ME4640 ROBOT HOMEWORK A (Rev 2)

Spring 2026

Introduction:

The goals of this assignment are to:

- Calibrate a set of 4 servomotors so that you can control them using your Arduino Uno,
- Construct a MeArm V3.0 miniature 4-degree-of-freedom robot arm, to use for further project assignments throughout the semester.

Deliverables Checklist (More details on the last page!):

[ ] Calibration plot

[ ] Sinusoidal Motion Plot

[ ] Arduino control code (PDF)

[ ] MATLAB/Python plotting Code (PDF)

[ ] 30-Sec servo motion video

[ ] Photo of assembled robot.

Some tips on soldering:

- Check your solder's required flow temps. Standard lead-free 60/40 solder wire typically flows well with an iron at 600–750°F (340–400°C).
- Use *helping hands* or *blue tac* to assist in holding components together while you solder.
- Less is more when soldering. Using too much solder at a join can cause bridging between other components.
- Take your time, and work from the least expensive to the most expensive components.
- I recommend watching this brief intro video on soldering iFixit's Soldering 101: Beginners Guide.
- Another useful guide to soldering is available at the following link: Adafruit Soldering Guide
- Before soldering, ensure you have all your components and understand how they will be placed onto your PCBs.

Phase 1 – Electronic Build:

Part 1: Check Kit Contents

Your kit should contain all laser-cut parts, electronic components, and hardware needed to complete projects over the semester. Refer to Canvas for the updated hardware cheat sheets to confirm that you have all the required components in your kit, and let your instructor know if you are missing any items.

Part 2: Solder MeArm PCB

In this part, you will prepare the MeArm PCB (Printed Circuit Board), which will route power and signal to each of the four servo motors and act as a structural part of the robot assembly.

1. Break header pins into appropriate lengths for the MeArm PCB. You will need 6, 3x1 sections.
    a. Note: needle-nose pliers are helpful for this.
2. Solder the header pins to the board as shown in Figure 1.
    a. Note that this board includes a location for an optional capacitor. We will skip it for now and include it on the HRIB board later.

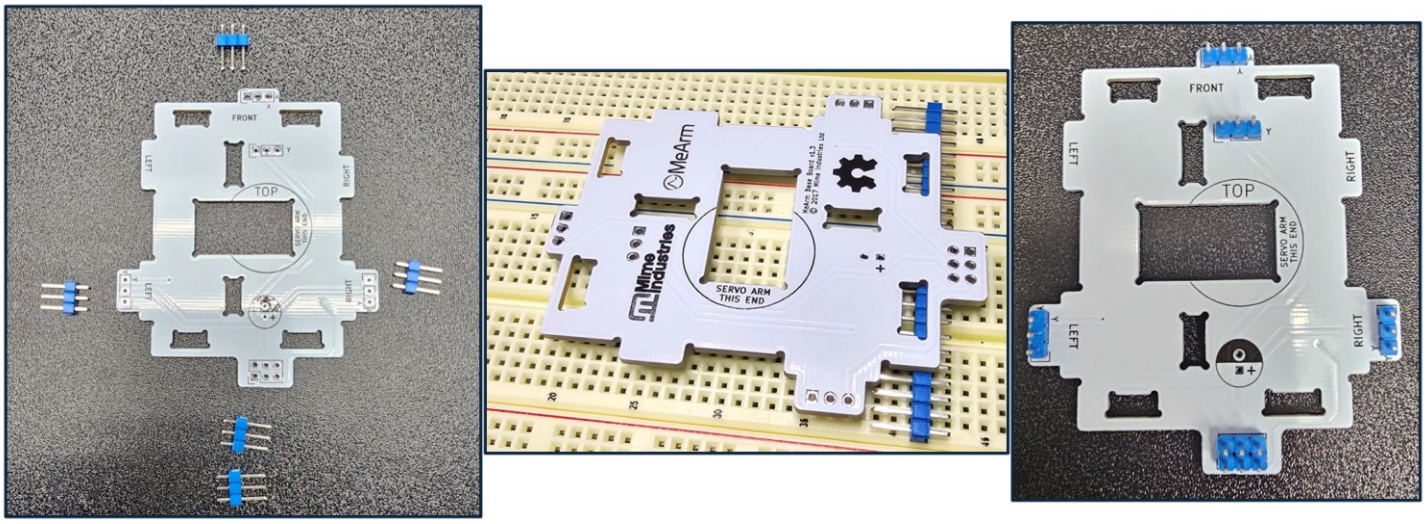

Figure 1. Soldering the MeArm Board Header Pins.

