ROBOTIC SORTING OF LAUNDRY

By

Danny Coombs
Anusha Nagabandi
Sid Sethupathi

Final Report for ECE 445, Senior Design, Spring 2015

TA: Kevin Chen

03 May 2015
Project No. 31
Abstract

We have designed and implemented an automated robotic arm that sorts small-scale models of colored clothing. Our aim was to combine the concepts of kinematics and embedded systems in order to create this framework for a sorter. This system consists of a mechanical arm, a microcontroller as the computing system, the Pixy CMUCam5 as the camera, a custom PCB with peripherals such as buttons and switches, and a power supply. Fully integrated, this robotic arm successfully identifies, gathers, and sorts clothing of three different colors. Additionally, while this describes our demo, our project is actually the framework for a generalized sorter that can easily be adapted to sort various items.
# Contents

1. Introduction ............................................................................................................................................. 1
   1.1 Purpose and Function of Project ........................................................................................................ 1
   1.2 Block Diagram .................................................................................................................................... 1

2 Design ....................................................................................................................................................... 2
   2.1 Power Supply ...................................................................................................................................... 2
      2.1.1 Design procedure ...................................................................................................................... 2
      2.1.2 Design details ............................................................................................................................ 3
   2.2 Microcontroller Board ....................................................................................................................... 5
      2.2.1 Design procedure ...................................................................................................................... 5
      2.2.2 Design details ............................................................................................................................ 5
   2.3 Kinematics ......................................................................................................................................... 7
      2.3.1 Forward Kinematics .................................................................................................................. 7
      2.3.2 Inverse Kinematics .................................................................................................................. 10
   2.4 Camera Module ............................................................................................................................... 12
   2.5 Software and Integration ................................................................................................................... 13

3. Design Verification ................................................................................................................................. 15
   3.1 Power Supply ...................................................................................................................................... 15
   3.2 Microcontroller Board ....................................................................................................................... 16
   3.3 Kinematics ......................................................................................................................................... 17
      3.3.1 Forward Kinematics .................................................................................................................. 17
      3.3.2 Inverse Kinematics .................................................................................................................. 17
   3.4 Camera Module ............................................................................................................................... 18
   3.5 Software Module and Integration ..................................................................................................... 18

4. Costs .......................................................................................................................................................... 20
   4.1 Parts .................................................................................................................................................... 20
   4.2 Labor .................................................................................................................................................. 21

5. Conclusion ............................................................................................................................................... 22
   5.1 Accomplishments .............................................................................................................................. 22
   5.2 Ethical Considerations ....................................................................................................................... 22
5.3 Future Work .................................................................................................................. 23

References............................................................................................................................ 24

Appendix A Requirement and Verification Table .................................................................. 25

Appendix B Schematics........................................................................................................... 29
1. Introduction

1.1 Purpose and Function of Project

The purpose of this project was to design and build a robotic system that combined the concepts of kinematics and embedded systems in order to automate the trivial task of sorting laundry. Automation is important as it frees up time to do other tasks, allowing the end-user to more effectively use his or her time. Additionally, many research groups have to buy expensive robots and systems before they can do even such a simple task as sorting; this project would allow groups to avoid those expenses, while also preventing them from spending weeks or months to build such a framework from scratch. This programmable sorter is the first step toward the bigger goal of laundry sorting and folding in general, and this project is also a general setup of the hardware and software framework which can be easily extended for future implementation.

1.2 Block Diagram

There are two main systems in this project: an electrical system and a mechanical system. The electrical system consists of the power supply, computing system, camera module, and the peripheral I/O subsystems. The mechanical system consists of the servos and the physical arm. A more detailed description of the design and verification of each subsystem is provided in the following sections, but the overall sequence of tasks for this system is as follows: the camera detects a colored object in the basket, the camera sends the information to the microcontroller, the microcontroller computes the angles values, the microcontroller sends PWM signals to the motors, the arm moves to the desired location, the claw picks up the object, the arms moves and drops the object in a specified location, and the process repeats until nothing remains in the basket.
2 Design

2.1 Power Supply
To power our robotic arm’s six servos, the microcontroller, the Pixy camera, and the LCD screen, we designed and built a power supply that plugs into any 115 V-120 V wall outlet.

2.1.1 Design procedure
The Servos, Pixy camera, and LCD screen all run at 5 V. Therefore, our power supply is designed to provide 5 V, and our microcontroller has its own 3.3 V linear regulator (see section 2.2). The maximum power consumption of the system's components determines the current rating of the power supply. Each servo consumes 2.5 A of current at 5 V when stalling, so we move only one servo at a time to allow us to design a 3 A power supply. The following equations summarize the power needs of our system:

\[
\text{Servo(stalling)} = 2.5A \\
\text{System(servos \_idle)} = uC + \text{Pixy} + IR + \text{LCD} \\
\text{System(servos \_idle)} = 5mA + 140mA + 30mA + 5mA = 180mA \\
\text{System(max)} = 2.5A + 180mA = 2.68A
\]

These equations show that our power supply needs to be rated to 3 A at 5 V. The ripple requirements are specified by the most sensitive device in the system, which in our case is the Pixy. Since the Pixy can only tolerate 5 V +/- 5% [1], our power supply's ripple requirement is also the same.

These design goals guided us toward the following architecture for the power supply. We use a transformer in our design in order to isolate our system from the high voltage of the outlet, mostly for safety reasons. The output of the transformer then goes through rectification and smoothing to produce a DC output. Since the ripple requirements of this power supply are so constrained, a regulation stage to the design is necessary. A switching regulator is preferred over a linear regulator due to its efficiency at the high current output that we needed. The final design architecture can be visualized by following the voltage in the block diagram for our power supply shown in Figure 2.

![Figure 2. Power Supply Block Diagram](attachment:image.png)
2.1.2 Design details

Designing the first stage of the power supply required choosing the correct transformer and smoothing capacitor for our system. The voltage is stepped down to 10 V AC in order to provide enough voltage for the input to our switching regulator, and this specifies our transformer. The smoothing capacitor’s value is decided using equations (5) and (6), and this value ensures that our ripple at this point is not too high for the switching regulator [2].

