Arcade Claw Game

ECE 230-04

Embedded Systems

Final Project Report

Winter 2022-2023

Instructor: Dr. Jianjian Song

Vineet Ranade

Yao Xiong

February 20, 2023

Contents

Introduction

Overview of Project Features

We created a variation of the traditional arcade claw game. The game serves as a microcontroller-based embedded system which acts on a player's inputs. Our inspiration for the project came from a suggested idea in the term project specification document. Upon further thought, we realized that implementing three-dimensional movement of the claw was surprisingly challenging. Although the software would be simple with the use of three stepper motors, the physical build would have been quite complex. Thus, we decided to implement two-dimensional movement of the claw: side-to-side motion and up-and-down motion. With more time, the project can be extended to add the third dimension.

In a normal claw game, the user uses the x- and y-coordinates of a joystick to move the claw along the upper plane of the build. A button is then pressed to prompt the claw to automatically descend and try to grab an object. However, we used the joystick's x-coordinate for the side-to-side movement and the y-coordinate for up-and-down movement. This gives the player more control over the claw and makes the game easier to win. The player is given 80 seconds each round. A round is won if the player can grab an object and drop it into a small box. The score will be equivalent to the time remaining when the round is won plus any added bonus, which depends on the height from which the object was dropped.

Alignment with ECE230 Project Requirements

The following subsections list the basic project requirements with a description of how we satisfied them. We have also elaborated on additional features we included to go beyond the minimum requirements.

Microcontroller: Must be $MSP432P4111$

The microcontroller used for the project was the $MSP432P4111$. It was mounted on the top portion of the build so it could be close to the stepper motors.

Sensors: At least 1

Evidently, the most important sensor in a claw game is the joystick. The analog signals generated by the x- and y- position of the joystick translate directly and linearly to the direction and speed of the claw. We were able to make use of the joystick's pushbutton to serve as a reset switch when the game is finished as well as the mechanism to open or close the claw during gameplay. Finally, two photoresistors were used in the small box. With a simple voltage divider, we could detect if an object was dropped in the box. The analog signal generated by one or both photoresistors would drop below a predefined cutoff voltage if an object obstructed light from reaching them. All ADC inputs were used in a 10-bit sequence-of-channels configuration.

Actuators: At least 1

Several actuators were used in the project. The LCD was configured in 4-bit mode. It reflected which state the program was in (resetting, countdown, gameplay, game win, or game loss). The resetting, countdown, and gameplay states all had a time countdown shown on the LCD. The game win and loss states prompt the user to press the pushbutton to play again. Additionally, the game win state displays the score of the most recent winner. Two stepper motors served as the actuators related to the joystick input, translating the claw in two dimensions to the player's intended destination. The motors spin faster the farther the joystick is pushed. A servo motor in the claw was toggled between two angles. In this way, we could open and close the claw in response to a button press during gameplay.

Timers: At least 1

All four timer modules provided by $Timer\ A$ and $SysTick$ were required. They were used to control the steppers' rotational velocity, generate the servo's PWM signal, track all time in the game, and create various delays.

Interrupts: At least 1

Several interrupts were used in tandem with the timers. They were used to act on pushbutton presses, schedule the next step of the stepper motors, time the positive pulse width and period of the PWM signal, and create a $\frac{1}{8}$-second cadence for the game to follow.

Additional Features

To track all real time in the game, we used $Timer32$. $Timer32$ is an $MSP432P4111$ peripheral which was never used in a lab exercise or project.

We also implemented a small bonus score feature in the game. While users have more control over our claw than in a traditional arcade game, we wanted to incentivize them to drop the object into the box from as high as possible (even though one can simply place the object on top of the photoresistor if desired). To gauge the height from which the object was dropped, we used a capture module of $Timer\ A0$ to detect the final button press. The magnitude of the bonus score depends on how much time elapsed between the drop and landing of the object. If the object was merely pushed or placed into the box, no bonus will be awarded. This is the first time we have used a $Timer\ A$ module in its capture configuration in the class.

One extended feature we began implementing was the use of an $ESP8266$ to host a webpage tracking all players' scores. We wanted to use SPI communication for this. Unfortunately, we had to prioritize dealing with power issues over this extra feature and did not have time to complete it to our satisfaction.

User's Manual

Thank you for using this claw game created by Vineet Ranade and Yao Xiong. This manual will primarily serve the arcade owner in setup. It will also include a description for the players of the game.

Before You Begin

Check that the components of the system are complete and intact. Figure 1 shows the entire setup.

Figure 1 – Complete Game Setup

Initialization

Locate the $5\ V$ power supply mounted on the microcontroller closer to the side with the cardboard box (this side of the setup will be referred to as the right side). Figure 2 shows what it looks like. Plug the supply into an outlet.

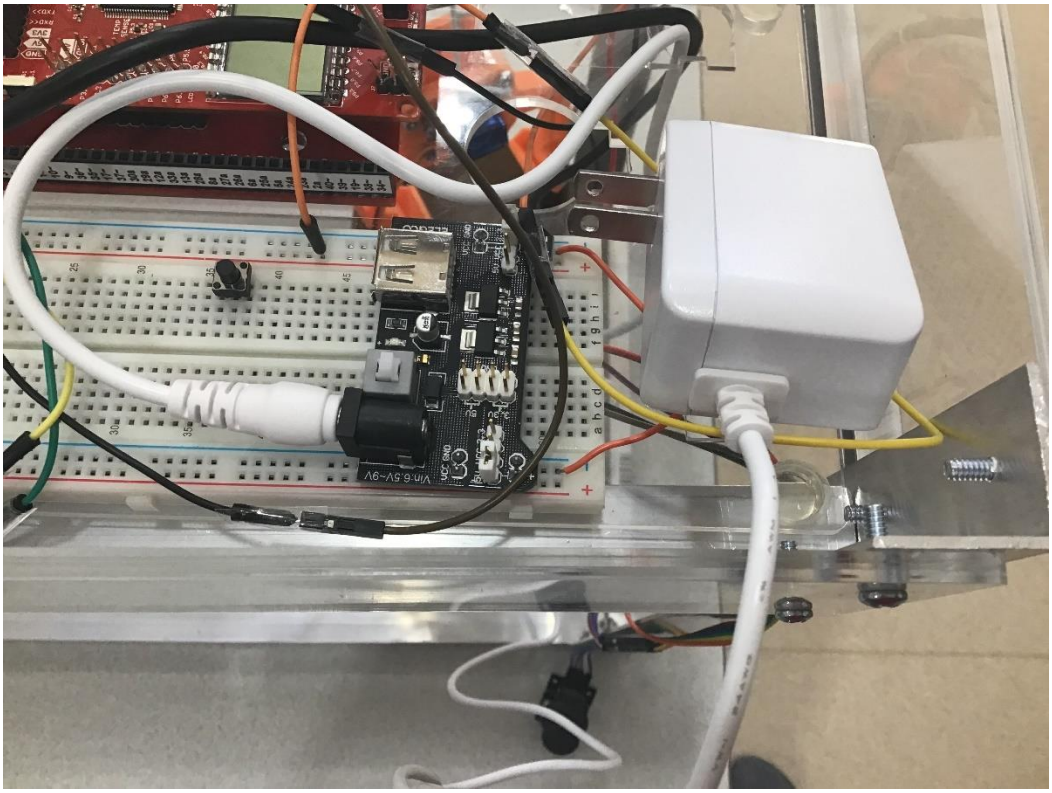

Figure 2 - 5 V Power Supply Connections

Locate the USB cord connected to the microcontroller on the left side of the setup. Connect the free end of this cord to a power bank or to a computer. The free end is shown in Figure 3.

