.65	$19.65	$19.65	Photon	PHOTONH	N/A	N/A	0		1	0	PHOTON1	link to product
Force Sensitve Resistor 1.5" Pad	1	$8.95	$8.95	$6.45	$6.45	Interlink Electronics	30-73258	Adafruit	FSR 406	1	09/25/2020	1	0	J35	link to product
Quadruple Half-H Drivers	1	$2.95	$2.95	$1.83	$1.83	Texas Instruments	L293D	DigiKey	L293D	0		2	1	U1	link to product
Dual General Purpose Low Power Op-Amp	1	$0.33	$0.33	$0.25	$0.25	Microchip Technology	MP6002-IP	DigiKey	MCP6002-I/P-ND	1	09/25/2020	2	1	U4	link to product
0.1uF Ceramic Nonpolar Capcitor	10	$0.14	$1.40	$0.14	$1.40	Uxcell	A14111100ux0177	N/A	N/A	0		10	0	C1,C4,C7,C8,C9,C10,C12,C13	link to product
0.33uF Ceramic Nonpolar Capcitor	1	$0.23	$0.23	$0.23	$0.23	Bplectronic	334PF	N/A	N/A	0		1	0	C2,C3	link to product
4 Phase Unipolar Stepper Motor	1	$4.95	$4.95	$3.96	$3.96	MikroElectronica	28BYJ-48 5V	Adafruit	858	0		1	0	J33	link to product
Micro Servo SG90	1	$5.95	$5.95	$5.95	$5.95	Tower Pro	SG90	N/A	N/A	0		1	0	J37	link to product
Through Hole Diffused 5mm LED	2	$0.20	$0.40	$0.15	$0.30	Fedy	FD-5WW35-1	Adafruit	4203	0		6	4	D1,D2	link to product
Silver Plated Top Actuated Plunger Tactile Switch	1	$0.44	$0.44	$0.23	$0.23	Omron Electronics	B3f-4055	DigiKey	SW414-ND	0		4	3	J38	link to product
1/4W 1K Ohm Resistor	4	$0.03	$0.12	$0.03	$0.12	Shenzhen Tai Electronic	CF1/4WMJ1K	Adafruit	4294	0		25	21	R10,R11,R12,R13,R15,R16	link to product
1/4W 10K Ohm Resistor	2	$0.03	$0.06	$0.03	$0.06	Shenzhen Tai Electronic	CF1/4WMJ10K	Adafruit	2784	0		25	23	R3,R4	link to product
1/4W 47K Ohm Resistor	2	$0.03	$0.06	$0.03	$0.06	Shenzhen Tai Electronic	CF1/4WMJ47K	Adafruit	2786	0		25	23	R2,R9	link to product

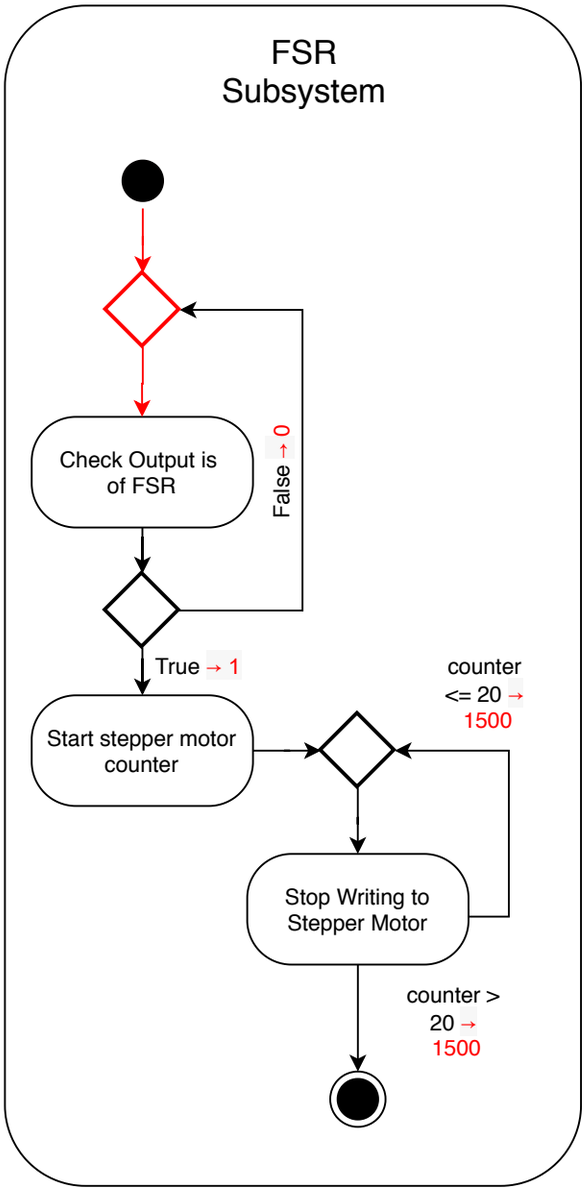

Photon Loop
Initialize Client/Server Class
Send UART updates to PSoC
Send/Receive Signal
Update Message State
While(true)