Part 3: Construct the HRIB Shield

Next, you will assemble the HRIB (Human-Robot Interface Board) shield, which plugs into the Arduino Uno. This board will route power to servos, provide connectors for motors, add joysticks and buttons for human control, and include capacitors to stabilize power and prevent servo noise from resetting the Arduino or affecting other servos.

Assembly Steps

1. Prepare and install motor connector header pins (J1 & J2)
    a. Break 8 additional 1x3 header pins from your headers strip (you'll need 4 sections for J1 and 4 for J2).
    b. These headers distribute 5V, GND, & signal to up to 6 servos. The last two provide secondary access to power and SDA/SCL pins for later expansion (Figure 2).
    c. Insert the header pins into J1 & J2 on the top side of the PCB, ensuring they are perpendicular to the board.
    d. **Tip:** Blue tac, helping hands can help hold pins in place while soldering.
    e. Solder all pins from the bottom side of the PCB.

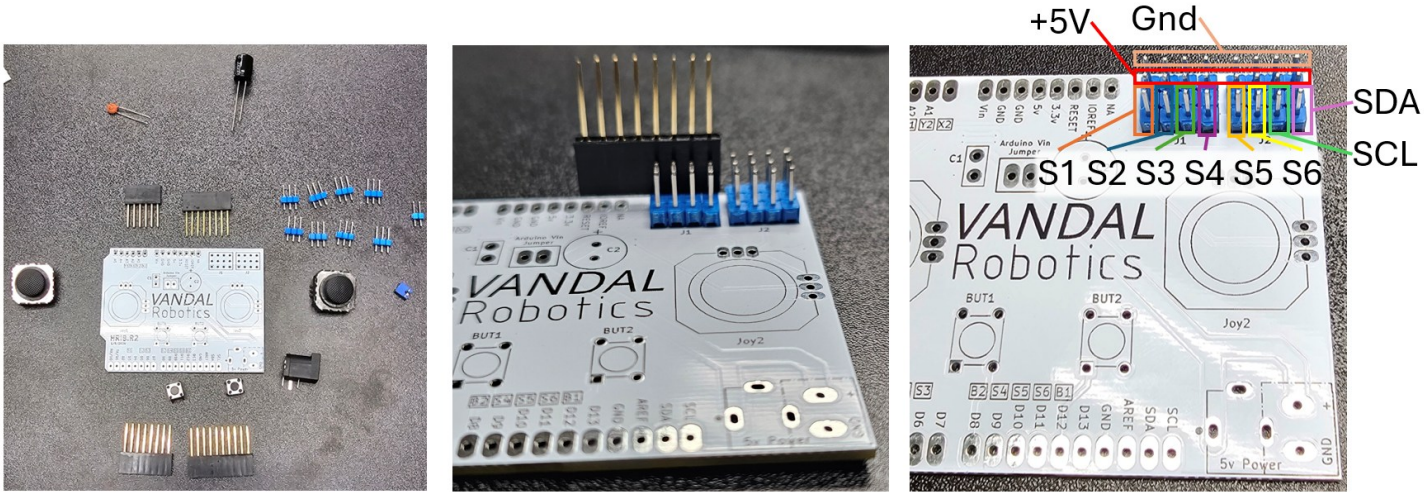

Figure 2. (Left) Components for the HRIB Shield, (Center) Aligning J1 and J2 pins using another header component, and (Right) pin out of J1 and J2.