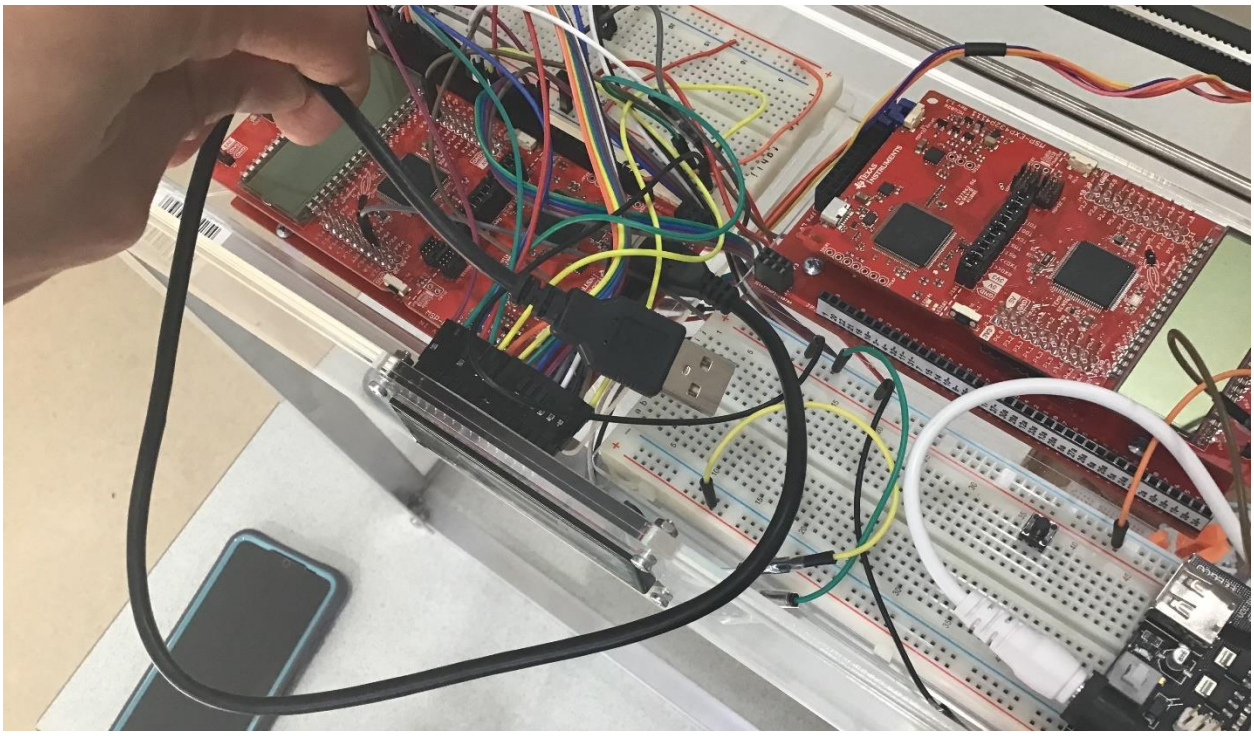

Figure 3 – USB Connector for the Left Microcontoller

Upon connecting the entire circuit to power, the LCD should light up, and the entire setup should be automatically resetting for a round of the game.

Operation as Arcade Owner

Set the build on a table and place a chair in front of it. Select a prize that players will win. Be sure to select a light, non-rigid object which can fit in the small box as well as the claw. Position these prizes on the table inside the build so that they are accessible by the claw's movement.

Ensure the small box is empty. When the first player comes by, explain the rules of the game to them, detailed in the following subsection. Perform the above initialization setups, which will automatically reset the game and begin a round.

When the round ends, inform the player of the result and tell them to remove their hands from the joystick. If the round was won, give them the prize that they dropped into the box. The number of follow-up attempts allowed after a round loss is up to the owner. For future rounds, simply prompt the next player to press the joystick to begin the automatic reset and the next round.

If players are struggling to grab prizes, you may use different ones or reposition the ones already in the game.

Operation as Player

The rules are simple. You have 80 seconds to navigate the claw from its reset position to a prize, grab the prize with the claw, and get the prize into the cardboard box. This can be done in any manner with the claw (dropping, placing, or sometimes even nudging the prize into the box with the claw) so long as you only use the joystick to control the claw. The challenge is that the claw moves very slowly.

Each round will start with 10 seconds of resetting, where the claw will move up and to the right automatically. Typically, the claw will have a home position. This is not the case for this game. The reason the resetting takes only 10 seconds is as follows: if the claw was near the small box at the end of the previous round, the player probably barely lost the game or won the game. In each case, it is evident that the player did not struggle much. Thus, if they choose to play again, the claw should be in the top right so that it is farthest from the prizes near the bottom left. If the claw was near the bottom left at the end of the previous round, the player was likely struggling to get a hold of a prize. Unlike the previous cases, a 10 second reset will not be enough for the claw to reach the top right of the setup. This is intentional. The next round will be easier to win since the claw will start closer to the prizes, saving time.

After the reset, you will be given a 3 second countdown, after which the gameplay begins. If you lose, the owner may give you another chance. If you win, the owner will give you a prize, and you may not play again that day. Do not touch any part of the setup other than the joystick. Do not reach into any part of the setup, even if you win.

We have put together some strategies to use for struggling players:

1. Move the claw down and to the left at the same time! This can be done with the joystick and will save time.
2. You can open the claw while moving it by pressing the button. This may save time.
3. The claw will rotate freely while you move it to the left. Make sure you pick a prize positioned such that you have a good angle of attack.

4. When you descend to grab an object, get low. This will give the claw a strong handle on the object so that the prize does not slip out of your grasp!
5. You need to land the prize directly on top of either photoresistor to win. Otherwise, you will lose. This is another part of the challenge, so make sure you position it correctly! Usually, an object successfully dropped in the box will trigger at least one photoresistor. Figure 4 below shows the box setup.
6. Even though you may gently place the object in the box, it is encouraged to drop the object instead for a chance to earn a couple of bonus points. They will be added to the time you had left to determine the final score!

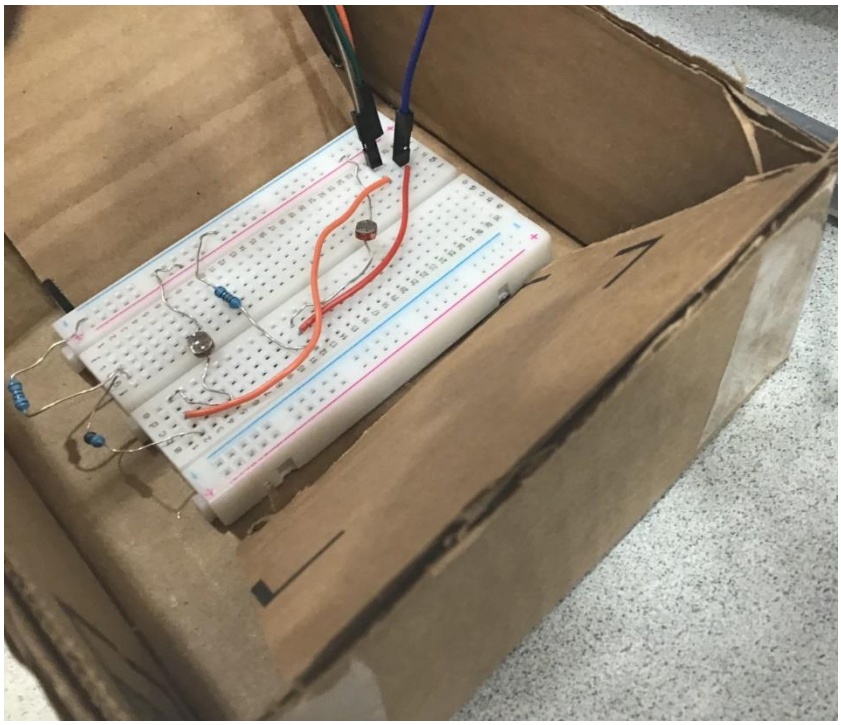

Figure 4 – Target Box for Dropping the Object

Hardware Design and Implementation

Figure 5 depicts the $3D$ model used for our claw game setup.

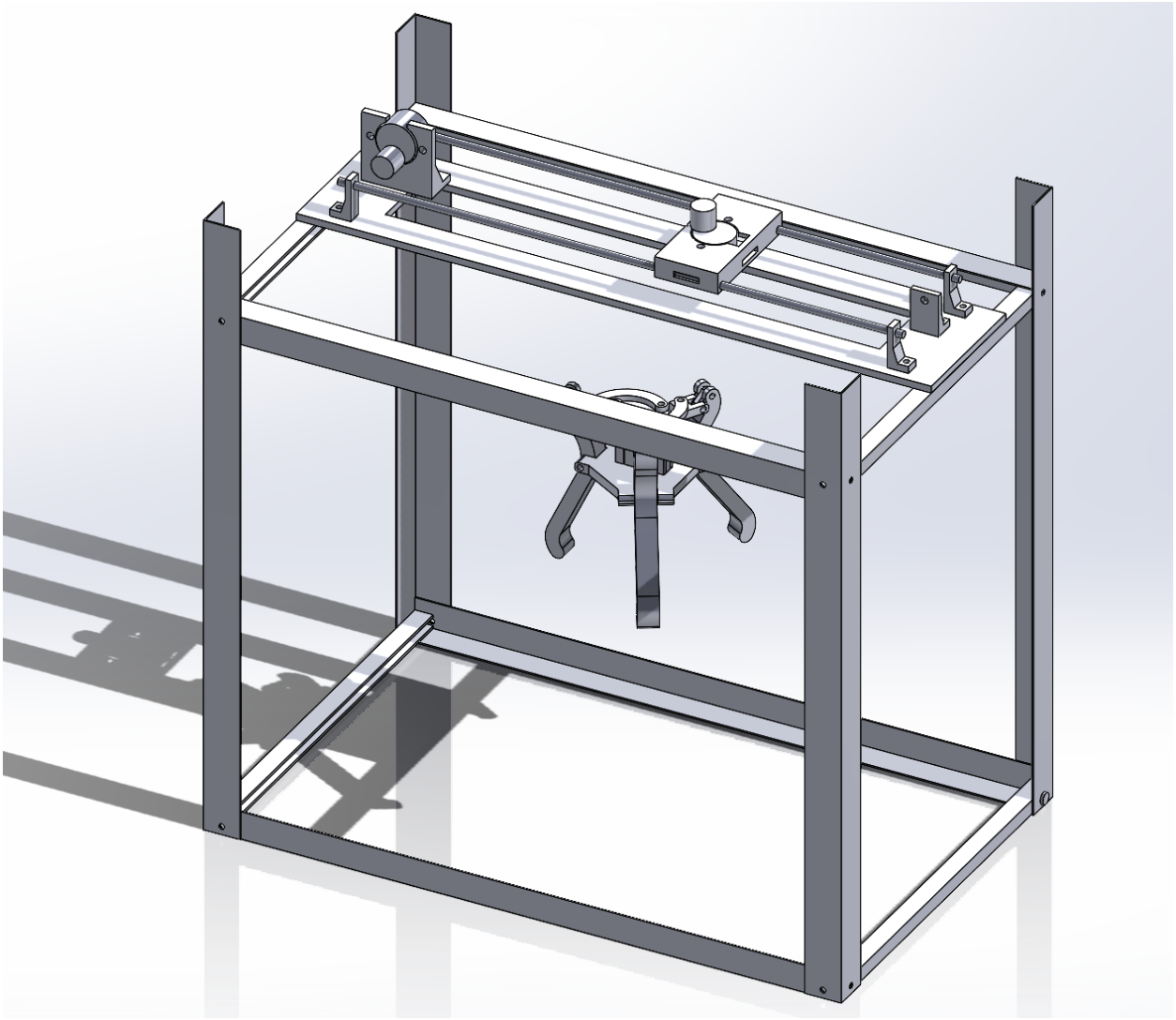

Figure 5 – 3D Model of Claw Game

Circuit Diagram

Figure 6 shows our overall system schematic.