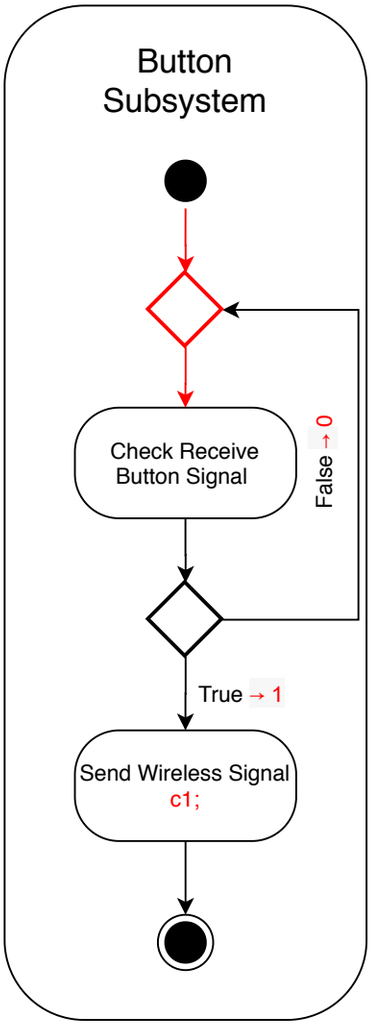

PSoC Main Loop
Initialize System
Photon Subsystem
Update Servo Subsystem
Update FSR Subsystem
Update Button Subsystem
While(true)