2. Install the center Arduino power jumper (2x1 header pin)
    a. This 2-pin male header allows you to bridge power from the HRIB shield to the Arduino Uno using a jumper cap
    b. When bridged, this connection powers the Arduino Uno through the shield, eliminating the need for a USB connection once your code is uploaded.
    c. Install the 2-pin header in the designated location (between capacitors 1 and 2).
    d. This jumper should be left open during initial programming and when Arduino is powered via USB.
3. Install capacitors C1 and C2.
    a. Locate capacitors C1 (0.1 μF ceramic) and C2 (1000 μF electrolytic) from your kit. The 0.1 μF capacitor is small, and the 1000 µF is larger and cylindrical.
    b. **Understanding Capacitor function**:
        - C1 filters high-frequency electrical noise. This prevents Arduino from resetting when servos suddenly draw current.
          C2 supplies short bursts of current when servos move suddenly. When servos draw current quickly, C2 supplies this current locally, preventing one servo's motion from causing jitter in other servos.
    c. **Critical C2 Polarity:** The 1000 µF capacitor is polarized and MUST be installed in the correct orientation
        - Identify the negative (-) lead by looking for the stripe marked with - symbols on the capacitor body.
        - The negative lead is also typically shorter than the positive lead.
        - Orient the negative lead toward the BOTTOM of the HRIB board.

d. Install both capacitors.
    - C1 can be installed in either direction
    - Bend the leads slightly on the back to hold the components in place, then solder.
    - Use a flush cutter to trim the capacitor leads close to the board after soldering.

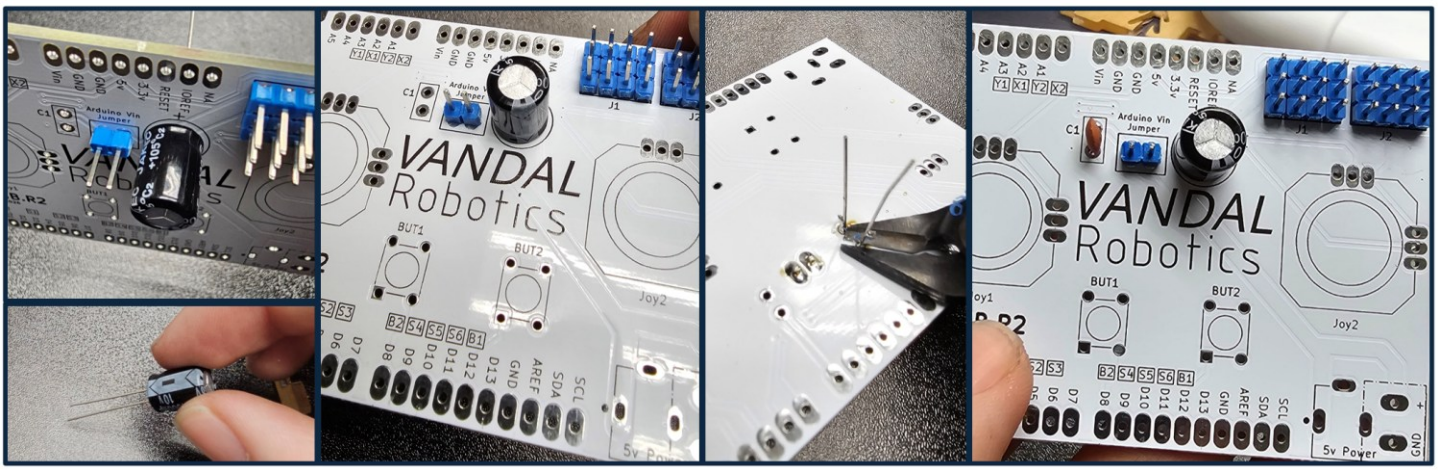

Figure 3. Installing capacitors C1 and C2.

4. Install joysticks and push buttons.
    a. Place the two joystick modules in their designated positions
    b. Ensure each joystick sits flush against the board before soldering
    c. Install the push buttons in their marked locations, also ensuring a flush fit
    d. Solder all pins from the back side of the board
5. Install Arduino shield headers (Stackable header pins)
    a. These long header pins connect the HRIB shield Arduino Uno's headers.
    b. **Critical – Alignment**: Install these headers as straight and perpendicular to the PCB as possible. Angled headers will make it difficult to properly seat the shield on the Arduino.
        - The pins can bend slightly if needed during installation, but proper initial alignment makes assembly much easier.