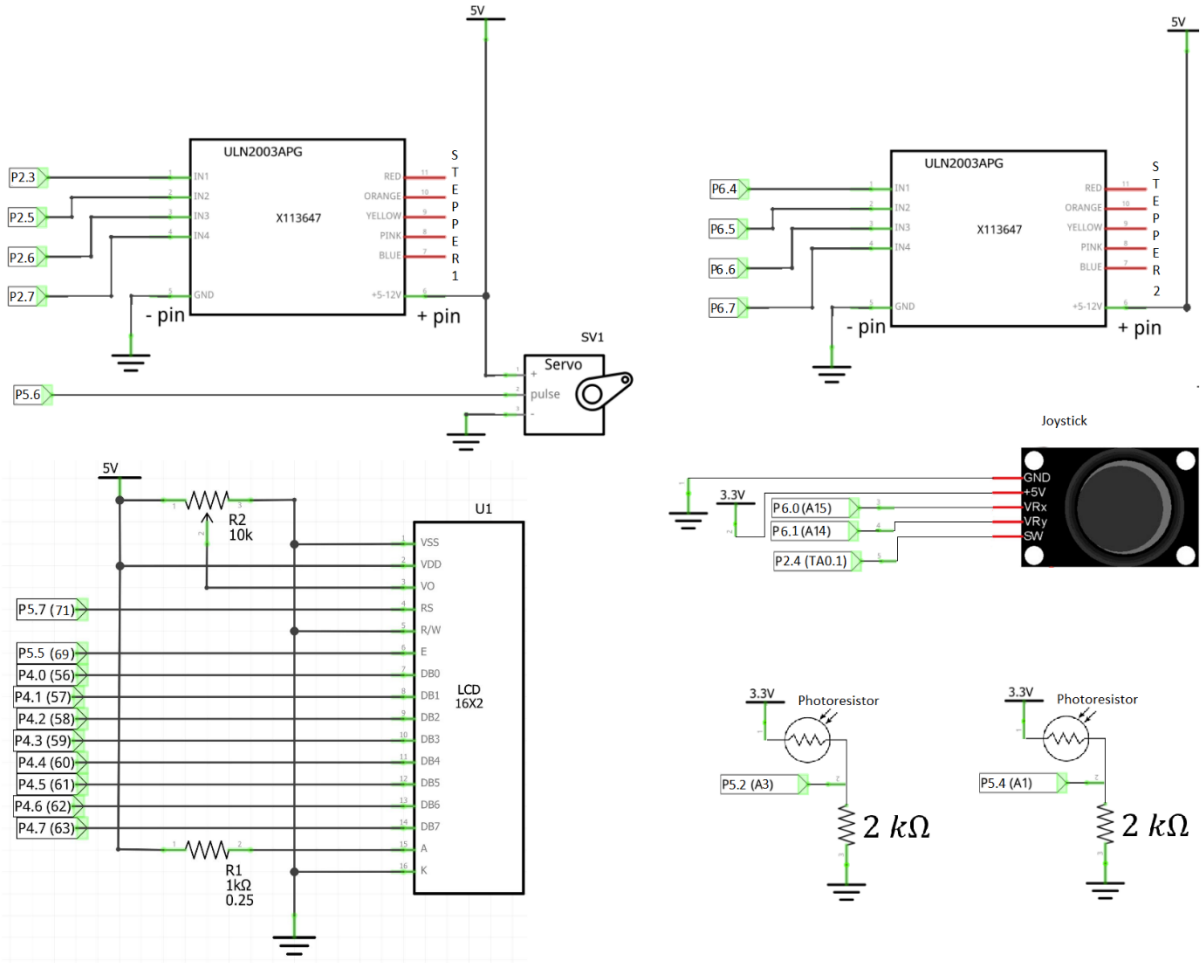

Figure 6 – Overall Circuit Schematic

Pin Assignments

Given the high number of circuit components used, we had to be very cognizant of which pins operate under which function. Table 1, Table 2, and Table 3 in the subsequent pages list each module in the diagram above. In the tables, we describe in further detail the purpose and configuration of each pin. The bolded signal name corresponds to the built-in $MSP432$ configuration that the pin was used for.

Module Name	Pin Number (PZ Package)	Signal Names	Description
Horizontal Stepper Motor	19	**P2.3 (RD)**	Stepper motor control signal, output to IN1
		PM_UCA1TXD	
		PM_UCA1SIMO	
		L8	
	21	**P2.5 (RD)**	Stepper motor control signal, output to IN2
		PM_TA0.2	
		PM_TA0.2	
		L22	
	22	**P2.6 (RD)**	Stepper motor control signal, output to IN3
		PM_TA0.3	
		L21	
	23	**P2.7 (RD)**	Stepper motor control signal, output to IN4
		PM_TA0.4	
		L20	
Vertical Stepper Motor	78	**P6.4 (RD)**	Stepper motor control signal, output to IN1
		UCB1SIMO	
		UCB1SDA	
		C1.3	
		L25	
	79	**P6.5 (RD)**	Stepper motor control signal, output to IN2
		UCB1SOMI	
		UCB1SCL	
		C1.2	
		L24	
	80	**P6.6 (RD)**	Stepper motor control signal, output to IN3
		TA2.3	
		UCB3SIMO	
		UCB3SDA	
		C1.1	
	81	**P6.7 (RD)**	Stepper motor control signal, output to IN4
		TA2.4	
		UCB3SOMI	
		UCB3SCL	
		C1.0	

Table 1 – Stepper Motors' Pin Assignment

Module Name	Pin Number (PZ Package)	Signal Names	Description
Servo Motor	45	P5.6 (RD)	Servo motor control signal, PWM output
		TA2.1	
		VREF+	
		VeREF+	
		C1.7	
Photoresistor 1	43	P5.4 (RD)	First object fall sensor, ADC input
		A1	
Photoresistor 2	41	P5.2 (RD)	Second object fall sensor, ADC input
		A3	
Joystick X Position	54	P6.0 (RD)	X velocity sensor, ADC input
		A15	
		L15	
Joystick Y Position	55	P6.1 (RD)	Y velocity sensor, ADC input
		A14	
		L14	
Joystick Pushbutton	20	P2.4 (RD)	Stepper motor control signal, output to IN1
		PM_TA0.1	
		L23	

Table 2 – Servo, Photoresistor, and Joystick Pin Assignments

Module Name	Pin Number (PZ Package)	Signal Names	Description
LCD	56	**P4.0 (RD)**	LCD signal, output to DB0
		A13	
		L13	
	57	**P4.1 (RD)**	LCD signal, output to DB1
		A12	
		L12	
	58	**P4.2 (RD)**	LCD signal, output to DB2
		ACLK	
		TA2CLK	
		A11	
	59	**P4.3 (RD)**	LCD signal, output to DB3
		MCLK	
		RTCCLK	
		A10	
	60	**P4.4 (RD)**	LCD signal, output to DB4
		HSMCLK	
		SVMHOUT	
		A9	
	61	**P4.5 (RD)**	LCD signal, output to DB5
		A8	
	62	**P4.6 (RD)**	LCD signal, output to DB6
		A7	
	63	**P4.7 (RD)**	LCD signal, output to DB7
		A6	
	69	**P5.5 (RD)**	LCD enable signal
		A0	
	71	**P5.7 (RD)**	LCD RS signal
		TA2.2	
		VREF-	
		VeREF-	
		C1.6	

Table 3 – LCD Pin Assignments

Horizontal Stepper Movement System

The horizontal movement system setup was built with the help of $3D$ printed parts. It uses one stepper motor, two tracks, and one belt to move the claw left and right.

Vertical Stepper Movement System

The vertical stepper movement system includes one stepper motor, a long string, and a claw. When the stepper motor rotates clockwise or counterclockwise, the claw will move up and down. Figure 7 shows the combination of vertical and horizontal stepper movement system (except the string and claw in the system).