The schematic for this stage of the power supply is shown in Figure 3.

\[
RC \gg \frac{1}{f} \quad \text{where} \quad f = 2 \times \text{lineFrequency} = 120\text{Hz} \\
R_{\text{load}} = \frac{V}{I} = \frac{12.8V}{3A} = 4.2\Omega \\
C \gg 1984uF \quad \text{choose} \quad C = 3300uF
\] 

Figure 3. Power Supply Stage 1

The second stage of our system uses the TI LM676S-5.0 switching regulator. This regulator is rated to output up to 3 A at 5 V with minimal ripple. To use this regulator correctly, it needs input capacitors, a feedback capacitor, a clipping diode, an output inductor, and an output capacitor. All of these values can be chosen using the data sheet for the LM676 [3] with the design constraints of 13.5 V input and 5 V/3 A outputs of the regulator. The final values for these components can be found in the schematic in Figure 4.

Figure 4. Power Supply Stage 2
To finalize our design and convince ourselves that it would work, we simulated the power supply using PSPICE. A model for the LM676 [4] was found on TI’s website to be used with PSPICE. TI only provides a model for their 3.3 V version of the regulator, so this was used in simulation instead of a 5 V regulator. The first simulation that we completed was the output of the transformer with an input of 120 V AC at 60 Hz. The expected output of 10 V AC was observed, as shown in Figure 5.

![Figure 5. Transformer Simulation: Voltage (V) vs Time (ms)](image)

The next simulation that we completed was the output of the LM676 regulator with a 3.3 Ω load. We expected an output of 3.3 V with less than 200 mV of ripple and a current of 1 A. This expected output was observed, as shown in Figure 6.

![Figure 6. Switching Regulator Simulation](image)
2.2 Microcontroller Board

We designed our microcontroller board to house the microcontroller, to provide connections to the peripheral components, and to provide components for user interaction.

2.2.1 Design procedure

The first major design decision was the choice of microcontroller. The microcontroller for our project needs to have sufficient I/O capabilities in order to communicate with our large number of peripheral components. The microcontroller also needs to have at least 6 PWM outputs to control the servos. We also require an easy to use microcontroller that operates at relatively low power.

The microcontroller that we use is the TI MSP430F5529. This microcontroller has just enough I/O capabilities to work with our design. It can also be programmed with a TI Launchpad through SPY-BI Wire, and we can write code for it using the Energia platform. Since the image processing of our system is designed to take place on the Pixy camera, the microcontroller code is not computationally intensive and the MSP430’s processing power can easily handle the tasks for our application.

We then decided to design our own board to connect this microcontroller to all of its peripherals, such as buttons and switches. Having a custom board with headers enables us to easily use a large number of connections. Our board takes in a 5 V input from a power supply, and it has connections for all of our peripheral components; a summary of these inputs and outputs is shown in Figure 7.

![Microcontroller Inputs and Outputs](image)

**Figure 7. Microcontroller Inputs and Outputs**

2.2.2 Design details

A 5 V power supply is used to power the microcontroller board. Since the microcontroller runs at a voltage between 1.7 V and 3.5 V, a regulator is used to create this voltage from the 5 V supply. We use a linear regulator due to the low power consumption of the MSP430. Using equation (8), we see that the
power dissipation across the linear regulator is very low even at maximum current consumption of the microcontroller, so it is acceptable for our use.

\[
P_{\text{diss}} = (5V - 3.3V) \times 5 \text{mA} = 8.5mW
\] (8)

We use the TI TPS7933 linear regulator due to its ease of use and its small package size. Using input and output decoupling capacitors as designated in the TPS7933 data sheet [5], the circuit in Figure 8 powers our microcontroller.

![Linear Regulator Diagram](image)

Figure 8. Linear Regulator

The microcontroller board schematic design also includes .1” headers for the 6 servos, Pixy SPI, LDD screen, IR sensor, and programmer/debugger. At each of the power outputs of the board, a 100nF decoupling capacitor is placed to ensure that noise created by peripheral components does not couple onto the microcontroller board. Two buttons and a switch, with their accompanying pull-up/pull down resistors, are also on the board. Additionally, two LED’s are on the board; they have 470 Ω limiting resistors whose values are calculated using equation, where the forward drop is 2.2 V and the current is less than 3 mA (9).

\[
R > \frac{3.3V - 2.2V}{3mA} = 367\Omega
\] (9)

A full schematic of the board can be found in Appendix B. The board uses a large trace width for the 5 V power line since this line can possibly draw up to 3 A of current. This minimum trace width is determined using equation (10), where \(h\)= thickness in mil = 2, \(k = 0.048\), \(b=0.44\), \(c= 0.725\), and the maximum increase in temperature is 10 degrees Celsius [6].

\[
W = I \times h \left( (k \times \text{Temp})^b \right)^{1/c}
\]

\[
W > 24.4\text{mils}
\] (11)
2.3 Kinematics

We need a relationship between the angles of each motor and the end location of the claw: Forward and inverse kinematics address exactly this. The problem of forward kinematics addresses using the motor angles to determine the position and orientation of the claw. Inverse kinematics then takes the information developed here to derive the angle values that each motor should go to in order to reach the desired location.

2.3.1 Forward Kinematics

As mentioned above, the problem of forward kinematics uses the joint variables to determine the position and orientation of the end effector. In our case, the joint variables are the angles of each servo, and the end effector is the tip of the claw. The first step to formulating forward kinematics is to identify each joint and link in the system.