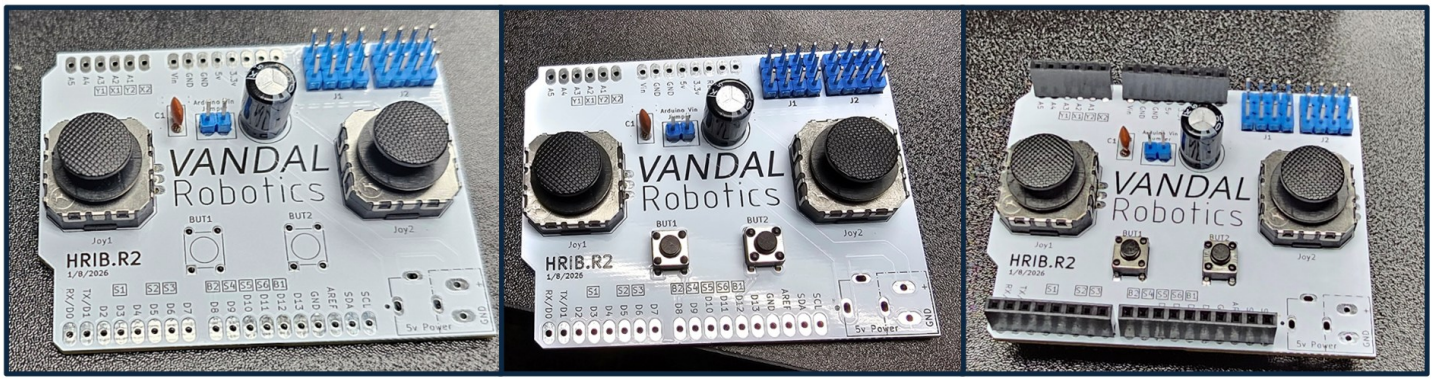

Figure 4. Installing Joysticks, Buttons, and Header Pins.

6. Install power input connector.
    a. **Primary Option – Barrel Jack:** Your kit includes a DC barrel connector (5.5 mm x 2.1 mm center positive). This is the most common connection type for wall adapter power supplies.
    b. Insert the barrel jack into the designated footprint on the PCB and solder.
    c. **Alternative power input options:** The HRIB shield includes extra power input pins for alternate supplies. Your alternate options are:
        - A 5mm pitch screw terminals
        - 0.1” header pins with Dupont connector:
    d. **Power supply requirements:**
        - Voltage: 5-6V DC
        - Current: Minimum 2A, recommended 4A
        - Polarity: Center-positive!
    e. **Electrical Safety:**
        - **Polarity:** Double-check polarity before applying power. Reversed polarity WILL destroy sevos.
        - If making custom cables: work only with the LOW VOLTAGE (DC) side. Never cut or modify the AC (Wall plug) side of a power adapter. This is dangerous and unnecessary
        - **Prevent shorts:** Use heat shrink or electrical tape to insulate all exposed wire connections
        - **Fire Safety:** Never create a short circuit between + and – leads. This can cause rapid overheating and a fire hazard.
7. Inspect and verify your HRIB shield
    a. **Visual inspection checklist:**

    [ ] All solder joints are shiny and have good wetting to both pad and pin (not dull or "balled up")

    [ ] No solder bridges between adjacent pins, loose components or cold solder joints

    [ ] All components are seated flush against the PCB

    [ ] Capacitor C2 polarity is correct (negative stripe toward the bottom of the board)

    [ ] All pins are trimmed flush

    b. **Optional - Multimeter testing:**
        - Set a multimeter to continuity/resistance mode
        - Verify NO continuity (infinite resistance) between 5V and GND pins on motor connectors J1 and J2
        - Verify continuity (low resistance) between corresponding pins on J1 and J2 (they are wired in parallel)
        - If you find a short between 5V and GND, inspect for solder bridges

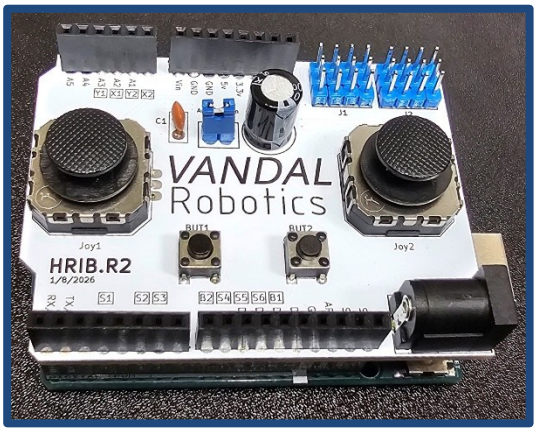

Figure 5. Complete HRIB + Arduino Uno Assembly

Part 4: Create a Servo Motor Circuit, Calibrate and Test Servos

Next, we will test our PCB and motors by creating a simple test circuit to power each servomotor and control them using an Arduino microcontroller