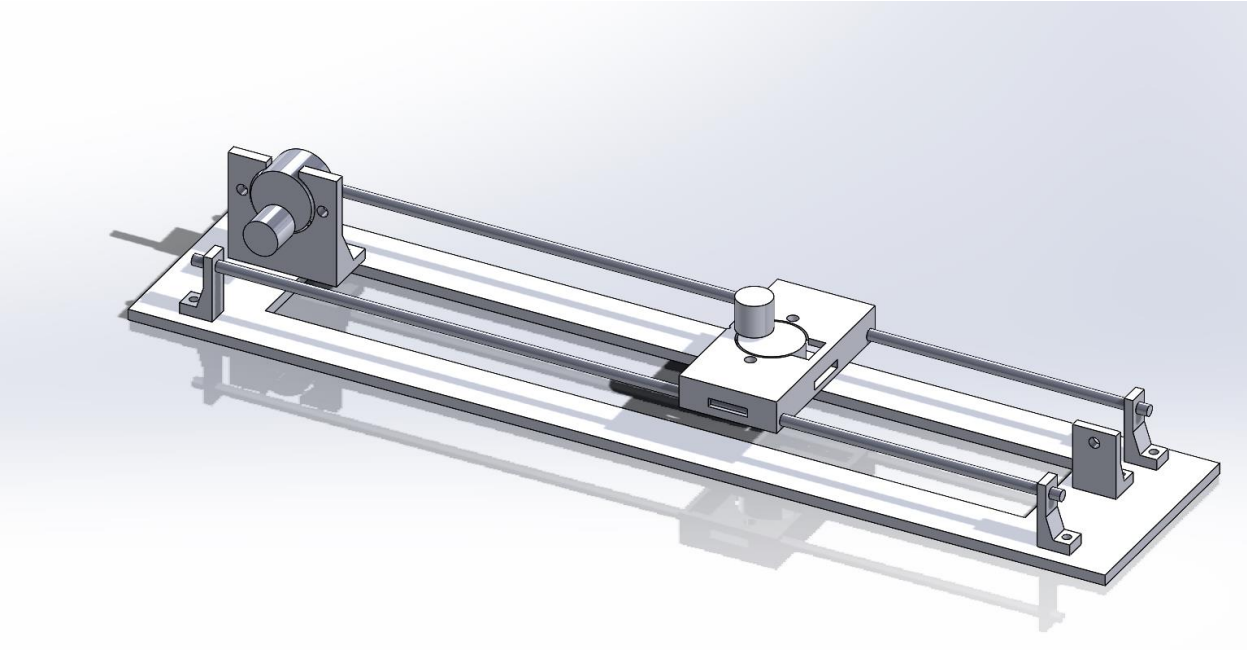

Figure 7 – Stepper Movement System

3.3 V & 5 V Power Supply

Because the $MSP432$ can only provide a maximum 450 mA (at 3.3 V), the output current is not enough for us to drive 2 stepper motors, 1 servo motor, 1 LCD, 1 joystick, and 2 photoresistors. Thus, we decide to add another external power supply to support the power. The power scheme is shown in Figure 8.

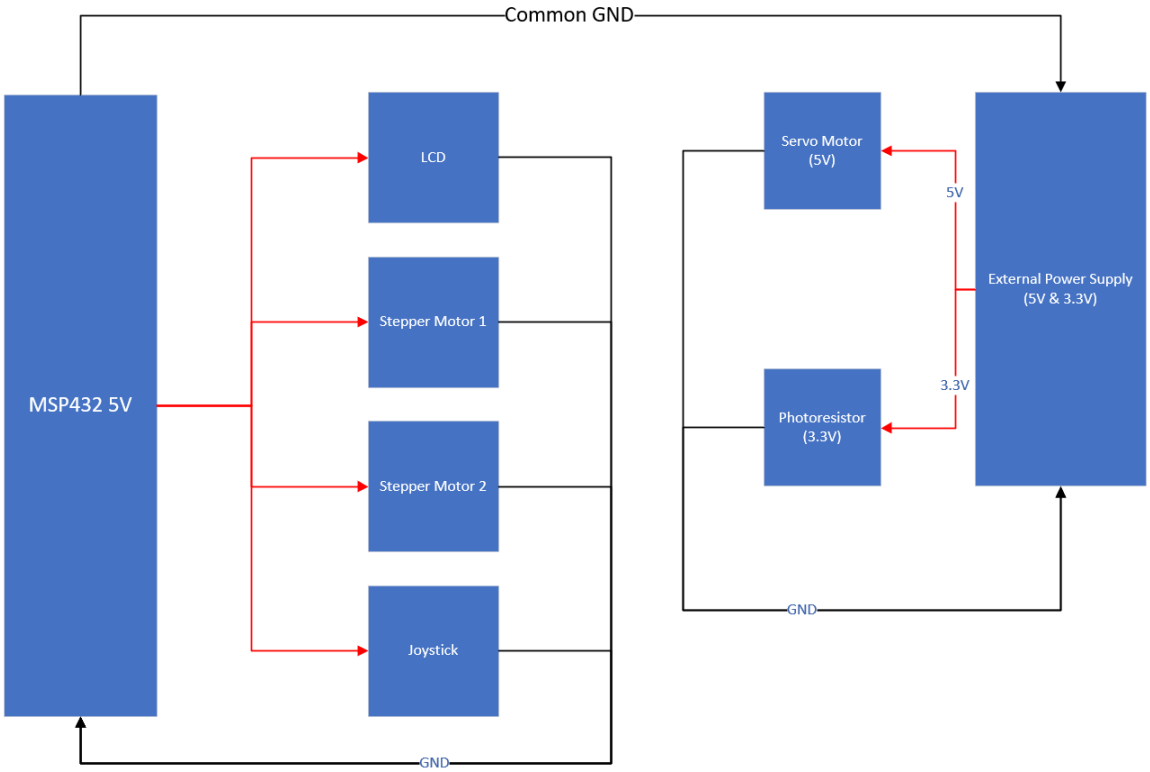

Figure 8 – Power Scheme for the Claw Game

Software Design and Implementation

Programming Style and Methodology

The overall coding style is called modularization development, which means we divide our code into different parts according to their function. Each module has a head file and a source code file. Our main program is running through a state machine, which will be explained in "State Machine Overview."

Timer Assignments

We use $Timer\ A3$ for a stepper motor, $Timer\ A1$ for another stepper motor, $Timer\ A2$ for the servo motor, and $Timer32$ for the game time counter, $Timer\ A0$ for the joystick pushbutton interrupt, and $SysTick$ for delay time.

$Timer\ A0$

$Timer\ A0$ is in continuous mode with source $ACLK$, prescale $1:1$, and interrupt enabled. $Timer\ A0$ is used to capture the push button's signal and debounce for it.

$Timer\ A1$

We set $Timer\ A1$ in $CCR0$ register for Up Mode and configure $CCR0$ for compare mode with interrupt enabled. In $Timer\ A1's$ Stop Mode, we use source $SMCLK$, prescale $3:1$, and interrupt disabled. In $Timer\ A1$'s IRQ handler, we use it to make stepper motor rotate one step either clockwise or counterclockwise.

$Timer\ A2$

We use $Timer\ A2$ to set the servo motor's angle. We configure the $Timer\ A2$ $CCR1$ register with No Capture Mode, reset/set Output Mode, and Interrupt disable. Furthermore, we set the period of $Timer\ A2$ in the $CCR0$ register for Up Mode and initialized the positive pulse-width of the PWM signal in the $CCR1$ register. In the end, we configure $Timer\ A2$ in Up Mode, with source $SMCLK$, prescale $6:1$, and interrupt disabled.

$Timer\ A3$

$Timer\ A3$ is used for the other stepper motor, so the configuration is exactly the same as that of $Timer\ A1$.

$Timer32$

We use $Timer32$ to count down the game time. With our reload value, $Timer32$ counts for $1/8$ second in each period, so once $Timer32$ hits 8 interrupts, the game time will count down by 1 second.

$SysTick$

$SysTick$ is a simple timer used for various delay times. It was used in initializations such as that of the LCD. While we initially used $SysTick$ for debounce delays, we settled on a lazy debounce instead. This is because pressing the pushbutton rapidly with the $SysTick$ configuration sometimes sent the program control to a Default IRQ Handler and froze the program.

State Machine Overview

The high-level state diagram is shown in Figure 9.