Note that when representing the arm as a series of joints and links, link $i$ connects joint $i$ and joint $i+1$. Next, we assign a coordinate system at each joint such that regardless of the motion of link $i$, the coordinates on link $i$ are constant when expressed in the $i^{th}$ coordinate frame. While we can go ahead and arbitrarily assign coordinate systems and develop transformation matrices, we instead use the Denavit-Hartenberg convention [7]. This convention of DH parameters adopts certain rules and specifications that lead to a universal method for people to follow, as well as a simplified analysis of the problem itself. The first convention sets $z_i$ as the axis of revolution of joint $i+1$, as shown in Figure 10.
Next, we assign the y and x axes for each of our joints, which fall under one of the two following cases.

**Case 1:** For \( z_i \) being parallel with \( z_{i-1} \)
- Choose \( x_i \) to go from \( o_i \) to \( o_{i-1} \)
- Then specify \( y_i \) using RHR
- Note that here,
  \[ d_i = \text{link offset} = 0 \]
  \[ \alpha_i = \text{link twist} = 0 \]

**Case 2:** For \( z_i \) intersecting \( z_{i-1} \)
- Choose \( x_i \) perpendicular to the plane created by \( z_i \) and \( z_{i-1} \)
- Then specify \( y_i \) using RHR and note that here, \( a_i = \text{link length} = 0 \)

![Diagram of assigning coordinate systems to each reference frame](image)

**Figure 11. Assigning coordinate systems to each reference frame**

After the set up of the coordinate systems, we then use them to specify the DH parameters, as explained below.

- **a:** Link length. Distance along \( x_i \) from the intersection of the \( x_i \) and \( z_{i-1} \) axes to \( o_i \)
- **d:** Link offset. Distance along \( z_{i-1} \) from \( o_{i-1} \) to the intersection of the \( x_i \) and \( z_{i-1} \) axes
- **\( \alpha \):** Link twist. Angle from \( z_{i-1} \) to \( z_i \), measured about \( x_i \)
- **\( \theta \):** Joint angle (the variable). Angle from \( x_{i-1} \) to \( x_i \), measured about \( z_{i-1} \)

<table>
<thead>
<tr>
<th>Link</th>
<th>( a_i )</th>
<th>( d_i )</th>
<th>( \alpha_i )</th>
<th>( \theta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-90</td>
<td>( \theta_1 )</td>
</tr>
<tr>
<td>2</td>
<td>( a_2 )</td>
<td>0</td>
<td>0</td>
<td>( \theta_2 )</td>
</tr>
<tr>
<td>3</td>
<td>( a_3 )</td>
<td>0</td>
<td>0</td>
<td>( \theta_3 )</td>
</tr>
<tr>
<td>4</td>
<td>( a_4 )</td>
<td>0</td>
<td>0</td>
<td>( \theta_4 )</td>
</tr>
</tbody>
</table>

*Table 1. DH Parameters for the links of interest in the arm*

Knowing these parameters, we have all of the information needed for forward kinematics. Now we set up \( T_i \), which is the transformation matrix that expresses the position and orientation of \( o_x y z_i \) with respect to \( o_x y z_i \). As explained in the explanation of forward kinematics, we desire \( T_{04} \), which is the position and orientation of the end effector with respect to the base frame. These expressions are shown below in equations (12) and (13).
\[ T_j^i = A_{i+1}A_{i+2} \cdots A_jA_j \]  
\[ T_4^0 = A_1A_2A_3A_4 \]

Note that \( A_i \) is given by the expression in equation (14), where \( c\theta_i \) means \( \cos(\theta_i) \) and \( s\theta_i \) means \( \sin(\theta_i) \). Using the DH parameters from Table 1 along with the equations above, the following matrices can be written [7].

\[
A_i = \begin{bmatrix}
c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\alpha_i \\
s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\alpha_i \\
0 & sai & cai & di \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
A_1 = \begin{bmatrix}
c\theta_1 & 0 & -s\theta_1 & 0 \\
s\theta_1 & c\theta_2 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
A_2 = \begin{bmatrix}
c\theta_2 & -s\theta_2 & 0 & a_2 c\theta_2 \\
s\theta_2 & c\theta_2 & 0 & a_2 s\theta_2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
A_3 = \begin{bmatrix}
c\theta_3 & -s\theta_3 & 0 & a_3 c\theta_3 \\
s\theta_3 & c\theta_3 & 0 & a_3 s\theta_3 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
A_4 = \begin{bmatrix}
c\theta_4 & -s\theta_4 & 0 & a_4 c\theta_4 \\
s\theta_4 & c\theta_4 & 0 & a_4 s\theta_4 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The significance of the elements of the T matrix is shown below [7], where the 3x3 matrix of r-values represents the orientation of the end effector in the base frame, and the 3x1 matrix of d-values represents the coordinates of the end effector location in the base frame.

\[
T_j^i = \begin{bmatrix}
r_{11} & r_{12} & r_{13} & dx \\
r_{21} & r_{22} & r_{23} & dy \\
r_{31} & r_{32} & r_{33} & dz \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Using the equations above, we derive the following equations for the location of the tip of the claw.

\[
x = a_4 c_1 c_2 c_3 - a_4 c_2 c_3 s_2 c_4 - a_4 c_1 c_2 s_3 s_4 - a_4 c_1 c_2 c_3 s_4 + a_3 c_1 c_2 c_3 - a_3 c_1 c_2 s_3 - a_2 c_1 c_2 \]

\[
y = a_4 s_1 c_2 c_4 - a_4 s_1 c_3 s_2 c_4 - a_4 s_1 c_2 s_3 s_4 - a_4 s_1 c_2 c_3 s_4 + a_3 s_1 c_2 c_3 - a_3 s_1 c_2 s_3 - a_2 s_1 c_2 \]

\[
z = -a_4 s_2 c_3 c_4 - a_4 s_2 c_3 s_4 - a_4 s_2 s_3 - a_4 c_2 c_3 - a_3 c_2 c_3 - a_3 c_2 s_3 - a_2 c_2 \]

As illustrated in this section, forward kinematics allows us to use the motors angles to determine the location and orientation of our end effector. Next, we formulate the inverse kinematics for the system.
2.3.2 Inverse Kinematics

Numerous methods exist for solving inverse kinematics, including the Jacobian inverse technique and other approximations. We use a geometric approach instead, because we want a less computationally intense solution. Furthermore, a geometric solution is not difficult, given that we can reduce the complexity of the system.