1. Connect power to the HRIB Shield.
    a. Double-check polarity at this point before installing servo motors!
2. Connect the 4 servo motors to the ME-Arm PCB.
    a. Note the locations of motors 1, 2, 3, and 4 (Base, Right, Left, and Claw).
    b. Properly orient ***servo horns***:
        - Servo horns are the plastic part that connects to the output. They should be centered such that the servos have the required range of motion, as shown in Figure 6
        - You will need to control the servo positions using an Arduino. Commanding a position of 1500 µSec will center each motor.
    c. These are **standard positional servos** (~180° range). Do not force them against their mechanical limits. Backdriving should generally be avoided, as it can damage the servo gearing.
    d. You might find it helpful to label each servo to keep them organized later.

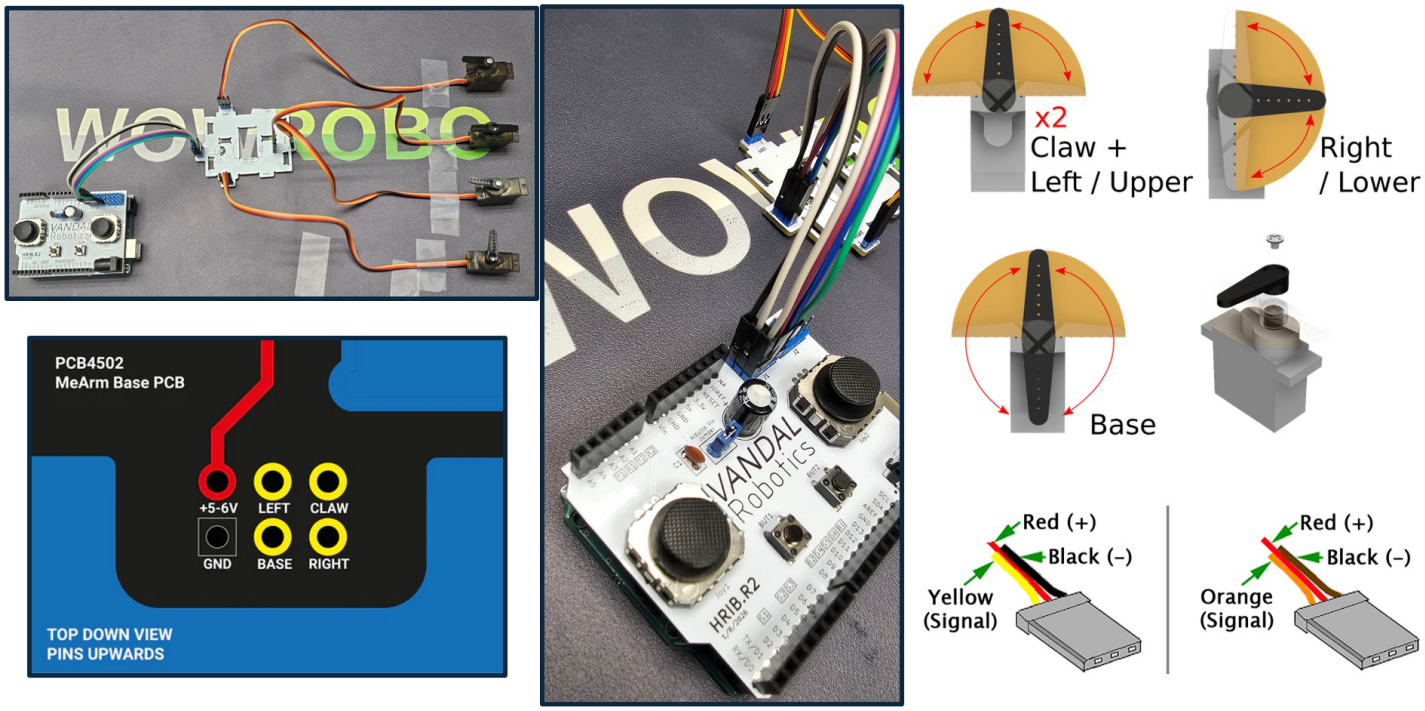

Figure 6. Assembling the Servo Motor Test Circuit. Note the orientation of the servo horns (you may need to adjust them after calibration to center them). Some tape can make it easier to manage the motor wires.