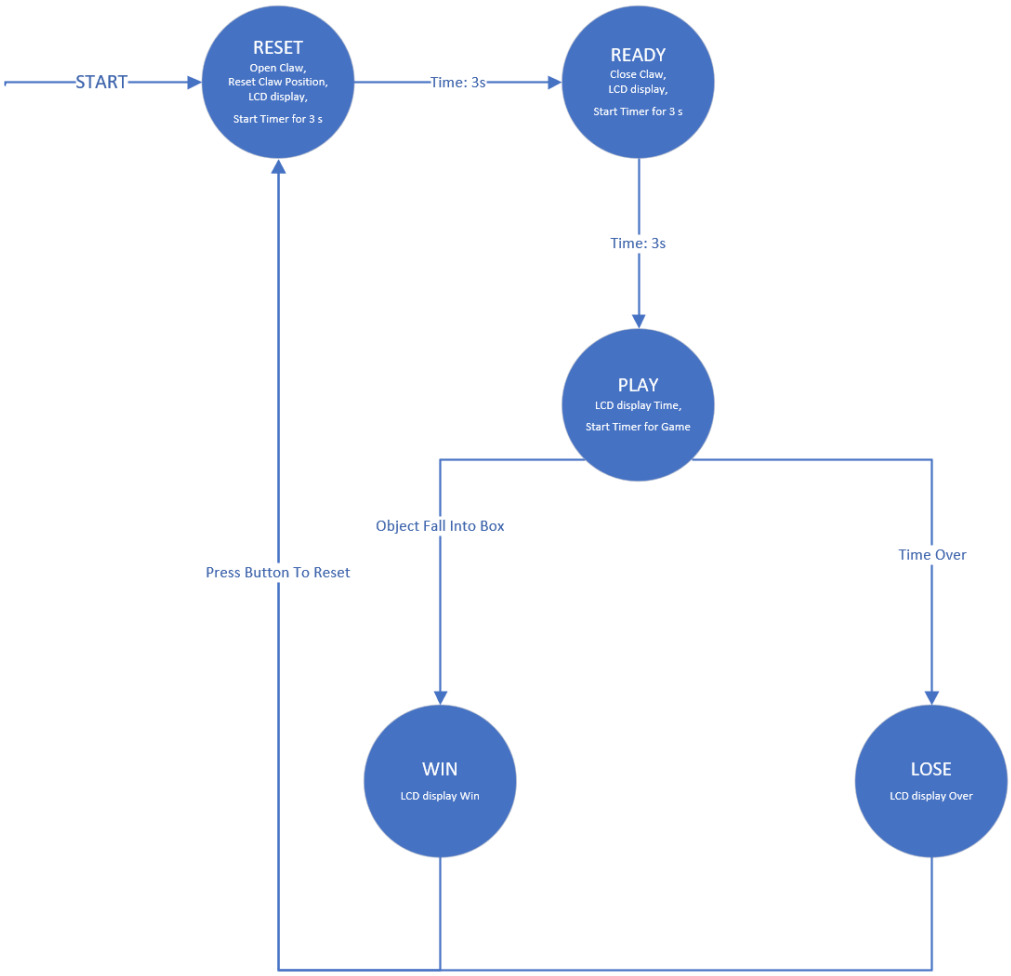

Figure 9 – Overall State Machine Flowchart

Figure 10 shows a more detailed, low level flowchart for the overall system.

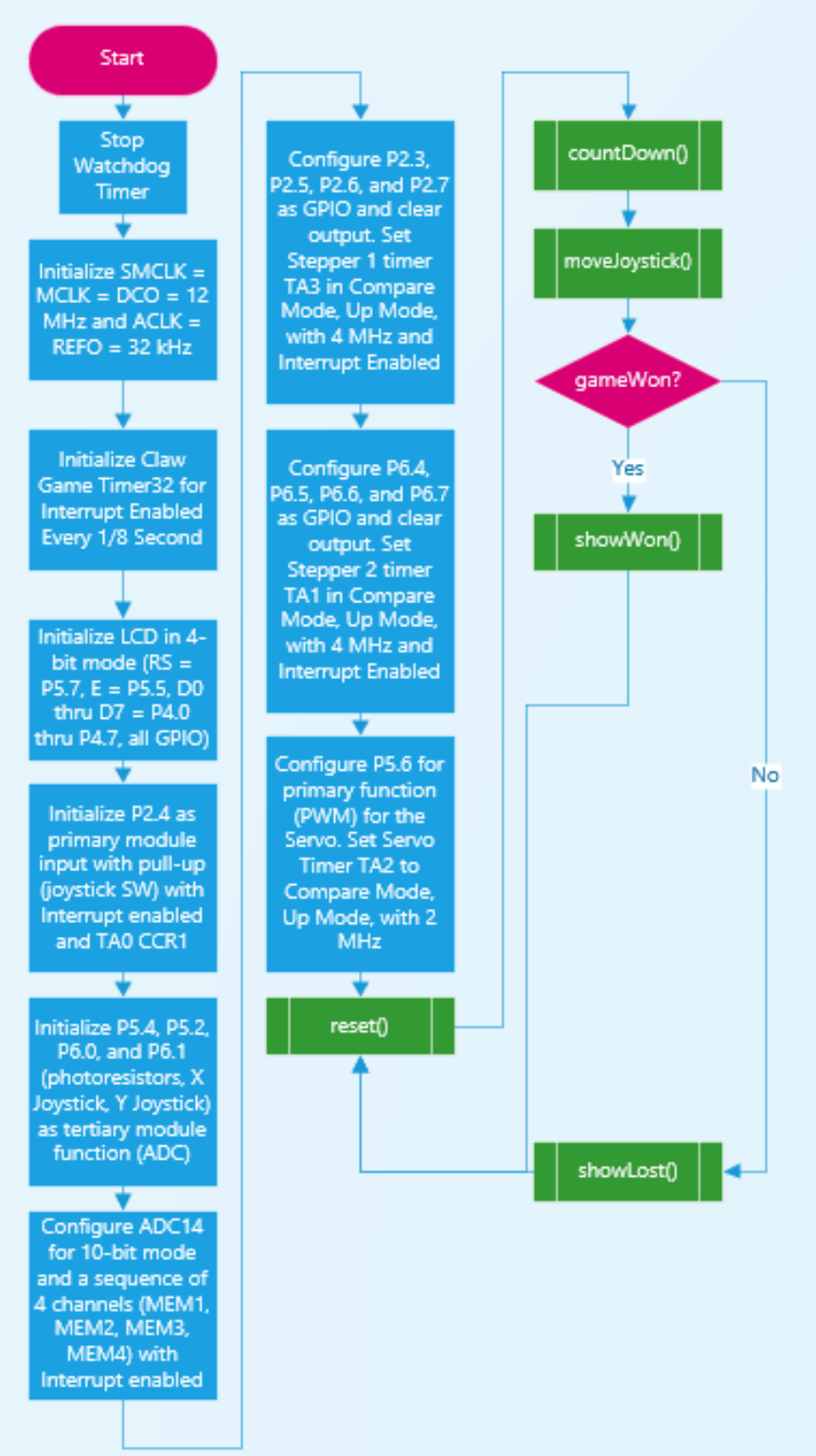

Figure 10 – Lower-Level State Machine Flowchart

Resetting the Game

In rest stage, we make our claw open, move the claw to the initial position, and display "Resetting…" on the LCD screen. We set a 10 second timer for this resetting stage. Figure 11 shows a flowchart of this state, but it abstracts away some details.

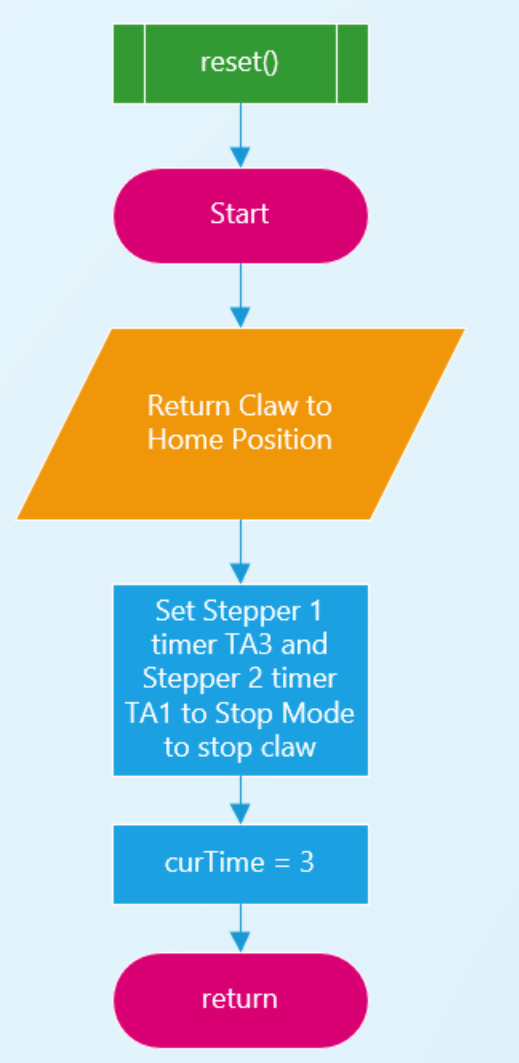

Figure 11 – RESET Flowchart

Starting the Countdown

When claw moves to the initial position, the game will go to READY stage. At this time, the claw will be open, and the LCD will show a counting down time for 3 seconds to tell the user to be ready for the game. Figure 12 shows a flowchart of this state, but it abstracts away some details.

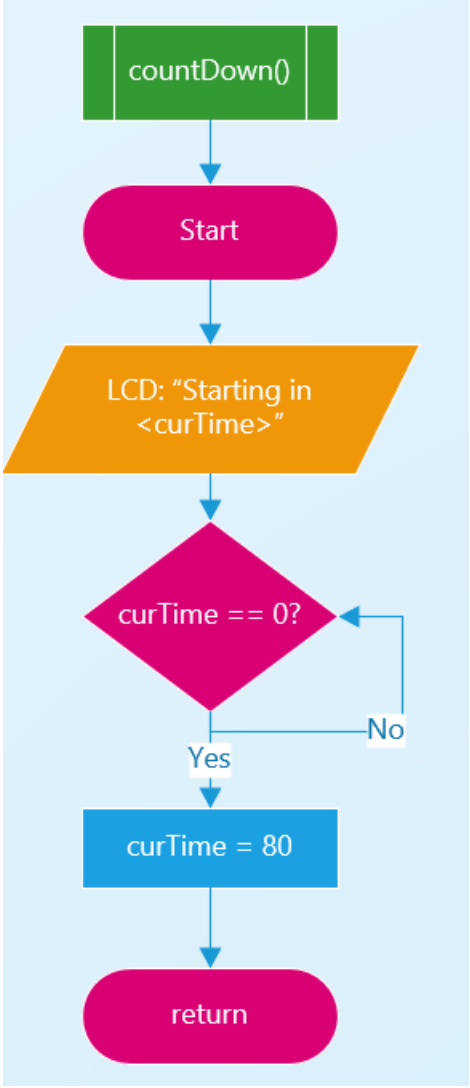

Figure 12 – READY Flowchart

Handling Gameplay with Claw Movement

After the counting down time becomes 0, the game comes to GAME stage. LCD screen will notify the user that game starts. The user can move the claw up and down, left and right, and open and close through the joystick. To win the game, the user should catch the ball and drop it to the box within a period. Figure 13 shows a flowchart of this state, but it abstracts away some details.

