MEASURING THE TURRET ANGLE

1. SENSING THE ROTATIONAL ANGLE

In a control feedback system, three essential actions take place: 1) sense, 2) decide, and 3) actuate. We have tackled the actuation portion of the system (that is, we can specify a duty cycle and direction and move the turret system via the DC motor and TD340 motor driver). Now, we must be able to sense what *angle* the motor turret has rotated to (we will denote this angle as θ).

4 wire cable coming from the rotary position sensor.

Figure 1: Turret sub-system

Q1: What are two types of <u>*sensors*</u> that can measure rotary angle?

Q2: What type of sensor is attached to the back of the GMX-6MP009A DC motor?

Q3: For the sensor connected to this DC motor, how many wires are required to relay the information to the mBed?

Q4: For each wire required, describe the signal that is being conducted (also make sure to note the orientation of all the wires – Norm Tyson has utilized the black wire in the cable as GND or COMMON as a reference).

Q5: For *ANY* rotational sensor, there has to be a calibration expression/relationship. Specifically, you are looking for an expression that relates the resolution of the rotational angle displacement to the type of measurement of the sensor (e.g., voltage or pulse or something else). From the data sheet, locate this information and express it below.

2. CONNECTION TO THE MBED

As mentioned above, the sensor utilizes four wires. Shown in Figure 2 is a FIVE pin male header (there will be one no connect (NC) pin within that bundle) that connects the mBed processor with the rotary sensor.

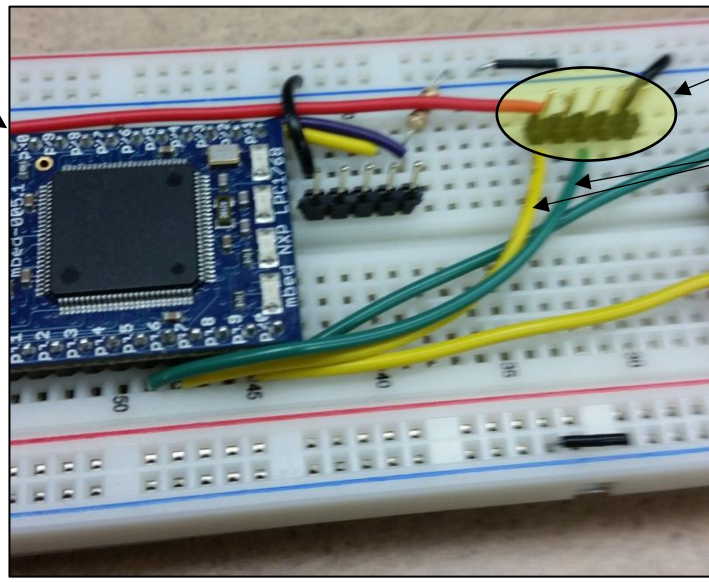


Figure 2: Connection to mBed

Select two pins of the mBed to serve as inputs for Channel A and Channel B of the encoder. Connect your GND and 5V lines appropriately.

IMPORTANT: The header for the TD340 motor driver looks just like the header pin for the encoder interface. It is recommended that you **label each header** in some fashion so that you don't inadvertently plug the encoder into the interface for the TD340 board.

Also, the header pins are not KEYED, so make note of the orientation of the cable so that you don't plug the cable in BACKWARDS. Use the GND wire as a reference.

Once complete, your system will look similar to Figure 3.

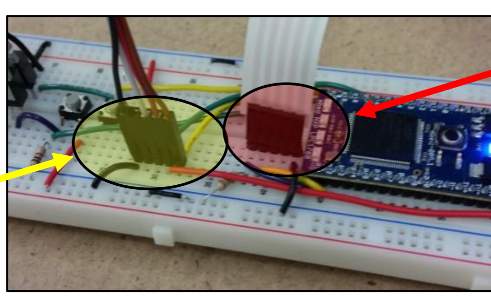


Figure 3: Encoder connection to mBed processor

3. READING AN ENCODER ON THE MBED

NEW TOOLS
QEI encoder library

In order to read the encoder, the library QEI will need to be imported into your project.

Online, you can quickly locate the documentation and an example program for the QEI library by searching "QEI library mbed".

Your task now is to write a simple program that will read the encoder sensor and then convert that measurement into a calibrated rotational angle $\theta(rad)$. You will need the calibration relationship discussed above. Use the example program that you found online as a guide.

IMPORTANT: To be consistent, we will adopt a CLOCKWISE (CW) rotation (as viewed looking DOWN on the turret) as positive rotation.