We first decouple the claw from the arm to make the geometry easier to visualize. This technique also allows us to place the constraint that the claw should always be pointing down. Step one for this technique is to solve for angles 1, 2, and 3 that lead to the third link ending in position (x, y, z). Step two is to then sweep through possible values for angle 4, plugging each value into forward kinematics and stopping when the end of the fourth link is directly below the end of the third link. The figure below shows the end of the third link, and the end of the fourth link.

Figure 12. Separating the problem into two main steps

Solving for the first three angles deals with looking at a system as shown in Figure 13a. Angle 1 comes naturally from looking at a top view of the arm, as shown in figure 13b, whereas angle 3 comes from looking at a side view of the arm as shown in figure 13c.

![Diagram showing angles and links](image)

Figure 13. a) Solving the simpler problem of 3 links b) Top view, solving for angle 1 c) Side view, solving for angle 3

From the diagrams labeled in Figure 13, we employ the law of cosines and other basic geometry to derive the following equations to express angle 1 and angle 3.
\[
\theta_1 = \tan^{-1}\frac{y}{x}
\]  
(23)

\[
2a_2a_3 \cos \theta_3 = r^2 + z^2 - a_2^2 - a_3^2
\]  
(24)

\[
\sin \theta_3 = \sqrt{1 - \cos^2 \theta_3}
\]  
(25)

\[
\theta_3 = \tan^{-1}\frac{\sin \theta_3}{\cos \theta_3}
\]  
(26)

Next, angle 2 is calculated in a similar way. After testing these equations using real numbers, as explained in the design verification section 3.3, we discovered that there are actually two possible configurations for this angle.

![Diagram](image)

**Figure 14.** a) Case 1, reaching upward  
b) Case 2, reaching downward  
c) Separating the angle \( \theta_2 \) into two parts

From the diagrams shown in figure 14a and 14b, we again employ the law of cosines to derive the following equations for angle 2.

\[
a = \tan^{-1}\frac{z}{r}
\]  
(27)

\[
\sin b = a_3 \sin \theta_3
\]  
(28)

CASE 1: \( \cos b = a_3 \cos \theta_3 + a_2 \)  
(29)

CASE 2: \( \cos b = a_3 \cos \theta_3 + a_2 \)  
(30)

\[
b = \tan^{-1}\frac{\sin b}{\cos b}
\]  
(31)

\[
\theta_2 = a + b
\]  
(32)

Finally, the last step in our technique for formulating the inverse kinematics of the arm involves finding angle 4. As mentioned above, the procedure for this involves a for-loop, sweeping through possible values.
of angle 4; for each angle value, we plug the four angles into the forward kinematics equations. As shown above in Figure 12, the correct value for angle 4 will be that which gives us an (x,y,z) for the 4th link that is directly below the (x,y,z) for the 3rd link. Shown below in Figure 15 is a series of images showing the sweeping of angle 4; the correct angle is the middle image, because the end of the 4th link is directly below the end of the 3rd link. In other words, this is the correct angle 4 for which the claw will be pointing straight down.

![Figure 15. Sweeping through values for angle 4](image)

**2.4 Camera Module**

This subsystem looks for objects in the robot's field of vision. Part of the logic for the camera module, although the actual calculations are done on the microcontroller, is to translate pixels from the camera's point of view into coordinates in the robot’s point of view. The camera that is used in this project is a Pixy CMUCam5 from CharmedLabs at Carnegie Mellon University [8]. The Pixy is a USB camera that stores color signatures based on hue in its onboard flash memory. Information about detected objects, based on color signature matches, is sent over SPI (Serial Peripheral Interface) from the Pixy to the microcontroller. The data sent over SPI is shown in Table 2.

<table>
<thead>
<tr>
<th>Word 0</th>
<th>Sync (0xAA88)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>Checksum of words 2-6</td>
</tr>
<tr>
<td>Word 2</td>
<td>Signature Number</td>
</tr>
<tr>
<td>Word 3</td>
<td>x center of object</td>
</tr>
<tr>
<td>Word 4</td>
<td>y center of object</td>
</tr>
<tr>
<td>Word 5</td>
<td>width of object</td>
</tr>
<tr>
<td>Word 6</td>
<td>height of object</td>
</tr>
</tbody>
</table>

*Table 2. Data block sent over SPI from Pixy to microcontroller [8]*

Using the signature, center coordinates, width, and height of the object, the Pixy detects objects in terms of pixels. The microcontroller converts the coordinates from the Pixy point of view to the robot point of view. In order to do this calibration, we use a calibration tool as shown in Figure 16, which has markers with known coordinates.
Upon program initialization, this tool is placed in front of the robot. Since the two markers are fixed, a one-to-one mapping between camera coordinates and robot coordinates is determined for these markers. A pixel-per-centimeter (ppc) value is calculated by the following equation

\[ ppc = \frac{py_1 - py_2}{ry_1 - ry_2} \]  

(33)

where \( ry_x \) is the robot’s y coordinate for marker \( x \) and \( py_x \) is the Pixy’s y coordinate for marker \( x \). Using this information, dead reckoning of a random object placed in the field of view is calculated by the following equations

\[
\begin{align*}
  r_{\text{final}_y} &= \frac{p_x - p_{\text{marker}_x}}{ppc} + r_{\text{marker}_y} \\
  r_{\text{final}_x} &= \frac{(200 - p_y) - p_{\text{marker}_y}}{ppc} + r_{\text{marker}_x}
\end{align*}
\]

(34)  

(35)

where \( p_{\text{marker}_i} \) is the ‘i’ coordinate in the Pixy point of view and \( r_{\text{marker}_i} \) is the respective ‘i’ coordinate in the robot point of view. In equation (33), an offset of 200 is used in order to account for the height of the camera’s output. In this way, we translate a pixel value into an x, y, z location.