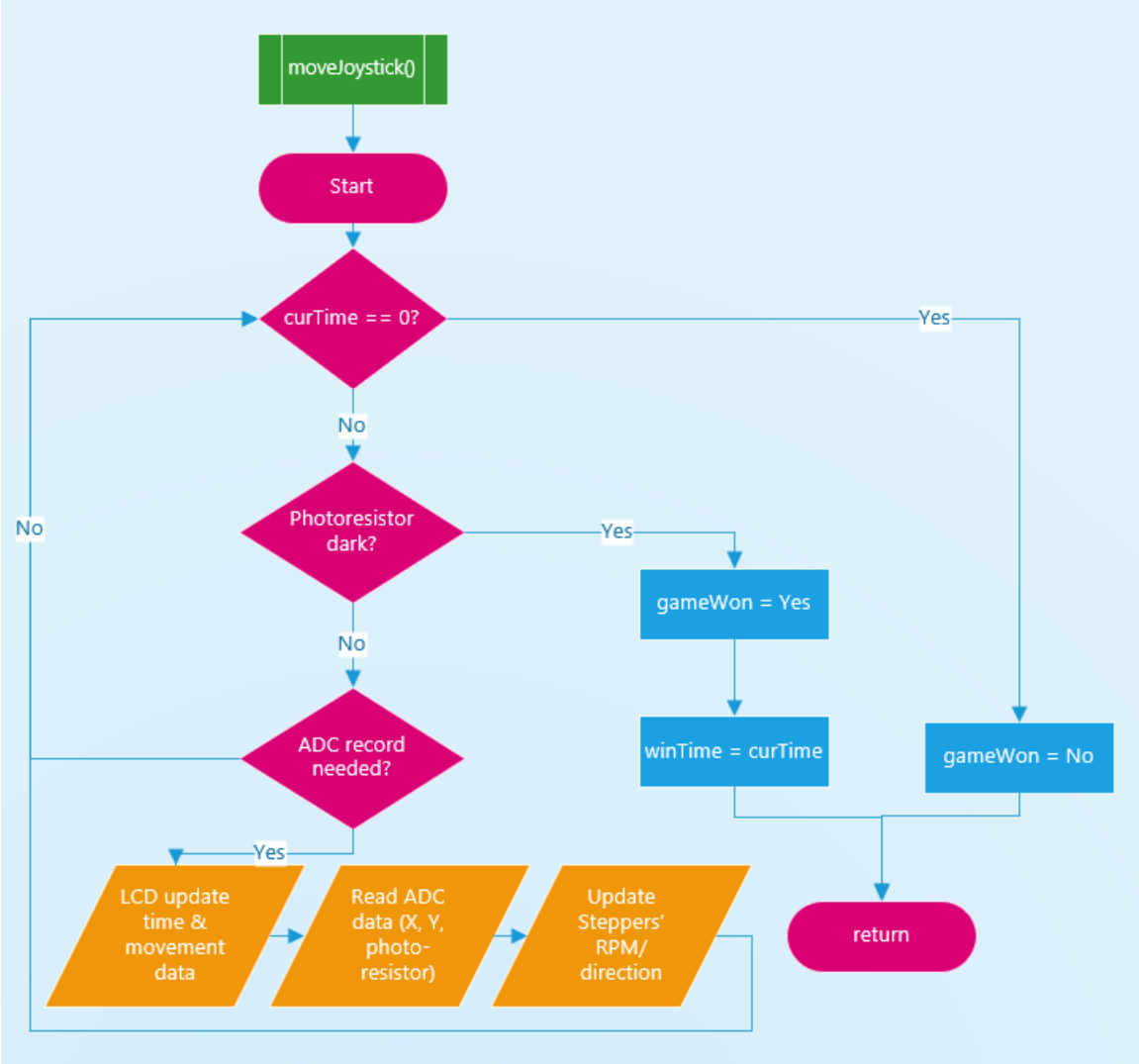

Figure 13 – GAME Flowchart

Winning the Game

If the user drops the ball successfully, he or she will win the game. The LCD will show the user's score, which is how many times are left. If the user presses the button, the game will go to the RESET stage.

Losing the Game

If the user cannot finish the task in the given of time, the user will lose this game. Like the WIN stage, the LCD will display the user is lost. If the user presses the button, the game will

go to the RESET stage. Figure 14 shows a flowchart of WIN and LOSE, but it abstracts away some details.

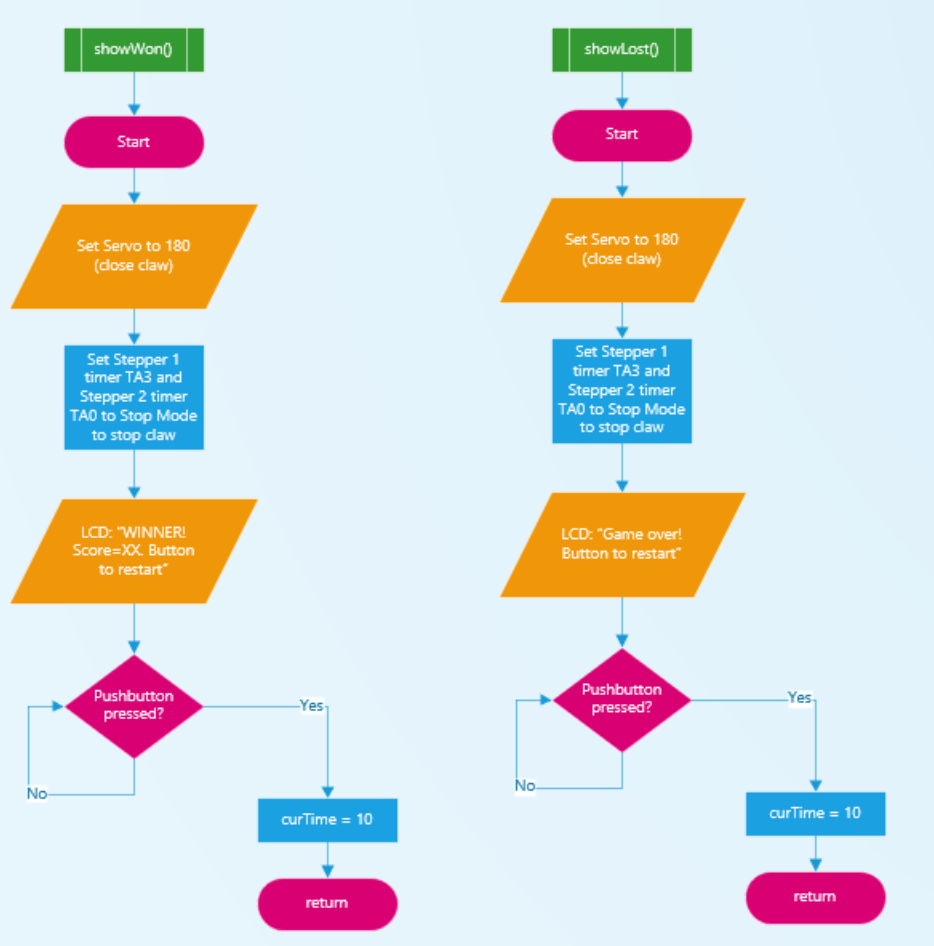

Figure 14 – WIN and LOSE Flowcharts

Test Plan and Test Results

Prepared by: Vineet Ranade & Yao Xiong

Date: 02/16/2023

Requirement #1

- Position of servo shall transition between $+120°$ (open) and $+40°$ (closed) with positive pulse width within $\pm 1\%$ accuracy
  - Period shall be 25 milliseconds
  - Positive pulse-width shall be 0.95 ms for min position ($+40°$) and 1.83 ms for max position ($+120°$)

Equipment needed:

- UUT
  - Launchpad
  - Wired circuit ($P5.6$ for servo motor)
- Oscilloscope from Digilent Waveform Board kit (DWB)
- Jumper wires
- Computer with working code

Setup and assumptions:

- Connect $CH1$ of the scope to $P5.6$ (servo control pin)
- Connect ground of the scope to the ground rail on the main circuit
- Uncheck Channels $2, 3$, and 4 so that only 1 is displayed on the scope
- Set the time interval on the scope to 5 ms and the trigger source to $CH1$, rising edge, and at about 0.25 V
- Add the following measurements to the scope screen
  - $CH1\ Period$
  - $CH1\ PosWidth$
- Assume that oscilloscope horizontal captures are within accuracy of 1 μs