3. Use four F/F jumper cables to connect the servomotor signal wires:
    a. (Base, Right, Left, and Claw) to pins 3, 9, 10, & 11 (S1, S2, S3 & S4 on the HRIB Board) on the Arduino microcontroller via the bottom row of the J1 pins on the HRIB board.
    b. The signal line for these servos is the yellow wires on the servomotor cable. The yellow cables are routed through the MeArm PCB, as shown in Figure 6.
4. Use 2 F/F jumper cables to connect 5V and ground to the ME arm PCB.
    a. Use a top row pin from the HRIB board's J1 for ground and a middle row pin for 5V. These 5V pins are tied to the 1000 µF capacitor.
    b. **Do not** use the 5V supply from Arduino or USB. This will not supply sufficient current.

Phase 2: Servo Motor Control

Part 5: Calibrate the Servomotors

For this part, you will use curve fitting to derive a function to map desired angles (independent variable) to servo microseconds (dependent variable), because future control will use angle commands.

1. The servomotors can be controlled using the servo library described on the Arduino website (Servo | Arduino Documentation).
    a. For the robot assignments, we will use the 'servoname.writeMicroseconds(value)' command.
    b. This command has a better resolution than the '**servoname.write(value)**' command. Sample code for this library is available on Canvas.
2. Each servomotor has a range of slightly less than 180°, corresponding to a range of about 500 to 2500 microseconds. These are approximations; thus, we will perform calibration to get more accuracy.
3. Use the provided calibration Arduino program to command the servomotor to hold at a constant microsecond value between 500 and 2500 microseconds.
    a. Collect at least 10-15 data points across the full range (e.g., every 100 – 200 microseconds).
    b. At each commanded microseconds value, measure the output angle in degrees.
    c. You can print a protractor image to help take angle measurements (See Figure 7)
    d. Record both "upscale" measurement (going from low to high microseconds) and a "downscale" measurement (going from high to low) to check for hysteresis.
4. Use curve fitting (e.g. MATLAB's "basic fitting" or "polyfit" command or Python's "numpy.polyfit") to find servo microseconds as a function of the desired angle in degrees.
    a. Set the middle of the servo's range to zero degrees. In future assignments, we'll adjust the zero point of each servo to match the robot's home position.
    b. The servo should then have a range of a little less than ±90°. The rotation axis should be assumed to be positive, extending out from the top of the servo.
    c. Ideally, the servomotors will have a linear relationship between microseconds and angle. Start with a linear fit. Only use higher order (quadratics, cubics, etc.) is residuals are clearly nonlinear. Sample fitting code is provided in Figures 7 and 8.
5. Create a calibration plot of microseconds vs. degrees, using MATLAB or Python, for all four servo motors.
    a. For your raw data points, use markers (use markers +/o/*/x and colors red/green/blue/black for servos 1/2/3/4, respectively)
    b. Plot your calibration curve as a line (match the color to the marker color).
    c. Ensure you properly annotate your graph (legend, axis labels with units, title).
    d. Put your fitted calibration equation(s) on the plot using the "text" command in MATLAB or the "text()" or "annotate" function in Python.
    e. Save your figure as a **.PDF** or **.png** for submission on Canvas.

MATLAB Calibration Example Code:

```
close all

data = [800,36,38;
        900,48.5,49;
        1000,59,60.5;
        1100,71.5,73;
        1200,83.5,85;
        1300,95.5,96.5;
        1400,107,108;
        1500,118,119.5;
        1600,129,129.5;
        1700,139,140;
        1800,148.5,149;
        1900,157.5,158.5;
        2000,165.5,166.5;
        2100,174.5,175;
        2180,180,180];

fitorder = 2;
p = polyfit([data(:,2) data(:,3)],[data(:,1) data(:,1)],fitorder);

figure(1)
hold on
grid on
% ylim([0 180])
plot(data(:,2),data(:,1),'bx','Markersize',8,'Linewidth',2)
plot(data(:,3),data(:,1),'g+','Markersize',8,'Linewidth',2)
thetas = linspace(data(1,2),data(end,2),100);

if fitorder == 1
    plot(thetas,p(1)*thetas + p(2),'r-','Linewidth',2)
elseif fitorder == 2
    plot(thetas,p(1)*thetas.^2 + p(2)*thetas + p(3),'r-','Linewidth',2)
elseif fitorder == 3
    plot(thetas,p(1)*thetas.^3 + p(2)*thetas.^2 + p(3)*thetas + p(4),'r-','Linewidth',2)
end

xlabel('Servomotor Angle (\circ)')
ylabel('Servomotor PWM command (\mus)')