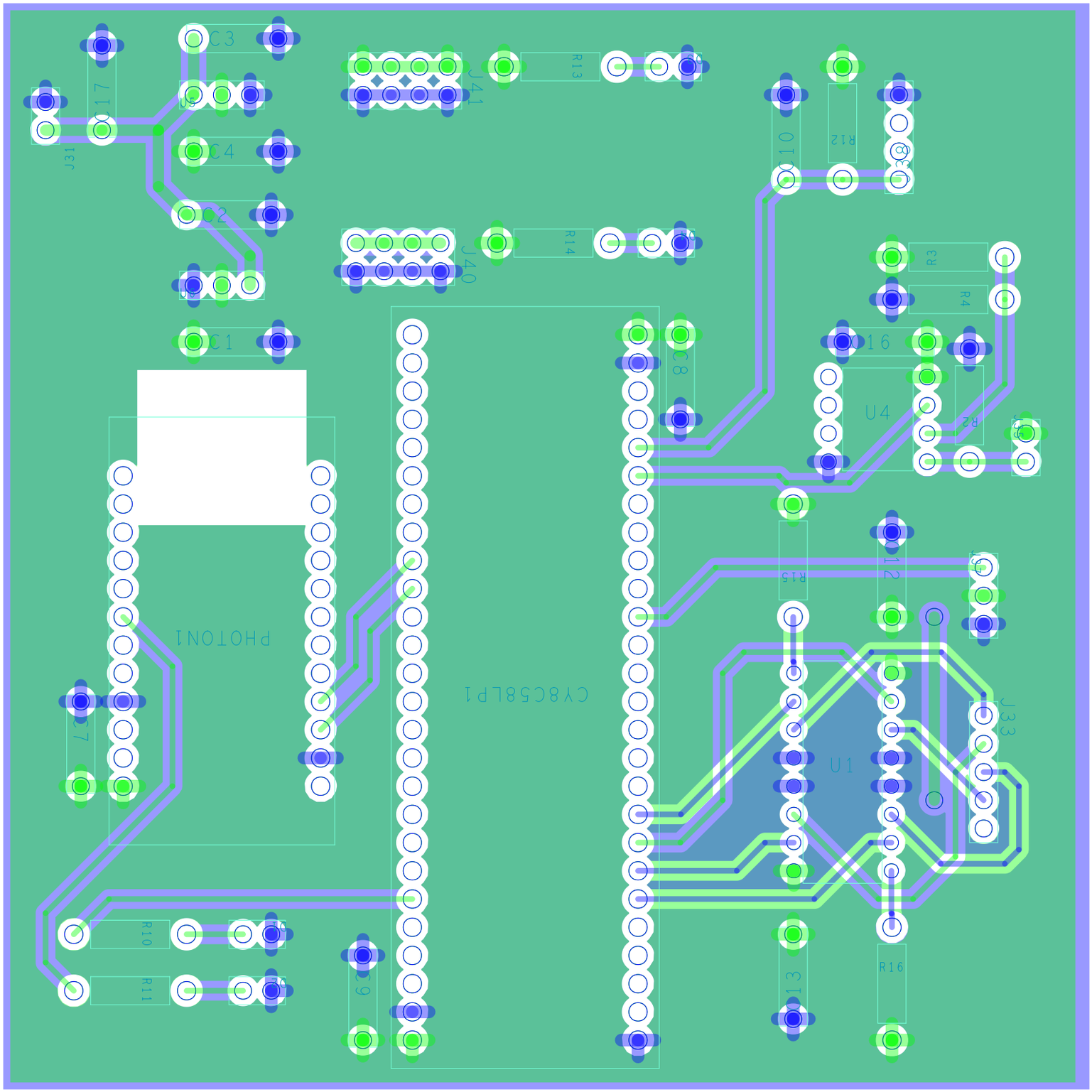

Initialize System
Set Hardware Registers
Reset Internal Variables and Servo Position
Start PWM and UART subsystems
Establish UART Communication


Photon Subsystem
Send UART updates to Photon
Send/Receive Signal
Update Message State


Servo Subsystem
Reset Servo to Initial Position
Check Signal to Start
False → b0;
True → b1;
Start PWM block Signal for one step Write to PWM via compare function

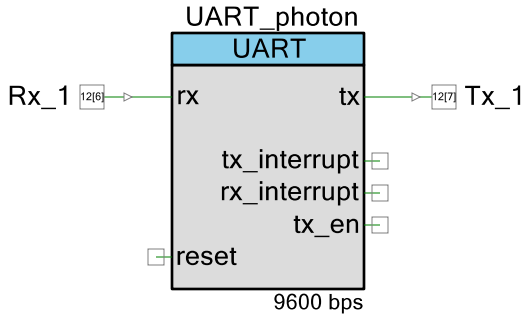

FSR Subsystem
Check Output is of FSR
False → 0
True → 1
Start stepper motor counter
counter <= 20 → 1500
Stop Writing to Stepper Motor
counter > 20 → 1500

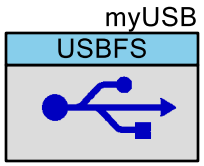

Button Subsystem
Check Receive Button Signal
False → 0
True → 1
Send Wireless Signal c1;



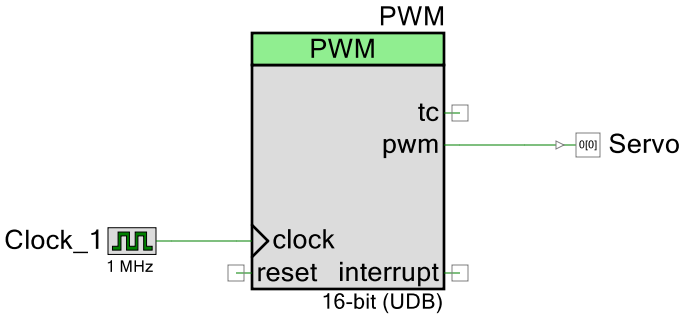
Blue: Ground Plane / Green: 5V Plane

Serial and UART Communication





Servo/PWM Subsystem



DI Pins from Sensors

FSR_OUTB 0[5]

TACT_SW 0[6]

DO Pin for LED

0[7] LED

DO Pins for Motor Driver

3[4] Motor_IN1

3[5] Motor_IN2

3[6] Motor_IN3

3[7] Motor_IN4

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
//Include standard libraries
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "project.h"

//internal variable
char c;

//pointer to USB serial data
char rxdata[100];
char txdata[100];
char out[64];

//declare some counters
int ii=0;

//declare pointer for use in conversion
char *ptr;

//declare values recieved from TxRx
int read_val = 0;
long lval = 0;

//length variable
uint16 len;

//compare value for servo
int PWM_compare = 2000;

void clear_buf(char * buffer,int len)
{
    for (int ii=0;ii<len;ii++)
    {
        buffer[ii]=0;
    }
}

void Servo_RST(PWM_compare)
{
    if(PWM_compare == 750)
```

```
            {
                PWM_compare = 2000;
            }
    PWM_WriteCompare(PWM_compare);

    CyDelay(100);
}

int main(void)
{
    //the following code gets run once

    //enable global interrupts
    CyGlobalIntEnable;

    //Start the USB subsystem
    UART_photon_Start();