Test procedure:

- Start scanning on the scope
- Take note of the following measurements of interest: T_{120} ($CH1\ Period$) and PW_{120} ($CH1\ PosWidth$)
- Toggle the claw with gameplay
- Repeat the above measurement for the closed claw (T_{40} and PW_{40})

Pass criteria:

- Calculations
    - Ideal period is $25\ ms$ for both
    - Since positive pulse width ranges from $0.51\ ms$ to $2.49\ ms$ for $0°$ to $180°$, following are the positive pulse widths required
        - Closed ($+40°$)
            - Ideal positive width is $0.95\ ms$
        - Open ($+120°$)
            - Ideal positive width is $1.83\ ms$
- Final criteria using scope assumption and 1% accuracy
    - $24.751\ ms \leq T_i \leq 25.249$ for $i = 40, 120$
    - $0.9415\ ms \leq PW_{40} \leq 0.9585\ ms$
    - $1.8127\ ms \leq PW_{40} \leq 1.8473\ ms$

Measurements/calculations:

- Measurements are inside red box towards the right top of the images on the following page

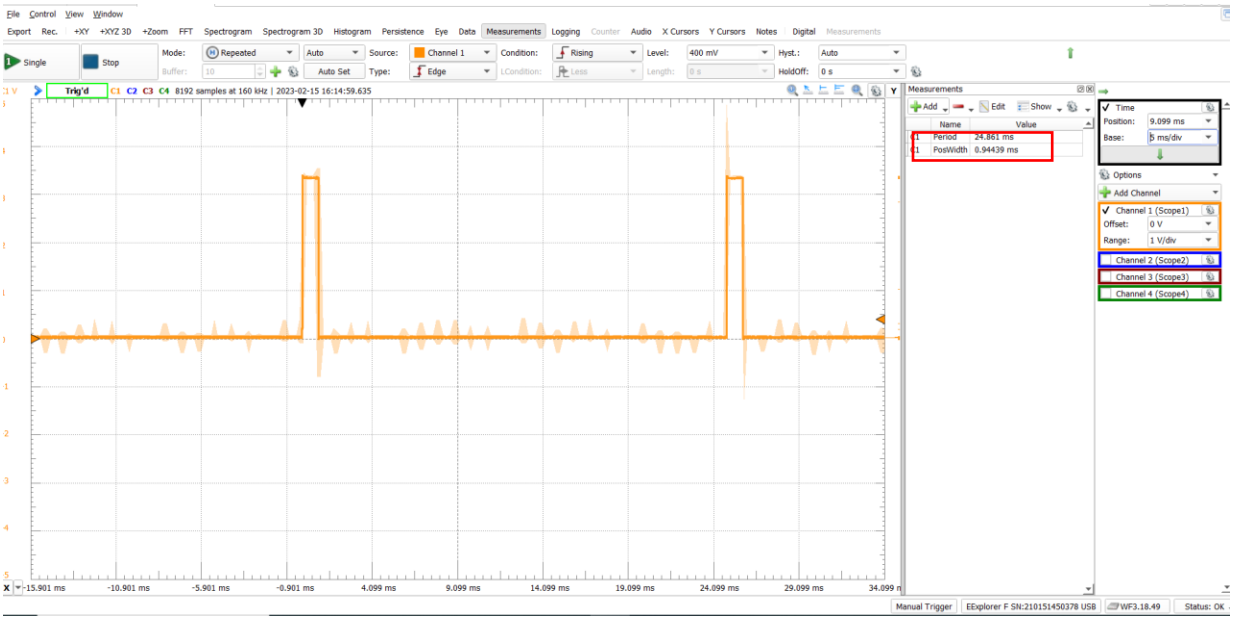

Figure $-T_{40} = 24.861\ ms, PW_{40} = 0.94439\ ms$

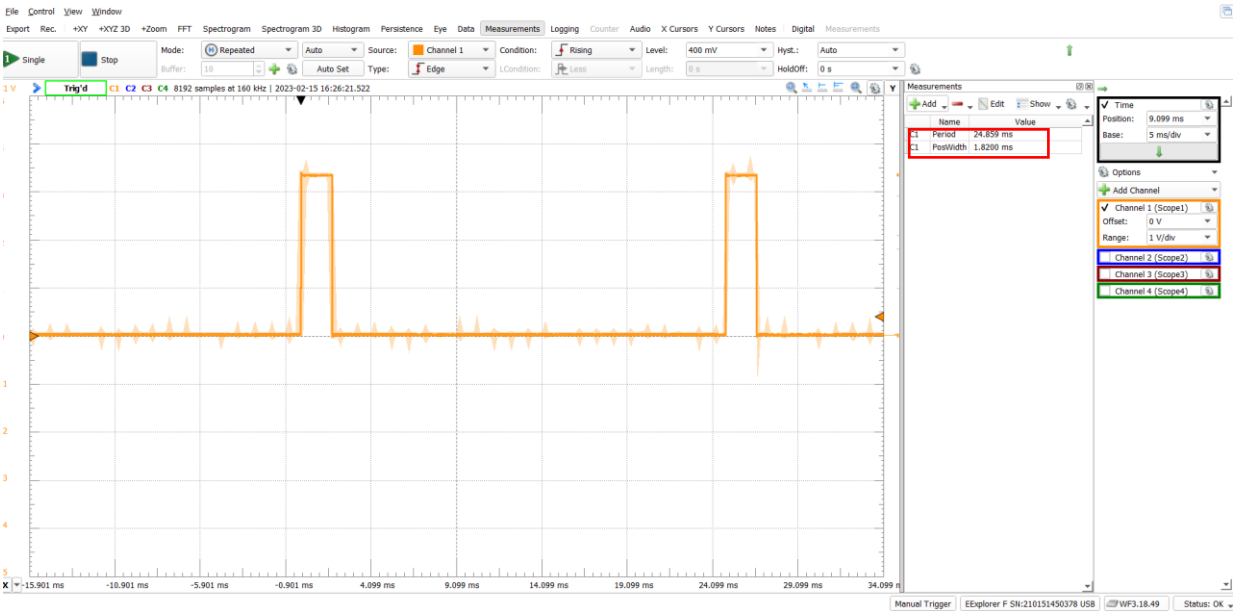

Figure 15 $-T_{120} = 24.859\ ms, PW_{120} = 1.8200\ ms$

Conclusions:

- $+40°$
  - $24.751\ ms \leq 24.861\ ms \leq 25.249\ ms$ ~ **PASS**
  - $0.9415\ ms \leq 0.94439\ ms \leq 0.9585\ ms$ ~ **PASS**
- $+120°$
  - $24.751\ ms \leq 24.859\ ms \leq 25.249\ ms$ ~ **PASS**
  - $1.8127\ ms \leq 1.8200\ ms \leq 1.8473\ ms$ ~ **PASS**

Requirement #2

- Rotation rate of horizontal stepper motor shall transition linearly with joystick input voltage, with speed of 1 RPM for 1.691 V and 15 RPM for 3.3 V (accuracy $\pm 1\%$)
- Obtain measurements for the following joystick inputs: 1.692 V, 3.3 V
  - Note that 1.692 V is used instead of 1.65 V because we have implemented an error bound for the resting position of the joystick
  - The minimum digital value accepted is 525, which means the minimum analog voltage accepted is $525 * \frac{3.3}{1024} = 1.692\ V$

Equipment needed:

- Same as Requirement #1
- However, the wired circuit includes a joystick connected to $P6.1$ for sideways movement

Setup and assumptions:

- Connect $CH1$ of the scope to $P2.4$ (first stepper control pin)
- Connect ground of the scope to the ground rail on the main circuit
- Uncheck Channels 2,3, and 4 so that only 1 is displayed on the scope
- Set the time interval on the scope to 5 or 15 ms and the trigger source to $CH1$, rising edge, and at about 0.25 V
- Add the following measurements to the scope screen
  - $CH1\ PosWidth$
- Same assumptions as Requirement #1

Test procedure:

- Start scanning on the scope
- Change code to force $yVal$ to be 525 since $yVal$ controls the horizontal step motor
- Take note of $PW_{1.692}$ ($CH1\ PosWidth$)
- Repeat the above for $yVal = 1023$ and $PW_{3.3}$

Pass criteria:

- Calculations
  - Note that $P2.4$ is high for exactly 3 steps in the half-step sequence (see line 18 in $stepperMotor.c$)
    - Therefore, to extract time per step (TS) from $PW_{3.3}$, use $TS_{3.3} = PW_{3.3}/3$
  - $1.692\ V$
    - $TS_{1.692} = \frac{1\ min}{1\ rot} * \frac{60\ sec}{1\ min} * \frac{1\ rot}{64*32*2\ steps} = \frac{14.6484\ ms}{step}$
  - $3.3\ V$
    - $TS_{3.3} = \frac{1\ min}{15\ rot} * \frac{60\ sec}{1\ min} * \frac{1\ rot}{64*32*2\ steps} = \frac{975.563\ \mu s}{step}$

- Final criteria using scope assumption and 1% accuracy
  - $14.5030\ ms \leq TS_{1.692} \leq 14.7939\ ms$
  - $967.797\ \mu s \leq TS_{3.3} \leq 985.328\ \mu s$