### 2.5 Software and Integration

After the design and verification of each individual component of our project, the integration is the next step. Some notable steps include calibrating motors, assembling the arm, and physically connecting the components into a system. Note that the camera is not mounted on the arm due to the small size of the arm, which is why we need the calibration, as mentioned above is section 2.4.
The computing system is a main part of integration, and it deals with data acquisition, data processing, and also data actuation. The computing system also keeps the general flow of events, as is shown below in Figure 17.

A few mappings have already been discussed: calibration to go from pixels to coordinates, and a linear transformation from calculation angles to motor angles. The remaining step involves a mapping from desired location to motor angles, which is exactly what we solved for using inverse kinematics. Since the MSP430F5529 has no native floating point architecture, angles are calculated offline using inverse kinematics and then uploaded into the microcontroller’s RAM (Random Access Memory) as a lookup table. The inputs to the lookup table are the desired location, and the values stored inside are the four angle values. This cache approach is used due to the technical limitations of the MSP430F5529, which prevents us from calculating the angle values in real time. Furthermore, since the scale of the robot is small, rounding errors forced the need to make minor adjustments to the calculated angles, so storing all of these values enabled us to minimize errors. The verification for these integration steps is explained further in section 3.5.
3. Design Verification
The full table of requirements and verifications is shown in Table 5 in the Appendix. A summary of the conducted tests and results is given in this section.

3.1 Power Supply
To verify that the power supply met our requirements, we connected a 5 Ω load to the outputs of the supply. A 5 Ω load was used since we estimated the average current draw of our system to be around 1 A while the arm is moving. The board was then plugged into the wall and two major test points were used for verification.

The first test point was at the output of the rectification and smoothing stage. At this point the voltage was expected to average around 13.5 V (see figure 2) and with a range no larger than 10-15 V. An oscilloscope was used to measure the voltage at this point over time. After measuring voltage over periods of 1 second for 25 periods the scope capture in Figure 18 was the result. From the chart at the bottom of the scope capture, it can be seen that the average voltage is 13.3 V with a ripple averaging at 3.5 V. Both of these results fall within the 5% tolerance that was set in our requirements and verifications table, which is listed in Appendix A.

![Figure 18. Power supply stage 1 scope capture](image)

The second test point for the power supply circuit was at its output. The oscilloscope was set up to measure the voltage across the 5 Ω load. Again, a voltage measurement over periods of 1 second was used. This time the test was run for 100 seconds and the results can be seen in Figure 19. The chart at the bottom of the figure shows an average maximum voltage of 5.04 V, an average minimum voltage of 4.87 V, and an average ripple of 172.1 mV. Each of these results fits into our requirement of a voltage output of 5 V +/- 5%.
3.2 Microcontroller Board
The verification of the microcontroller board functionality involved a series of pass/fail tests showing whether or not the board could communicate with its peripherals. These tests could be done in such a binary fashion because every component of this board either functioned or did not, without any gray area. By the end of the verification phase, the board was passing every requirement. The major tests included:

- Programmability: Attempt to program the microcontroller with any code. If the programming completes (does not fail in Energia), programmability is verified.
- UART communication with computer: Program the microcontroller to count the number of times it has gone through a loop (with some loop delay) and send that number to a computer through UART. If the computer’s terminal sees an increasing series of numbers, without skips, then UART is verified.
- Blink LED: Program the microcontroller to blink both LED’s. If this output is observed, the LED’s are verified.
- Read switch and button values: Have the microcontroller print the values of the buttons and switch to UART. If the correct values are shown on the screen, while the user is changing them, then the switch and buttons are verified.
- Display “Hello World” on LCD screen: Program the microcontroller the print “Hello World” to the LCD screen. If this is seen, the LCD screen is functional.
- Communicate with Pixy via SPI: Program the Pixy to send data to the microcontroller via SPI. Program the microcontroller to send that data to the computer via UART. If the computer sees the correct data, SPI communication with the Pixy works.
- Servo control x6: One at a time, program the microcontroller to move each servo to a specified location. If the location is correct, servo control works.
3.3 Kinematics

3.3.1 Forward Kinematics
Verification of forward kinematics was straightforward. We simply confirmed that various configurations of the angles gave reasonable results. The inputs to the code were the four angle values, and the outputs of the code were the x, y, and z end location of the claw. Test cases included sweeping through all values, including the zeros, lower limits, and upper limits of each variable. Shown below in Figure 20 are a few of the test cases used during verification of the functionality of the forward kinematics equations. For each pair of test cases, we compared the configurations to validate that the x, y, and z values for each case increased or decreased, as we would expect. For example, from Figure 20b and Figure 20c, we saw that lowering angle 3 and increasing angle 4 leads to an increase in z and a slight decrease in x, which is logical.

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
a) 0, 0, 0, 0 & x= 12 \\
 & y= 0 \\
 & z= 0 \\
\hline
b) 30, 0, 0, -45 & x= 9.4 \\
 & y= 5.4 \\
 & z= 2.8 \\
\hline
c) -30, 0, -45, 0 & x= 8.4 \\
 & y= -4.8 \\
 & z= 5.7 \\
\hline
d) 0, 0, -90, 0 & x= 4 \\
 & y= 0 \\
 & z= 8 \\
\hline
e) 0, 0, 0, -90 & x= 8 \\
 & y= 0 \\
 & z= 4 \\
\hline
f) 0, 40, 0, 0 & x= 9.2 \\
 & y= 0 \\
 & z= -7.7 \\
\hline
g) 0, 0, -40, -90 & x= 4.5 \\
 & y= 0 \\
 & z= 5.6 \\
\hline
h) 0, 0, 0, -135 & x= 5.2 \\
 & y= 0 \\
 & z= 2.8 \\
\hline
\end{tabular}
\caption{Eight test cases for verification of forward kinematics}
\end{figure}

3.3.2 Inverse Kinematics
Verification of inverse kinematics was similar to that of forward kinematics. We used the same test cases as above, but we just did it in reverse and then compared the results to what we expected from forward kinematics. The inputs to the code were the x, y, and z end location of the claw, and the outputs of the code were the four angle values. Shown below in Figure 21 are a few of the test cases used during verification of the functionality of the inverse kinematics equations. Figure 21b, for example, has the claw in the same end location as Figure 20b, and 21b results in the same angles as specified in the forward kinematics of Figure 20b.
3.4 Camera Module

The main requirement for the camera module is that it can communicate over SPI with the microcontroller. Without this requirement, our systems would not have the necessary communication lines. In order to test this requirement, we loaded a known debug program onto the microcontroller and flashed one color signature to the Pixy. We expected the output from the camera to the microcontroller to be information about the block of color that the camera saw, including the center position, the width, and the height of the block: This is exactly the output that we received on the serial monitor, as shown in Figure 22, so our communication line proved to be functional.