```

Figure 7. Example calibration code. The first column of data is the commanded microseconds, followed by measured angles from an upscale calibration test in column 2, and a downscale calibration test in column 3. The polyfit command uses all of the data by concatenating both columns into a single list. Changing the variable fitorder will change the order of the fit (1 = linear, 2 = quadratic, 3 = cubic, etc.)

Python Calibration Example Code:

```python
# Python Example (using NumPy and Matplotlib)
import numpy as np
import matplotlib.pyplot as plt

data = np.array([[800, 36, 38],
                 [900, 48.5, 49],
                 [1000, 59, 60.5],
                 # ... (complete data)
                 [2180, 180, 180]])

fitorder = 2
# Concatenate angle data from both columns
angles = np.concatenate([data[:, 1], data[:, 2]])
microseconds = np.concatenate([data[:, 0], data[:, 0]])

# Fit polynomial
p = np.polyfit(angles, microseconds, fitorder)

plt.figure(1)
plt.plot(data[:, 1], data[:, 0], 'bx', markersize=8, linewidth=2, label='Upscale')
plt.plot(data[:, 2], data[:, 0], 'g+', markersize=8, linewidth=2, label='Downscale')

thetas = np.linspace(data[0, 1], data[-1, 2], 100)
fit_values = np.polyval(p, thetas)
plt.plot(thetas, fit_values, 'r-', linewidth=2, label='Fit')

# Add equation to plot
equation_text = f'y = {p[0]:.4f}x² + {p[1]:.4f}x + {p[2]:.4f}'
plt.text(0.05, 0.95, equation_text, transform=plt.gca().transAxes,
         verticalalignment='top', bbox=dict(boxstyle='round', facecolor='wheat'))

plt.xlabel('Servomotor Angle (°)')
plt.ylabel('Servomotor PWM command (µs)')
plt.grid(True)
plt.legend()
plt.savefig('calibration_plot.pdf')
plt.show()
```

Figure 8. Example calibration code implemented in Python.

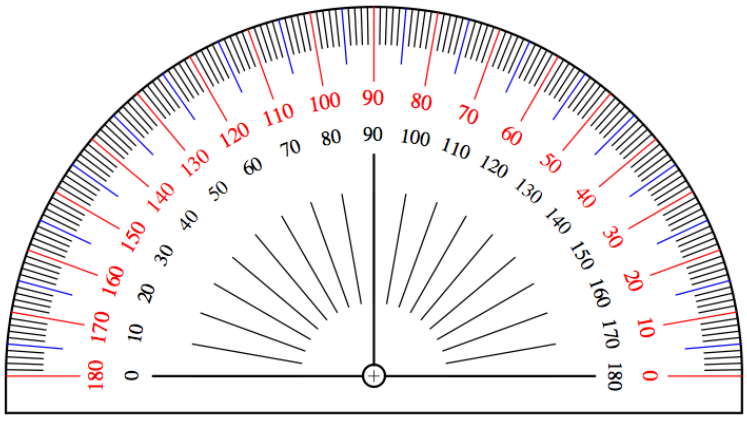

Figure 9. Protractor image.

Part 6: Create an Arduino program to test the servos

In this part, you will create a program to move the servos in sinusoidal patterns at different frequencies.

