Robotic Arm Project

Introductions

The aim of this project was to develop a robotic arm that is able to be controlled and move to a desired position. To achieve this, I used an MSP430 microcontroller to receive input coordinates via a serial terminal and rotate motors in the arm accordingly. The arm is able to rotate around the x and y axes, allowing it to move in a range of directions. Our program was designed to accept input coordinates in the range of [-5,5] and can rotate the motors clockwise or counterclockwise as needed.

Hardware Components

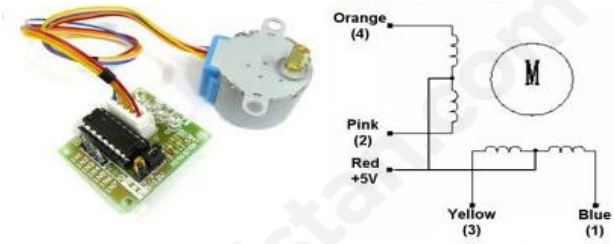
- Msp430G2553 development board
- 2 x 28BYJ-48 step motor
- 2 x ULN2003 motor driver
- 2 x Custom 3d printed robot arm and gear system
- Jumper cables
- Buzzer

For this project, I chose to use 28BYJ-48 step motors to control the movement of the robotic arm. These motors are able to rotate multiple times 360 degrees in both the clockwise and counterclockwise directions, making them well-suited for our needs. In addition, they are relatively affordable and offer good performance. To drive the motors, I used ULN2003 motor drivers. These drivers are compatible with the 28BYJ-48 step motors and can be powered by a voltage range of 5-12V, allowing us to drive them using the MSP430 microcontroller without the need for an external power supply.

Since I were unable to find suitable mechanical components for the arm online, I designed and printed our own using a 3D printer. I also added a simple buzzer to the system to make it more noticeable when in operation.

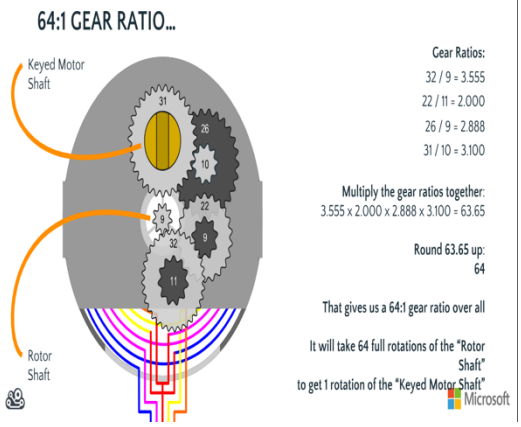
Software Design

To control the step motors in the robotic arm, I wrote a custom library with functions for clockwise and counterclockwise rotation and parameters to adjust the motor speed. I based the library on the datasheet for the motors and learned about how step motors work. Step motors have four inputs that control the electromagnets around the rotor, and they can be driven in either full step or half step mode. I implemented both modes in the library, but ultimately chose to use full step mode because it provides more torque at the cost of speed



Half-Step Switching Sequence

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-



To achieve one full rotation of the motor shaft, the inner rotor must complete 512 cycles of the sequence described above, based on the gear ratio shown in the image. In the software architecture, the program keeps track of the current arm position and uses while loops to move the arm to the desired position, which must be within the range of [-5,5]. The reset position for the arm is (0,0), and the program allows the user to input new coordinates via the serial terminal. When the button on the MSP430 is pressed, the program moves the arm to the reset position (0,0).

The x arm has priority, so the y arm will only move after the x arm has reached its destination. If the reset button is pressed at any time, the reset process will wait until the current process is completed before initiating.

In terms of code, there are three files for this project. The main.c file contains the main program, which includes the oscillator setup, motor pin setup, RX TX serial terminal pin setup, and interrupt setups for the x arm, y arm, and reset interrupt. I also included the main while loop, which waits for input from the user to get the x and y coordinates. If-else statements are used to initiate software interrupts for the x and y arms.

The my_motor_lib.h file is the header file, which has definitions and declarations. The third file is the source file my_robot_lib.c which contains the code I wrote to drive the motors. It has two functions for each motor to rotate in the +x, -x directions and +y, -y directions. The interrupts for P1 pins, which have the button and x arm interrupt, and the interrupt for P2 pin, which has the y arm interrupt, are also in this file.

It also has a parse string function, which takes user input and checks to make sure it is in the correct format. The input string must be five characters long, with the first and fourth characters being a sign (+ or -) that indicates the direction, and the second and fifth characters being a number between 0 and 5 that indicates the coordinate. The third character must be a comma to separate the x and y values.

Additionally, each motor function includes a buzzer effect with an adjustable ratio using a divider variable, and there is a reset buzzer function to create an effect when the reset occurs.

How it Works

When the program starts, it waits for input. If the input is not in the correct format, the user receives an error message and the program continues to wait for a properly formatted input. Once the correct input is received, the program compares the current position of the arms with the desired coordinates. If the coordinates are different, arm interrupts are initiated with x having priority, meaning the x arm will move first until it reaches its desired position, then the y arm will do the same. If the button is pressed at any time, a button interrupt is initiated, signaling that the user wants to move the arms to the reset position (0,0). However, this interrupt will only be processed after the current motor rotations are complete.

While the motors and arms are in motion, there is a buzzer creating a sound to draw attention. When the reset occurs, a different pattern of sound is produced. After every movement and interrupt is completed, the program waits for new input.


ÇUKUROVA
ÜNİVERSİTESİ
ÖĞRENCİ
STUDENT