![Figure 22. Output over SPI from Pixy to microcontroller](image)

This data output is then verified by individually checking the signature value, the x and y coordinates of the center, and the relative width and height of the object. Aside from the data transfer, it is also important for the camera module to correctly identify objects based on colors. In order to test this, distinctly colored objects were saved on the Pixy’s flash memory. Then, using PixyMon, a tool to monitor the Pixy via USB, we verified that the colored objects were correctly identified. We then restarted the Pixy and repeated the test. The first issue we saw when performing this test was that the Pixy’s settings, such as “auto white balance upon startup,” was causing color signatures to auto adjust and no longer match the intended objects. In order to fix this, we edited some settings for the Pixy and moved our setup to a room with a more stable and controllable lighting system.

3.5 Software and Integration

The main requirement for verifying the integration of our subsystems is that the robotic arm correctly detects, moves to, picks up, and sorts objects. This verification relies heavily on the correct integration of the camera module and kinematics, using the microcontroller as the module that brings these two pieces together. In order to verify that the camera module and kinematics integrated correctly, we test whether an object placed in the same location elicits the same movement from the robotic arm. Because of the scale and floating point limitations of our setup, the arm does not always move to the same location. A difference of as low as two pixels in the Pixy’s point of view could translate to a difference of an entire centimeter due to rounding error. Rounding errors also affect the translation coordinates in the robot’s point of view to joint angles. Again, since we cannot apply a floating point
value to a servo angle, we often see that rounding angles moves the arm to positions up to one centimeter away from the desired target, especially since errors propagate up the different joints.

In order to resolve this issue, we sweep through positions and visually inspect and manually adjust the angles in our mapping function.
4. Costs

4.1 Parts

Table 3. Parts Cost

<table>
<thead>
<tr>
<th>Power Supply</th>
<th>Part Number</th>
<th>Distributor</th>
<th>Quantity</th>
<th>Unit Cost</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>F20-1800-C2</td>
<td>Digike</td>
<td>1</td>
<td>$14.89</td>
<td>$14.89</td>
</tr>
<tr>
<td>Rectifier</td>
<td>GBU6J-BPMS-ND</td>
<td>Digike</td>
<td>2</td>
<td>$0.88</td>
<td>$1.76</td>
</tr>
<tr>
<td>Smoothing Capacitor</td>
<td>P14433-ND</td>
<td>Digike</td>
<td>2</td>
<td>$1.82</td>
<td>$3.64</td>
</tr>
<tr>
<td>Output Capacitor</td>
<td>P16460CT-ND</td>
<td>Digike</td>
<td>2</td>
<td>$2.36</td>
<td>$4.72</td>
</tr>
<tr>
<td>Inductor</td>
<td>SRR1210-180MCT-ND</td>
<td>Digike</td>
<td>2</td>
<td>$1.06</td>
<td>$2.12</td>
</tr>
<tr>
<td>Diode</td>
<td>B340LA-FDICT-ND</td>
<td>Digike</td>
<td>2</td>
<td>$0.54</td>
<td>$1.08</td>
</tr>
<tr>
<td>Fuse</td>
<td>F2313-ND</td>
<td>Digike</td>
<td>4</td>
<td>$0.72</td>
<td>$2.88</td>
</tr>
<tr>
<td>10nF Ceramic</td>
<td>490-1664-1-ND</td>
<td>Digike</td>
<td>10</td>
<td>$0.10</td>
<td>$1.00</td>
</tr>
<tr>
<td>4.7uF Ceramic</td>
<td>490-5422-1-ND</td>
<td>Digike</td>
<td>10</td>
<td>$0.28</td>
<td>$2.80</td>
</tr>
<tr>
<td>100nF Ceramic</td>
<td>478-3755-1-ND</td>
<td>Digike</td>
<td>10</td>
<td>$0.10</td>
<td>$1.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$35.89</td>
</tr>
</tbody>
</table>

**Microcontroller Board**

<table>
<thead>
<tr>
<th>Part</th>
<th>Distributor</th>
<th>Quantity</th>
<th>Unit Cost</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>MSP430F5529</td>
<td>TI</td>
<td>1</td>
<td>$4.95</td>
</tr>
<tr>
<td>Linear Regulator</td>
<td>TPS79933</td>
<td>TI</td>
<td>1</td>
<td>$1.95</td>
</tr>
<tr>
<td>Power Jack</td>
<td>CP-002AH-ND</td>
<td>Digike</td>
<td>1</td>
<td>$0.98</td>
</tr>
<tr>
<td>Electrolytic Capacitor</td>
<td>PCE4156CT-ND</td>
<td>Digike</td>
<td>1</td>
<td>$1.52</td>
</tr>
<tr>
<td>.1' Header</td>
<td>N/A</td>
<td>Digike</td>
<td>50</td>
<td>$0.01</td>
</tr>
<tr>
<td>Green LED</td>
<td>67-1557-1-ND</td>
<td>Digike</td>
<td>1</td>
<td>$1.75</td>
</tr>
<tr>
<td>Red LED</td>
<td>67-1556-1-ND</td>
<td>Digike</td>
<td>1</td>
<td>$2.00</td>
</tr>
<tr>
<td>Switch</td>
<td>N/A</td>
<td>Digike</td>
<td>1</td>
<td>$0.10</td>
</tr>
<tr>
<td>Button</td>
<td>N/A</td>
<td>Digike</td>
<td>2</td>
<td>$0.10</td>
</tr>
<tr>
<td>Ceramic Capacitors</td>
<td>N/A</td>
<td>Digike</td>
<td>15</td>
<td>$0.05</td>
</tr>
<tr>
<td>Resistors</td>
<td>N/A</td>
<td>Digike</td>
<td>3</td>
<td>$0.05</td>
</tr>
<tr>
<td>Potentiometer</td>
<td>3314J-103ECT-ND</td>
<td>Digike</td>
<td>1</td>
<td>$2.08</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td>$16.93</td>
</tr>
</tbody>
</table>

