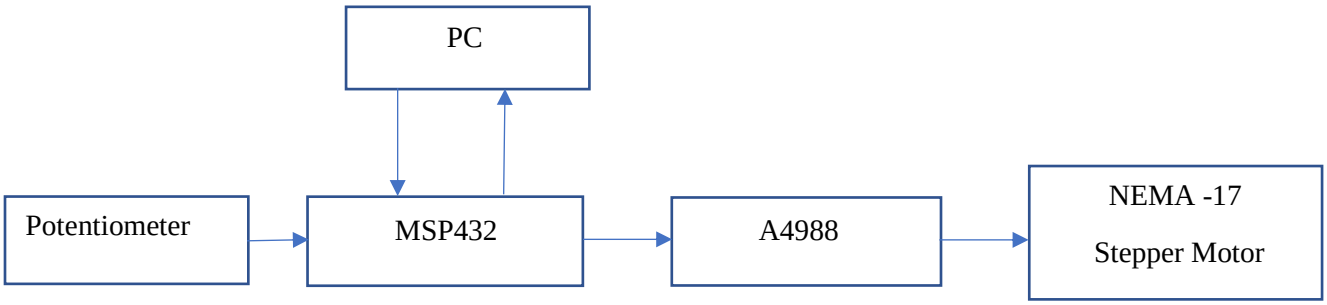
# CE6302 PROJECT REPORT

**Submitted By**
**Gokul Balagopal**
**gxb161030**

## Introduction

The aim of the project is to control a stepper motor using MSP432 and to send the updates regarding the direction changes of the stepper motor over the serial port using UART communication protocol. The motor used is a NEMA17 stepper motor. It is a two-phase stepper motor. The speed of the motor is increased /decreased using a potentiometer. The direction of the motor is switched using a button interrupt. This direction change is transmitted to the PC through the serial port

## Block Diagram




**Potentiometer**

10K Potentiometer is connected to Pin 5.4 to adjust voltage, send to the ADC of MSP432. One end is connected to VCC (3.3V) and other end is connected to the ground.

**MSP432**

It is programmed in embedded C to control the functionalities of the motor and to send the data over the serial port using UART communication protocol.

**Bipolar Stepper Motor (NEMA - 17)**

It has 4 wire connections. It needs voltage reversal to reverse the direction. It has high torque because it makes full use of the coil windings within it. This has a slower maximum speed because the coils have more inductance. This requires a more advanced controller because of the voltage reversal.

**A4998**

The A4988 is a complete micro stepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A.

**PC**

The code is run on CCstudio software in the PC and the data indicating the motor direction is sent by the MSP432 to the PC via the serial port. It is displayed in the terminal.

## Working and Solution

### Software

We start off by loading the msp432 libraries. We initialize the flag variable *fl* for detecting the change in direction of the motor. I have also initialized string variable *TXData* for sending the message over the serial port. Next the Watch Dog Timer is stopped. We activate the GPIO ports, for changing the motor direction(Port 4.6),providing the signal to the ADC (Port 5.4), generating the PWM wave (Port 2.7), for data transmission (Port 1.3) through the serial port and for generating Button Interrupt (Port 1.4).

*Timer A Configuration*

We use Timer A for controlling the PWM wave produced by Port 2.7. We use the SM clock in this case. The Timer A is configured as follows: Counting mode is set as UP Mode, the TA0R register is cleared and we use CCR4 register in reset/set output mode.

*UART Configuration*

We configure the serial module eUSCI_A0 by putting it in reset. We use 9600as our Baud rate. We turn off the modulation. Now we initialize the serial module.

*ADC (Analog to Digital Converter) Configuration*

We configure the ADC by setting the sample and hold timer, sample and hold pulses (they are in pulse sample mode) and the ADC control is turned ON when a valid conversion is triggered. Using the sampling timer, we do the 12 bit conversion. The ADC input channel is selected. The reference voltage selected is 3.3V. We enable the global interrupt for the ADC (Analog to digital converter). We enable the interrupt for the ADC. The ADC interrupt occurs once the sample conversion is complete, till then it remains in low power mode 0 (LPM0). The converted data is written to the ADC memory. Once we return from the interrupt it comes out of the LPM0.

*Interrupts and Serial Communication*

From the ADC memory we check if the results we obtain by conversion lies within a range. If so, we adjust the CCR0 value to increase/ decrease the speed of the stepper motor and adjust the duty cycle by changing the CCR4 value (this is not mandatory, we can provide a constant duty cycle).Next we check if a button press occurs by checking whether the button interrupt flag is set. If it is set, we flip the motor direction by flipping the existing value in port 4.6. Once that is done, we flip the *fl* flag. If the *fl* flag is low, then we know that the motor is rotating in the anticlockwise direction else the motor is rotating in the clockwise direction. We copy this message onto the variable called *TXData.* Next, we check if the interrupt flag for the serial communication and the transmit interrupt flag is set or not. If not set, we send the data through the serial port and it can be seen in the Serial Port of our PC. Hence the message which indicates the direction is transmitted for each button press and it changes the direction of rotation of the motor. This data can be viewed in Putty/Terminal of CC Studio. We also provide delays after each button press to avoid de-bouncing. We also clear the button interrupt flag for each button press.