    //Start the USB Serial Peripheral
    myUSB_Start(0,myUSB_5V_OPERATION);

    //Starts PWM subsystem
    PWM_Start();

    //Variables used for Switch, FSR, and Stepper
    int SW_State;
    int FSR_State;
    int motor_State = 0;
    int delay = 3;
    int sm_Start = 0;

    //wait until the USB configuration is loaded
    /*while(0 == myUSB_GetConfiguration())
    {
    }
    //prepare the USB Serial System for receiving data
    myUSB_CDC_Init();*/

    //Reset Servo position when machine is started
    Servo_RST(PWM_compare);

    //this is like a while loop
    for(;;)
    {
        //Updating states of Switch and FSR
        SW_State = TACT_SW_Read();
        FSR_State = FSR_OUTB_Read();



        //if there are characters in the UART receive buffer
        while(UART_photon_GetRxBufferSize()>0)
```

```
        {
            //retrieve one character and save in c
            c = UART_photon_GetChar();

            //make a decision based on the character received
            //if you get newline characters, reset the counter, you are going ↵
to receive a new message
            if(c=='\n' || c=='\r')
            {
                //don't process rxdata buffer
                read_val = 0;
                //clear the rxdata buffer
                clear_buf(rxdata,100);
                //reset the ii counter
                ii=0;
            }
            //if you get terminating characters, stop reading into your rx ↵
buffer and process the incoming number.
            else if(c==';' || c=='.')
            {
                //process rxdata buffer
                read_val = 1;
                //reset the ii counter
                ii=0;
                break;
            }
            //otherwise, add the character to your rxdata buffer and ↵
increment your ii counter.
            else
            {
                //don't process rxdata buffer
                read_val = 0;
                //save c into rxdata buffer
                rxdata[ii]=c;
                //increment the ii counter
                ii++;
            }
        }
        //check if read_val value is nonzero
        if (read_val!=0)
        {
            //read the first character of the rxdata buffer.  If it is an l, ↵
process it as a long int.
            if (rxdata[0]=='b')
            {
                //clear out the next character in rxdata to ensure it's ↵
treated as an end of line.
                rxdata[ii]=0;
                //convert string in rxdata to a long int, starting with the ↵
first character after the 0th position
                lval = strtol( rxdata+1, &ptr, 10 );    // convert message to ↵
an integer
```

```
            //clear the rxdata buffer
            clear_buf(rxdata,100);
            //reset the ii counter
            ii=0;

        }

    }


    //Write to servo motor if reieves gorrect sensor value from Jiaji
    if(lval == 1)
    {
        if(PWM_compare == 2000)
        {
            PWM_compare = 750;
        }
        PWM_WriteCompare(PWM_compare);

    }

    //If get FSR singal, signal to to start motor
    if(FSR_State == 1)
    {
        motor_State = 1;
    }


    //Writes to stepper motor if FSR pad is pressed
    if(motor_State == 1 && sm_Start <= 1500)
    {
            Motor_IN1_Write(1);
            Motor_IN2_Write(0);
            Motor_IN3_Write(0);
            Motor_IN4_Write(0);
            CyDelay(delay);

            Motor_IN1_Write(0);
            Motor_IN2_Write(1);
            Motor_IN3_Write(0);
            Motor_IN4_Write(0);
            CyDelay(delay);

            Motor_IN1_Write(0);
            Motor_IN2_Write(0);
            Motor_IN3_Write(1);
            Motor_IN4_Write(0);
            CyDelay(delay);

            Motor_IN1_Write(0);
            Motor_IN2_Write(0);
            Motor_IN3_Write(0);
```

```
            Motor_IN4_Write(1);
            CyDelay(delay);

            sm_Start++;
        }
        else
        {
            sm_Start = 0;
            motor_State = 0;
        }



        //Test for FSR sensor circuit
        /*if(FSR_State == 1)
        {
         LED_Write(1);

        CyDelay(100);

        LED_Write(0);

        CyDelay(100);
        }*/

        //LED turns on to signify microcontroller is running
        LED_Write(1);

        //Writes char out for message to send to Tx pin state of Tactile Switch
        sprintf(out,"c%d;\r",SW_State);

        //Verify Correct values being assigned to lval variable
        //sprintf( txdata, "lval = %ld\r\n", lval);
        //myUSB_PutString(txdata);

        //only update everytime button is pressed
        if (SW_State == 1)
        {
            //wait until USB system is ready
            //comment out when running off of 5V w/o USB
            //while (!myUSB_CDCIsReady());
            //write txdata to USB
            UART_photon_PutString(out);
        }

    }
}

/* [] END OF FILE */
```