**Other**

<table>
<thead>
<tr>
<th>Part</th>
<th>Distributor</th>
<th>Quantity</th>
<th>Unit Cost</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robotic Arm</td>
<td>N/A</td>
<td>eLab Peers</td>
<td>1</td>
<td>$120.00</td>
</tr>
<tr>
<td>Pixy Camera</td>
<td>N/A</td>
<td>Adafruit</td>
<td>1</td>
<td>$75.00</td>
</tr>
<tr>
<td>PCB</td>
<td>N/A</td>
<td>Advanced Circuits</td>
<td>1</td>
<td>$50.00</td>
</tr>
<tr>
<td>Camera Mount</td>
<td>N/A</td>
<td>ECE Machine Shop</td>
<td>1</td>
<td>$70.00</td>
</tr>
<tr>
<td><strong>Grand Total</strong></td>
<td></td>
<td></td>
<td></td>
<td>$367.82</td>
</tr>
</tbody>
</table>
## 4.2 Labor

Table 4. Labor Costs

<table>
<thead>
<tr>
<th>Name</th>
<th>Hourly Rate</th>
<th>Total Hours Invested</th>
<th>Total = Hourly Rate x 2.5 x Total Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anusha Nagabandi</td>
<td>$30</td>
<td>150</td>
<td>$11,250</td>
</tr>
<tr>
<td>Danny Coombs</td>
<td>$30</td>
<td>150</td>
<td>$11,250</td>
</tr>
<tr>
<td>Sid Sethupathi</td>
<td>$30</td>
<td>150</td>
<td>$11,250</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>450</td>
<td>$33,750</td>
</tr>
</tbody>
</table>
5. Conclusion

5.1 Accomplishments
We succeeded in combining concepts of kinematics, embedded systems, and hardware design to create a sorter. Our verifications show the functionality of each of our components. Furthermore, our demo shows that our system can recognize various colored objects, do the calculations to physically move the arm to that point, pick up and drop objects is specified end locations, and distinguish objects to sort them into different locations. The complete integration of our components was successful, aside from the current spikes that made it unsafe for us to connect our power supply to our system. Possible future work to further enhance our design is explained below in section 5.4.

5.2 Ethical Considerations
Operating and conducting ourselves in an ethical manner during all phases of projects is a crucial part of being engineers. To help our decision-making, we consult and follow the IEEE code of ethics [9]. Below are a few of the points that are directly relevant to our project.

1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.
We take many steps to reduce any risk to the user that is presented by our project. In addition to careful design of the electronics to mitigate risk to the user, we lay out possible safety hazards for the user. The two main safety hazards come from the 120V power source, and the moving parts of the arm. First, we minimize the risk of electric shock: We follow proper techniques such as grounding the circuits, and we ensure that all users adhere to general policies such as no water near the circuit. Second, we minimize any risk from the mechanical system. In general, we just follow proper safe etiquette such as being cautious to not bump into the arm while its running, keeping long hair tied back, and rolling up long sleeves. We also urge users to do the same. We ensure that long cords and wires are all tied back properly. Also, the area in which we run the arm is clutter-free so nothing can get tangled in it during operation. Additionally, we have a power button so the user can start or stop the operation of the arm at any time they want.

3. To be honest and realistic in stating claims or estimates based on available data.
We reported our data honestly and did not falsify any data.

5. To improve the understanding of technology; its appropriate application, and potential consequences.
We worked with many components and many different types of technology, so we definitely worked to improve the understanding and integration of technology. Furthermore, as explained in the introduction section of this document, we hope that this project can be applied to more complex situations. We aim to have this framework serve as an easy stepping-stone for further development, so we are working toward improving the application of such technology.

7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.
We received feedback from our TA, other TA's, professors, and other students. We both sought out and appreciated this criticism of our work, and we appropriately applied the feedback to our project.

5.3 Future Work
As mentioned in the introduction, this project serves as a general setup of the hardware and software framework, which can be easily extended for future implementation. Future work includes editing code to adapt this programmable sorter to sort various objects by size, shape, or color. This could include Legos, coins, wires, or even more complicated objects such as silverware, pencils and pens, and various types of shoes. We can also integrate the IR depth sensor with our system to enable working with piles of clothing, instead of dealing only with objects that are flat on the bottom of the basket.

Additionally, we can use a more powerful microcontroller. As discussed in the section 3.5, the MSP430 has no native floating point numbers, which led to some rounding errors with our values. Using the new MSP432 would reduce our errors and allow us improved functionality.