**Hardware**

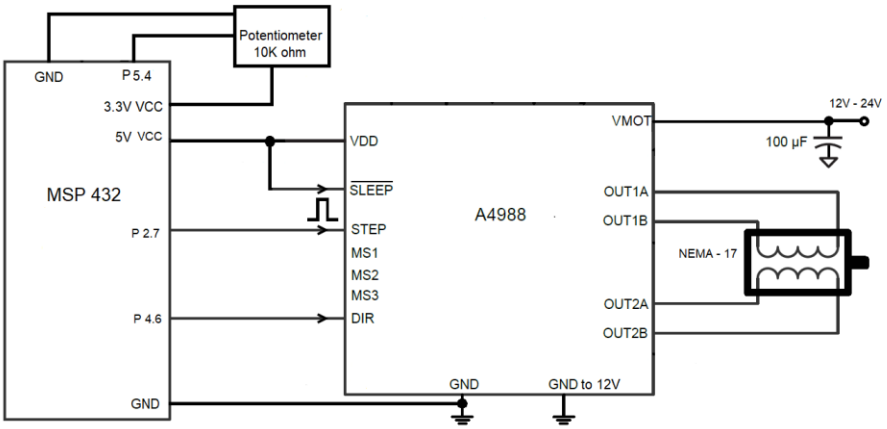



Fig.1

The circuit is set up as shown in the figure. Here we use a potentiometer to adjust the DC signal flowing into the MSP432s ADC which in turn triggers the ADC to produce the ADC counts. One end of the potentiometer is grounded while the other end is given to the 3.3V on the board. (The maximum operating voltage for MSP432 is 3.7V). The ADC counts are used for adjusting the motor speed by controlling the PWM wave. The PWM wave is given to the A4988 stepper motor driver. The motor is powered by supplying 12V to VMOT of A4988 through a 100µF capacitor. The other end of the capacitor is connected to the ground (Note: Don't short it with the ground of the µC). The direction of the stepper motor is controlled by connecting the Port 4.6 to the DIR pin to the controller. So, when the signal is high the motor rotates in the "Clockwise direction". When it is low it rotates in the "Anticlockwise direction". We set the SLEEP pin of the controller high because by default it is active low. Here, we are not using the MS1, MS2, MS3 pins (used to control step sizes). The pins A+, A-, B+, B-, are connected to the output pins of the driver

## Challenges / Problems Experienced

To start off with, the lack of access to the lab, getting the components on time was a big challenge. I started ordering all the components on November 10th. I received most of them by November 17th. I had to buy a DSO and power supply to do the project which was a bit costly.

Secondly designing the circuit to drive the motor was a big challenge as no similar designs for MSP432 was available online. I had to drive a NEMA 17 stepper motor, whose current rating was 2A and the operating voltage range was from 12V to 24V. So, I followed steps required to run a stepper motor using Arduino. Initially I was using a motor driver called L293D which was commonly used with Arduinos to drive stepper motor. While using that driver my MSP432 stopped working due to some short circuit. This happened on 21st of November, I had to order a new MSP432 which came around 27th. This caused a delay in the project. I ordered the TB660 and A4988 along with this. The TB660 did not work. I used the A4988 to control the motor.

Upon discussing with Dr. Bennet on November 30th, we decided to add a new functionality to the project. A message indicating the change in direction of motor upon button press, will be sent to

the PC via the serial port. The motor direction will be displayed in terminal / putty. While doing this my second board stopped working as I unknowingly over clocked it while sending the data over the serial port. I had to reorder a new board again. This time I got the board a bit earlier than usual as I had to pay extra money for express delivery. My DSO also stopped working few days back. I also had issues figuring out the required baud rate as the TI example codes where all designed for high frequency applications. Not only did it not work, as well as caused so many issues with my board.

## Overcoming Difficulties and things learned

The concepts of UART, ADC, Timers, Low power modes, PWM wave generation, and interrupts were reviewed through this project. I was able to understand the working of UART transmission through this project. Initially I had trouble figuring out the working of UART due to the lack of proper example codes in the TI website. The one's they had did not working when I ran them due to the clocking issue. So, I had to go through the TRM to figure out the issue with my Baud rate. The clocking in the examples were all for SM clock at 48MHz.While my SM clock only delivered 3Mhz.It helped me overcome some of the issues. I also had issues with sending out string data. Initially I assigned the character updating logic in the wrong part of the loop. But later I fixed it. I removed the overmodulation as it was not required for lower frequencies like 3MHz.I realized this after going through the reference manual. When it came to the hardware, I had issue trying to run the motor with A4988 as the current flow to the motors where low. I had to supply the adequate amount of current by adjusting the potentiometer on the A4988 board. I also realized I had to turn the sleep pin high every time I run the motor. I was able to figure this out after going through the A4988 data sheet as well as some videos related to this in YouTube.

## Conclusion

All the major concepts relating to MSP432 that was taught during semester was implemented in this project. It helped me review the concepts of Interrupt, ADC, Low Power Modes, Timers, and UART. I was able to do some hardware design for this project by figuring the voltage and current requirements of the Nema17 stepper motor. Even though I struggled a lot with getting the components and trying to fix the MSP432 board that crashed multiple times. I believe it was worth it. As I was able to do the following: Increase/Decrease the speed of the motor using a potentiometer. The direction of the motor was reversed for each button interrupt. I was able to implement transmission of data to the PC using the concept of UART in MSP432

## Extension/Improvement to the Project

My thoughts regarding the extension and improvement of the project are as follows:

- Provide a proper voltage regulator to stabilize the input.
- Use an LCD to display the statistics of the motor.
- Use this motor to control gears for a 3D printer/robot buy adjusting its speed, direction, step size as per our requirement.

# References Used


- https://www.ti.com/lit/ug/slau356i/slau356i.pdf?ts=1607468093499
- https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf
- https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/
- https://www.omc-stepperonline.com/download/17HS16-2004S1_Torque_Curve.pdf
- https://www.omc-stepperonline.com/download/17HS16-2004S1.pdf