1. The servomotors should follow a sine wave, covering a range of (±60°) at the following frequencies:
    a. Servomotor 1: 0.25 Hz
    b. Servomotor 2: 0.5 Hz
    c. Servomotor 3: 1 Hz
    d. Servomotor 4: 2 Hz
2. Use your calibration equation to convert from a desired angle (in degrees) to a desired microsecond value.
3. In your Arduino code:
    a. Calculate the desired angle for each servo using: angle = amplitude × sin(2π × frequency × time)
    b. Convert the angle to microseconds using your calibration equation.
    c. Write the time (in milliseconds) and commanded microseconds for each servomotor to the serial monitor.
    d. Target a sampling rate of ~ 25 – 50 Hz (e.g., a **delay()** of between ~ 20-50 ms).
4. Run your Arduino program and record serial monitor data for approximately 4 seconds. Copy this data to a text file for analysis.
5. In MATLAB or Python
    a. Import the data from your serial monitor output.
    b. Convert the microseconds of each servomotor back to degrees using your calibration equations.
    c. Create a single plot showing degrees vs. time (as lines, use colors redm green, blue, black for servos 1/2/3/4, respectively).
    d. Include proper axis labels, legend, title, and grid.
    e. Save as a PDF or PNG for submission.
6. Create a short video (<30 seconds) of your 4 servomotors following the sinusoidal motions.
    a. Talk during the video: state the project assignment name/number and your name.
    b. Show all four servos moving simultaneously.

Phase 3: Build the Robot

Part 7: Assemble the Robot

Now that you have confirmed the servos and PCB work properly, you assemble you ME arm.

1. Collect your tools. Your kit includes a hex key. You will also need a small philips driver. These are available in the machine shop & design suite.
2. Follow the assembly instructions provided online at the instructables link:
   Pocket Sized Robot Arm - MeArm V3.0 - Small, Hackable, Open Source : 18 Steps (with Pictures) - Instructables
3. **Important Note:** If you do need to back-drive the servos during assembly. Apply torque slowly and carefully to avoid damaging the internal servo gearing. **Excessive back-driving can damage Servo Gearing.**
4. Test the Robot
    a. Reconnect your motors and use your Arduino code to drive each motor and verify the range of motion.
    b. Test that each joint moves through its full range without binding or obstruction.
    c. If the robot functions as expected, congratulations, you have successfully built the MeArm V3!
    d. If not, double-check your assembly. We will reserve some class time for troubleshooting.

WHAT TO SUBMIT

Complete the above and submit ***the following files*** (*do not zip*!) on Canvas. Use the specified naming convention with no spaces in filenames and replace my initials with your own.

1. **RobotA_Lastname_CalibrationPlot.pdf** (or .png)**:** A single PDF or image file with your calibration plot showing all four servos. The plot should include:
    a. Raw data points with appropriate markers and colors
    b. Fitted curves for each servo
    c. Calibration equations displayed on the plot
    d. Proper axis labels, legend, and title
    e. Font size of 12 pts or larger
2. **RobotA_Lastname_TestingPlot.pdf** (or .png)**:** A plot showing degrees vs. time for all four servos during the sinusoidal test. Should include proper formatting as described above.
3. **RobotA_Lastname_video.mp4** (or.mov): A short video (<30 seconds) of the servos tracking sinusoidal waves. State your name and the assignment name at the beginning.
4. **RobotA_Lastname_Arduino.pdf:** A single PDF document of your Arduino code with your name at the top (commented out). To create a PDF your Arduino code with formatting intact:
    a. From the Arduino IDE, select your code and click Edit -> Copy as HTML
    b. Paste the text into a text file and save it with a .html extension
    c. Open the HTML file in your web browser and copy all text.
    d. Paste into your document in Microsoft Word (or similar) for conversion to PDF.
5. **RobotA_Lastname_PlottingCode.pdf:** A single PDF document of your MATLAB or Python plotting code used to create both plots.
    a. In MATLAB: Highlight the code and paste it into Word using “keep source formatting”.
    b. In Python: Copy code and paste with formatting preserved.
    c. Put your full name at the top of the program (commented out).
6. **RobotA_LastName_RobotPicture.pdf** (or .png): A clear photo of your fully assembled robot arm.

GRADING RUBRIC

1. Part 1-4: PCB assembly and servo connection (15%)
2. Part 5: Servo calibration and plot (25%)
3. Part 6: Sinusoidal testing, plot, and video (25%)
4. Part 7: Robot assembly and photo (15%)
5. Code quality and documentation (10%)
6. Proper file naming and submission format (10%)