Furthermore, we can make changes to allow the use of a custom power supply. As discussed in section 3.5, our current power supply cannot handle current spikes greater than 3A, so in order to safely use it with our system, we could redesign the power supply with a higher current rating. An even better approach is to redesign our microcontroller board with a servo controller, or any similar device, to completely detach the microcontroller from the servos during reprogramming or power-up of the microcontroller. This way, we limit our current spikes, and we would also be able to use our power supply to power our system.
References


### Appendix A  Requirement and Verification Table

Table 5. System Requirements and Verifications

<table>
<thead>
<tr>
<th>Module</th>
<th>Requirement</th>
<th>Verification</th>
<th>Verification status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power System</td>
<td>1. Wall outlet transform to correct voltage:</td>
<td>1a. Probe output of smoothing capacitors (input to switching regulator) with voltage meter and output to oscilloscope</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a) V&lt;sub&gt;out&lt;/sub&gt; = 12VDC ± 5%</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>2. Transformed down voltage regulates to correct voltage on 5V line through switching regulator</td>
<td>2a. Probe input of switching regulator with voltage meter and output to oscilloscope</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>a) Vin = 12VDC ± 5%</td>
<td>2b. Probe output of switching regulator circuit with voltage meter and output to oscilloscope</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b) Vout = 5VDC ± 5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. The 5V regulates down to 3.3V through linear regulator</td>
<td>3a. Probe output of the switching regulator circuit with voltage meter and output to oscilloscope</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>a) Vin = 5VDC ± 5%</td>
<td>3b. Probe output of linear regulator with voltage meter and output to oscilloscope</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b) Vout = 3.3VDC ± 5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peripheral IO</td>
<td>3. Start button communicates with microcontroller correctly</td>
<td>3a. Leave the button unpressed and probe START_BTN (bouncing is okay - debounced in software). Ensure that state is not changing by looking at status LEDs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a) When the button is unpressed, microcontroller program state is unchanging</td>
<td>3b. Push the button and probe START_BTN (bouncing is okay - debounced in software). Ensure that microcontroller program state has reset by looking at status LEDs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b) When the button is pressed, microcontroller program state resets to initial state</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Mode select communicates with the microcontroller correctly.</td>
<td>4a. Flip mode select switch to Mode 0. Probe MODE_0 and MODE_1 with voltage meter. Verify MODE_0 is</td>
<td>Y</td>
</tr>
</tbody>
</table>
2. When Mode 1 is physically selected, uC is provided with Mode 1 designation.

4b. Flip mode select switch to Mode 1. Probe MODE_0 and MODE_1 with voltage meter. Verify MODE_0 is digital low and MODE_1 is digital high. Verify through uC code that Mode 0 is selected.

5a. Write, load, and run microcontroller program to send high signal to LED 0. Visually inspect that LED 0 lights up.

5b. Write, load, and run microcontroller program to send low signal to LED 0. Visually inspect that LED 0 stays off.

6a. Write, load, and run microcontroller program to send high signal to LED 0. Visually inspect that LED 0 lights up.

6b. Write, load, and run microcontroller program to send low signal to LED 0. Visually inspect that LED 0 stays off.

7. Connect Pixy CMUcam5 to the SPI header on the board. Load program on uC to capture object block specified by Table 1. Verify data in object block is valid.

8a. Connect Pixy CMUcam5 to computer. Run PixyMon. Hold white object in front of camera module and run blob detection algorithm. Verify object identified as blue. Repeat 100 times.
objects with 95% accuracy
3. Identifies red objects with 95% accuracy

8b. Connect Pixy CMUcam5 to computer. Run PixyMon. Hold white object in front of camera module and run blob detection algorithm. Verify object identified as green. Repeat 100 times.

8c. Connect Pixy CMUcam5 to computer. Run PixyMon. Hold white object in front of camera module and run blob detection algorithm. Verify object identified as red. Repeat 100 times.

<table>
<thead>
<tr>
<th>Computing System Module</th>
<th>9. Given servo motor angles, communicate correctly with motors to send appropriate PWM signals (angles should map to a position in the basket).</th>
</tr>
</thead>
<tbody>
<tr>
<td>9. Intelligently guess motor angles and physical representation of the angles. Input angles into uC code. Verify physical representation is similar to result of the movement.</td>
<td></td>
</tr>
<tr>
<td>10. Input (x,y) to microcontroller. Calculate/lookup angles. Set motors to correct angles. Verify that arm moves to desired (x,y) position.</td>
<td></td>
</tr>
<tr>
<td>11. Given a image of the basket from above, match camera coordinate system with robot coordinate system through calibration routine (mapping pixels to centimeters and matching axes).</td>
<td></td>
</tr>
<tr>
<td>11. Place object in a specific position in the robot coordinate system. View marker from camera. Transform camera coordinates of marker to robot coordinates. Verify that calculated coordinates matches known coordinates</td>
<td></td>
</tr>
<tr>
<td>12. Maintain program flow during execution.</td>
<td></td>
</tr>
<tr>
<td>12. Ensure that nothing happens until after calibration occurs. Ensure that nothing moves until after start button is pressed. Ensure that calibration does not occur</td>
<td></td>
</tr>
<tr>
<td>Mechanical System</td>
<td>13. The apparatus is assembled correctly. This entails that the three following components are addressed: The arm is attached to the basket; the camera is attached to a rigid holder looking down on the basket.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>14. System is physical stable while motors are moving.</td>
<td>14. After attaching the arm to the basket, move the motors and ensure that neither the arm falls due to its weight, nor the basket moves due to the arm's movement.</td>
</tr>
<tr>
<td>15. Claw opens and closes on command.</td>
<td>15. Send PWM commands from microcontroller to the servo that controls the claw. Ensure that the claw was assembled correctly by confirming that the servo opens and closes the claw.</td>
</tr>
<tr>
<td>16. Arm can lift objects without having the objects slip out. Lift our largest piece of clothing (15cm long) &lt;15% drop rate (metric to determine if the arm picks up an object and drops it before correctly identifying it and placing it in the right bin)</td>
<td>16. Control the motors to make the arm lift objects of various size, shape, and texture and ensure that it can actually lift the objects. If things are slipping out, add material to the claw for more friction.</td>
</tr>
<tr>
<td>17. Motors can move for claw to reach any part of the basket.</td>
<td>17. Use computing system to move servos across their entire range of motion. Measure this 3D range and compare to dimensions of basket. Results can be specified as numbers.</td>
</tr>
</tbody>
</table>
Appendix B  Schematics

Figure 23. Microcontroller schematic page 1
Figure 25. Power supply schematic