Measurements/calculations:

- Measurements are inside red box towards the right top of the images

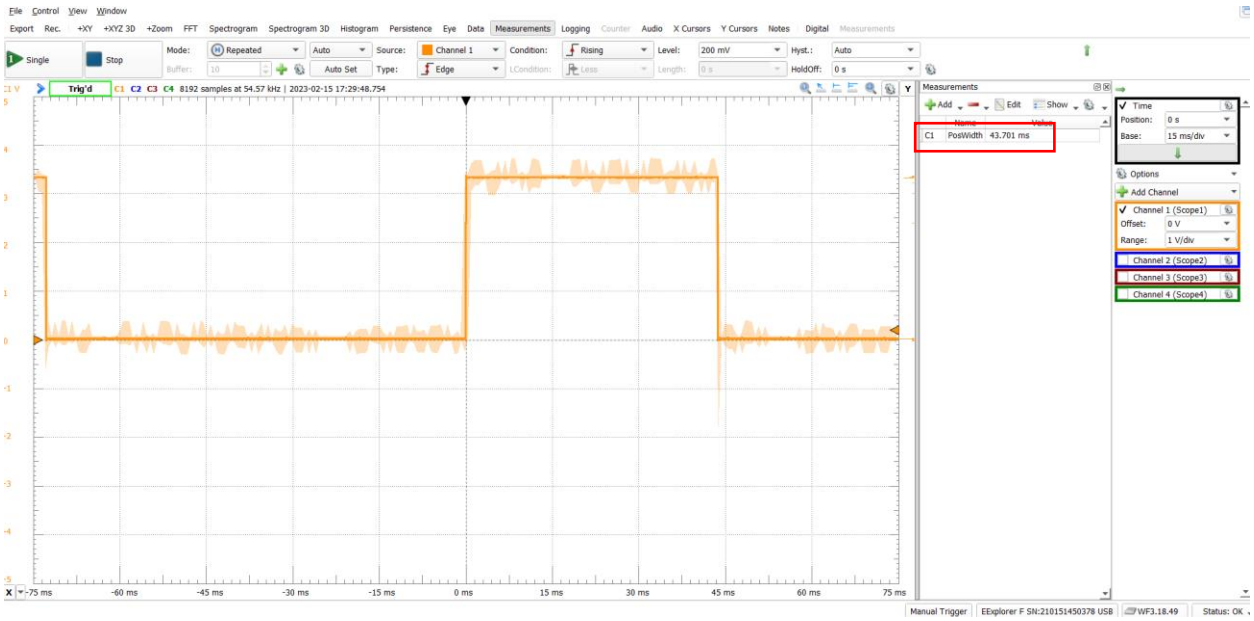

Figure 16 – $PW_{1.692} = 43.701\ ms$, $TS_{1.692} = 14.567\ ms$

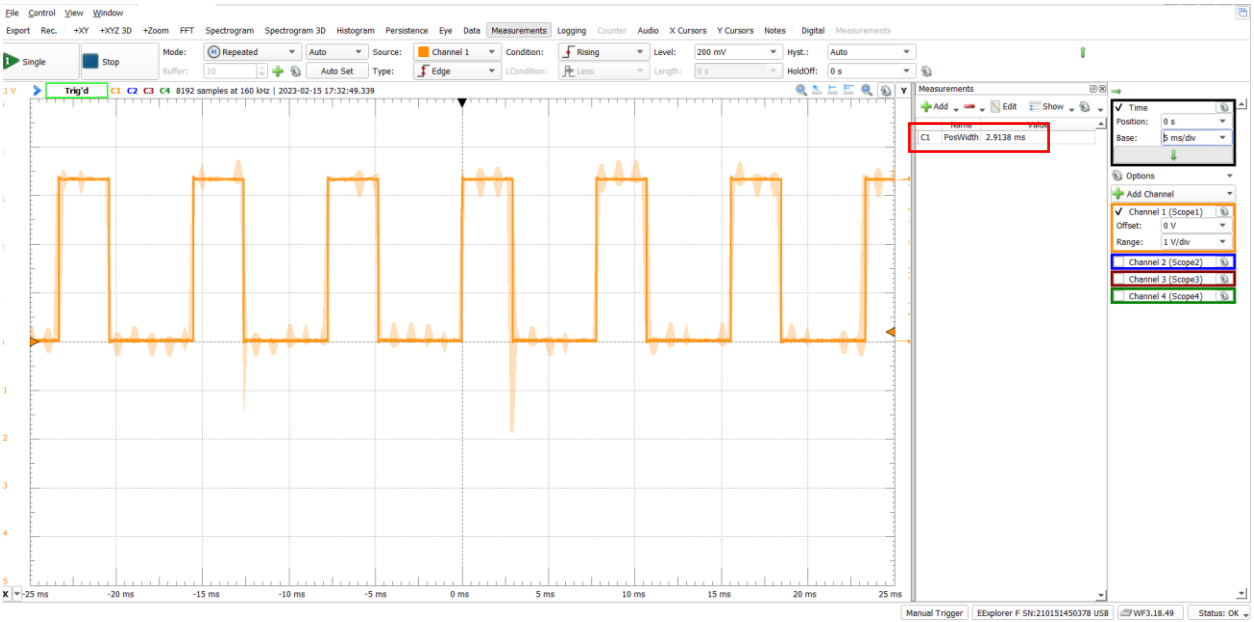

Figure 17 – $PW_{3.3} = 2.9138\ ms$, $TS_{3.3} = 971.267\ \mu s$

Conclusions:

- $1.692\ V$
  - $14.5030\ ms \leq 14.567\ ms \leq 14.7939\ ms$ ~ **PASS**
- $3.3\ V$
  - $967.797\ \mu s \leq 971.267\ \mu s \leq 985.328\ \mu s$ ~ **PASS**

Requirement #3

- Photoresistor ADC value drops below 150 when object falls on it (to detect a game win)
  - ADC input shall be configured for 10 bits, so 200 corresponds to $0.645\ V$

Equipment needed:

- Same as Requirement #1
- One $2k\Omega$ resistor and 1 photoresistor as the circuit

Setup and assumptions:

- Set a breakpoint at line 105 in file $stateMachine.c$
- Add $photoVal$ to the variable watch list

Test procedure:

- Load debug model to $MSP432$
- Click continue until the board stops at the breakpoint, dropping an object on photoresistor
- Check the variable $photoVal$ in the variable watch list to verify the object is detected
- Repeat these steps 10 times

Pass criteria:

- Photoresistor ADC value ≤ 150
- Photoresistor passes at least 9 of 10 trials

Measurements/calculations:

Trial	Photoresistor ADC Value	Pass
1	78	≤ 150
2	85	≤ 150
3	23	≤ 150
4	136	≤ 150
5	49	≤ 150
6	8	≤ 150
7	56	≤ 150
8	83	≤ 150
9	90	≤ 150
10	8	≤ 150

Table 4 – Results of Photoresistor Testing Indicate Success in 10 / 10 *trials*

Conclusions:

Pass Number: $10 \geq 9$ ~ **PASS**

Bill of Materials

Name	Number	Price	Note
Stepper Motor	2	$0	Included in Kit
ECE160 Claw	1	$0	Lent by Jack
15in-5mm Solid Rob	2	$11.99	Amazon
Belt + 5mm Pulley	1	$14.99	Amazon
5m Steel Channel	4	$22.12	MENDARDS
Acrylic Sheet	5	$0	Given by Gary
Cardboard	many	$0	Acquired
M4 Screw + Nut	many	$0	Borrow from friend
String for Claw	1	$2	Walmart
Joystick	1	$0	Included in Kit
Photoresistor	2	$0	Included in Kit
2K Resistor	2	$0	Included in Kit
LCD-HD44780	1	$0	Included in Kit
Jumper Wires	many	$0	Given by Jack
3D print Models	many	$0	Given by Gary
SG90 Servo Motor	1	$0	Included in Claw
MSP432 Dev board	2	$0	Include in Kit
	Total:	$51.10	

References and Acknowledgements

Datasheets and Materials

During the development of our project, we referred to the following materials:

- MSP432P4xx Technical Reference Manual
- MSP432P411x Data Sheet
- Stepper Motor Data Sheet
- ULN2003APG Data Sheet
- SG90 Data Sheet
- HD44780 Data Sheet
- LCD Data Sheet

Individuals

We would like to thank the following individuals for their invaluable help to us over not only this project but also the course of the entire class:

- Mr. Gary Meyer
  - Provided and cut all acrylic sheet in our setup
  - Helped us model and $3D$ print necessary parts for claw movement
  - Gave us several free parts from $E107$
- Mr. Jack Shrader
  - Lent us the ECE160 Robot Project gripper as our claw
  - Provided us with several wires so that we could chain them together and run various lines throughout our setup
- Dr. Jianjian Song
  - Reinforced all concepts during lecture and helped with labs and projects
  - Advice on power issues in our term project
  - Template code for various subroutines (not used but appreciated)
  - Documents and data sheets for parts not used before in the class
- Dr. Christopher Miller
  - Constructed the Moodle course which helped us learn concepts
  - Answered clarifying questions

Appendix A: GitHub Link

Considering the length of our source code, we decided to upload it to GitHub. You can download our code through this link:

https://github.com/peter-bear/ECE230_Term_Project_Claw_Game.git