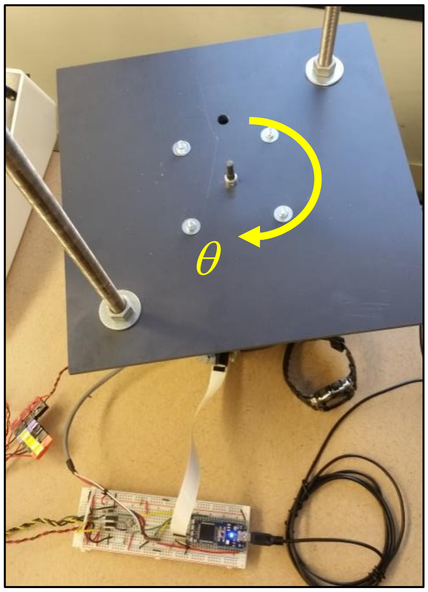


Figure 4: Positive rotation for turret

Q6: If your measurement is backwards (your θ is negative for CW rotation), describe a SOFTWARE method that can remedy the situation.

Demonstrate to your instructor the calibrated angle measurement by rotating the turret positive $90(\deg)$ (manually) and displaying the measured angle in Terra Term.

4. LOGGING EXPERIMENTAL DATA

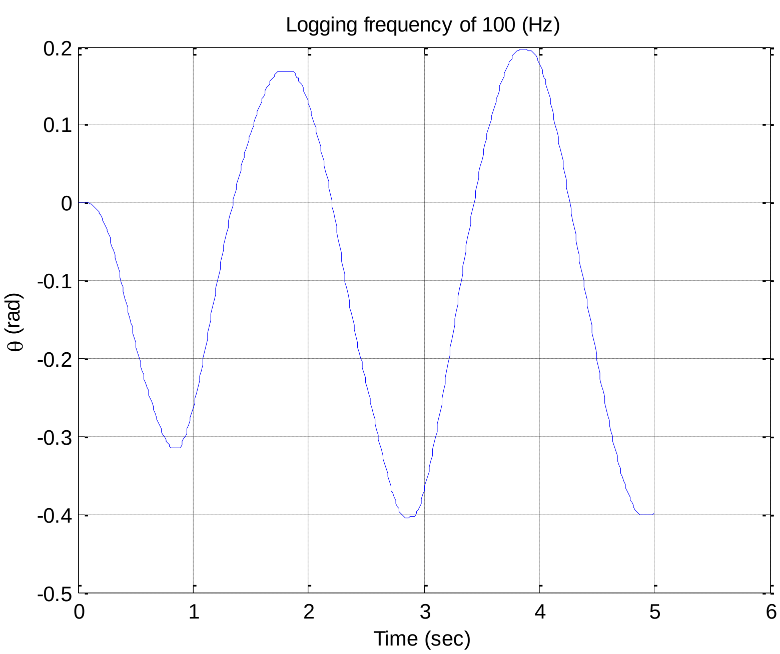
We are now at a point where we may wish to save measurements from the encoder, export them from the mBed, and then analyze them in MATLAB.

NEW TOOLS

The *ticker* class
The *timer* class

Your tasks are to:

1.) Measure <u>exactly</u> 5.0(sec) of a *manually* moved (approximate) sinusoidal motion of the rotor displacement angle θ.
2.) Save/log displacement angle θ and the TIME at which it was logged. We need to get measurements at a rate of 100.0(Hz).
3.) Export the data from your program in an efficient manner for plotting in MATLAB.
4.) Import the data into MATLAB.
5.) Plot the displacement angle θ vs. time in a MATLAB figure. Like this:



6.) Show to your instructor that the data was actually logged at a rate of 100.0(Hz). (the difference in time between each data point should be 0.01 sec). Use the MATLAB command: `diff(logged_time)`
7.) Prepare a brief memo that includes:
   i) your documented code
   ii) a labeled graph of the output of the encoder in MATLAB
   iii) a description of the method you used to collect the data from the mbed and import it into MATLAB
   iv) the answers to questions 1-6, typed

<u>**Things to think about once you get your first method to work:**</u>

- Do you have ROBUST code? If you run your program for 10(sec), will it continue to attempt to log the data? We don't want that. Why?

- The mBed also functions like a USB drive. You can store files on it. Can we take advantage of that?
- The printf command allows you to FORMAT your print statements. What if we made output look like MATLAB commands? Would that streamline the process?